KANIYAM FOUNDATION

கணியம் அறக்கட்டளை

Python Tips

v 1.0

Compiled by

T Shrinivasan

tshrinivasan@gmail.com

https://twitter.com/clcoding

https://codecut.ai/posts-by-categories/

```
pip install countryinfo
```

Country Details using Python

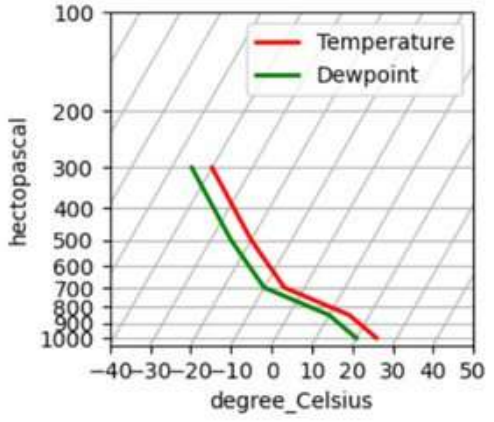
[8]:

```
from countryinfo import CountryInfo
country = CountryInfo(input("Enter Country Name:"))

# Various information about the country
print("Country Name:", country.name())
print("Capital:", country.capital())
print("Population:", country.population())
print("Area (in square kilometers):", country.area())
print("Region:", country.region())
print("Subregion:", country.subregion())
print("Demonym:", country.demonym())
print("Currency:", country.currencies())
print("Languages:", country.languages())
print("Borders: ", country.borders())
#clcoding.com
```

```
Enter Country Name: USA
Country Name: united states
Capital: Washington D.C.
Population: 319259000
Area (in square kilometers): 9629091
Region: Americas
Subregion: Northern America
Demonym: American
Currency: ['USD', 'USN', 'USS']
Languages: ['en']
Borders:  ['CAN', 'MEX']
```

Plotting Skew-T Log-P Diagram:

```python
import numpy as np
import matplotlib.pyplot as plt
from metpy.plots import SkewT

# Sample data for plotting
pressure_levels = np.array([1000, 850, 700, 500, 300]) * units.mbar
temperature_levels = np.array([25, 15, -5, -20, -40]) * units.degC
dewpoint_levels = np.array([20, 10, -10, -25, -45]) * units.degC

# Plotting
fig = plt.figure(figsize=(3, 3))
skew = SkewT(fig)

skew.plot(pressure_levels, temperature_levels, 'r', linewidth=2, label='Temperature')
skew.plot(pressure_levels, dewpoint_levels, 'g', linewidth=2, label='Dewpoint')

plt.legend()
plt.show()

#clcoding.com
```



Check Battery status using Python

[8]:

```
import psutil

battery = psutil.sensors_battery()

if battery is not None:
    print("Battery Percentage:", battery.percent, "%")

    print("Power plugged in:", battery.power_plugged)

    def convertTime(seconds):
        minutes, seconds = divmod(seconds, 60)
        hours, minutes = divmod(minutes, 60)
        return "%d:%02d:%02d" % (hours, minutes, seconds)

    print("Battery remaining time:", convertTime(battery.secsleft))
else:
    print("No battery information available.")

#source Code --> clcoding.com
```

```
No battery information available.
```

Detect the language using Python

```
pip install langdetect
```

[19]:

```
from langdetect import detect, DetectorFactory

DetectorFactory.seed = 0

text = input("Enter a senetence to detect: ")

detected_language = detect(text)

print(f"The detected language is: {detected_language}")

#source code --> clcoding.com
```

```
Enter a senetence to detect:  Hallo zusammen
The detected language is: de
```

Extract Text from Image using Python

[ ]:

```
pip install pytesseract Pillow
```

[1]:

```
from PIL import Image
Image.open("pyclcoding.jpg")
```

[1]:

[9]:

```
from PIL import Image
import pytesseract

# Open an image file
image_path = 'clcodingjpg.jpg'
image = Image.open(image_path)
extracted_text = pytesseract.image_to_string(image)
print(extracted_text)

#clcoding.com
```

```
at py Diwali

CLCODING.COM
```

/clcoding

/Pythoncoding

/Pythonclcoding

7 LESSER-KNOWN PYTHON TECHNIQUES ABOUT LISTS

1. Flatten a Nested List

Flatten a deeply nested list into a single list of elements.

[6]:

```
from collections.abc import Iterable

def flatten(lst):
    for item in lst:
        if isinstance(item, Iterable) and not isinstance(item, str):
            yield from flatten(item)
        else:
            yield item

nested_list = [1, [2, 3, [4, 5]], 6]
flat_list = list(flatten(nested_list))
print(flat_list)
```

```
[1, 2, 3, 4, 5, 6]
```


2. List of Indices for Specific Value

Get all indices of a specific value in a list.

[9]:

```
lst = [10, 20, 10, 30, 10, 40]
indices = [i for i, x in enumerate(lst) if x == 10]
print(indices)
```

```
[0, 2, 4]
```

3. Transpose a List of Lists (Matrix)

Transpose a matrix-like list (switch rows and columns).

[12]:

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
transposed = list(map(list, zip(*matrix)))
print(transposed)
```

```
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```


7 LESSER-KNOWN PYTHON TECHNIQUES ABOUT LISTS

4. Rotate a List

Rotate the elements of a list by n positions.

```python
def rotate(lst, n):
    return lst[-n:] + lst[:-n]

lst = [1, 2, 3, 4, 5]
rotated_lst = rotate(lst, 2)
print(rotated_lst)
```

```
[4, 5, 1, 2, 3]
```

5. Find Duplicates in a List

Identify duplicate elements in a list.

```python
from collections import Counter

lst = [1, 2, 2, 3, 4, 4, 4, 5]
duplicates = [item for item, count in Counter(lst).items() if count > 1]
print(duplicates)
```

```
[2, 4]
```

6. Chunk a List

Split a list into evenly sized chunks.

```python
def chunk(lst, n):
    for i in range(0, len(lst), n):
        yield lst[i:i + n]

lst = [1, 2, 3, 4, 5, 6, 7, 8]
chunks = list(chunk(lst, 3))
print(chunks)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8]]
```

7. Remove Consecutive Duplicates

Remove consecutive duplicates from a list, preserving order.

```python
from itertools import groupby

lst = [1, 2, 2, 3, 3, 3, 4, 4, 5]
result = [key for key, _ in groupby(lst)]
print(result)
```

```
[1, 2, 3, 4, 5]
```


```
pip install winshell
```

Create a Shortcut on Desktop

[71]:

```
import winshell
import os

desktop = winshell.desktop()
shortcut_path = os.path.join(desktop, "clcoding.lnk")
shortcut = winshell.shortcut(shortcut_path)
shortcut.path = r"C:\Program Files\SomeApp\app.exe"
shortcut.write()
print("Shortcut created on Desktop")

#source Code: clcoding.com
```

```
Shortcut created on Desktop
```

Integral of a Trigonometric function

[4]:

```
# Define a trigonometric function
g = sp.sin(x)

# Compute the indefinite integral
G = sp.integrate(g, x)

print("∫sin(x) dx =", G)

#clcoding.com
```

```
∫sin(x) dx = -cos(x)
```

Barcode using Python

```
pip install python-barcode
```

```
import barcode
from barcode.writer import ImageWriter
from IPython.display import Image, display

barcode_format = barcode.get_barcode_class('ean13')

barcode_number = '123456789012'

barcode_image = barcode_format(barcode_number, writer=ImageWriter())

barcode_filename = 'barcode_image'
barcode_image.save(barcode_filename)

display(Image(filename=f'{barcode_filename}.png'))

#source code --> clcoding.com
```

Convert Video Files to a Gif in Python

[ ]:

```
pip install moviepy
```

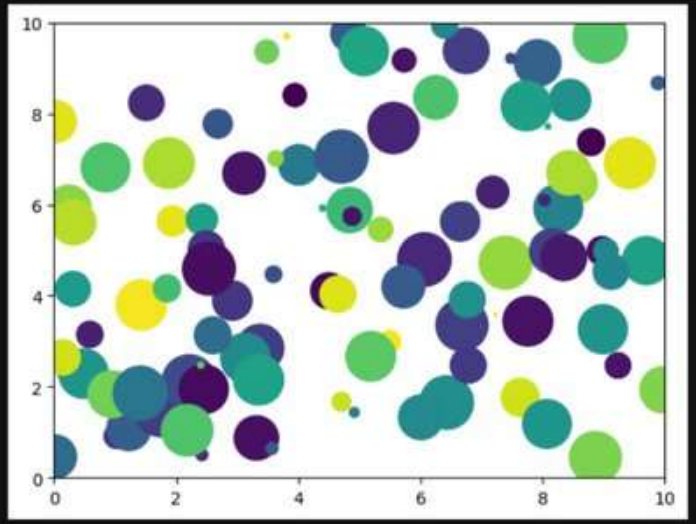
[*]:

```
from moviepy.editor import VideoFileClip

videoClip = VideoFileClip("p3.mp4")

videoClip.write_gif("p3.gif")

#clcoding.com
```

```
MoviePy - Building file p3.gif with imageio.
```

Animated scatter plot using Python

```python
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import numpy as np

num_points = 100
x, y = np.random.rand(2, num_points) * 10
colors = np.random.rand(num_points)
sizes = np.random.rand(num_points) * 1000

fig, ax = plt.subplots()
scat = ax.scatter(x, y, c=colors, s=sizes)
ax.set_xlim(0, 10)
ax.set_ylim(0, 10)

def animate(i):
    scat.set_offsets(np.c_[x + np.random.randn(num_points) * 0.1, y + np.random.randn(num_points) * 0.1])
    return scat,

ani = animation.FuncAnimation(fig, animate, frames=100, interval=50)
plt.show()
```



Stock Chart Plot using Python

```python
import yfinance as yf
import matplotlib.pyplot as plt

# Download historical data of STOCK.NS
ticker = input("Enter Stock Name : ")
data = yf.download(ticker, start="2021-01-01", end="2023-10-13")

# Plot the stock chart
plt.figure(figsize=(10, 5))
plt.plot(data["Close"])
plt.title(f"{ticker} Stock Chart")
plt.xlabel("Date")
plt.ylabel("Price (INR)")
plt.show()
#clcoding.com
```

```
Enter Stock Name : INFY
[**********************100%**********************]  1 of 1 completed
```

Periodic Table Data in Python

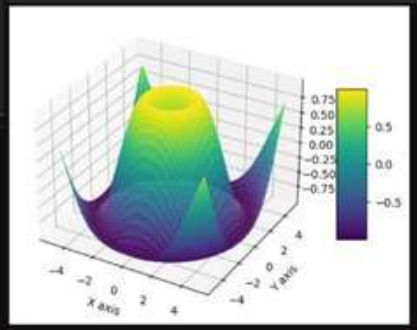
pip install chempy

```
from chempy.util import periodic

n = int(input("Enter number to see the table: "))
print("Atomic No.\tName\t\tSymbol\t\tMass")

for i in range(1, n + 1):
    print(i, end="\t\t")
    if len(periodic.names[i]) > 7:
        print(periodic.names[i], end="\t")
    else:
        print(periodic.names[i], end="\t\t")
    print(periodic.symbols[i], end="\t\t")
    print(periodic.relative_atomic_masses[i])
#source code --> clcoding.com
```

```
Enter number to see the table:  5
Atomic No.      Name            Symbol          Mass
1               Helium          He              4.002602
2               Lithium         Li              6.94
3               Beryllium       Be              9.0121831
4               Boron           B               10.81
5               Carbon          C               12.011
```

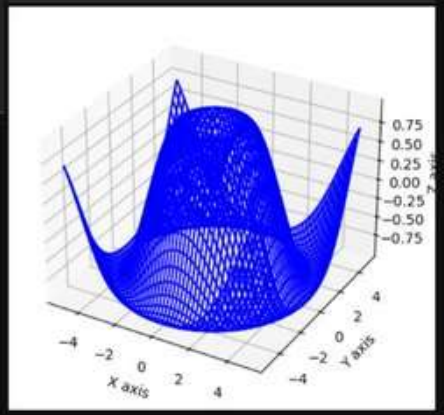
Surface Plot in Python

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x = np.linspace(-5, 5, 50); y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surface = ax.plot_surface(X, Y, Z, cmap='viridis')
fig.colorbar(surface, ax=ax, shrink=0.5, aspect=5)

ax.set_xlabel('X axis'); ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
plt.show()
#source Code --> clcoding.com
```



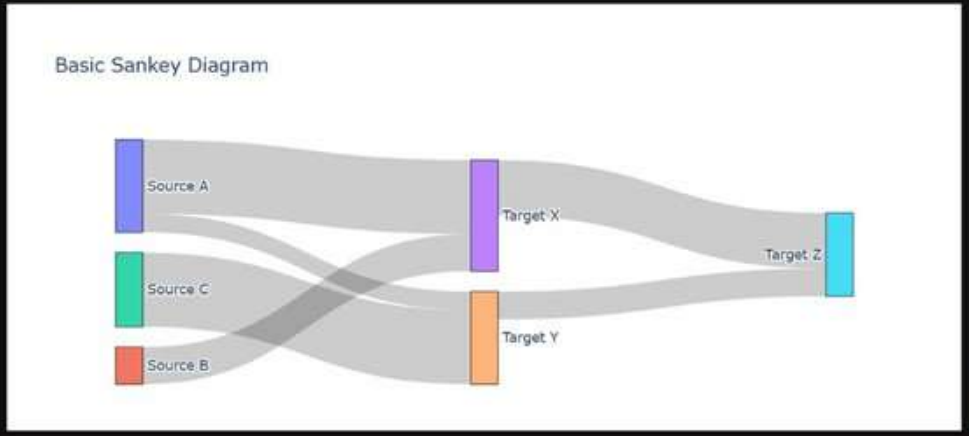
Wireframe Plot

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x = np.linspace(-5, 5, 50); y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(X, Y, Z, color='blue')

ax.set_xlabel('X axis',); ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
plt.show()
#source Code --> clcoding.com
```



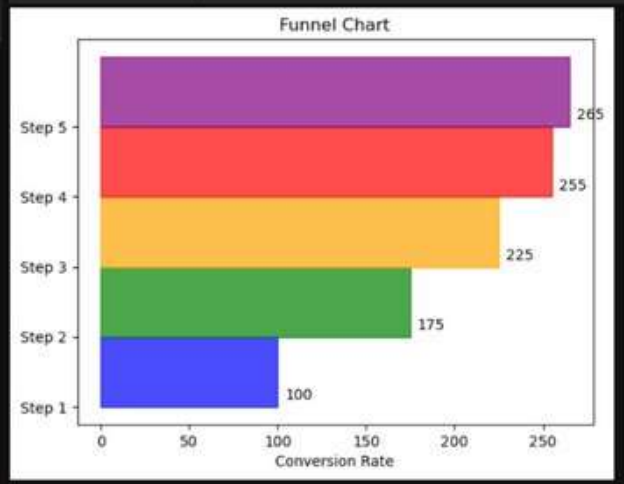
Sankey diagram in Python

```python
import plotly.graph_objects as go

labels = ["Source A", "Source B", "Source C", "Target X", "Target Y", "Target Z"]

source = [0, 1, 0, 2, 3, 3, 4]
target = [3, 3, 4, 4, 5, 5, 5]
values = [8, 4, 2, 8, 4, 2, 3]

fig = go.Figure(data=[go.Sankey(
    node=dict(
        pad=15,
        thickness=20,
        line=dict(color="black", width=0.5),
        label=labels
    ),
    link=dict(
        source=source,
        target=target,
        value=values
    )
)])

fig.update_layout(title_text="Basic Sankey Diagram", font_size=10)
fig.show()
```



Create a funnel chart using Matplotlib

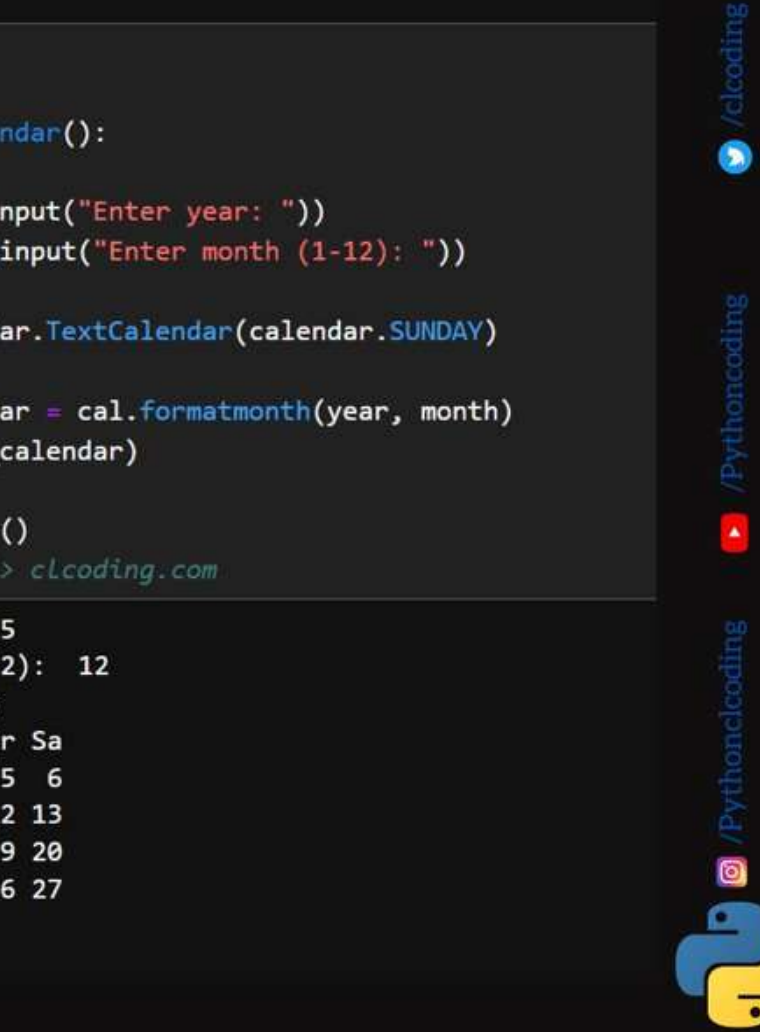
```python
import matplotlib.pyplot as plt

labels = ['Step 1', 'Step 2', 'Step 3', 'Step 4', 'Step 5']
values = [100, 75, 50, 30, 10]

cumulative_values = [sum(values[:i+1]) for i in range(len(values))]

colors = ['blue', 'green', 'orange', 'red', 'purple']

fig, ax = plt.subplots()
for i in range(len(labels)):
    ax.fill_betweenx([i, i + 1], 0, cumulative_values[i], step='mid', alpha=0.7, color=colors[i])

ax.set_yticks(range(len(labels)))
ax.set_yticklabels(labels)
ax.set_xlabel('Conversion Rate')

for i, value in enumerate(cumulative_values):
    ax.annotate(str(value), xy=(value, i), xytext=(5, 5), textcoords='offset points')

plt.title('Funnel Chart')
plt.show()
#Source Code --> clcoding.com
```



Calendar month using Python

```python
import calendar

def display_calendar():

    year = int(input("Enter year: "))
    month = int(input("Enter month (1-12): "))

    cal = calendar.TextCalendar(calendar.SUNDAY)

    month_calendar = cal.formatmonth(year, month)
    print(month_calendar)

display_calendar()
# source code --> clcoding.com
```

```
Enter year:  2025
Enter month (1-12):  12
   December 2025
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```



HOW TO USE PYTHON F-STRINGS?

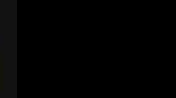
Multi-line F-Strings

F-strings can also be used with multi-line strings:

[11]:

```
name = "Alice"
age = 30

info = (
    f"Name: {name}\n"
    f"Age: {age}\n"
    f"Location: Wonderland"
)
print(info)
```

```
Name: Alice
Age: 30
Location: Wonderland
```



Combining F-Strings with Other String Formatting

F-strings can be combined with other string formatting methods if necessary:

[13]:

```
name = "Alice"
age = 30

combined = f"Name: {name}, " + "Age: {}".format(age)
print(combined)
```

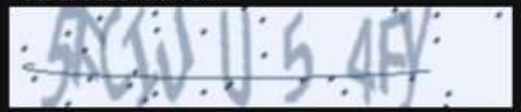
```
Name: Alice, Age: 30
```


CAPTCHA in Python using captcha Library

```
pip install captcha
```

```python
from captcha.image import ImageCaptcha
from PIL import Image
import random

def generate_captcha_text(length=10):
    return ''.join(random.choices(string.ascii_letters
                                  + string.digits, k=length))

def generate_captcha_image(captcha_text,image_width=300):
    image = ImageCaptcha(image_width)
    image_file = f"{captcha_text}.png"
    image.write(captcha_text, image_file)
    return image_file

captcha_text = generate_captcha_text()
image_file = generate_captcha_image(captcha_text)

print(f"Generated CAPTCHA")
Image.open(image_file)

#source Code --> clcoding.com
```

Generated CAPTCHA



Pencil Sketch using Python

[*]:

```python
import cv2
import numpy as np

def pencil_sketch(image_path, output_path):
    image = cv2.imread(image_path)

    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    inverted_gray_image = cv2.bitwise_not(gray_image)

    blurred_image = cv2.GaussianBlur(inverted_gray_image, (21, 21), 0)

    inverted_blurred_image = cv2.bitwise_not(blurred_image)
    pencil_sketch_image = cv2.divide(gray_image, inverted_blurred_image, scale=256.0)
    cv2.imwrite(output_path, pencil_sketch_image)

    cv2.imshow('Pencil Sketch', pencil_sketch_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

input_image_path = 'w2.jpg'
output_image_path = 'w2_sketch.jpg'
pencil_sketch(input_image_path, output_image_path)

#source Code --> clcoding.com
```

Original

Sketch

/clcoding

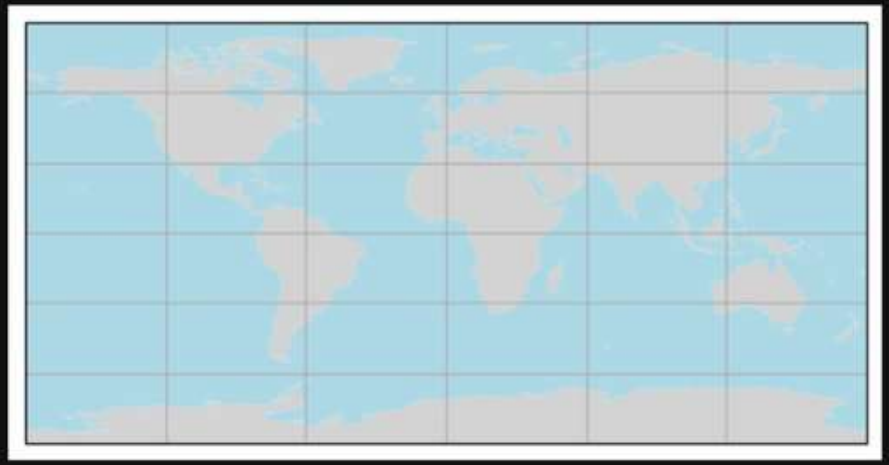
/Pythoncoding

/Pythonclcoding

World map using Python

pip install cartopy

```python
import cartopy.crs as ccrs
import cartopy.feature as cfeature
import matplotlib.pyplot as plt

projection = ccrs.PlateCarree()

fig, ax = plt.subplots(subplot_kw={'projection': projection})
ax.set_extent([-180, 180, -90, 90], crs=ccrs.PlateCarree())

ax.add_feature(cfeature.LAND, facecolor='lightgray')
ax.add_feature(cfeature.OCEAN, facecolor='lightblue')

ax.gridlines()
plt.show()

#Source Code --> clcoding.com
```



/clcoding

/Pythoncoding

/Pythonclcoding

Movie Information using Python

```python
import imdb

ia = imdb.Cinemagoer()
Movie = input("Enter a movie name: ")
items = ia.search_movie(Movie)
print("\nSearch results:")
for index, movie in enumerate(items):
    print(f"{index + 1}. {movie['title']} ({movie['year']})")

movie_index = int(input("\nEnter the number of the movie you want to get info for: ")) - 1
movie_id = items[movie_index].movieID
movie_info = ia.get_movie(movie_id)

print("\nMovie Information:")
print(f"Title: {movie_info.get('title')}")
print(f"Year: {movie_info.get('year')}")
print(f"Rating: {movie_info.get('rating')}")
print(f"Genres: {', '.join(movie_info.get('genres', []))}")
print(f"Director(s): {', '.join(str(d) for d in movie_info.get('directors', []))}")
print(f"Cast: {', '.join(str(c) for c in movie_info.get('cast', [])[:5])}...")
print(f"Plot: {movie_info.get('plot outline')}")
print(f"Runtime: {movie_info.get('runtimes', ['N/A'])[0]} minutes")
print(f"Country: {', '.join(movie_info.get('countries', []))}")
print(f"Language: {', '.join(movie_info.get('languages', []))}")

#Source Code --> clcoding.com
```

```
Enter a movie name:  Lift

Search results:
1. Lift (2024)
2. Elevator to the Gallows (1958)
3. Lift (2021)
4. The Lift (1983)
5. Lift (2001)
6. The Big Lift (1950)
7. Lift (2022)
8. Lift (2016)
9. Lift Off (1992)
10. How Heavy Are the Dumbbells You Lift? (2019)
11. Lift (1989)
12. Lift Me Up (2015)
13. The Lift Boy (2019)
14. Lift (2019)
15. Lift (2017)
16. Lift (2018)
17. The Lift (2022)
18. Lift Off (1969)
19. LIFT (2019)
20. Lift Off (2021)

Enter the number of the movie you want to get info for:  1

Movie Information:
Title: Lift
Year: 2024
Rating: 5.5
Genres: Action, Comedy, Crime, Drama, Thriller
Director(s): F. Gary Gray
Cast: Kevin Hart, Gugu Mbatha-Raw, Sam Worthington, Vincent D'Onofrio, Úrsula Corberó...
Plot: Cyrus, an international thief and his crew specialize in stealing extremely costly art pieces and resort to k
idnapping if necessary. Cyrus's ex girlfriend Abby, an Interpol agent convinces him to steal a huge consignment of
gold being sent by plane from London to Zurich in return for immunity from arrest . Cyrus knows this is an almost i
mpossible task but decides to go ahead since it means freedom for him and his crew not to speak of his getting toge
ther with Abby again.
Runtime: 107 minutes
Country: United States
Language: English, Italian
```

Avatar Logo using Python

[*]:

```
from turtle import *
speed(0)
bgcolor('black')
color('orange')
hideturtle()
n=1
p=True
while True:
    circle(n)
    if p:
        n=n-1
    else:
        n=n+1
    if n==0 or n==60:
        p=not p
    left(1)
    forward(3)
#Source Code --> clcoding.com
```

/clcoding

/Pythoncoding

/Pythonclcoding

4. Extract All Emojis from a Text

This program extracts all the emojis from a given text string.

[9]:

```
import demoji

text = "Enjoy your meal! 🍽️🍕🍔🍟"

emojis = demoji.findall(text)
print("Emojis found:", list(emojis.keys()))

#source Code --> clcoding.com
```

```
Emojis found: ['🍕', '🍽️', '🍟', '🍔']
```


5. Create an Emoji Frequency Dictionary

This program creates a dictionary that maps each emoji in a text to the number of times it appears.

[11]:

```
import demoji

text = "I love 🍎 and 🍌. Do you like 🍎 too? 🍎🍎"

emoji_freq = {}
emojis = demoji.findall(text)
for emoji in emojis.keys():
    emoji_freq[emoji] = text.count(emoji)

print("Emoji Frequency:", emoji_freq)

#source Code --> clcoding.com
```

```
Emoji Frequency: {'🍌': 1, '🍎': 4}
```

3. Count the Number of Emojis in a Text

This program counts how many emojis are present in a text string.

[7]:

```
import demoji

text = "Coding is awesome! 👨‍💻🚀🤯"

emoji_count = len(demoji.findall(text))
print(f"Number of emojis: {emoji_count}")

#source Code --> clcoding.com
```

```
Number of emojis: 3
```


Convert emoji into text in Python

pip install demoji

[5]:

```
import demoji

text="IN 📚 ❤️ 🌺 🌹 👶"
demoji.findall(text)

#clcoding.com
```

[5]:

```
{'👶': 'baby',
 '❤️': 'red heart',
 '📚': 'books',
 '🌺': 'hibiscus',
 '🌹': 'rose',
 'IN': 'flag: India'}
```

1. Saving and Loading a List

This program saves a list to a file and then loads it back.

[3]:

```
import pickle

my_list = ['apple', 'banana', 'cherry']

with open('list.pkl', 'wb') as file:
    pickle.dump(my_list, file)

with open('list.pkl', 'rb') as file:
    loaded_list = pickle.load(file)

print("Loaded List:", loaded_list)

#source code --> clcoding.com
```

```
Loaded List: ['apple', 'banana', 'cherry']
```


2. Saving and Loading a Dictionary

This program demonstrates saving a dictionary to a file and loading it back.

[5]:

```
import pickle

my_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

with open('dict.pkl', 'wb') as file:
    pickle.dump(my_dict, file)

with open('dict.pkl', 'rb') as file:
    loaded_dict = pickle.load(file)

print("Loaded Dictionary:", loaded_dict)

#source code --> clcoding.com
```

```
Loaded Dictionary: {'name': 'John', 'age': 30, 'city': 'New York'}
```

3. Saving and Loading a Custom Object

This program saves an instance of a custom class to a file and loads it back.

[7]:

```
import pickle

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __repr__(self):
        return f"Person(name={self.name}, age={self.age})"

person = Person('Alice', 25)

with open('person.pkl', 'wb') as file:
    pickle.dump(person, file)

with open('person.pkl', 'rb') as file:
    loaded_person = pickle.load(file)

print("Loaded Person:", loaded_person)

#source code --> clcoding.com
```

```
Loaded Person: Person(name=Alice, age=25)
```


4. Saving and Loading a Tuple

This program saves a tuple to a file and loads it back.

[9]:

```
import pickle

my_tuple = (10, 20, 30, 'Hello')

with open('tuple.pkl', 'wb') as file:
    pickle.dump(my_tuple, file)

with open('tuple.pkl', 'rb') as file:
    loaded_tuple = pickle.load(file)

print("Loaded Tuple:", loaded_tuple)

#source code --> clcoding.com
```

```
Loaded Tuple: (10, 20, 30, 'Hello')
```


5. Saving and Loading Multiple Objects

This program saves multiple objects to a single file and loads them back.

[11]:

```
import pickle

list_data = [1, 2, 3]
dict_data = {'a': 1, 'b': 2}
string_data = "Hello, World!"

with open('multiple.pkl', 'wb') as file:
    pickle.dump(list_data, file)
    pickle.dump(dict_data, file)
    pickle.dump(string_data, file)

with open('multiple.pkl', 'rb') as file:
    loaded_list = pickle.load(file)
    loaded_dict = pickle.load(file)
    loaded_string = pickle.load(file)

print("Loaded List:", loaded_list)
print("Loaded Dictionary:", loaded_dict)
print("Loaded String:", loaded_string)

#source code --> clcoding.com
```

```
Loaded List: [1, 2, 3]
Loaded Dictionary: {'a': 1, 'b': 2}
Loaded String: Hello, World!
```


10 ESSENTIAL USE CASES OF PYTHON'S ZIP() FUNCTION WITH EXAMPLES

1. Combining Two Lists

Use case: Merging two lists element-wise.

```
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]

combined = list(zip(names, ages))
print(combined)
```

```
[('Alice', 25), ('Bob', 30), ('Charlie', 35)]
```

2. Unzipping Lists

Use case: Splitting paired data into separate lists.

```
combined = [('Alice', 25), ('Bob', 30), ('Charlie', 35)]

names, ages = zip(*combined)
print(names)
print(ages)
```

```
('Alice', 'Bob', 'Charlie')
(25, 30, 35)
```


3. Iterating Over Multiple Lists Simultaneously

Use case: Useful when you need to iterate through multiple lists at the same time.

```
subjects = ['Math', 'Science', 'English']
scores = [88, 92, 85]

for subject, score in zip(subjects, scores):
    print(f"{subject}: {score}")
```

```
Math: 88
Science: 92
English: 85
```

4. Creating Dictionaries

Use case: Creating dictionaries from two lists: one for keys, one for values.

```
keys = ['name', 'age', 'city']
values = ['Alice', 25, 'New York']

dictionary = dict(zip(keys, values))
print(dictionary)
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

5. Combining Multiple Lists

Use case: Zipping more than two lists together.

```
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']
list3 = [True, False, True]

combined = list(zip(list1, list2, list3))
print(combined)
```

```
[(1, 'a', True), (2, 'b', False), (3, 'c', True)]
```

6. Handling Different Length Iterables

Use case: When lists have different lengths, zip() stops at the shortest one.

```
list1 = [1, 2, 3]
list2 = ['a', 'b']

combined = list(zip(list1, list2))
print(combined)
```

```
[(1, 'a'), (2, 'b')]
```


7. Working with Ranges

Use case: Zipping together ranges or sequences.

```
numbers = range(1, 4)
letters = ['a', 'b', 'c']

result = list(zip(numbers, letters))
print(result)
```

```
[(1, 'a'), (2, 'b'), (3, 'c')]
```

8. Comparing Elements of Two Lists

Use case: Zipping two lists to compare elements.

```
list1 = [1, 2, 3]
list2 = [1, 4, 3]

comparison = [a == b for a, b in zip(list1, list2)]
print(comparison)
```

```
[True, False, True]
```


9. Transpose a Matrix

Use case: Use zip() to transpose rows and columns of a 2D matrix.

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

transposed = list(zip(*matrix))
print(transposed)
```

```
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
```

10. Zipping with Enumerate

Use case: Zipping with an enumerated list for indexed pairings.

```
data = ['apple', 'banana', 'cherry']
indexed_data = list(zip(range(1, len(data) + 1), data))
print(indexed_data)
```

```
[(1, 'apple'), (2, 'banana'), (3, 'cherry')]
```


Auto copy paste using Python

[3]:

```
import pyperclip as pc

text1 = input("Enter a text : ")

pc.copy(text1)

text2 = pc.paste()

print(text2)

#Source Code --> clcoding.com
```

```
Enter a text :  Hello Clcoding
Hello Clcoding
```

/clcoding

/Pythoncoding

/Pythonclcoding

Enhancing Python Scripts with Alive-Progress: 5 Practical Examples

1. Simple Progress Bar

```python
from alive_progress import alive_bar
import time

items = range(100)

with alive_bar(len(items)) as bar:
    for item in items:
        time.sleep(0.05)
        bar()
```

```
|████████████████████████████████████████| 100/100 [100%] in 5.0s (19.83/s)
```

2. Progress Bar with Custom Text

```python
from alive_progress import alive_bar
import time

tasks = ['task1', 'task2', 'task3', 'task4']

with alive_bar(len(tasks), title='Processing Tasks') as bar:
    for task in tasks:
        time.sleep(0.5)
        bar.text = f'Working on {task}'
        bar()
```

```
Processing Tasks |████████████████████████████████████████| 4/4 [100%] in 2.0s (2.00/s)
```


3. Nested Progress Bars

```python
from alive_progress import alive_bar
import time

outer_items = range(3)
inner_items = range(5)

with alive_bar(len(outer_items) * len(inner_items), title='Processing') as bar:
    for _ in outer_items:
        for _ in inner_items:
            time.sleep(0.1)  # Simulate work
            bar()
```

```
Processing |████████████████████████████████████| 15/15 [100%] in 1.5s (9.94/s)
```

4. Download Simulation with Progress Bar

```python
from alive_progress import alive_bar
import time

file_size = 1024
chunk_size = 64

with alive_bar(file_size // chunk_size, title='Downloading') as bar:
    for _ in range(0, file_size, chunk_size):
        time.sleep(0.1)
        bar()
```

```
Downloading |████████████████████████████████████| 16/16 [100%] in 1.6s (9.92/s)
```

4. Download Simulation with Progress Bar

```python
from alive_progress import alive_bar
import time

file_size = 1024
chunk_size = 64

with alive_bar(file_size // chunk_size, title='Downloading') as bar:
    for _ in range(0, file_size, chunk_size):
        time.sleep(0.1)
        bar()
```

```
Downloading |████████████████████████████████████████| 16/16 [100%] in 1.6s (9.83/s)
```

5. Progress Bar with Percentage and ETA

```python
from alive_progress import alive_bar
import time

total_items = 50

with alive_bar(total_items, title='Processing',
               spinner='dots_waves', bar='smooth') as bar:
    for _ in range(total_items):
        time.sleep(0.1)
        bar()
```

```
Processing |████████████████████████████████████████| 50/50 [100%] in 5.0s (9.95/s)
```

Check Internet Speed using Python

pip install speedtest-cli

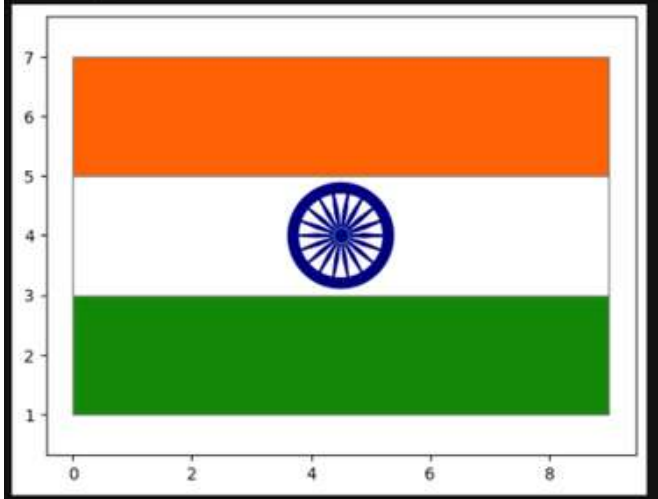
[5]:

```
import speedtest as st

def Speed_Test():
    test = st.Speedtest()

    down_speed = test.download()
    down_speed = round(down_speed / 10**6, 2)
    print("Download Speed in Mbps: ", down_speed)

    up_speed = test.upload()
    up_speed = round(up_speed / 10**6, 2)
    print("Upload Speed in Mbps: ", up_speed)

    ping = test.results.ping
    print("Ping: ", ping)
Speed_Test()

#source code -> clcoding.com
```

```
Download Speed in Mbps:  20.57
Upload Speed in Mbps:  18.47
Ping:  26.177
```

Indian Flag using Python

```
import numpy as np
import matplotlib.pyplot as py
import matplotlib.patches as patch

a = patch.Rectangle((0,1), width=9, height=2, facecolor='#138808', edgecolor='grey')
b = patch.Rectangle((0,3), width=9, height=2, facecolor='#ffffff', edgecolor='grey')
c = patch.Rectangle((0,5), width=9, height=2, facecolor='#FF6103', edgecolor='grey')
m,n = py.subplots()
n.add_patch(a)
n.add_patch(b)
n.add_patch(c)

radius=0.8
py.plot(4.5,4, marker = 'o', markerfacecolor = '#000080', markersize = 9.5)
chakra = py.Circle((4.5, 4), radius, color='#000080', fill=False, linewidth=7)
n.add_artist(chakra)

for i in range(0,24):
   p = 4.5 + radius/2 * np.cos(np.pi*i/9 + np.pi/48)
   q = 4.5 + radius/2 * np.cos(np.pi*i/9 - np.pi/48)
   r = 4 + radius/2 * np.sin(np.pi*i/9 + np.pi/48)
   s = 4 + radius/2 * np.sin(np.pi*i/9 - np.pi/48)
   t = 4.5 + radius * np.cos(np.pi*i/9)
   u = 4 + radius * np.sin(np.pi*i/9)
   n.add_patch(patch.Polygon([[4.5,4], [p,r], [t,u],[q,s]], fill=True, closed=True, color='#000080'))
py.axis('equal')
py.show()
#clcoding.com
```



Chi-Square Test with Python

```python
import numpy as np
from scipy.stats import chi2_contingency

# Example contingency table
data = np.array([[30, 10],   # Category A
                 [20, 25]])  # Category B

# Perform the Chi-Square test
chi2_stat, p_val, dof, expected = chi2_contingency(data)

print(f"Chi-Square Statistic: {chi2_stat}")
print(f"P-value: {p_val}")
print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies:")
print(expected)
```

```
Chi-Square Statistic: 6.949930555555551
P-value: 0.00838224869725173
Degrees of Freedom: 1
Expected Frequencies:
[[23.52941176 16.47058824]
 [26.47058824 18.52941176]]
```

Definite Integration using Python

[38]:

```
import sympy as sp

x = sp.Symbol('x')
f = input("Enter the function(in terms of x):")

F_indefinite = sp.integrate(f, x)
print("Indefinite integral ∫f(x) dx =", F_indefinite)

a = float(input("Enter the lower limit of integration: "))
b = float(input("Enter the upper limit of integration: "))

F_definite = sp.integrate(f, (x, a, b))
print(f"Definite integral ∫f(x) dx from {a} to {b} =", F_definite)

#clcoding.com
```

```
Enter the function(in terms of x): x^2 + 3
Indefinite integral ∫f(x) dx = x**3/3 + 3*x
Enter the lower limit of integration:  9
Enter the upper limit of integration:  27
Definite integral ∫f(x) dx from 9.0 to 27.0 = 6372.00000000000
```

Indefinite Integral of a polynomial function

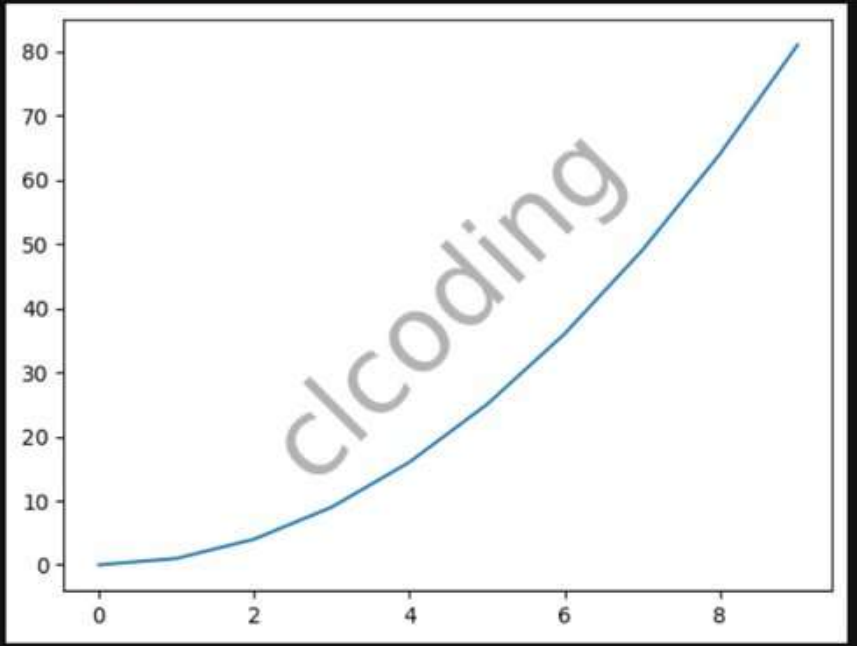
[2]:

```
import sympy as sp

# Define the variable and function
x = sp.Symbol('x')
f = x**2 + 3*x + 2

# Compute the indefinite integral
F = sp.integrate(f, x)

print("∫f(x) dx =", F)

#clcoding.com
```

```
∫f(x) dx = x**3/3 + 3*x**2/2 + 2*x
```

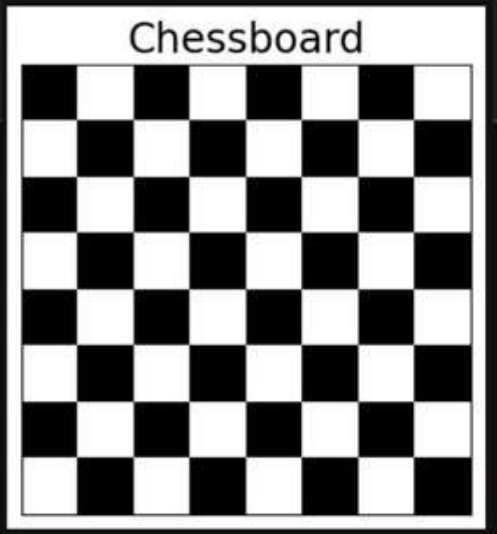
Adding a Watermark to a Plot in Matplotlib

```python
import matplotlib.pyplot as plt

x = range(10)
y = [i**2 for i in x]

plt.plot(x, y)

# Add watermark
plt.text(0.5, 0.5, 'clcoding', alpha=0.3, fontsize=50, rotation=45,
         ha='center', va='center', transform=plt.gca().transAxes)

plt.show()

#clcoding.com
```



Python Code to Draw a Chessboard

```python
import matplotlib.pyplot as plt
import numpy as np

def draw_chessboard():
    chessboard = np.zeros((8, 8))
    chessboard[1::2, ::2] = 1
    chessboard[::2, 1::2] = 1

    plt.figure(figsize=(4, 4))
    plt.imshow(chessboard, cmap='gray', interpolation='nearest')

    plt.xticks([])
    plt.yticks([])
    plt.title('Chessboard', fontsize=20)

    plt.show()
draw_chessboard()
#clcoding.com
```



Find Weather using Python

pip install beautifulsoup4

[3]:

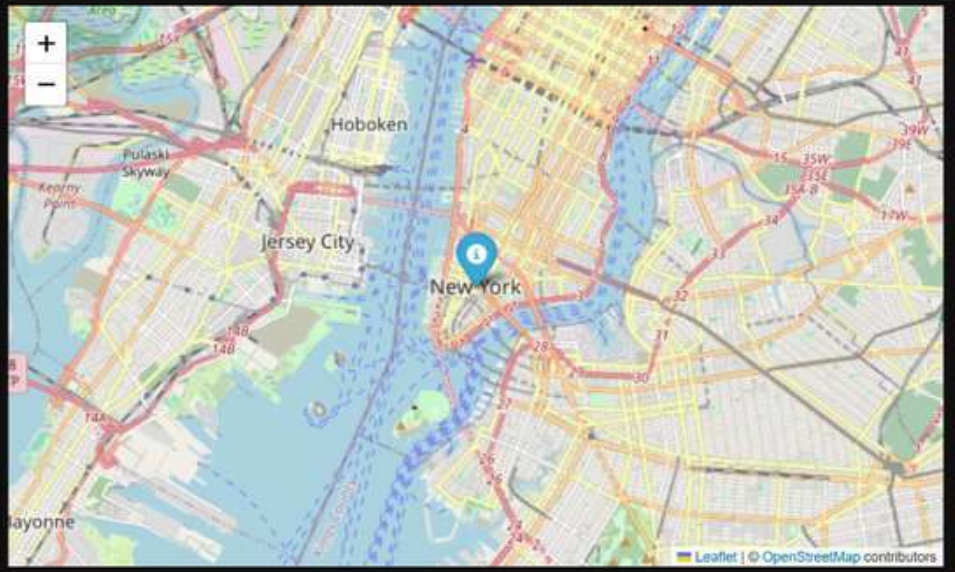
```
import requests
from bs4 import BeautifulSoup

city = input("Enter City Name: ")
city_formatted = city.lower().replace(" ", "-")

url = f"https://www.timeanddate.com/weather/usa/{city_formatted}"
response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

try:
    temperature = soup.find("div", class_="h2").get_text(strip=True)
    description = soup.find("div", class_="h2").find_next("p").get_text(strip=True)

    print(f"Weather in {city}:")
    print(f"Temperature: {temperature}")
    print(f"Condition: {description}")

except AttributeError:
    print("Please check the city name and try again.")
```

```
Enter City Name:  Newyork
Weather in Newyork:
Temperature: 19 °C
Condition: Light rain. Low clouds.
```

Create a map using Python

```
pip install folium
```

```
import folium
from IPython.display import display

map_center = [40.7128, -74.0060]
mymap = folium.Map(location=map_center, zoom_start=12)

folium.Marker(
    [40.7128, -74.0060],
    popup="New York",
    icon=folium.Icon(color="blue", icon="info-sign")
).add_to(mymap)

display(mymap)
#clcoding.com
```



Find Position of a Planet using Python

[ ]:

```
pip install astropy
```

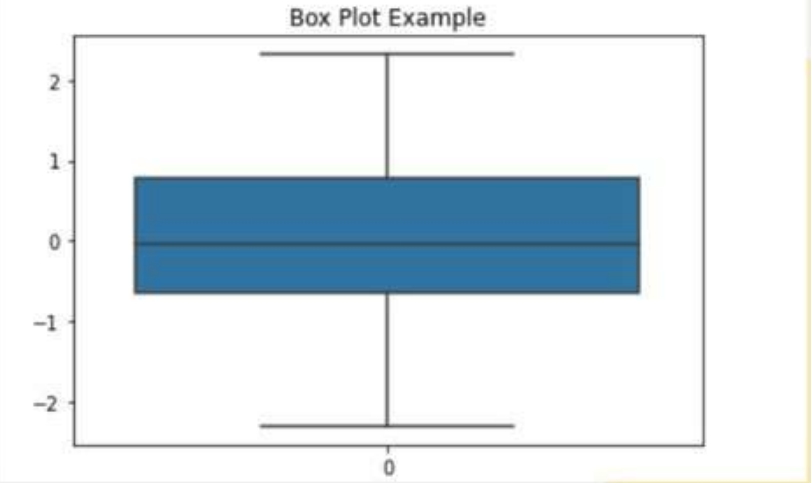
[4]:

```
from astropy.coordinates import get_body, EarthLocation
from astropy.time import Time

now = Time.now()

location = EarthLocation.of_site('greenwich')

planet_name = input("Enter the name of the planet: ").lower()

planet_position = get_body(planet_name, now, location)

print(f"{planet_name.capitalize()} "
      f"Position: RA = {planet_position.ra}, "
      f"Dec = {planet_position.dec}")

#clcoding.com
```

```
Enter the name of the planet:  mars
Mars Position: RA = 70.60887634363304 deg, Dec = 21.7475767689189 deg
```

Box Plot using Python

```python
import seaborn as sns
import numpy as np

# Generate random data
data = np.random.randn(100)

# Create a box plot
sns.boxplot(data=data)
plt.title('Box Plot Example')
plt.show()
```



LaTeX math problems using Python

/Pythoncoding

```
pip install latexify-py
```

```
import math
import latexify
```

Some math symbols are converted automatically

```
@latexify.function(use_math_symbols=True)
def greek(alpha, beta, gamma, Omega):
  return alpha * beta + math.gamma(gamma) + Omega
greek        #clcoding.com
```

$$\text{greek}(\alpha, \beta, \gamma, \Omega) = \alpha\beta + \Gamma(\gamma) + \Omega$$

```
# Elif or nested else-if are unrolled.
@latexify.function
def fib(x):
  if x == 0:
    return 0
  elif x == 1:
    return 1
  else:
    return fib(x-1) + fib(x-2)
fib    #clcoding.com
```

$$\text{fib}(x) = \begin{cases} 0, & \text{if } x = 0 \\ 1, & \text{if } x = 1 \\ \text{fib}(x-1) + \text{fib}(x-2), & \text{otherwise} \end{cases}$$

/Pythonclcoding

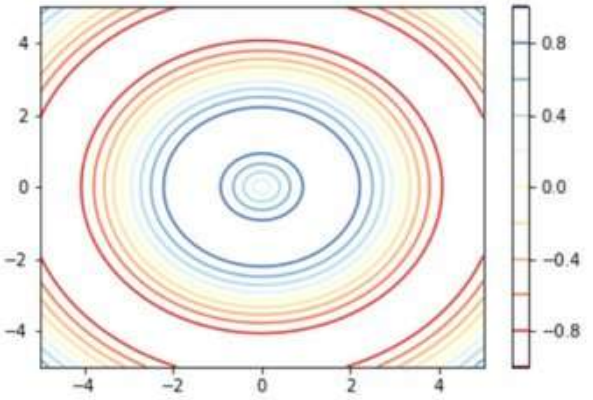
/clcoding

For Free Courses and Certifications check
Link in Bio

Contour lines in Python

/Pythoncoding

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate some sample data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))

# Create the contour plot
plt.contour(X, Y, Z, levels=10, cmap='RdYlBu')
plt.colorbar()

# Show the plot
plt.show()
```



/Pythonclcoding

/clcoding

For Free Courses and Certifications check
Link in Bio

[ ]:

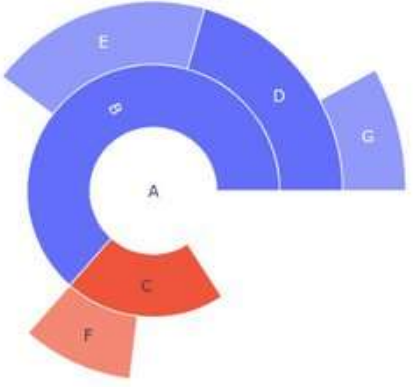
```
pip install wifi-qrcode-generator
```

[1]:

```
from wifi_qrcode_generator import wifi_qrcode

qr_code = wifi_qrcode("clcoding", hidden=False,
                      authentication_type="WPA", password="Cl@2014")

qr_code_img = qr_code.make_image()

qr_code_img.save("wifi_qr_code.png")

#clcoding.com
```

Sunburst Chart in Python

```python
import plotly.express as px

# Sample data
data = {
    'id': ["A", "B", "C", "D", "E", "F", "G"],
    'parent': ["", "A", "A", "B", "B", "C", "D"],
    'value': [10, 15, 7, 8, 12, 6, 5],
}

# Create a sunburst chart
fig = px.sunburst(data, names='id', parents='parent', values='value')

# Set the chart title
fig.update_layout(title_text="Sunburst Chart")

# Show the chart
fig.show()
#clcoding.com
```



Happy Friendship Day using Python

[6]:

```
from rich.console import Console
from rich.text import Text

console = Console()

# Create the message and color mapping
text = "Happy Friendship Day!"
colors = [
    "red", "yellow", "green", "cyan", "blue", "white", "magenta",
    "red", "yellow", "green", "cyan", "blue", "magenta", "red",
    "yellow", "green", "white", "cyan", "blue", "magenta", "red"
]

# Generate the colorful message
message = Text()
[message.append(char, style=color) for char,color in zip(text, colors)]

console.print(message)

#clcoding.com
```

Happy Friendship Day!

/clcoding

/Pythoncoding

/Pythonclcoding

Happy Friendship Day using Python

•[4]:

```
import pyfiglet
from termcolor import colored

text = "Happy Friendship Day!"
fonts = ["slant"]

for i, word in enumerate(text.split()):
    font = pyfiglet.Figlet(font=fonts[i % len(fonts)])
    color = ["red", "green", "yellow", "blue", "magenta"][i % 5]
    ascii_art = font.renderText(word)
    print(colored(ascii_art, color))

#clcoding.com
```

/clcoding

/Pythoncoding

/Pythonclcoding

Returning Multiple Values

What: Python allows functions to return multiple values as a tuple.

Why: Enables you to return complex data without creating a class or data structure.

[11]:

```
def get_name_age():
    name = "clcoding"
    age = 30
    return name, age

name, age = get_name_age()
print(name, age)
#clcoding.com
```

```
clcoding 30
```


Port Scanning using Python

Port scanning is a common task in cybersecurity to identify open ports on a network.

[2]:

```
import socket

def port_scanner(target, ports):
    clcoding = socket.gethostbyname(target)
    print(f"Scanning {target} ({clcoding})")

    for port in ports:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        socket.setdefaulttimeout(1)
        result = sock.connect_ex((clcoding, port))
        if result == 0:
            print(f"Port {port}: Open")
        sock.close()

# Example usage
target = "clcoding.com"
ports = [22, 80, 443, 8080]
port_scanner(target, ports)
```

```
Scanning clcoding.com (216.239.38.21)
Port 80: Open
Port 443: Open
```

```
pip install pyfiglet
```

```
pip install termcolor
```

Create Font Art using Python

```
import pyfiglet
from termcolor import colored

# User inputs
text = input("Enter the text you want to format: ")
font = input("Enter the font you want to use (press Enter for default): ")
color = input("Enter the color you want (e.g., red, green, yellow): ")

# Apply the font and color
if font:
    formatted_text = pyfiglet.figlet_format(text, font=font)
else:
    formatted_text = pyfiglet.figlet_format(text)

if color:
    colored_text = colored(formatted_text, color)
    print(colored_text)
else:
    print(formatted_text)
#clcoding.com
```

```
Enter the text you want to format:  Python Coding
Enter the font you want to use (press Enter for default):
Enter the color you want (e.g., red, green, yellow):  yellow
```

Password authentication process using Python

[2]:

```
import hashlib, os

def hash_psd(psd: str) -> str:
    salt = os.urandom(16)
    hashed_psd = hashlib.pbkdf2_hmac('sha256',
                                     psd.encode(), salt, 100000)
    return salt.hex() + hashed_psd.hex()

def verify_psd(stored_psd: str, provided_psd: str) -> bool:
    salt = bytes.fromhex(stored_psd[:32])
    stored_hash = stored_psd[32:]
    hashed_psd = hashlib.pbkdf2_hmac('sha256',
                                     provided_psd.encode(),salt,100000)
    return hashed_psd.hex() == stored_hash

if __name__ == "__main__":
    psd_to_store = input("Enter a Password: ")
    stored_psd = hash_psd(psd_to_store)
    print(f'Stored Password: {stored_psd}')

    psd_attempt = 'clcoding'
    is_valid = verify_psd(stored_psd, psd_attempt)
    print(f'Password is valid: {is_valid}')

#clcoding.com
```

```
Enter a Password:  clcoding
Stored Password: ec87cb3f526a3dc27e5a67fe7878f850a6b24c71c2941edc5bbe2c
3500afc164cebb1ec8bbbc6a2260faa4825307600e
Password is valid: True
```

Password Authentication using Python

/Pythoncoding

```python
import getpass
database = {"clcoding": "976729", "pythonclcoding": "2502"}
username = input("Enter Your Username : ")
password = getpass.getpass("Enter Your Password : ")
for i in database.keys():
    if username == i:
        while password != database.get(i):
            password = getpass.getpass("Enter Your Password Again : ")
        break
print("Verified")

#clcoding.com
```

```
Enter Your Username : pythonclcoding
Enter Your Password : ........
Enter Your Password Again : ........
Enter Your Password Again : ........
Verified
```

Unprinting stuff in Python

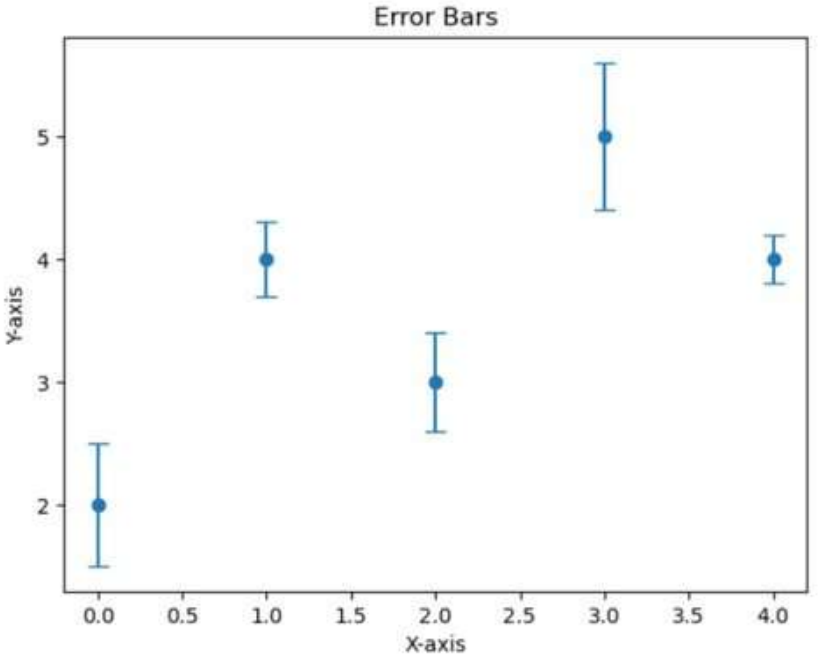
[1]:

```
a = "\033[1A"
b = "\x1b[2K"
print(a)
print(b)

#nothing is printed below
#clcoding.com
```

"""The "\033[1A" escape character moves our cursor up by 1 line, and the "\x1b[2K" character clears the entire current line. If we print them together, we can essentially 'unprint' an entire line in our terminal"""

Error Bars using Python

```python
import matplotlib.pyplot as plt

x = range(5)
y = [2, 4, 3, 5, 4]
errors = [0.5, 0.3, 0.4, 0.6, 0.2]

plt.errorbar(x, y, yerr=errors, fmt='o', capsize=5)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Error Bars')
plt.show()
#clcoding.com
```



Enter chemical Name to find formula

pip install pubchempy

```python
import pubchempy as pcp

# Take name as input
chemical_name = input("Enter chemical name: ")

try:
    # Search PubChem for the compound by its name
    compound = pcp.get_compounds(chemical_name, 'name')[0]

    # Display information about the compound
    print(f"IUPAC Name: {compound.iupac_name}")
    print(f"Common Name: {compound.synonyms[0]}")
    print(f"Molecular Weight: {compound.molecular_weight}")
    print(f"Formula: {compound.molecular_formula}")

    # You can access more properties as needed
except IndexError:
    print(f"No information found for {chemical_name}.")

    #clcoding.com
```

```
Enter chemical name:  Carbon dioxide
IUPAC Name: None
Common Name: carbon dioxide
Molecular Weight: 44.009
Formula: CO2
```

/clcoding

/Pythoncoding

/Pythonclcoding

Remove Image Background using Python

```python
from rembg import remove
from PIL import Image
input_path = 'p3.jpg'
output_path = 'p3.png'
inp = Image.open(input_path)
output = remove(inp)
output.save(output_path)
Image.open("p3.png")
#clcoding.com
```

/clcoding

/Pythoncoding

/Pythonclcoding

Program to Create a Countdown Timer

[1]:

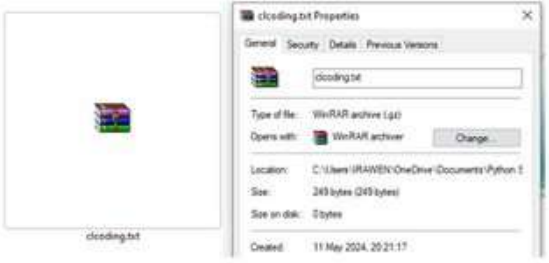
```
import time

def countdown(time_sec):
    while time_sec:
        mins, secs = divmod(time_sec, 60)
        timeformat = '{:02d}:{:02d}'.format(mins, secs)
        print(timeformat, end='\r')
        time.sleep(1)
        time_sec -= 1
    print("Time's up!")

num = int(input("Set Your Timer in Sec: "))
countdown(num)

#clcoding.com
```

```
Set Your Timer in Sec:  12
Time's up!
```

Automating File Compression with gzip

[5]:

```
import gzip
import shutil

def compress_file(input_file, output_file):
    with open(input_file, 'rb') as f_in:
        with gzip.open(output_file, 'wb') as f_out:
            shutil.copyfileobj(f_in, f_out)

# Example usage
input_filename = 'clcoding.txt'
output_filename = 'clcoding.txt.gz'
compress_file(input_filename, output_filename)

#clcoding.com
```



Search anything in Python

[7]:

```
import pywhatkit as kit
searchitem=(input("Enter the topic : "))
kit.info(searchitem,20)

#clcoding.com
```

```
Enter the topic :  Python Coding
Python is a high-level, general-purpose programming languag
e. Its design philosophy emphasizes code readability with t
he use of significant indentation.
Python is dynamically typed and garbage-collected. It suppo
rts multiple programming paradigms, including structured (p
articularly procedural), object-oriented and functional pro
gramming. It is often described as a "batteries included" l
anguage due to its comprehensive standard library.
Guido van Rossum began working on Python in the late 1980s
as a successor to the ABC programming language and first re
leased it in 1991 as Python 0.9.0. Python 2.0 was released
in 2000. Python 3.0, released in 2008, was a major revision
not completely backward-compatible with earlier versions. P
ython 2.7.18, released in 2020, was the last release of Pyt
hon 2.
Python consistently ranks as one of the most popular progra
mming languages, and has gained widespread use in the machi
ne learning community.
```

How to Convert two lists into a dictionary

Method 1: Using a Dictionary Comprehension

```python
keys = ["name", "age", "city"]
values = ["John", 30, "New York"]

# Create a dictionary using a dictionary comprehension
my_dict = {keys[i]: values[i] for i in range(len(keys))}

print(my_dict)
#clcoding.com
```

```
{'name': 'John', 'age': 30, 'city': 'New York'}
```

Method 2: Using the zip() Function

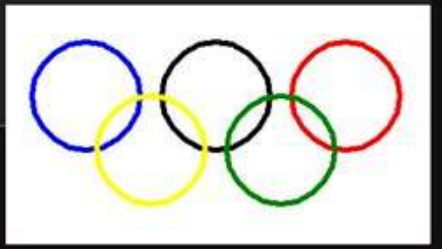
```python
keys = ["name", "age", "city"]
values = ["John", 30, "New York"]

# Create a dictionary using zip
my_dict = dict(zip(keys, values))

print(my_dict)
#clcoding.com
```

```
{'name': 'John', 'age': 30, 'city': 'New York'}
```

Olympics Logo using Python

[*]:

```
import turtle

def draw_ring(color, x, y):
    turtle.penup()
    turtle.color(color)
    turtle.goto(x, y)
    turtle.pendown()
    turtle.circle(50)

turtle.speed(5)
turtle.width(5)
draw_ring("blue", -120, 0)
draw_ring("black", 0, 0)
draw_ring("red", 120, 0)
draw_ring("yellow", -60, -50)
draw_ring("green", 60, -50)
turtle.hideturtle()
turtle.done()
#clcoding.com
```



/clcoding

/Pythoncoding

/Pythonclcoding

Convert Decimal into other number using Python

[2]:

```
dec = int(input("Enter a Decimal Number: "))

#decimal to binary
print(bin(dec), "in Binary.")

#decimal to octal
print(oct(dec), "in Octal.")

#decimal to Hexadecimal
print(hex(dec), "in Hexadecimal.")

#clcoding.com
```

```
Enter a Decimal Number:  143
0b10001111 in Binary.
0o217 in Octal.
0x8f in Hexadecimal.
```

Grammar Correction using Python

[ ]:

```
pip install textblob
```

[9]:

```
from textblob import TextBlob

def correct_grammar(text):
    blob = TextBlob(text)
    corrected_text = str(blob.correct())
    return corrected_text

text = input("Enter your Sentence: ")
corrected_text = correct_grammar(text)
print(f"Corrected: {corrected_text}")

#clcodig.com
```

```
Enter your Sentence:  mathemtics
Corrected: mathematics
```

Generate Image captcha using Python

[4]:

```
from captcha.image import ImageCaptcha
from PIL import Image
def generate_captcha_text(length):
    import string
    import random
    return ''.join(random.choices(string.ascii_letters
                                  + string.digits, k=length))

def generate_captcha(captcha_length=7,save_path='CAPTCHA.png'):
    image = ImageCaptcha(width=500, height=100)
    captcha_text = generate_captcha_text(captcha_length)
    data = image.generate(captcha_text)
    image.write(captcha_text, save_path)
    return captcha_text

if __name__ == "__main__":
    captcha_text = generate_captcha()
    print("CAPTCHA text:", captcha_text)
Image.open('CAPTCHA.png')
#clcoding.com
```

```
CAPTCHA text: E5XG6pa
```

[4]:

/clcoding

/Pythoncoding

/Pythonclcoding

```
pip install colorma
```

Printing coloured output in Python

```
from colorama import Fore
print(Fore.RED + "hello world")
print(Fore.BLUE + "hello world")
print(Fore.GREEN + "hello world")
print(Fore.YELLOW + "CLCODING.COM")
print(Fore.CYAN + "THANK YOU")
print(Back.BLUE + "CLCODING.COM")
#CLCODING.COM
```

```
hello world
hello world
hello world
CLCODING.COM
THANK YOU
CLCODING.COM
```

```
pip install periodictable
```

Python Code for Periodic Table Elements

```
import periodictable

# Get details of an element by atomic number
Atomic_No = int(input("Enter Element Atomic No :"))
element = periodictable.elements[Atomic_No]
print('Atomic number:', element.number)
print('Symbol:', element.symbol)
print('Name:', element.name)
print('Atomic mass:', element.mass)
print('Density:', element.density)
#clcoding.com
```

```
Enter Element Atomic No :26
Atomic number: 26
Symbol: Fe
Name: iron
Atomic mass: 55.845
Density: 7.874
```

Remove Image Background using Python

pip install rembg

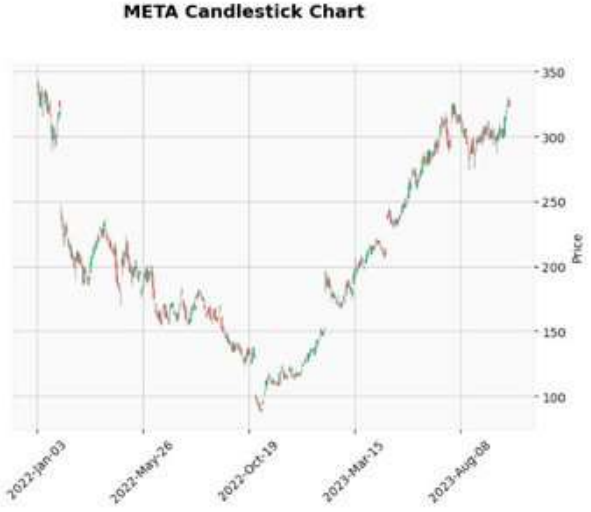
```
from rembg import remove
from PIL import Image
input_path = 'p22.jpg'
output_path = 'p22.png'
inp = Image.open(input_path)
output = remove(inp)
output.save(output_path)
#clcoding.com
```

/Pythonclcoding /Pythoncoding /clcoding

Candlestick Chart Plot using Python

```
pip install mplfinance
```

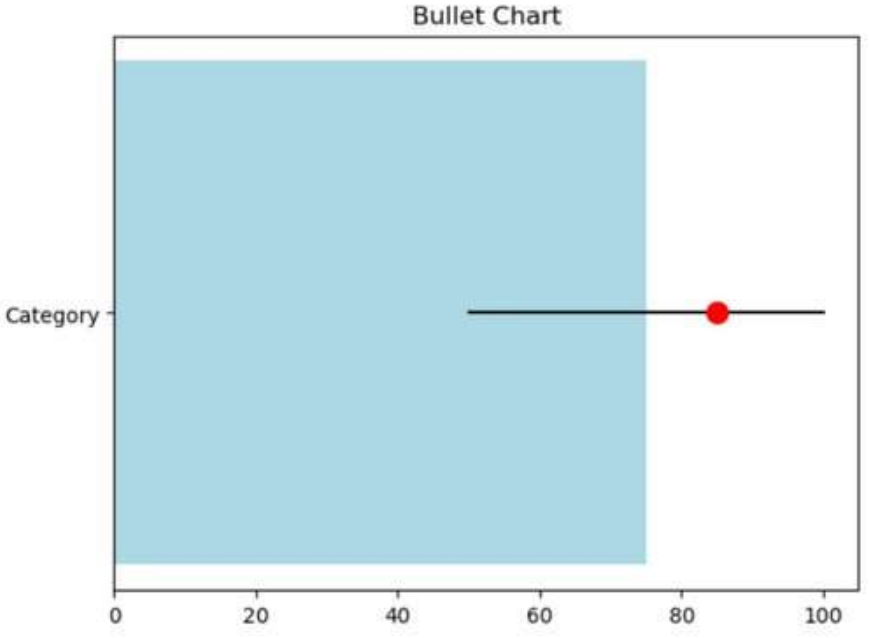
```
import yfinance as yf
import mplfinance as mpf
# Define the stock symbol and date range
symbol = input("Enter Stock Name : ")
start_date = '2022-01-01'
end_date = '2023-10-13'

# Fetch stock data
stock_data = yf.download(symbol, start=start_date, end=end_date)

# Create the candlestick chart
mpf.plot(stock_data, type='candle', style='yahoo', title=f'{symbol} Candlestick Chart')
#clcoding.com
```



Bullet Charts using Python

[6]:

```
import matplotlib.pyplot as plt
categories = ['Category']
values = [75]
ranges = [(50, 100)]
markers = [85]
fig, ax = plt.subplots()
ax.barh(categories, values, color='lightblue')
for i, (low, high) in enumerate(ranges):
    ax.plot([low, high], [i]*2, color='black')
    ax.plot([markers[i]], [i], marker='o', markersize=10, color='red')
plt.title('Bullet Chart')
plt.show()
#clcoding.com
```



Calculate derivatives in Python

```
import sympy as sym
```

```
x = sym.Symbol('x') # Symbolize X

func= x**4+4*x**2+5*x-6    # Function

sym.Derivative(func, x) # Derivative expression
```

$$\frac{d}{dx}\left(x^4 + 4x^2 + 5x - 6\right)$$

```
sym.Derivative(func, x, evaluate=True) # Calculate derivative of func
```

$$4x^3 + 8x + 5$$

```
func.diff(x)    # Or use this for the same
```

$$4x^3 + 8x + 5$$

```
# Create functions with lambdify

expr= sym.lambdify(x, func)

expr_der=sym.lambdify(x, func.diff(x))
```

```
print(f'value of func at x=5: {expr(5)}')
print(f'derivative of func at x=5: {expr_der(5)}')
```

```
value of func at x=5: 744
derivative of func at x=5: 545
```

Indefinite Integral of a polynomial function

[2]:

```
import sympy as sp

# Define the variable and function
x = sp.Symbol('x')
f = x**2 + 3*x + 2

# Compute the indefinite integral
F = sp.integrate(f, x)

print("∫f(x) dx =", F)

#clcoding.com
```

```
∫f(x) dx = x**3/3 + 3*x**2/2 + 2*x
```

Convert Video Files to a Gif in Python

[ ]:

```
pip install moviepy
```

[*]:

```
from moviepy.editor import VideoFileClip

videoClip = VideoFileClip("p3.mp4")

videoClip.write_gif("p3.gif")

#clcoding.com
```

```
MoviePy - Building file p3.gif with imageio.
```

Checking Stocks Price Using Python

pip install yfinance

```python
import yfinance as yf

# Prompting user for the share name
STK = input("Enter share name: ")

# Fetching historical market data
data = yf.Ticker(STK).history(period="1d")

# Extracting the last market price
last_market_price = data['Close'].iloc[-1]

# Displaying the last market price
print("Last market price:", last_market_price)

#clcoding.com
```

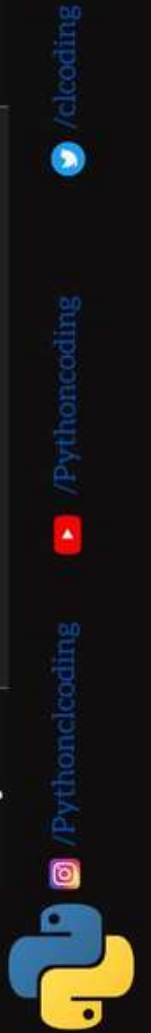
```
Enter share name:  META
Last market price: 481.07000732421875
```

Get Address details using Python

[4]:

```
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="geoapiExercises")

place = input("Enter the Place Name : ")
location = geolocator.geocode(place)

data = location.raw
loc_data = data['display_name'].split()
print("Full Location")
print(loc_data)
print("Zip code : ",loc_data[-2])

#clcoding.com
```

```
Enter the Place Name :  Banaras
Full Location
['Banaras', '(Manduadih),', 'SH74,', 'Bhullanpur,', 'Varanasi,
r,', 'Varanasi,', 'Uttar', 'Pradesh,', '221009,', 'India']
Zip code :  221009,
```



Downloading a YouTube Playlist using Python

[1]:

```
from pytube import Playlist

playlist_url = input('Enter Playlist:')
playlist = Playlist(playlist_url)

for video in playlist.videos:
    video.streams.get_highest_resolution().download()

#clcoding.com
```

```
Enter Playlist: https://www.youtube.com/watch?v=WmcyZPw3PdY
&list=PLeLGx0BaYD6aVCxN0Oj-85rYm3DdsCjBX&ab_channel=PythonC
oding
```

```
pip install geopy geocoder
```

Get PINCODE details using Python

```
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="geoapiExercises")

zip_data = input("Enter the zipcode : ")
zipcode = zip_data

location = geolocator.geocode(zipcode)

print("Zipcode:",zipcode)
print("Details of the Zipcode:")
print(location)

#clcoding.com
```

```
Enter the zipcode :  411044
Zipcode: 411044
Details of the Zipcode:
411044, Haveli, Pune, Maharashtra, India
```

[ ]:

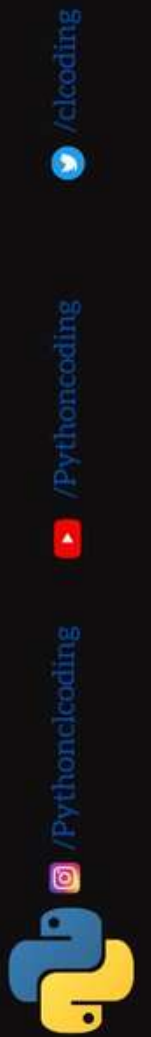
```
pip install pyshorteners
```

URL Shortener using Python - Tinyurl

[2]:

```
import pyshorteners

long_url = input("Enter the URL to shorten: ")

type_tiny = pyshorteners.Shortener()

short_url = type_tiny.tinyurl.short(long_url)

# Display the shortened URL
print("The Shortened URL is: " + short_url)

# clcoding.com
```

```
Enter the URL to shorten:  https://www.clcoding.com/2024/01/top-20-python-set-questions.html
The Shortened URL is: https://tinyurl.com/27cj4x59
```



```
pip install countryinfo
```

Country Details using Python

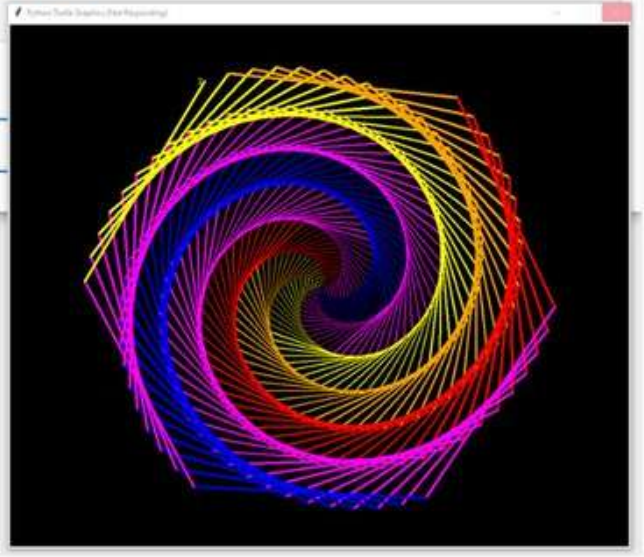
```
from countryinfo import CountryInfo
country = CountryInfo(input("Enter Country Name:"))

# Various information about the country
print("Country Name:", country.name())
print("Capital:", country.capital())
print("Population:", country.population())
print("Area (in square kilometers):", country.area())
print("Region:", country.region())
print("Subregion:", country.subregion())
print("Demonym:", country.demonym())
print("Currency:", country.currencies())
print("Languages:", country.languages())
print("Borders: ", country.borders())

#clcoding.com
```

```
Enter Country Name: Germany
Country Name: germany
Capital: Berlin
Population: 80783000
Area (in square kilometers): 357114
Region: Europe
Subregion: Western Europe
Demonym: German
Currency: ['EUR']
Languages: ['de']
Borders:  ['AUT', 'BEL', 'CZE', 'DNK', 'FRA', 'LUX', 'NLD', 'POL', 'CHE']
```

Colorful Galaxy using Python

```
import turtle
t = turtle.Turtle()
#clcoding.com
s = turtle.Screen()
colors=['orange', 'red', 'magenta', 'blue', 'magenta',
        'yellow', 'green', 'cyan', 'purple']
s.bgcolor('black')
t.pensize('2')
t.speed(0)
for x in range (360):
    t.pencolor(colors[x%6])
    t.width(x//100+1)
    t.forward(x)
    t.right(59)
    turtle.hideturtle()
#clcoding.com
```

```
[ ]:
```



/clcoding

/Pythoncoding

/Pythonclcoding

File Chooser using Python

[2]:

```
from plyer import filechooser

# Open a file chooser dialog
file_path = filechooser.open_file()
print("Selected file:", file_path)

# Open multiple files chooser dialog
files_path = filechooser.open_file(multiple=True)
print("Selected files:", files_path)

# Save file chooser dialog
save_path = filechooser.save_file()
print("Save file path:", save_path)

#clcoding.com
```

```
Selected file: []
Selected files: []
Save file path: ['C:\\Users\\IRAWEN\\Documents\\backup-hdd
\\New folder\\aa']
```

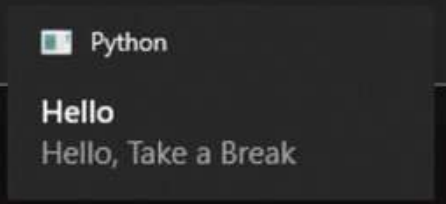
Displaying a Notification

[ ]:

```
pip install plyer
```

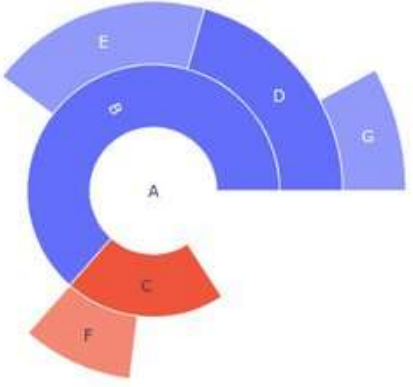
•[2]:

```
from plyer import notification

# Display a notification

notification.notify(
    title='Hello',
    message='Hello, Take a Break',
    app_name='Python Clcoding'
)

#clcoding.com
```



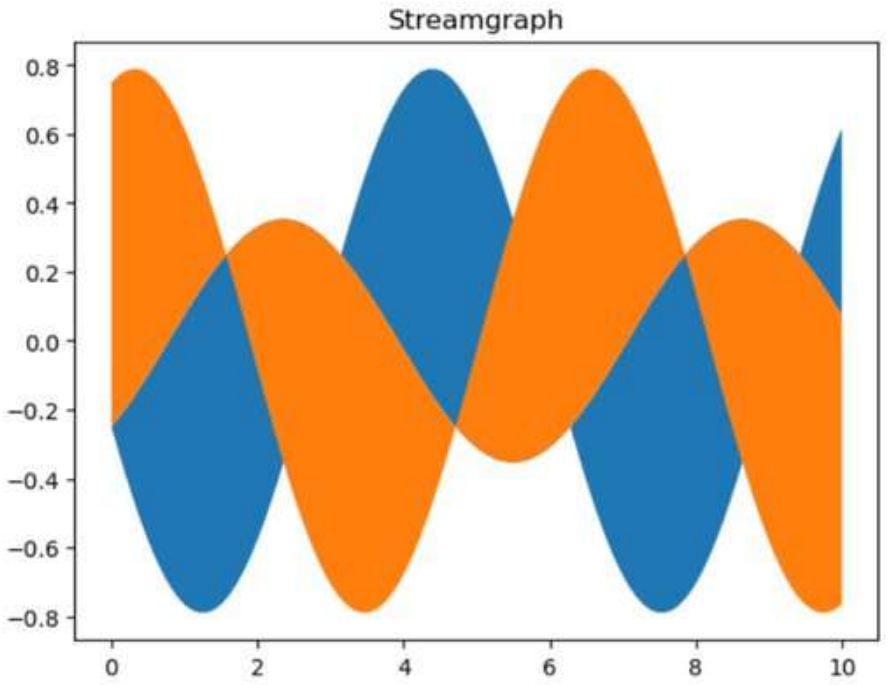
Sunburst Chart in Python

```python
import plotly.express as px

# Sample data
data = {
    'id': ["A", "B", "C", "D", "E", "F", "G"],
    'parent': ["", "A", "A", "B", "B", "C", "D"],
    'value': [10, 15, 7, 8, 12, 6, 5],
}

# Create a sunburst chart
fig = px.sunburst(data, names='id', parents='parent', values='value')

# Set the chart title
fig.update_layout(title_text="Sunburst Chart")

# Show the chart
fig.show()
#clcoding.com
```



Streamgraphs using Python

```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

plt.stackplot(x, y1, y2, baseline='wiggle')
plt.title('Streamgraph')
plt.show()
```



QR Code generation using Python

```
pip install pyqrcode
```

```
import pyqrcode
from PIL import Image
link = input("Enter anything to generate QR : ")
qr_code = pyqrcode.create(link)
qr_code.png("QRCode.png", scale=5)
Image.open("QRCode.png")

#clcoding.com
```

```
Enter anything to generate QR :  https://x.com/clcoding
```

/clcoding

/Pythoncoding

/Pythonclcoding

Generate Audio captcha in Python

[ ]:

```
pip install captcha
```

[1]:

```
import random
from captcha.audio import AudioCaptcha

def generate_random_captcha(length=6):
    characters =  '1234567890'
    captcha_text = ''.join(random.choices(characters, k=length))
    return captcha_text

audio = AudioCaptcha()
captcha_text = generate_random_captcha()
print("Generated CAPTCHA text:", captcha_text)
audio_captcha = audio.generate(captcha_text)
audio.write(captcha_text, 'Audio_Captcha.wav')
print("Audio CAPTCHA generated: Audio_Captcha.wav")

#clcoding.com
```

```
Generated CAPTCHA text: 447022
Audio CAPTCHA generated: Audio_Captcha.wav
```

```
pip install sounddevice
```

Voice Recorder in Python

[1]:

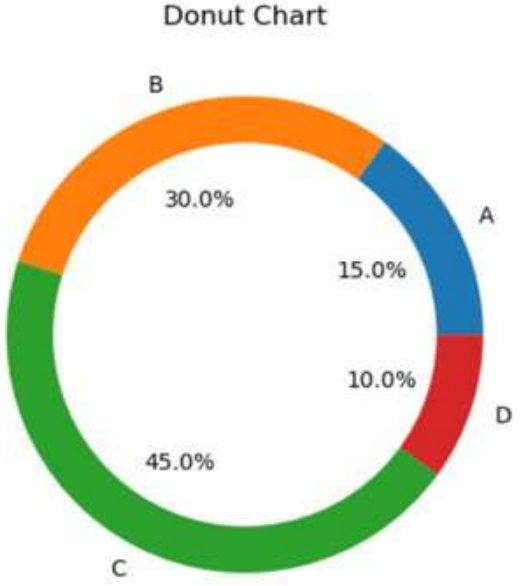
```
import sounddevice
from scipy.io.wavfile import write

#sample_rate
fs=44100

#Ask to enter the recording time
second=int(input("Enter the Recording Time in second: "))
print("Recording....\n")
record_voice=sounddevice.rec(int(second * fs),samplerate=fs,channels=2)
sounddevice.wait()
write("MyRecording.wav",fs,record_voice)
print("Recording is done Please check you folder to listen recording")

#clcoding.com
```

```
Enter the Recording Time in second:  10
Recording....

Recording is done Please check you folder to listen recording
```

Donut Charts using Python

[4]:

```
import matplotlib.pyplot as plt

sizes = [15, 30, 45, 10]
labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%')
circle = plt.Circle((0, 0), 0.8, color='white')
plt.gca().add_artist(circle)
plt.title('Donut Chart')
plt.show()

#clcoding.com
```



Convert PDF files using Python

1. Convert PDF to Audio (MP3) using gTTS:

[ ]:

```
from gtts import gTTS
from PyPDF2 import PdfReader

def pdf_to_text(pdf_file):
    text = ""
    with open(pdf_file, 'rb') as f:
        reader = PdfReader(f)
        for page_num in range(len(reader.pages)):
            page = reader.pages[page_num]
            text += page.extract_text()
    return text

def text_to_audio(text, output_file):
    tts = gTTS(text)
    tts.save(output_file)

# Example usage:
pdf_file = "clcoding.pdf"
output_audio_file = "clcoding_audio.mp3"

text = pdf_to_text(pdf_file)
text_to_audio(text, output_audio_file)

#clcoding.com
```

Convert PDF files using Python

2. Convert PDF to Images (PNG) using PyPDF2 Pillow:

```python
import os
from PyPDF2 import PdfReader
from pdf2image import convert_from_path

def pdf_to_images(pdf_file, output_dir):
    images = []
    with open(pdf_file, 'rb') as f:
        reader = PdfReader(f)
        for page_num, _ in enumerate(reader.pages):
            # Convert each PDF page to image
            img_path = os.path.join(output_dir, f"page_{page_num}.png")
            images.append(img_path)
    return images

# Example usage:
pdf_file = "clcoding.pdf"
output_dir = "output_images"

if not os.path.exists(output_dir):
    os.makedirs(output_dir)

pdf_to_images(pdf_file, output_dir)

#clcoding.com
```

Yellow Heart using Python

[*]:

```
import turtle

t = turtle.Turtle()
t.shapesize(0.2, 0.2)
s = turtle.Screen()
s.bgcolor('black')

t.fillcolor("yellow")
t.begin_fill()

t.left(50)
t.forward(240)
t.circle(90, 200)
t.left(221)
t.circle(90, 200)
t.forward(260)

t.end_fill()

turtle.done()
#clcoding.com
```

/Pythonclcoding /Pythoncoding /clcoding

Create Dummy Data using Python

[12]:

```
from faker import Faker
fake = Faker()
print(fake.name())
print(fake.address())
print(fake.text())
print(fake.email())
print(fake.country())
print(fake.latitude(), fake.longitude())
print(fake.url())

#clcoding.com
```

```
Danny Gonzalez
8292 King Fork
Alexanderview, AR 45700
Inside other series force hot operation health. Economic mo
dern as safe boy whether own.
Suggest indicate president class. Water receive term explai
n.
kbrown@example.com
Hungary
-52.0710205 -177.740046
http://reyes-smith.biz/
```

```
pip install pdf2docx
```

Convert PDF to docx using Python

[3]:

```
from pdf2docx import Converter
pdf_file = 'cloding.pdf'
docx_file = 'cloding.docx'
cv = Converter(pdf_file)
cv.convert(docx_file)
cv.close()

#clcoding.com
```


```
[INFO] Start to convert cloding.pdf
[INFO] [1/4] Opening document...
[INFO] [2/4] Analyzing document...
[INFO] [3/4] Parsing pages...
[INFO] (1/2) Page 1
[INFO] (2/2) Page 2
[INFO] [4/4] Creating pages...
[INFO] (1/2) Page 1
[INFO] (2/2) Page 2
[INFO] Terminated in 0.39s.
```

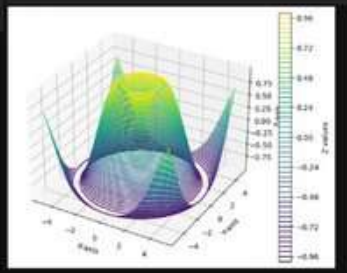
/clcoding

/Pythoncoding

/Pythonclcoding

3D contour plot using Python

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
def f(x, y):
    return np.sin(np.sqrt(x**2 + y**2))
Z = f(X, Y)
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
contour = ax.contour3D(X, Y, Z, 50, cmap='viridis')

# Add labels and a colorbar
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
fig.colorbar(contour, ax=ax, label='Z values')
plt.show()
```



/clcoding

/Pythoncoding

/Pythonclcoding

Pdf To Audio using Python

pip install PyPDF2

pip install pyttsx3

```python
# importing the modules
import PyPDF2,pyttsx3

# PDF file
path = open('clcoding.pdf', 'rb')

# creating a PdfFileReader object
pdfReader = PyPDF2.PdfFileReader(path)

# Get an engine instance for the speech synthesis
speak = pyttsx3.init()

for pages in range(pdfReader.numPages):
    text = pdfReader.getPage(pages).extractText()
    speak.say(text)
    speak.runAndWait()
speak.stop()
#clcoding.com
```

TO

Audio

1. Integers Are of Arbitrary Precision

In Python, integers are of arbitrary precision, meaning they can grow as large as the memory allows. This is different from many other programming languages where integers have fixed sizes.

[1]:

```
a = 10**100
print(a)

#clcoding.com
```

```
1000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000
000000000000000
```

2. Floating-Point Precision

Floating-point numbers in Python are based on the IEEE 754 double-precision standard, which can lead to precision issues.

•[2]:

```
a = 0.1 + 0.2
print(a)

#clcoding.com
```

```
0.30000000000000004
```

3. Complex Numbers

Python natively supports complex numbers, which can be created by adding a 'j' or 'J' suffix to a number.

•[3]:

```
a = 3 + 4j
print(a.real)
print(a.imag)

#clcoding.com
```

```
3.0
4.0
```


4. Built-In Functions for Numerical Operations

Python provides several built-in functions to perform various numerical operations, such as abs(), round(), pow(), and many more.

•[4]:

```
print(abs(-5))
print(round(3.14159, 2))
print(pow(2, 3))

#clcoding.com
```

```
5
3.14
8
```


6 Things I Wish I Knew Earlier About Python Numbers

5. Decimal Module for Precise Calculations

For financial and other applications requiring precise decimal representation, the decimal module is very useful.

•[5]:

```
from decimal import Decimal, getcontext

getcontext().prec = 6
a = Decimal('0.1')
b = Decimal('0.2')
c = a + b
print(c)

#clcoding.com
```

```
0.3
```

6 Things I Wish I Knew Earlier About Python Numbers

6. Fraction Module for Rational Numbers

The fractions module provides support for rational number arithmetic.

•[6]:

```
from fractions import Fraction

a = Fraction(1, 3)
b = Fraction(2, 3)
c = a + b
print(c)

#clcoding.com
```

```
1
```

WHERE MATH DOESN'T WORK IN PYTHON

1. Precision Issues:

[1]:

```
x = 1.0
y = 1e-16
result = x + y
print(result)

#clcoding.com
```

```
1.0
```


WHERE MATH DOESN'T WORK IN PYTHON

2. Comparing Floating-Point Numbers:

[2]:

```
a = 0.2 + 0.4
b = 0.6
print(a == b)

#clcoding.com
```

```
False
```


3. NaN (Not a Number) and Inf (Infinity):

[3]:

```
result = float('inf') / float('inf')
print(result)

#clcoding.com
```

```
nan
```

WHERE MATH DOESN'T WORK IN PYTHON

4. Large Integers:

[4]:

```
result = 2 ** 1000
print(result)

#clcoding.com
```

```
107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881569585812759467291755314682518714528569231404359845775746985748039345677748242309854210746050623711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316526243868372056680693776
```

/clcoding

/Pythoncoding

/Pythonclcoding

WHERE MATH DOESN'T WORK IN PYTHON

5. Round-off Errors:

[5]:

```
result = 0.1 + 0.2 - 0.3
print(result)

#clcoding.com
```

```
5.551115123125783e-17
```

NUMERICAL METHODS IN PYTHON

1. Newton-Raphson Method:

Used for finding successively better approximations to the roots (or zeroes) of a real-valued function.

[1]:

```
import numdifftools as nd

def newton_raphson(func, initial_guess, tolerance=1e-10, max_iterations=100):
    x0 = initial_guess
    for _ in range(max_iterations):
        fx0 = func(x0)
        if abs(fx0) < tolerance:
            return x0
        fprime_x0 = nd.Derivative(func)(x0)
        x0 = x0 - fx0 / fprime_x0
    raise ValueError("Failed to converge")

# Example usage:
import math

def f(x):
    return x**3 - 2*x - 5

root = newton_raphson(f, initial_guess=3)
print(f"Root found at x = {root}")

#clcoding.com
```

```
Root found at x = 2.0945514815423474
```

NUMERICAL METHODS IN PYTHON

2. Euler's Method:

A first-order numerical procedure for solving ordinary differential equations (ODEs).

•[2]:

```python
def euler_method(func, initial_x, initial_y, step_size, num_steps):
    x = initial_x
    y = initial_y
    for _ in range(num_steps):
        y += step_size * func(x, y)
        x += step_size
    return x, y

# Example usage:
def dy_dx(x, y):
    return x + y

x_final, y_final = euler_method(dy_dx, initial_x=0,
                                initial_y=1, step_size=0.1, num_steps=100)
print(f"At x = {x_final}, y = {y_final}")

#clcoding.com
```

```
At x = 9.99999999999998, y = 27550.224679644543
```

NUMERICAL METHODS IN PYTHON

3. Secant Method:

A root-finding algorithm that uses a succession of roots of secant lines to better approximate a root of a function.

[3]:

```
def secant_method(func, x0, x1, tolerance=1e-10, max_iterations=100):
    for _ in range(max_iterations):
        fx1 = func(x1)
        if abs(fx1) < tolerance:
            return x1
        fx0 = func(x0)
        denominator = (fx1 - fx0) / (x1 - x0)
        x_next = x1 - fx1 / denominator
        x0, x1 = x1, x_next
    raise ValueError("Failed to converge")

# Example usage:
def g(x):
    return x**3 - 2*x - 5

root = secant_method(g, x0=2, x1=3)
print(f"Root found at x = {root}")

#clcoding.com
```

```
Root found at x = 2.094551481542327
```

NUMERICAL METHODS IN PYTHON

4. Bisection Method:

A root-finding algorithm that repeatedly bisects an interval and then selects a subinterval in which a root must lie for further processing.

[4]:

```
def bisection_method(func, a, b, tolerance=1e-10, max_iterations=100):
    if func(a) * func(b) >= 0:
        raise ValueError("Function does not change sign over interval")

    for _ in range(max_iterations):
        c = (a + b) / 2
        if abs(func(c)) < tolerance:
            return c
        if func(c) * func(a) < 0:
            b = c
        else:
            a = c
    raise ValueError("Failed to converge")

# Example usage:
def h(x):
    return x**3 - 2*x - 5

root = bisection_method(h, a=2, b=3)
print(f"Root found at x = {root}")

#clcoding.com
```

```
Root found at x = 2.0945514815393835
```

NUMERICAL METHODS IN PYTHON

5. Runge-Kutta Method (RK4):

A fourth-order numerical method for solving ordinary differential equations (ODEs), more accurate than Euler's method for many types of problems.

•[5]:

```
def runge_kutta_4(func, initial_x, initial_y, step_size, num_steps):
    x = initial_x
    y = initial_y
    for _ in range(num_steps):
        k1 = step_size * func(x, y)
        k2 = step_size * func(x + step_size / 2, y + k1 / 2)
        k3 = step_size * func(x + step_size / 2, y + k2 / 2)
        k4 = step_size * func(x + step_size, y + k3)
        y += (k1 + 2*k2 + 2*k3 + k4) / 6
        x += step_size
    return x, y

# Example usage:
def dy_dx(x, y):
    return x + y

x_final, y_final = runge_kutta_4(dy_dx, initial_x=0,
                                 initial_y=1, step_size=0.1, num_steps=100)
print(f"At x = {x_final}, y = {y_final}")

#clcoding.com
```

```
At x = 9.99999999999998, y = 44041.593801752446
```

Download YouTube videos

[1]:

```
from pytube import YouTube

video_url = input(" Enter Youtube Video link:")

yt = YouTube(video_url)

stream = yt.streams.get_highest_resolution()

stream.download()

print("Download completed!")

#clcoding.com
```

```
 Enter Youtube Video link: https://www.youtube.com/watch?v=6VYtgkl3HPQ
Download completed!
```

In Comprehensions:

While not a direct use of else, Python comprehensions can incorporate conditional logic that mimics if-else behavior.

[7]:

```
numbers = [1, 2, 3, 4, 5]
even_odd = ["Even" if num % 2 == 0
            else "Odd" for num in numbers]
print(even_odd)

#clcoding.com
```

```
['Odd', 'Even', 'Odd', 'Even', 'Odd']
```

In Context Managers:

Although not a common practice, else can be used in conjunction with context managers to execute code based on the successful completion of the context block.

•[5]:

```
class CustomContextManager:
    def __enter__(self):
        print("Entering context")
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        if exc_type is None:
            print("Exiting context successfully")
        else:
            print("Exiting context with exception:", exc_type)

with CustomContextManager():
    print("Inside context block")

#clcoding.com
```

```
Entering context
Inside context block
Exiting context successfully
```

With Functions and Returns:

You can use the else statement to provide alternative return paths in functions, making the logic more readable and explicit.

•[3]:

```
def check_even(number):
    if number % 2 == 0:
        return True
    else:
        return False

print(check_even(4))
print(check_even(5))

#clcoding.com
```

```
True
False
```

Program to Check Whether a Number is Even or Odd

[3]:

```
num = int(input("Enter a number: "))
if num % 2 == 0:
    print(num, "is even")
else:
    print(num, "is odd")
```

```
Enter a number:  12
12 is even
```

Program to Check Whether a Character is a Vowel or Consonant

[4]:

```
char = input("Enter a character: ").lower()
if char in 'aeiou':
    print(char, "is a vowel")
else:
    print(char, "is a consonant")
#clcoding.com
```

```
Enter a character:  c
c is a consonant
```

Using in for Substring Checks:

Checking if a substring exists within a string using the in keyword is simple and effective.

[6]:

```
text = "The quick brown fox"
print("quick" in text)  # True
print("slow" in text)  # False

#clcoding.com
```

```
True
False
```

Program to Demonstrate the Working of Keyword long

In Python, all integers are of type int and can be arbitrarily large.

[1]:

```
num = 12345678901234567890
print("Long integer:", num)
```

```
Long integer: 12345678901234567890
```

Program to Swap Two Numbers

•[2]:

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
a, b = b, a
print("After swapping: a =", a, "b =", b)

#clcoding.com
```

```
Enter first number:  5
Enter second number:  6
After swapping: a = 6 b = 5
```

Program to Find the Size of int, float, double, and char

•[6]:

```
import sys

print("Size of int:", sys.getsizeof(int()))
print("Size of float:", sys.getsizeof(float()))
print("Size of double:", sys.getsizeof(float()))
print("Size of char:", sys.getsizeof(chr(0)))


#clcoding.com
```

```
Size of int: 28
Size of float: 24
Size of double: 24
Size of char: 50
```

Program to Compute Quotient and Remainder

[5]:

```
dividend = int(input("Enter dividend: "))
divisor = int(input("Enter divisor: "))
quotient = dividend // divisor
remainder = dividend % divisor
print("Quotient:", quotient)
print("Remainder:", remainder)

#clcoding.com
```

```
Enter dividend:  16
Enter divisor:  5
Quotient: 3
Remainder: 1
```

Program to Add Two Integers

[3]:

```
a = int(input("Enter first integer: "))
b = int(input("Enter second integer: "))
sum = a + b
print("Sum:", sum)
```

```
Enter first integer:  15
Enter second integer:  5
Sum: 20
```

Program to Multiply Two Floating-Point Numbers

•[4]:

```
a = float(input("Enter first floating-point number: "))
b = float(input("Enter second floating-point number: "))
product = a * b
print("Product:", product)
#clcoding.com
```

```
Enter first floating-point number:  1.2
Enter second floating-point number:  2.4
Product: 2.88
```

Different Ways to Format and Print Characters in Python

1. Using f-strings (Python 3.6+)

```
[1]: print(f'[{chr(65)}]')
```

```
[A]
```

2. Using str.format method

```
[2]: print('[{}]'.format(chr(65)))
```

```
[A]
```

3. Using format function with placeholders

```
[3]: print('[{:c}]'.format(65))
```

```
[A]
```

4. Using concatenation with chr function

```
[4]: print('[' + chr(65) + ']')
```

```
[A]
```

5. Using old-style string formatting

```
[5]: print('[%c]' % 65)
```

```
[A]
```

Different Ways to Print Binary Values as Decimals in Python

1. Directly using the binary literal

```
[1]: print(0b101)
```

```
5
```

2. Using int function with a binary string

```
[2]: print(int('101', 2))
```

```
5
```

3. Using format function for binary to decimal conversion

```
[3]: print(int(format(0b101, 'd')))
```

```
5
```

4. Using f-strings (Python 3.6+)

```
[5]: binary_value = 0b101
     print(f'{binary_value}')
```

```
5
```

5. Using str.format method

```
[6]: binary_value = 0b101
     print('{}'.format(binary_value))
```

```
5
```

Without Currying in Python

The function total_cost takes all three arguments (price, tax_rate, and discount) at once. It performs the calculation directly and returns the result.

[2]:

```
def total_cost(price, tax_rate, discount):
    return price + (price * tax_rate) - discount

# Using the function
price = 100
tax_rate = 0.05
discount = 10
print(total_cost(price, tax_rate, discount))

#clcoding.com
```

```
95.0
```

With Currying in Python

The function total_cost takes one argument price and returns another function apply_tax. apply_tax takes one argument tax_rate and returns another function apply_discount. apply_discount takes one argument discount and performs the calculation.

[1]:

```
def total_cost(price):
    def apply_tax(tax_rate):
        def apply_discount(discount):
            return price + (price * tax_rate) - discount
        return apply_discount
    return apply_tax

# Using the curried function
price = 100
tax_rate = 0.05
discount = 10

curried_total_cost = total_cost(price)
apply_tax = curried_total_cost(tax_rate)
apply_discount = apply_tax(discount)
print(apply_discount)
```

```
95.0
```

/clcoding

/Pythoncoding

/Pythonclcoding

Rank of Matrix

[3]:

```
import numpy as np

x = np.matrix("4,5,16,7;2,-3,2,3;,3,4,5,6;4,7,8,9")
print(x)
```

```
[[ 4  5 16  7]
 [ 2 -3  2  3]
 [ 3  4  5  6]
 [ 4  7  8  9]]
```

[4]:

```
#numpy.linalg.matrix_rank() - return a rank of a matrix
# Syntax: numpy.linalg.matrix_rank(matrix)
```

[5]:

```
rank_matrix = np.linalg.matrix_rank(x)
print(rank_matrix)
```

```
4
```

Inverse of a Matrix

inverse formula = A^{-1} = (1/determinant of A) * adj A

numpy.linalg.inv() - return the multiplicative inverse of a matrix Syntax: numpy.linalg.inv(matrix)

[5]:

```
A = np.matrix("3,1,2;3,2,5;6,7,8")
print(A)
```

```
[[3 1 2]
 [3 2 5]
 [6 7 8]]
```

[6]:

```
Inv_matrix = np.linalg.inv(A)
print(Inv_matrix)
```

```
[[ 0.57575758 -0.18181818 -0.03030303]
 [-0.18181818 -0.36363636  0.27272727]
 [-0.27272727  0.45454545 -0.09090909]]
```

Downloading a Audio file from Youtube Video

[2]:

```
from pytube import YouTube

url = input('Enter a Video to Download Audio:')
yt = YouTube(url)

# Filter audio streams and download the first one
audio_stream = yt.streams.filter(only_audio=True).first()
audio_stream.download()
```

```
Enter a Video to Download Audio: https://youtu.be/6VYtgkl3HPQ
```

[2]:

```
'C:\\Users\\IRAWEN\\OneDrive\\Documents\\Python Scripts\\The for loop in Python.mp4'
```

Hierarchy of Operators

Hierarchy of arithmetic operators

Decreasing order of precedence

- Parentheses -> ()
- Exponent -> **
- Division -> /
- Multiplication -> *
- Addition and Subtraction -> +,-

[9]:

```
A = 7-2*(27/3**2)+4
print(A)
```

```
5.0
```

Assignment Operator

=, +=, -=, *=, /=

[16]:

```
a = 5
b = 2
a += b # a = a + b
print(a)
a -= b
print(a)
a *= b
print(a)
a /= b
print(a)
```

```
7
5
10
5.0
```

Comparison Operators

<, <=, >, >=, ==

x = 17 y = 17 print(x <= y) print(x >= y) print(x == y)

Logical Operators

and, or, not

[36]:

```
print(x <= y and x == y)
```

```
True
```

[34]:

```
print(x <= y or x == y)
```

```
True
```

[37]:

```
print(not(x <= y or x == y))
print(not(x <= y and x == y))
```

```
False
False
```

Membership Operator

in , not in

[41]:

```
a = [1,2,3]
print(5 in a)
print(5 not in a)
```

```
False
True
```

Identity Operators

is, is not

[43]:

```
a = 5
b = 5
print(a is b)
print(a == b)
```

```
True
True
```

[44]:

```
p = [1,2,3]
q = [1,2,3]
print(p is q)
print(p == q)
```

```
False
True
```

Bitwise Operators

|, &

[46]:

```
print(6|3)

#  0000000000000010 -> 2
```

7

[ ]:

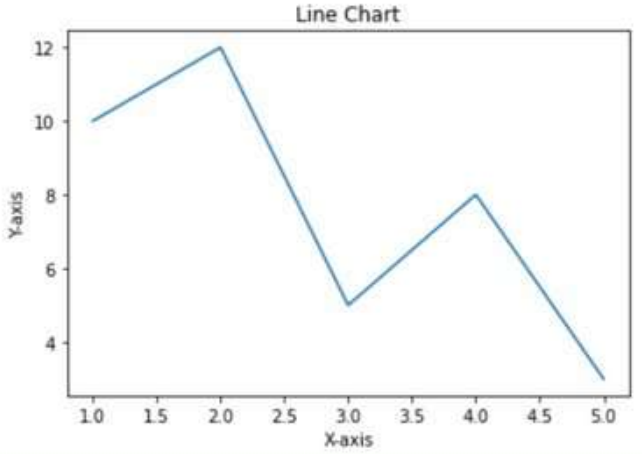
```
#  0000000000000110 - 6
#  0000000000000011 - 3
#  0000000000000111 - 7
```

10 different data charts using Python

10 different data charts using Python

1. Line Chart:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 12, 5, 8, 3]

plt.plot(x, y)
plt.title('Line Chart')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
#clcoding.com
```



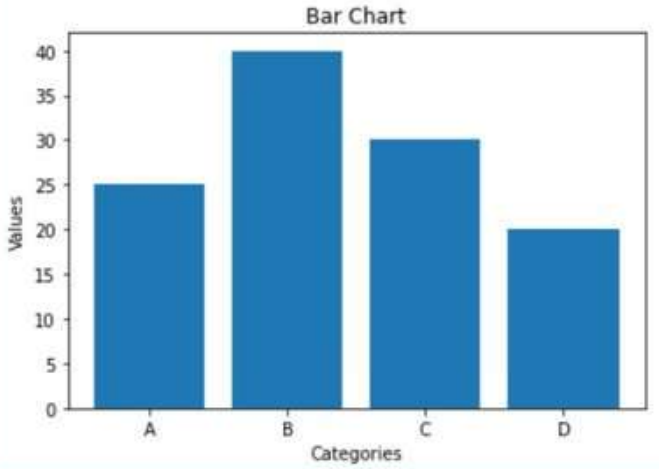
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

2. Bar Chart:

```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D']
values = [25, 40, 30, 20]

plt.bar(categories, values)
plt.title('Bar Chart')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
#clcoding.com
```



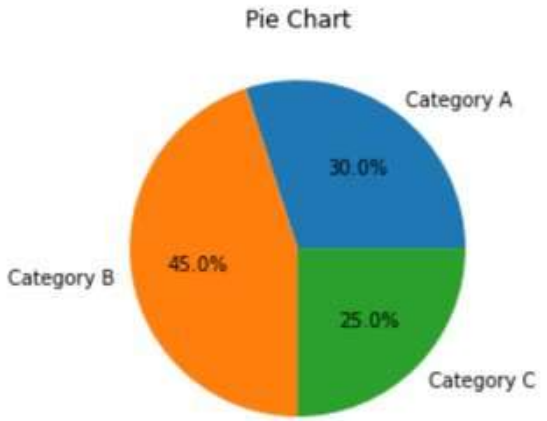
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

3. Pie Chart:

```python
import matplotlib.pyplot as plt

labels = ['Category A', 'Category B', 'Category C']
sizes = [30, 45, 25]

plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title('Pie Chart')
plt.show()
#clcoding.com
```



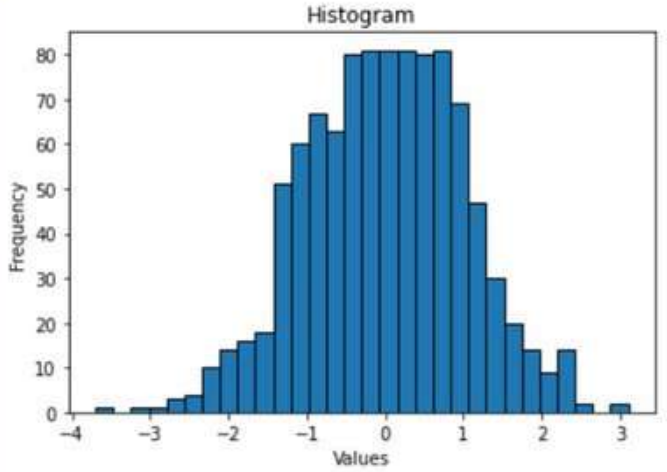
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

4. Histogram:

```python
import matplotlib.pyplot as plt
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, edgecolor='black')
plt.title('Histogram')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
#clcoding.com
```



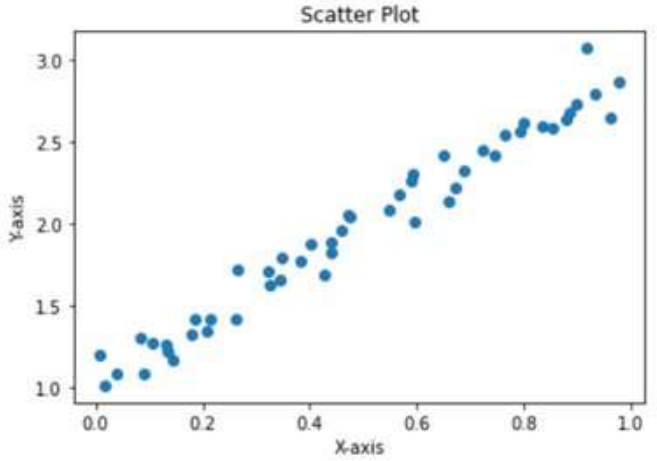
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

5. Scatter Plot:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.rand(50)
y = 2 * x + 1 + 0.1 * np.random.randn(50)

plt.scatter(x, y)
plt.title('Scatter Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
#clcoding.com
```



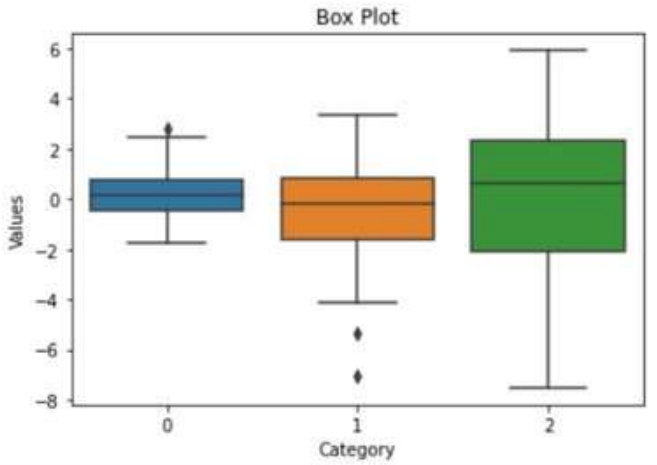
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

6. Box Plot:

```
import seaborn as sns
import numpy as np

data = [np.random.normal(0, std, 100) for std in range(1, 4)]

sns.boxplot(data=data)
plt.title('Box Plot')
plt.xlabel('Category')
plt.ylabel('Values')
plt.show()
#clcoding.com
```



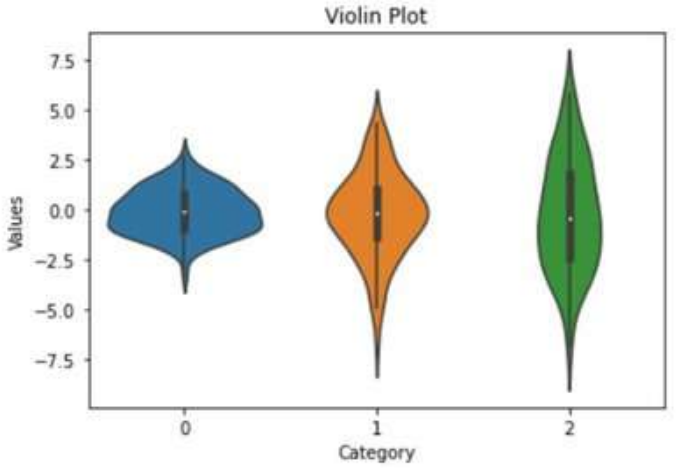
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

7. Violin Plot:

```
import seaborn as sns
import numpy as np

data = [np.random.normal(0, std, 100) for std in range(1, 4)]

sns.violinplot(data=data)
plt.title('Violin Plot')
plt.xlabel('Category')
plt.ylabel('Values')
plt.show()
#clcoding.com
```



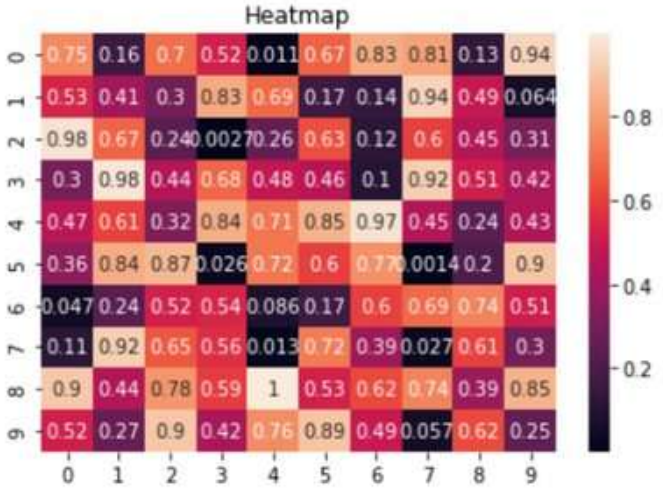
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

8. Heatmap:

```python
import seaborn as sns
import numpy as np

data = np.random.rand(10, 10)

sns.heatmap(data, annot=True)
plt.title('Heatmap')
plt.show()
#clcoding.com
```



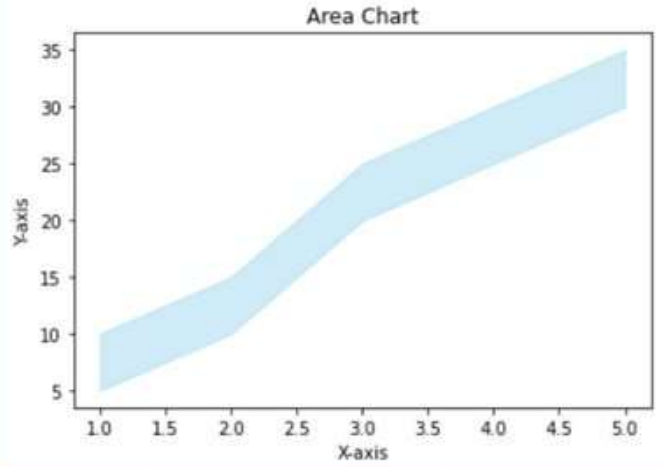
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

9. Area Chart:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y1 = [10, 15, 25, 30, 35]
y2 = [5, 10, 20, 25, 30]

plt.fill_between(x, y1, y2, color='skyblue', alpha=0.4)
plt.title('Area Chart')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
#clcoding.com
```



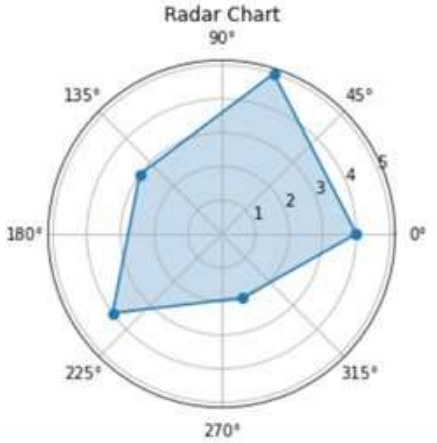
Learn More

/Pythonclcoding /Pythoncoding /clcoding

10 different data charts using Python

10. Radar Chart:

```python
import matplotlib.pyplot as plt
import numpy as np

labels = np.array([' A', ' B', ' C', ' D', ' E'])
data = np.array([4, 5, 3, 4, 2])

angles = np.linspace(0, 2 * np.pi, len(labels), endpoint=False)
data = np.concatenate((data, [data[0]]))
angles = np.concatenate((angles, [angles[0]]))

plt.polar(angles, data, marker='o')
plt.fill(angles, data, alpha=0.25)
plt.title('Radar Chart')
plt.show()
#clcoding.com
```



Learn More

/Pythonclcoding /Pythoncoding /clcoding

5 Levels of Writing Python Lists

5 Levels of Writing Python Lists

Level 1: Basic List Creation and Access

[1]:

```
# Creating a list
fruits = ['apple', 'banana', 'cherry']

# Accessing elements
print(fruits[0])

# Adding an element
fruits.append('date')
print(fruits)

# Removing an element
fruits.remove('banana')
print(fruits)

#clcoding.com
```

```
apple
['apple', 'banana', 'cherry', 'date']
['apple', 'cherry', 'date']
```

5 Levels of Writing Python Lists

Level 2: List Slicing and Iteration

[2]:

```
# Slicing a list
print(fruits[1:3])  # Output: ['cherry', 'date']

# Iterating over a list
for fruit in fruits:
    print(fruit)

#clcoding.com
```

```
['cherry', 'date']
apple
cherry
date
```

Level 3: List Comprehensions

List comprehensions offer a concise way to create lists. They can replace loops for generating list elements.

[3]:

```
# Creating a list of squares using list comprehension
squares = [x**2 for x in range(10)]
print(squares)

#clcoding.com
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

5 Levels of Writing Python Lists

Level 4: Nested Lists and Multidimensional Arrays

[4]:

```
# Creating a nested list (2D array)
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Accessing elements in a nested list
print(matrix[1][2])  # Output: 6

# Iterating over a nested list
for row in matrix:
    for elem in row:
        print(elem, end=' ')
    print()
#clcoding.com
```

```
6
1 2 3
4 5 6
7 8 9
```

5 Levels of Writing Python Lists

Level 5: Advanced List Operations

[7]:

```
# List comprehension with conditionals
even_squares = [x**2 for x in range(10) if x % 2 == 0]
print(even_squares)

# Using higher-order functions: map, filter, and reduce
from functools import reduce

numbers = [1, 2, 3, 4, 5]

# Map: Apply a function to all items in the list
doubled = list(map(lambda x: x * 2, numbers))
print(doubled)  #

# Filter: Filter items based on a condition
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)

#clcoding.com
```

```
[0, 4, 16, 36, 64]
[2, 4, 6, 8, 10]
[2, 4]
```

7 level of writing Python Dictionary

7 Levels of Writing Python Dictionary

Level 1: Basic Dictionary Creation

Create a simple dictionary with key-value pairs.

•[1]:

```
# Creating a basic dictionary
person = {
    "name": "Alice",
    "age": 30,
    "city": "New York"
}
print(person)

#Clcoding.com
```

```
{'name': 'Alice', 'age': 30, 'city': 'New York'}
```

7 Levels of Writing Python Dictionary

Level 2: Accessing and Modifying Values

Access values using keys, and modify existing key-value pairs.

•[2]:

```
# Accessing values
print(person["name"])

# Modifying values
person["age"] = 31
print(person["age"])

#Clcoding.com
```

```
Alice
31
```

7 Levels of Writing Python Dictionary

Level 3: Adding and Removing Key-Value Pairs

Add new key-value pairs and remove existing ones.

•[3]:

```
# Adding a new key-value pair
person["email"] = "alice@example.com"
print(person)

# Removing a key-value pair
del person["city"]
print(person)

#Clcoding.com
```

```
{'name': 'Alice', 'age': 31, 'city': 'New York', 'email':
'alice@example.com'}
{'name': 'Alice', 'age': 31, 'email': 'alice@example.com'}
```

7 Levels of Writing Python Dictionary

Level 4: Dictionary Methods

Use dictionary methods like keys(), values(), items(), get(), and pop()

•[4]:

```
# Getting all keys
print(person.keys())
# Getting all values
print(person.values())
# Getting all key-value pairs
print(person.items())
# Using get() method
print(person.get("name"))
print(person.get("city", "Not Found"))
# Using pop() method
email = person.pop("email")
print(email)
print(person)
```

```
dict_keys(['name', 'age', 'email'])
dict_values(['Alice', 31, 'alice@example.com'])
dict_items([('name', 'Alice'), ('age', 31), ('email', 'alice@example.co
m')])
Alice
Not Found
alice@example.com
{'name': 'Alice', 'age': 31}
```

7 Levels of Writing Python Dictionary

Level 5: Dictionary Comprehensions

Create dictionaries using dictionary comprehensions for more concise and readable code.

•[5]:

```
# Dictionary comprehension
squares = {x: x*x for x in range(6)}
print(squares)

#Clcoding.com
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

7 Levels of Writing Python Dictionary

Level 6: Nested Dictionaries

Work with dictionaries within dictionaries to represent more complex data structures.

•[6]:

```
# Nested dictionary
people = {
    "person1": {
        "name": "Alice",
        "age": 30
    },
    "person2": {
        "name": "Bob",
        "age": 25
    }
}
print(people)

# Accessing nested dictionary values
print(people["person1"]["name"])

#Clcoding.com
```

```
{'person1': {'name': 'Alice', 'age': 30}, 'person2': {'name': 'Bob', 'age': 25}}
Alice
```

7 Levels of Writing Python Dictionary

Level 7: Advanced Dictionary Operations

Using advanced features like merging dictionaries, using defaultdict from collections, and performing operations with dict and zip

•[7]:

```
# Merging dictionaries (Python 3.9+)
dict1 = {"a": 1, "b": 2}
dict2 = {"b": 3, "c": 4}
merged_dict = dict1 | dict2
print(merged_dict)  # Output: {'a': 1, 'b': 3, 'c': 4}

# Using defaultdict
from collections import defaultdict

dd = defaultdict(int)
dd["key1"] += 1
print(dd)  # Output: defaultdict(<class 'int'>, {'key1': 1})

# Creating a dictionary from two lists using zip
keys = ["name", "age", "city"]
values = ["Charlie", 28, "Los Angeles"]
person = dict(zip(keys, values))
print(person)
#Clcoding.com
```

```
{'a': 1, 'b': 3, 'c': 4}
defaultdict(<class 'int'>, {'key1': 1})
{'name': 'Charlie', 'age': 28, 'city': 'Los Angeles'}
```

Image Processing in Python

6. Image Blurring (using a filter) with Pillow:

```python
from PIL import Image, ImageFilter

def blur_image(image_path, output_path, radius):
    image = Image.open(image_path)
    blurred_image = image.filter(ImageFilter.GaussianBlur(radius))
    blurred_image.save(output_path)

# Example usage:
blur_image('clcodingmr.jpg', 'blurred_output.jpg', 5)
Image.open('blurred_output.jpg')
```

Learn More

/Pythonclcoding /Pythoncoding /clcoding

The zip function in Python

THE ZIP FUNCTION IN PYTHON

Example 1: Basic Usage of zip

```python
# Basic usage of zip
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]

# Combining lists using zip
combined_data = zip(names, ages)

# Displaying the result
for name, age in combined_data:
    print(f"Name: {name}, Age: {age}")

    #clcoding.com
```

```
Name: Alice, Age: 25
Name: Bob, Age: 30
Name: Charlie, Age: 35
```

THE ZIP FUNCTION IN PYTHON

Example 2: Different Lengths in Input Iterables

```python
# Zip with different lengths in input iterables
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30]

# Using zip with different lengths will stop at the shortest iterable
combined_data = zip(names, ages)

# Displaying the result
for name, age in combined_data:
    print(f"Name: {name}, Age: {age}")

#clcoding.com
```

```
Name: Alice, Age: 25
Name: Bob, Age: 30
```

THE ZIP FUNCTION IN PYTHON

Example 3: Unzipping with zip

```python
# Unzipping with zip
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]

# Combining lists using zip
combined_data = zip(names, ages)

# Unzipping the result
unzipped_names, unzipped_ages = zip(*combined_data)

# Displaying the unzipped data
print("Unzipped Names:", unzipped_names)
print("Unzipped Ages:", unzipped_ages)

#clcoding.com
```

```
Unzipped Names: ('Alice', 'Bob', 'Charlie')
Unzipped Ages: (25, 30, 35)
```

THE ZIP FUNCTION IN PYTHON

Example 4: Using zip with Dictionaries

```
# Using zip with dictionaries
keys = ["name", "age", "city"]
values = ["Alice", 25, "New York"]

# Creating a dictionary using zip
person_dict = dict(zip(keys, values))

# Displaying the dictionary
print(person_dict)
#clcoding.com
```

```
{'name': 'Alice', 'age': 25, 'city': 'New York'}
```

THE ZIP FUNCTION IN PYTHON

Example 5: Transposing a Matrix with zip

```python
# Transposing a matrix using zip
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Using zip to transpose the matrix
transposed_matrix = list(zip(*matrix))

# Displaying the transposed matrix
for row in transposed_matrix:
    print(row)
#clcoding.com
```

```
(1, 4, 7)
(2, 5, 8)
(3, 6, 9)
```

THE ZIP FUNCTION IN PYTHON

Example 6: Using zip with enumerate

```python
# Using zip with enumerate
names = ["Alice", "Bob", "Charlie"]

# Combining with enumerate to get both index and value
indexed_names = list(zip(range(len(names)), names))

# Displaying the result
for index, name in indexed_names:
    print(f"Index: {index}, Name: {name}")
#clcoding.com
```

```
Index: 0, Name: Alice
Index: 1, Name: Bob
Index: 2, Name: Charlie
```

Python — Using reduce()

The reduce() function is a powerful tool from Python's functools module. It allows you to apply a function cumulatively to the items of a sequence, from left to right, reducing the sequence to a single value

Importing reduce

To use reduce(), you need to import it from the functools module:

[1]:

```
from functools import reduce
```

Example 1: Sum of Elements

Let's start with a simple example: calculating the sum of all elements in a list.

•[2]:

```
numbers = [1, 2, 3, 4, 5]
result = reduce(lambda x, y: x + y, numbers)
print(result)

#clcoding.com
```

```
15
```

PYTHON — USING REDUCE()

Example 2: Product of Elements

Similarly, you can calculate the product of all elements in a list.

•[3]:

```
numbers = [1, 2, 3, 4, 5]
result = reduce(lambda x, y: x * y, numbers)
print(result)

#clcoding.com
```

```
120
```

Example 3: Finding the Maximum Element

You can use reduce() to find the maximum element in a list.

•[4]:

```
numbers = [1, 2, 3, 4, 5]
result = reduce(lambda x, y: x if x > y else y, numbers)
print(result)

#clcoding.com
```

```
5
```

Example 4: Concatenating Strings

You can use reduce() to concatenate a list of strings into a single string.

•[5]:

```
words = ["Hello", "World", "from", "Python"]
result = reduce(lambda x, y: x + " " + y, words)
print(result)

#clcoding.com
```

```
Hello World from Python
```

Example 5: Using an Initial Value

You can provide an initial value to reduce(). This initial value is placed before the items of the sequence in the calculation and serves as a default when the sequence is empty.

•[6]:

```
numbers = [1, 2, 3, 4, 5]
result = reduce(lambda x, y: x + y, numbers, 10)
print(result)

#clcoding.com
```

```
25
```

PYTHON — USING REDUCE()

Example 6: Flattening a List of Lists

You can use reduce() to flatten a list of lists into a single list.

•[7]:

```
lists = [[1, 2, 3], [4, 5], [6, 7, 8]]
result = reduce(lambda x, y: x + y, lists)
print(result)

#clcoding.com
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Example 7: Finding the Greatest Common Divisor (GCD)

You can use reduce() with the gcd function from the math module to find the GCD of a list of numbers.

[8]:

```
import math

numbers = [48, 64, 256]
result = reduce(math.gcd, numbers)
print(result)
```

```
16
```

PYTHON — USING REDUCE()

Example 8: Combining Dictionaries

You can use reduce() to combine a list of dictionaries into a single dictionary.

•[9]:

```
dicts = [{'a': 1}, {'b': 2}, {'c': 3}]
result = reduce(lambda x, y: {**x, **y}, dicts)
print(result)

#clcoding.com
```

```
{'a': 1, 'b': 2, 'c': 3}
```

Example 9: Custom Function with reduce()

You can also use a custom function with reduce(). Here's an example that calculates the sum of squares of elements in a list.

•[10]:

```
def sum_of_squares(x, y):
    return x + y**2

numbers = [1, 2, 3, 4]
result = reduce(sum_of_squares, numbers, 0)
print(result)

#clcoding.com
```

```
30
```

10 Level of writing Python List

10 LEVEL OF WRITING PYTHON LIST

Level 1: Basic List Creation

```
# Creating a simple list
my_list = [1, 2, 3, 4, 5]
print(my_list)

#clcoding.com
```

```
[1, 2, 3, 4, 5]
```

Level 2: Accessing List Elements

```
# Accessing elements by index
my_list = [1, 2, 3, 4, 5]
first_element = my_list[0]
last_element = my_list[-1]
print( "First:", first_element, "Last:", last_element)

#clcoding.com
```

```
First: 1 Last: 5
```

10 LEVEL OF WRITING PYTHON LIST

Level 3: List Slicing

```python
# Slicing a list
my_list = [1, 2, 3, 4, 5]
sub_list = my_list[1:4]
print(sub_list)

#clcoding.com
```

```
[2, 3, 4]
```

Level 4: Adding Elements to a List

```python
# Appending and extending a list
my_list = [1, 2, 3, 4, 5]
my_list.append(6)
my_list.extend([7, 8])
print(my_list)

#clcoding.com
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

10 LEVEL OF WRITING PYTHON LIST

Level 5: Removing Elements from a List

```python
# Removing elements
my_list = [1, 2, 3, 4, 5]
my_list.remove(2)
popped_element = my_list.pop()
print(my_list)
print(popped_element)

#clcoding.com
```

```
[1, 3, 4]
5
```

Level 6: List Comprehensions

```python
# Using list comprehensions
my_list = [1, 2, 3, 4, 5]
squared_list = [x**2 for x in my_list]
print(squared_list)

#clcoding.com
```

```
[1, 4, 9, 16, 25]
```

10 LEVEL OF WRITING PYTHON LIST

Level 7: Nested Lists

```python
# Creating and accessing nested lists

nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
element = nested_list[1][2]
print(nested_list, element)

#clcoding.com
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]] 6
```

Level 8: Using List Functions and Methods

```python
# Using built-in functions and methods

my_list = [1, 2, 3, 4, 5]
length = len(my_list)
sorted_list = sorted(my_list)
my_list.sort(reverse=True)
print(length, sorted_list, my_list)

#clcoding.com
```

```
5 [1, 2, 3, 4, 5] [5, 4, 3, 2, 1]
```

10 LEVEL OF WRITING PYTHON LIST

Level 8: Using List Functions and Methods

```python
# Using built-in functions and methods

my_list = [1, 2, 3, 4, 5]
length = len(my_list)
sorted_list = sorted(my_list)
my_list.sort(reverse=True)
print(length, sorted_list, my_list)

#clcoding.com
```

```
5 [1, 2, 3, 4, 5] [5, 4, 3, 2, 1]
```

Level 9: List Iteration

```python
# Iterating over a list
for element in my_list:
    print(element)
```

```
5
4
3
2
1
```

10 LEVEL OF WRITING PYTHON LIST

Level 10: Advanced List Operations

•[10]:

```python
# Using advanced operations like map, filter, and reduce

my_list = [1, 2, 3, 4, 5]
from functools import reduce

# Map: applying a function to each element
doubled_list = list(map(lambda x: x * 2, my_list))
print(doubled_list)

# Filter: filtering elements based on a condition
even_list = list(filter(lambda x: x % 2 == 0, my_list))
print(even_list)

# Reduce: reducing the list to a single value
sum_of_elements = reduce(lambda x, y: x + y, my_list)
print(sum_of_elements)

#clcoding.com
```

```
[2, 4, 6, 8, 10]
[2, 4]
15
```

Function Interfaces in Python

FUNCTION INTERFACES IN PYTHON

1. Basic Function Definition

•[1]:

```
def add(a, b):

    """Add two numbers and
    return the result."""

    return a + b

result = add(3, 5)
print(result)

# clcoding.com
```

```
8
```

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

2. Keyword Parameters

[2]:

```
def greet(name, greeting="Hello"):
    """Greet someone with a provided
    greeting or a default greeting."""
    return f"{greeting}, {name}!"

message = greet("Clcoding")
print(message)

message = greet("Python", "Hi")
print(message)

# clcoding.com
```

```
Hello, Clcoding!
Hi, Python!
```

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

3. Arbitrary Positional Parameters

•[3]:

```
def sum_all(*args):

    """Sum all given arguments."""

    return sum(args)

total = sum_all(1, 2, 3, 4, 5)
print(total)

# clcoding.com
```

```
15
```

FUNCTION INTERFACES IN PYTHON

4. Arbitrary Keyword Parameters

•[4]:

```
def print_info(**kwargs):

    """Print key-value pairs
    from keyword arguments."""

    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="Clcoding", age=30, city="Earth")

# clcoding.com
```

```
name: Clcoding
age: 30
city: Earth
```

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

5. Function Annotations

•[5]:

```
def multiply(x: int, y: int) -> int:

    """Multiply two integers
    and return the result."""

    return x * y

result = multiply(3, 4)
print(result)

# clcoding.com
```

```
12
```

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

6. Docstrings

[6]:

```
def divide(x, y):
    """
    Divide x by y and return the result.

    :param x: numerator
    :param y: denominator
    :return: division result
    """
    return x / y

result = divide(10, 2)
print(result)

# clcoding.com
```

```
5.0
```


FUNCTION INTERFACES IN PYTHON

7. First-Class Functions

[7]:

```
def square(x):
    """Return the square of x."""
    return x * x

def apply_function(func, value):
    """Apply a function to a value
    and return the result."""
    return func(value)

result = apply_function(square, 4)
print(result)

# clcoding.com
```

16

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

8. Lambda Functions

[8]:

```
square = lambda x: x * x
print(square(5))

# clcoding.com
```

```
25
```

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

9. Decorators

[9]:

```
def debug(func):
    """A decorator that prints the function
    signature and return value."""
    def wrapper(*args, **kwargs):
        print(f"Calling {func.__name__} with {args} and {kwargs}")
        result = func(*args, **kwargs)
        print(f"{func.__name__} returned {result}")
        return result
    return wrapper

@debug
def add(a, b):
    """Add two numbers."""
    return a + b

add(3, 4)

# clcoding.com
```

```
Calling add with (3, 4) and {}
add returned 7
[9]:
7
```

/clcoding

/Pythoncoding

/Pythonclcoding

FUNCTION INTERFACES IN PYTHON

10. Combining Concepts

[11]:

```python
def trace(func):

    """A decorator that traces function calls."""

    def wrapper(*args, **kwargs):
        print(f"TRACE: calling {func.__name__}() with {args}, {kwargs}")
        result = func(*args, **kwargs)
        print(f"TRACE: {func.__name__}() returned {result}")
        return result
    return wrapper

@trace
def compute_area(length: float, width: float) -> float:
    """Compute the area of a rectangle."""
    return length * width

area = compute_area(3.0, 4.5)

# clcoding.com
```

```
TRACE: calling compute_area() with (3.0, 4.5), {}
TRACE: compute_area() returned 13.5
```

5 Practical Examples to Master Python Decorators

1. Basic Decorator

A simple example to understand the basic structure and functionality of decorators.

•[1]:

```
def simple_decorator(func):
    def wrapper():
        print("Before the function runs")
        func()
        print("After the function runs")
    return wrapper

@simple_decorator
def say_hello():
    print("Hello!")

say_hello()

#clcoding.com
```

```
Before the function runs
Hello!
After the function runs
```

2. Decorator with Arguments

A decorator that accepts arguments to customize its behavior.

•[2]:

```
def repeat(n):
    def decorator(func):
        def wrapper(*args, **kwargs):
            for _ in range(n):
                func(*args, **kwargs)
        return wrapper
    return decorator

@repeat(3)
def greet(name):
    print(f"Hello, {name}!")
greet("Clcoding")

#clcoding.com
```

```
Hello, Clcoding!
Hello, Clcoding!
Hello, Clcoding!
```

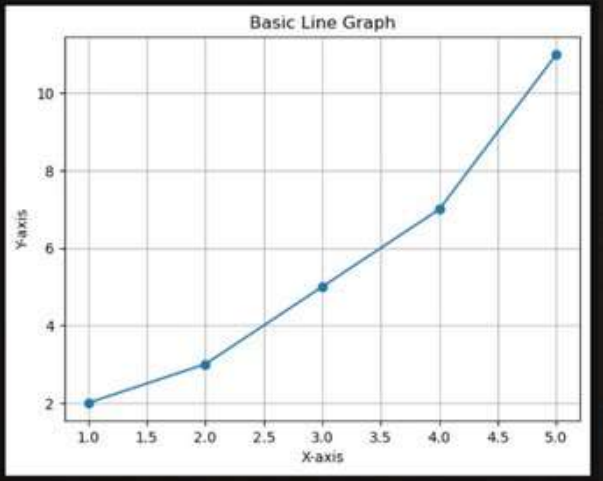
Different Line graph plot using Python

LINE GRAPHS PLOT USING PYTHON

1. Basic Line Graph using Matplotlib

•[1]:

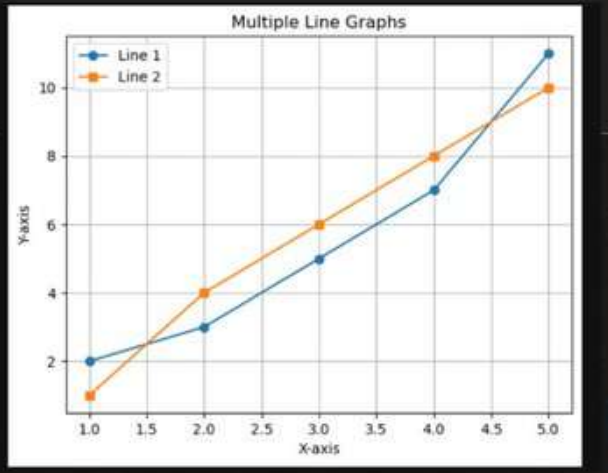
```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Plotting the line graph
plt.plot(x, y, marker='o')
plt.title('Basic Line Graph')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
#clcoding.com
```



2. Multiple Line Graphs using Matplotlib

•[2]:

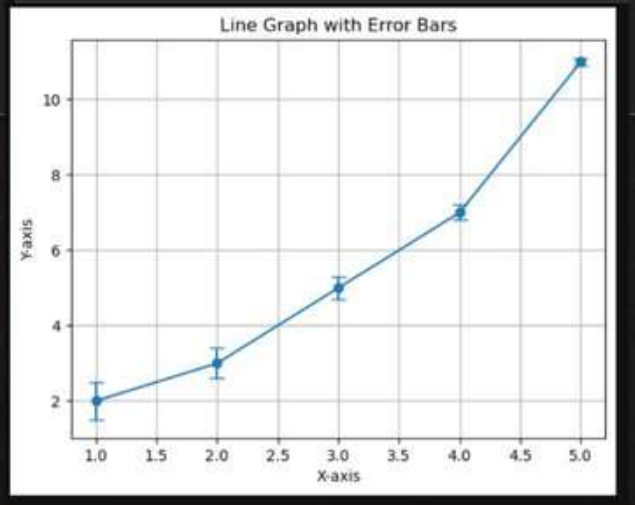
```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y1 = [2, 3, 5, 7, 11]
y2 = [1, 4, 6, 8, 10]

# Plotting multiple line graphs
plt.plot(x, y1, label='Line 1', marker='o')
plt.plot(x, y2, label='Line 2', marker='s')
plt.title('Multiple Line Graphs')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.show()
#clcoding.com
```



3. Line Graph with Error Bars using Matplotlib

•[3]:

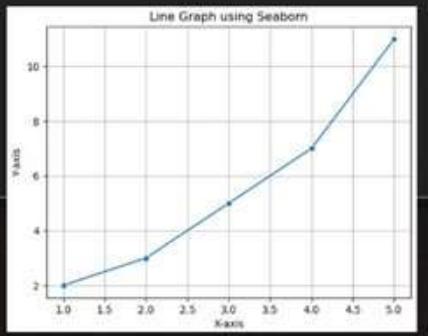
```
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 3, 5, 7, 11])
yerr = np.array([0.5, 0.4, 0.3, 0.2, 0.1])

# Plotting the line graph with error bars
plt.errorbar(x, y, yerr=yerr, fmt='-o', capsize=5)
plt.title('Line Graph with Error Bars')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
#clcoding.com
```



4. Line Graph using Seaborn

•[4]:

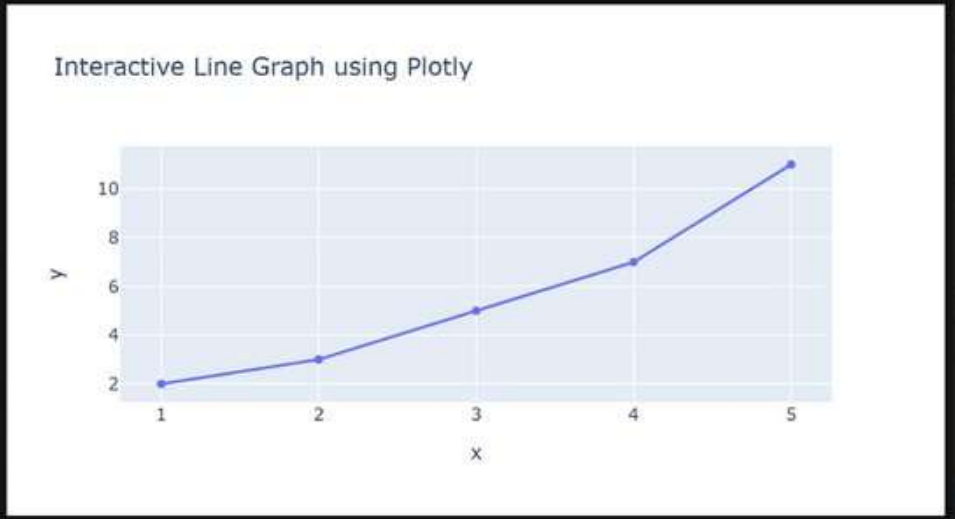
```
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = {
    'x': [1, 2, 3, 4, 5],
    'y': [2, 3, 5, 7, 11]
}

# Creating a Seaborn line plot
sns.lineplot(x='x', y='y', data=data, marker='o')
plt.title('Line Graph using Seaborn')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.grid(True)
plt.show()
#clcoding.com
```



5. Interactive Line Graph using Plotly

[6]:

```
import plotly.express as px

# Sample data
data = {
    'x': [1, 2, 3, 4, 5],
    'y': [2, 3, 5, 7, 11]
}

# Creating an interactive line plot
fig = px.line(data, x='x', y='y',
              title='Interactive Line Graph using Plotly', markers=True)
fig.show()

#clcoding.com
```



Practical Uses of continue and break Statements

Example 1: Skipping Even Numbers

Using continue to skip even numbers in a loop.

•[1]:

```
for i in range(1, 11):
    if i % 2 == 0:
        continue
    print(i)

#clcoding.com
```

```
1
3
5
7
9
```

Example 2: Stopping at a Specific Number

Using break to stop the loop when encountering the number 5.

•[2]:

```
for i in range(1, 11):
    if i == 5:
        break
    print(i)

#clcoding.com
```

```
1
2
3
4
```

Example 3: Skipping a Specific Number and Stopping at Another

Combining continue and break to skip the number 3 and stop at 7.

•[3]:

```
for i in range(1, 11):
    if i == 3:
        continue  # Skip the number 3
    if i == 7:
        break  # Stop the loop when i is 7
    print(i)

#clcoding.com
```

```
1
2
4
5
6
```

Example 4: Finding the First Negative Number

Using break to find and print the first negative number in a list.

•[4]:

```
numbers = [10, 20, -5, 30, -10]

for num in numbers:
    if num < 0:
        print(f"First negative number is: {num}")
        break

#clcoding.com
```

```
First negative number is: -5
```

Example 5: Summing Non-Negative Numbers

Using continue to skip negative numbers and sum the non-negative ones.

•[5]:

```
numbers = [10, -5, 20, -10, 30]
total = 0

for num in numbers:
    if num < 0:
        continue  # Skip negative numbers
    total += num

print(f"Total sum of non-negative numbers is: {total}")

#clcoding.com
```

```
Total sum of non-negative numbers is: 60
```

Why you should use PEP 8 guidelines ?

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

1. Consistency

Reason: Consistent code is easier to read and understand. PEP 8 provides a standard style guide that promotes consistency across different projects and among different developers.

Without PEP 8:

[2]:

```
def my_function():print("Hello"); print("World")
my_function()
```

```
Hello
World
```

With PEP 8:

[3]:

```
def my_function():
    print("Hello")
    print("World")

my_function()
```

```
Hello
World
```

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

2. Readability

Reason: Readable code is easier to understand and debug. PEP 8 encourages practices that make your code more readable.

Without PEP 8:

[4]:

```
def add(a,b):return a+b
```

With PEP 8:

[5]:

```
def add(a, b):
    return a + b
```

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

3. Collaboration

Reason: When everyone on a team follows the same style guide, it's easier for team members to read and understand each other's code, making collaboration smoother.

Without PEP 8:

[6]:

```
def fetchData():
  # fetch data from API
  data = {"name":"John","age":30}
  return data
```

With PEP 8:

[7]:

```
def fetch_data():
    # Fetch data from API
    data = {"name": "John", "age": 30}
    return data
```

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

4. Maintainability

Reason: Code that adheres to a standard style is easier to maintain and update. PEP 8's guidelines help you write code that is more maintainable in the long run.

Without PEP 8:

[8]:

```
def processData(data):
  result = data["name"].upper() + " is " + str(data["age"]) + " years old"
  return result
```

With PEP 8:

[18]:

```
def process_data(data):
    result = f"{data['name'].upper()} is {data['age']} years old"
    return result

#clcoding.com
```

/clcoding

/Pythoncoding

/Pythonclcoding

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

5. Professionalism

Reason: Following PEP 8 shows that you care about writing high-quality code. It demonstrates professionalism and attention to detail, which are valuable traits in any developer.

Without PEP 8:

[10]:

```
def square(x):return x*x
```

With PEP 8:

[11]:

```
def square(x):
    return x * x
```

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

6. Community Standard

Reason: PEP 8 is the de facto standard for Python code. Following it ensures your code aligns with what other Python developers expect, making it easier for others to read and contribute to your projects.

Without PEP 8:

[12]:

```
class Person:
  def __init__(self,name,age):
    self.name=name
    self.age=age
  def getDetails(self):
    return self.name + " is " + str(self.age)
```

With PEP 8:

[13]:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def get_details(self):
        return f"{self.name} is {self.age}"
```

WHY YOU SHOULD USE PEP 8 GUIDELINES ?

7. Error Prevention

Reason: PEP 8 includes guidelines that help prevent common errors, such as mixing tabs and spaces for indentation.

Without PEP 8:

[16]:

```
def calculate_sum(a, b):
  return a + b
    print("Sum calculated")
```

```
  Cell In[16], line 3
    print("Sum calculated")
    ^
IndentationError: unexpected indent
```

With PEP 8:

[17]:

```
def calculate_sum(a, b):
    return a + b

print("Sum calculated")
```

```
Sum calculated
```

5 Hidden Gems in Pandas You Should Start Using Today

5 Hidden Gems in Pandas You Should Start Using Today

1. query() Method for Filtering Data

What it is: The query() method allows you to filter data in a DataFrame using a more readable and concise string-based expression.

Why it's useful: It avoids the verbosity of standard indexing and makes the code more readable, especially for complex conditions.

[18]:

```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, 3, 4],
                   'B': [10, 20, 30, 40]})
result = df.query('A > 2 & B < 40')
print(result)

#clcoding.com
```

```
   A   B
2  3  30
```

2. eval() Method for Efficient Calculations

What it is: The eval() method evaluates a string expression within the context of a DataFrame, allowing for efficient computation.

Why it's useful: It can speed up operations involving arithmetic or logical operations on DataFrame columns, especially with large datasets.

[5]:

```
df['C'] = df.eval('A + B')
print(df)

#clcoding.com
```

```
   A   B   C
0  1  10  11
1  2  20  22
2  3  30  33
3  4  40  44
```


3. at and iat for Fast Access

What it is: at and iat are optimized methods for accessing scalar values in a DataFrame.

Why it's useful: These methods are much faster than using .loc[] or .iloc[] for individual cell access, making them ideal for performance-critical code.

[7]:

```
value = df.at[2, 'B']
print(value)

#clcoding.com
```

```
30
```

4. pipe() Method for Method Chaining

What it is: The pipe() method allows you to apply a function or sequence of functions to a DataFrame within a method chain.

Why it's useful: It improves code readability by keeping the DataFrame operations within a single fluent chain.

[9]:

```
def add_constant(df, value):
    return df + value

df = df.pipe(add_constant, 10)
print(df)

#clcoding.com
```

```
    A   B   C
0  11  20  21
1  12  30  32
2  13  40  43
3  14  50  54
```


5. explode() for Expanding Lists in Cells

What it is: The explode() method expands a list-like column into separate rows.

Why it's useful: This is particularly useful when working with data that has embedded lists within cells and you need to analyze or visualize each item individually.

[20]:

```
df = pd.DataFrame({'A': [1, 2],
                   'B': [[10, 20, 30], [40, 50]]})
df_exploded = df.explode('B')
print(df_exploded)

#clcoding.com
```

```
   A   B
0  1  10
0  1  20
0  1  30
1  2  40
1  2  50
```

8 Levels of Writing Python Functions

8 LEVELS OF WRITING PYTHON FUNCTIONS

Level 1: Basic Function Definition

Goal: Learn how to define simple functions with def.

[3]:

```
def greet():
    return "Hello, World!"
```

Concepts: Function definition, return statement, and basic usage.

Level 2: Function Arguments

Goal: Understand how to pass data to functions using arguments.

[7]:

```
def greet(name):
    return f"Hello, {name}!"
```

Concepts: Positional arguments, basic string formatting.

8 LEVELS OF WRITING PYTHON FUNCTIONS

Level 1: Basic Function Definition

Goal: Learn how to define simple functions with def.

[3]:

```
def greet():
    return "Hello, World!"
```

Concepts: Function definition, return statement, and basic usage.

Level 2: Function Arguments

Goal: Understand how to pass data to functions using arguments.

[7]:

```
def greet(name):
    return f"Hello, {name}!"
```

Concepts: Positional arguments, basic string formatting.

Level 3: Default Arguments

Goal: Use default argument values to make functions more flexible.

[11]:

```
def greet(name="World"):
    return f"Hello, {name}!"
```

Concepts: Default values, handling optional parameters.

Level 4: Variable-Length Arguments

Goal: Handle an arbitrary number of arguments using *args and **kwargs.

[15]:

```
def greet(*names):
    return "Hello, " + ", ".join(names) + "!"
```

Concepts: *args for positional arguments, **kwargs for keyword arguments.

Level 3: Default Arguments

Goal: Use default argument values to make functions more flexible.

[11]:

```
def greet(name="World"):
    return f"Hello, {name}!"
```

Concepts: Default values, handling optional parameters.

Level 4: Variable-Length Arguments

Goal: Handle an arbitrary number of arguments using *args and **kwargs.

[15]:

```
def greet(*names):
    return "Hello, " + ", ".join(names) + "!"
```

Concepts: *args for positional arguments, **kwargs for keyword arguments.

8 LEVELS OF WRITING PYTHON FUNCTIONS

Level 5: Return Multiple Values

Goal: Return multiple values from a function using tuples.

```python
def divide(a, b):
    quotient = a // b
    remainder = a % b
    return quotient, remainder
```

Concepts: Tuple unpacking, returning multiple values.

Level 6: First-Class Functions

Goal: Treat functions as first-class citizens by passing them as arguments or returning them.

```python
def apply_function(func, value):
    return func(value)

def square(x):
    return x * x

result = apply_function(square, 5)
```

Concepts: Functions as arguments, higher-order functions.

8 LEVELS OF WRITING PYTHON FUNCTIONS

Level 6: First-Class Functions

Goal: Treat functions as first-class citizens by passing them as arguments or returning them.

```python
def apply_function(func, value):
    return func(value)

def square(x):
    return x * x

result = apply_function(square, 5)
```

Concepts: Functions as arguments, higher-order functions.

Level 7: Lambda Functions

Goal: Use lambda expressions for short, unnamed functions.

```python
def apply_function(func, value):
    return func(value)

result = apply_function(lambda x: x * x, 5)
```

Concepts: Lambda expressions, anonymous functions.

8 LEVELS OF WRITING PYTHON FUNCTIONS

Level 6: First-Class Functions

Goal: Treat functions as first-class citizens by passing them as arguments or returning them.

```python
def apply_function(func, value):
    return func(value)

def square(x):
    return x * x

result = apply_function(square, 5)
```

Concepts: Functions as arguments, higher-order functions.

Level 7: Lambda Functions

Goal: Use lambda expressions for short, unnamed functions.

```python
def apply_function(func, value):
    return func(value)

result = apply_function(lambda x: x * x, 5)
```

Concepts: Lambda expressions, anonymous functions.

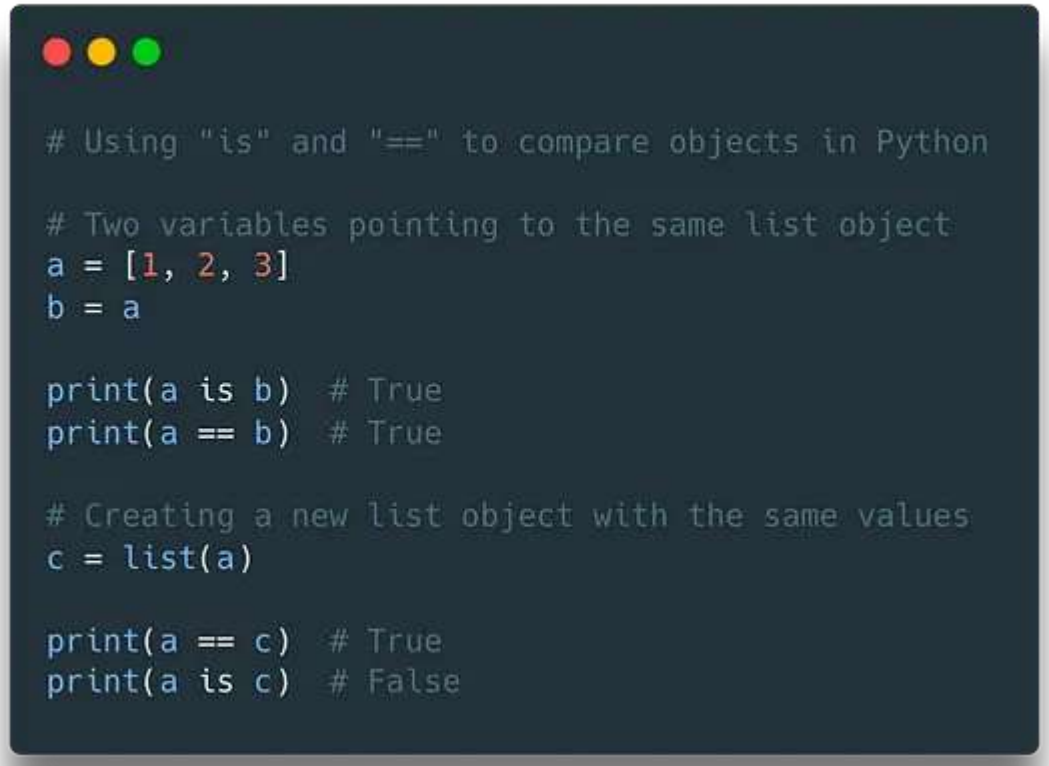
Level 8: Decorators

Goal: Modify the behavior of functions using decorators.

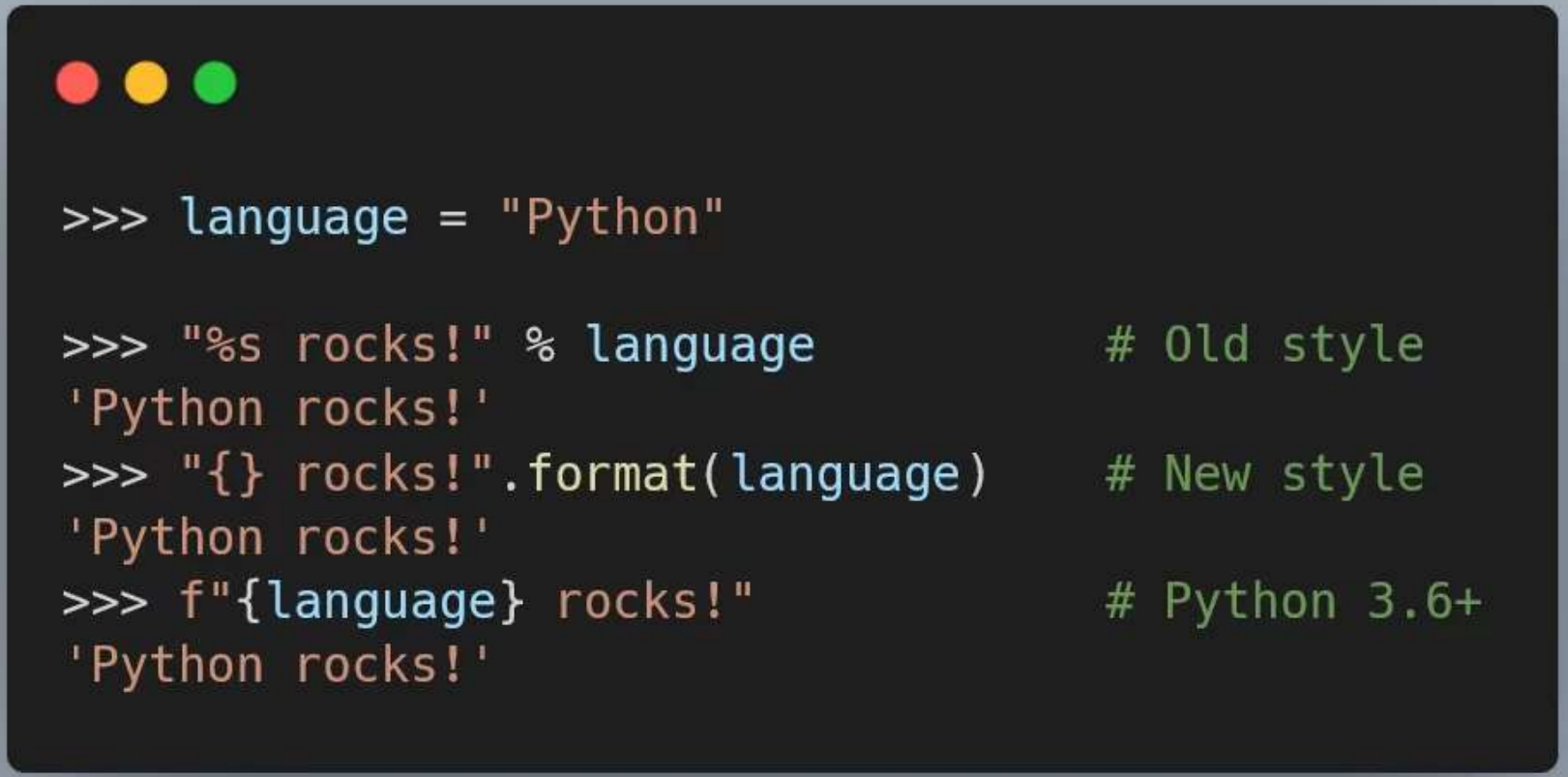
```python
def decorator(func):
    def wrapper(*args, **kwargs):
        print("Before the function")
        result = func(*args, **kwargs)
        print("After the function")
        return result
    return wrapper

@decorator
def greet(name):
    return f"Hello, {name}!"

greet("World")
```

```
Before the function
After the function
'Hello, World!'
```

Concepts: Decorators, wrapping functions, modifying behavior.



```python
# Using "is" and "==" to compare objects in Python

# Two variables pointing to the same list object
a = [1, 2, 3]
b = a

print(a is b)  # True
print(a == b)  # True

# Creating a new list object with the same values
c = list(a)

print(a == c)  # True
print(a is c)  # False
```


>>> language = "Python"
>>> "%s rocks!" % language # Old style
'Python rocks!'
>>> "{} rocks!".format(language) # New style
'Python rocks!'
>>> f"{language} rocks!" # Python 3.6+
'Python rocks!'

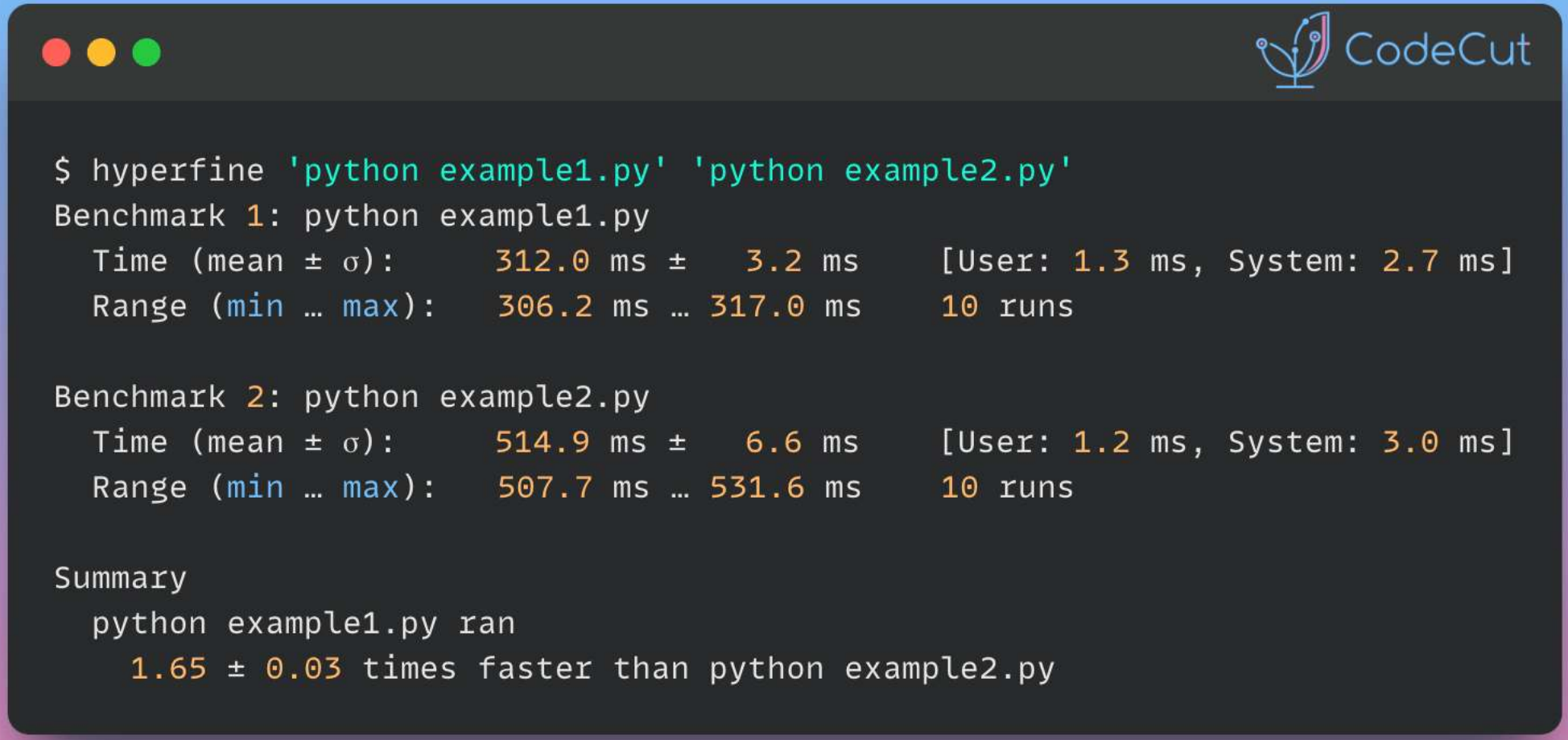
How to Load SQL Tables into Pandas DataFrames

CodeCut

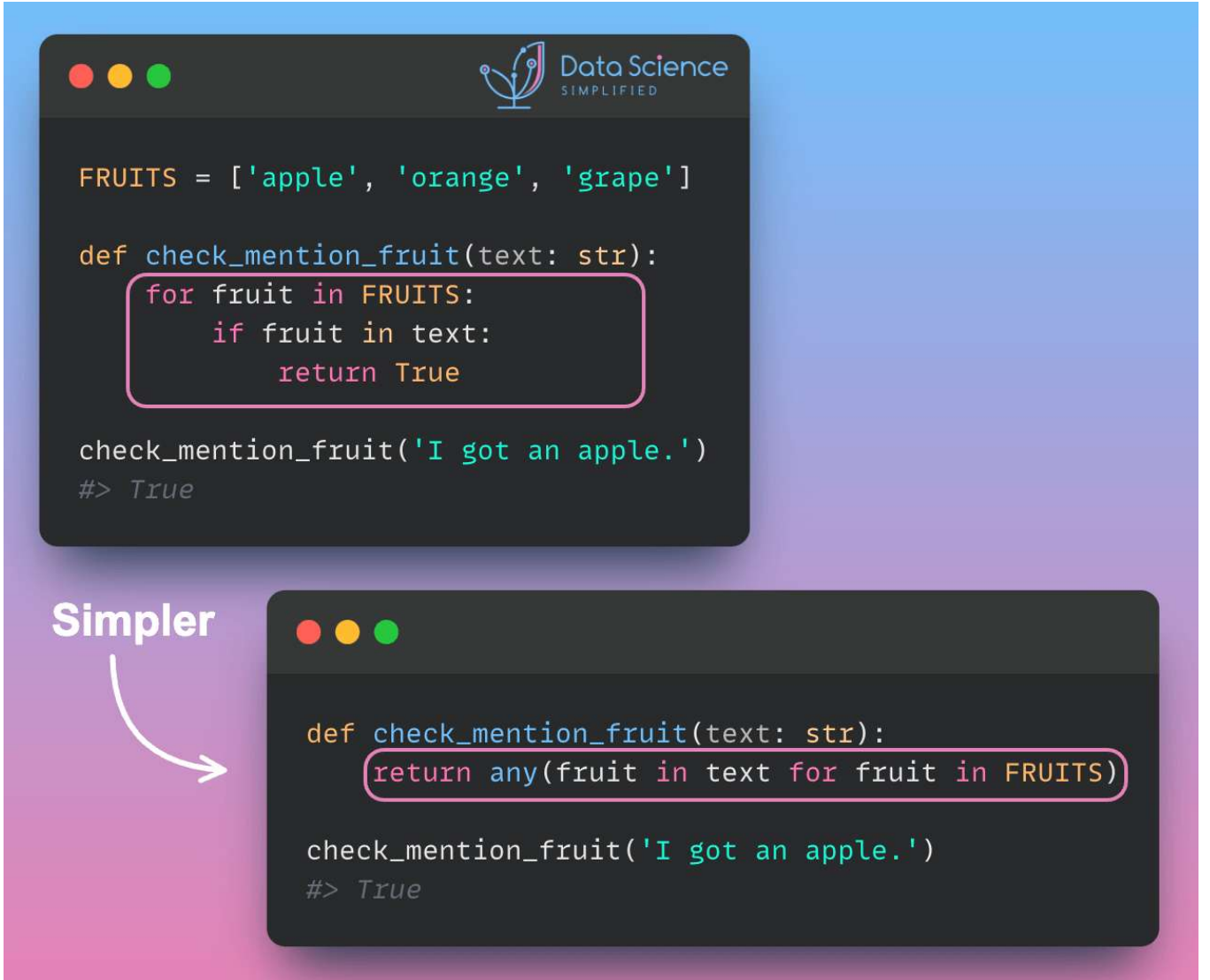
```python
import pandas as pd
import sqlalchemy

# Create a SQLAlchemy engine
engine = sqlalchemy.create_engine(
    "postgresql://username:password@host:port/database_name"
)

# Read a SQL table into a DataFrame
df = pd.read_sql("SELECT * FROM table_name", engine)
```

TextBlob: Processing Text in One Line of Code

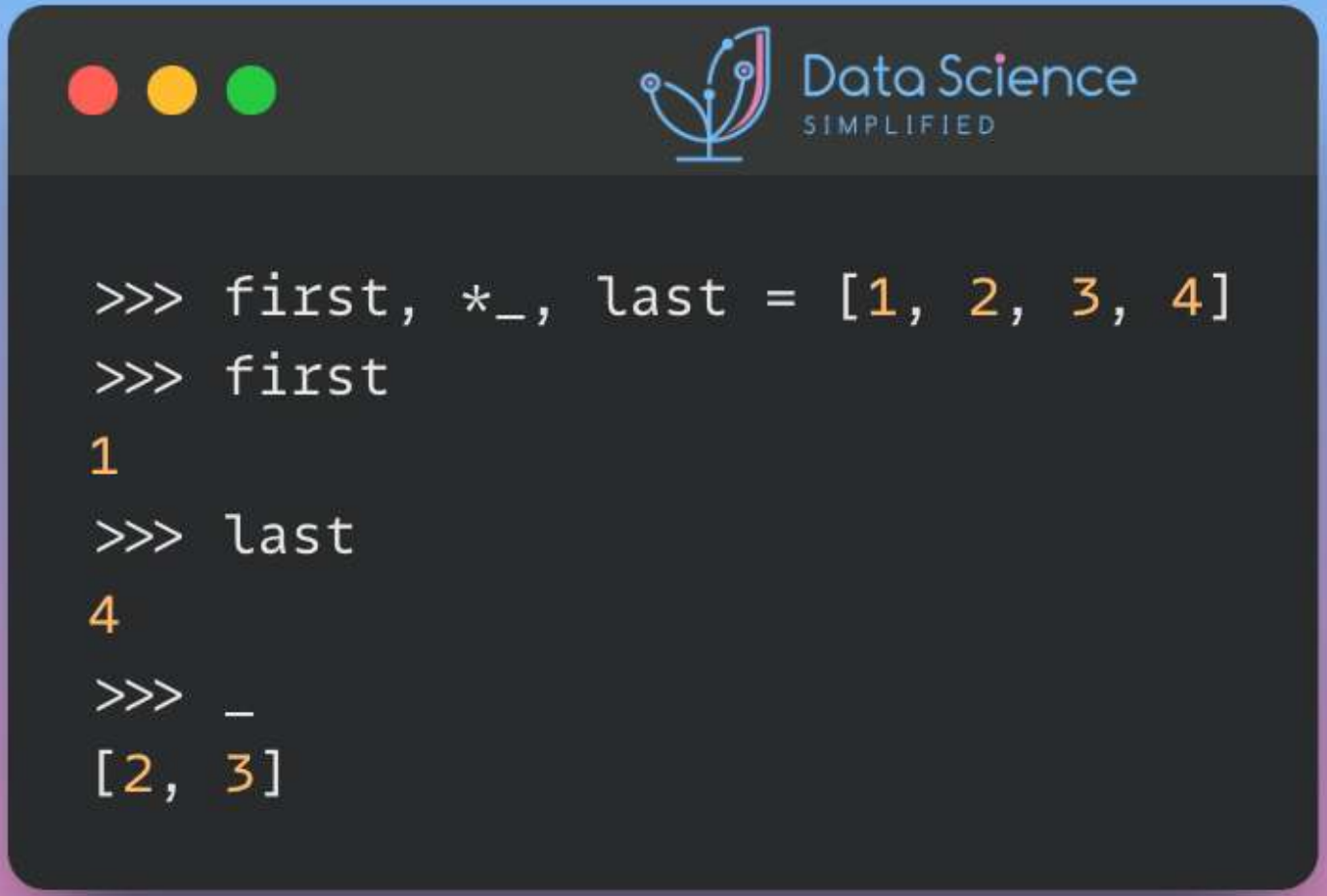
Hyperfine: Compare the Speed of Two Commands



Simplify List Condition Evaluation with any and List Comprehensions

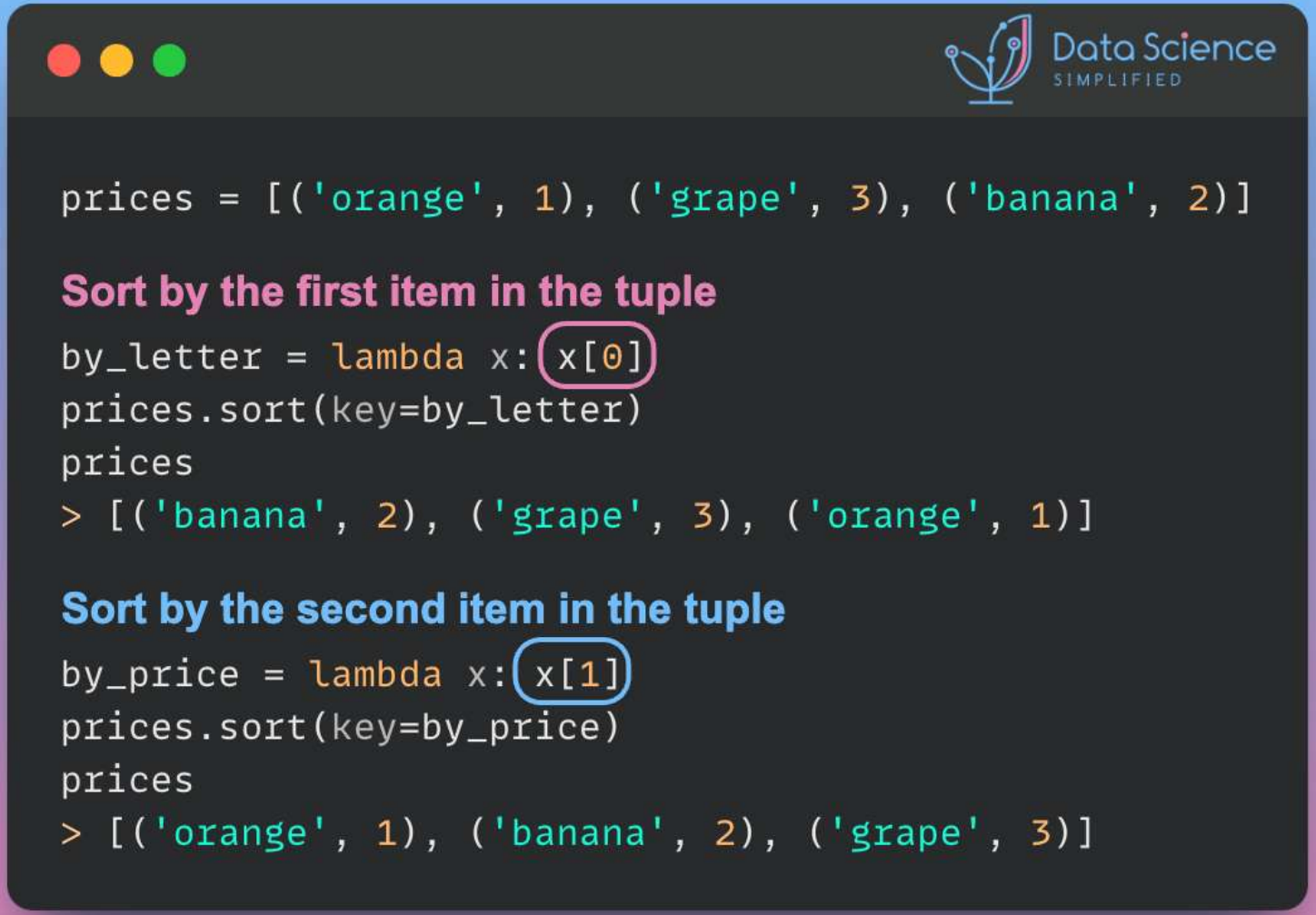


Ignore Unneeded Values During Python Iterable Unpacking

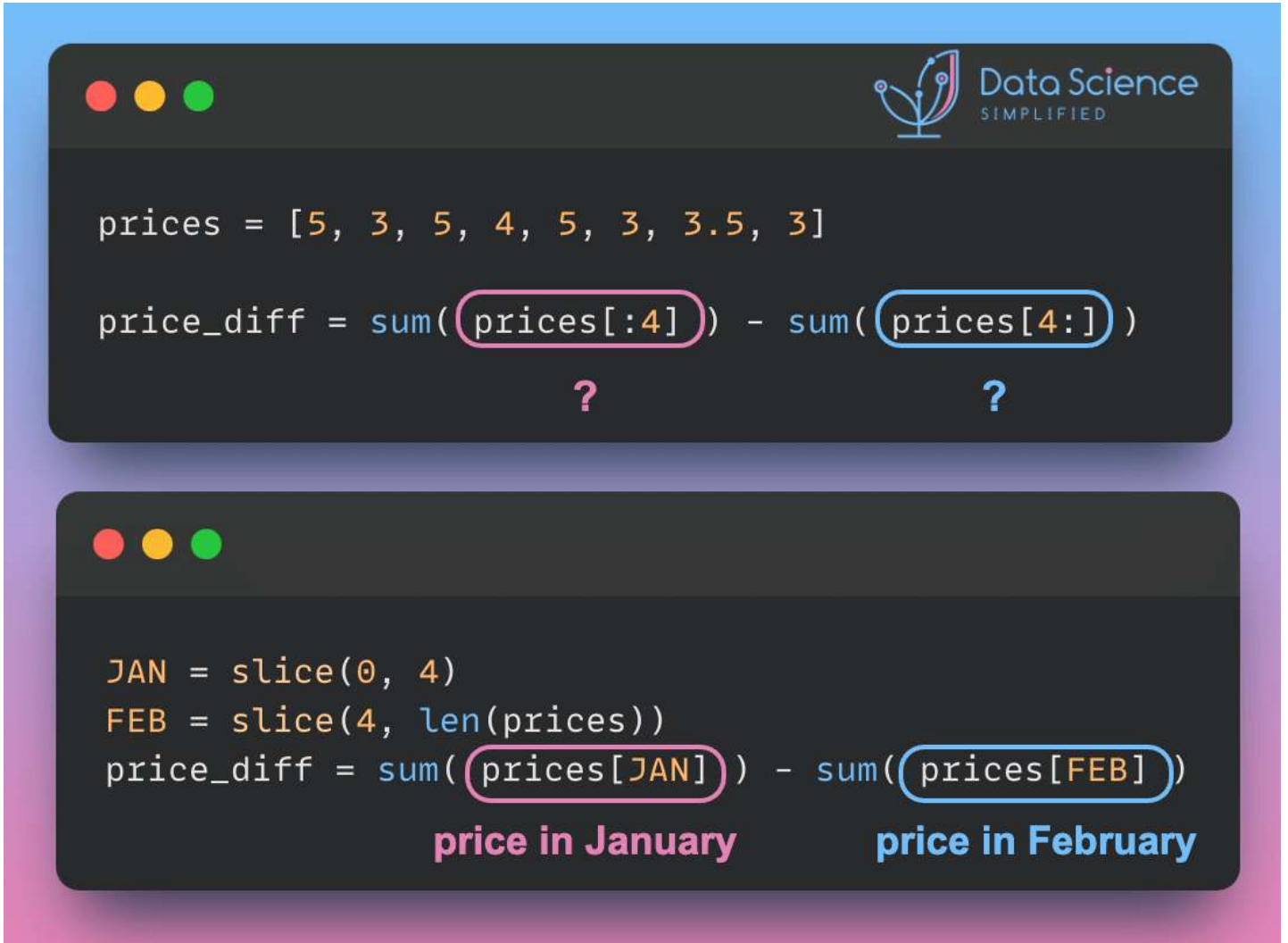


Convert number to words

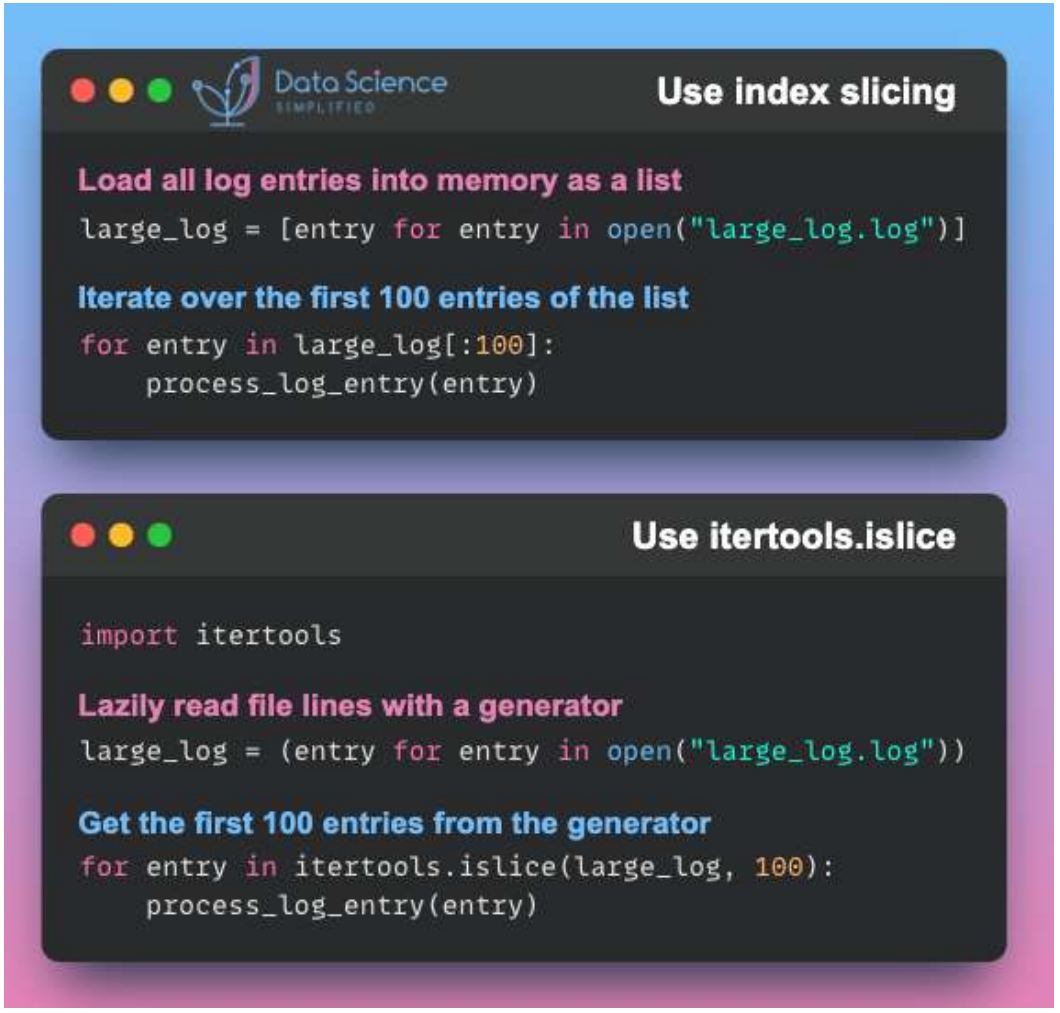
How to Sort a List of Tuples in Python



slice: Make your Indices more Readable by Naming your Slice



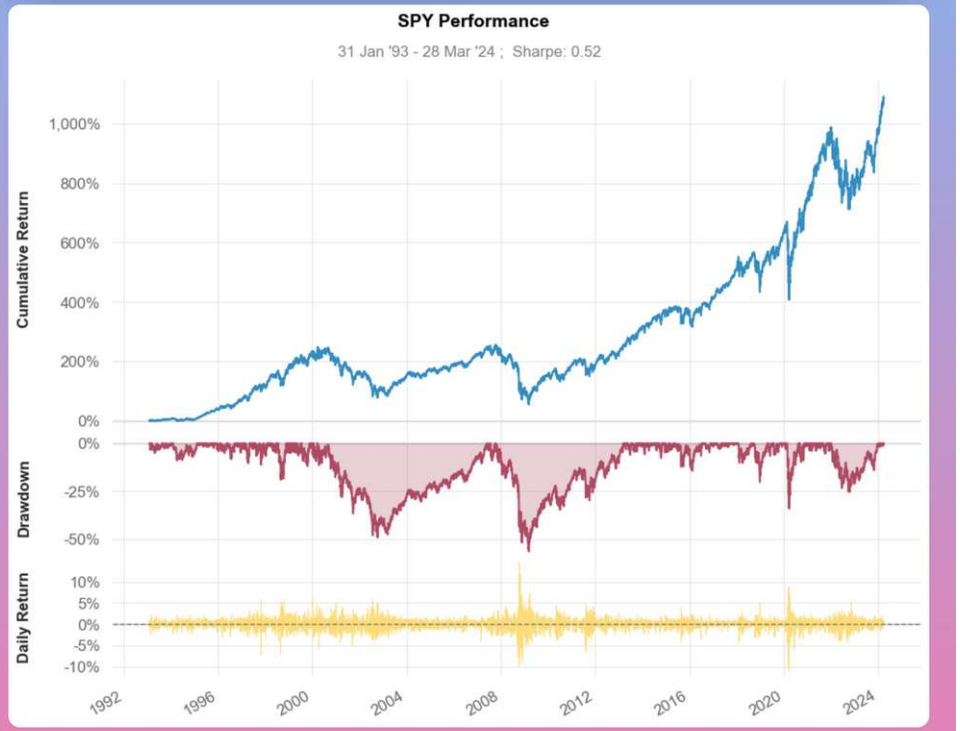
itertools.islice: Efficient Data Processing for Large Data Streams



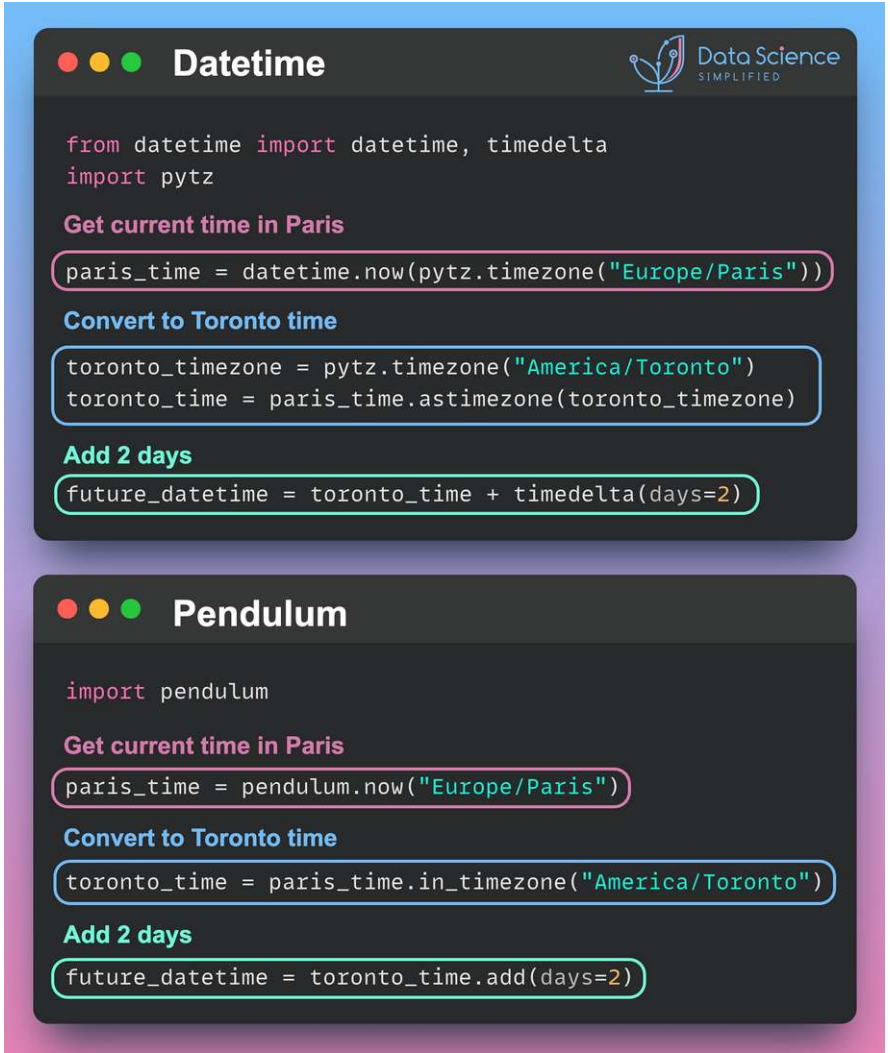
Simplify Conditional Execution with Match Statements

QuantStats: Simplify Stock Performance Analysis in Python

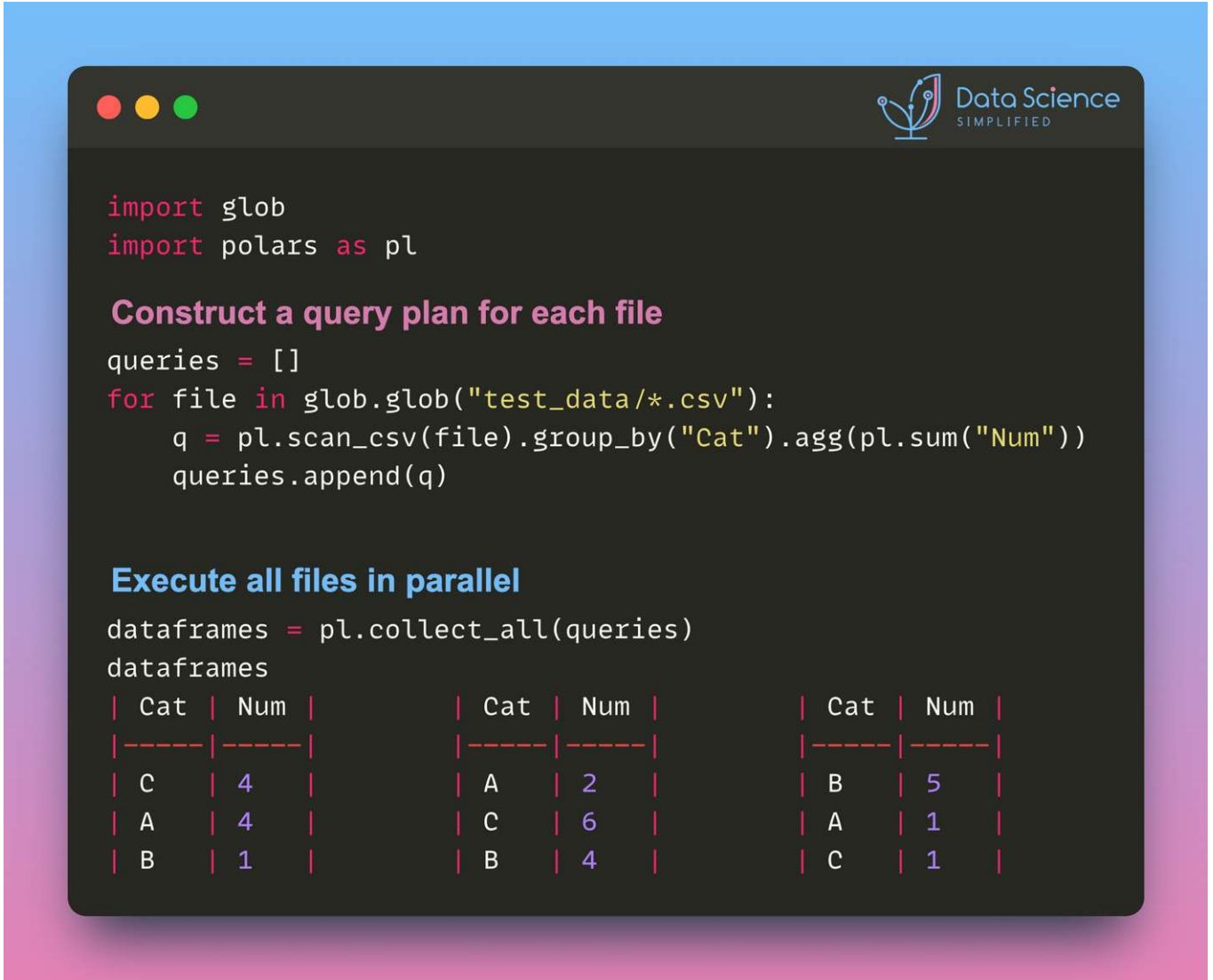
```python
import quantstats as qs

stock = qs.utils.download_returns('SPY')
qs.plots.snapshot(stock, title='SPY Performance', show=True)
```



Pendulum: Python Datetimes Made Easy



Parallel Execution of Multiple Files with Polars

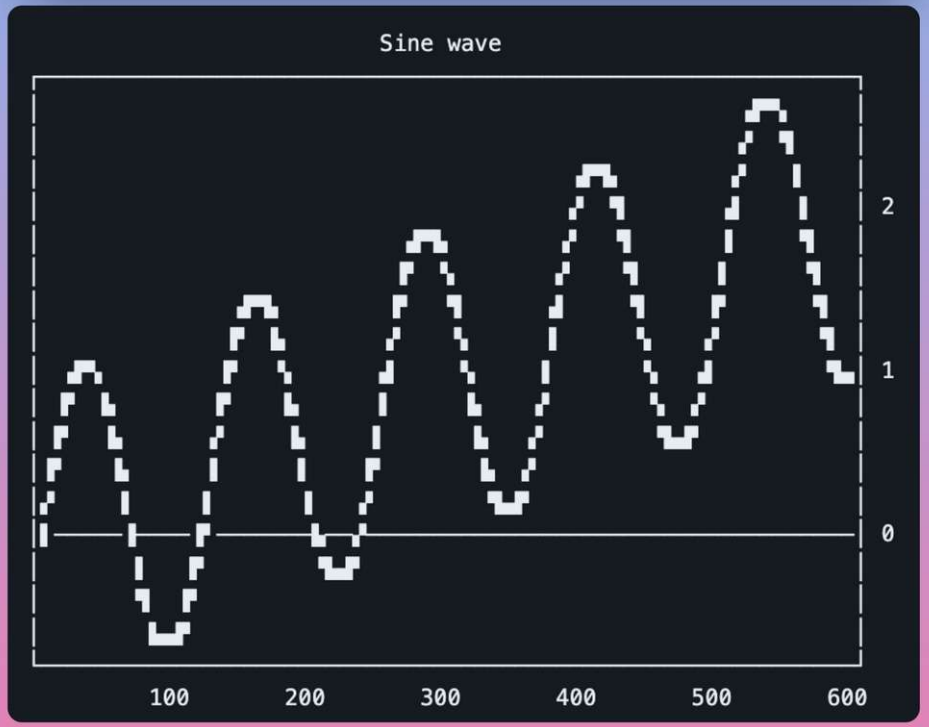


Uniplot: Terminal-Based Plotting for Enhanced Data Science Pipelines

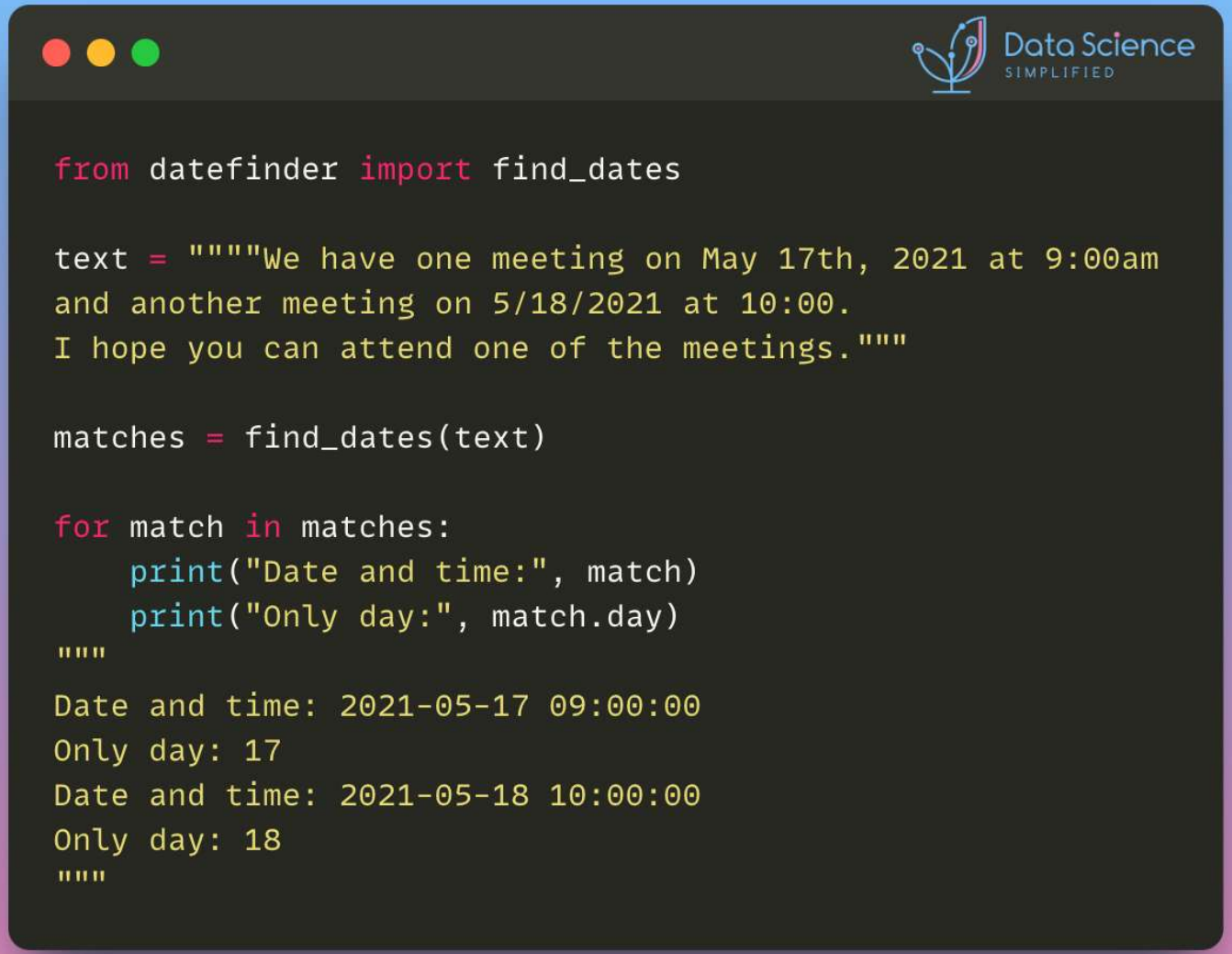
terminal.sh

Data Science SIMPLIFIED

```
>>> import math
>>> x = [math.sin(i/20)+i/300 for i in range(600)]
>>> from uniplot import plot
>>> plot(x, title="Sine wave")
```

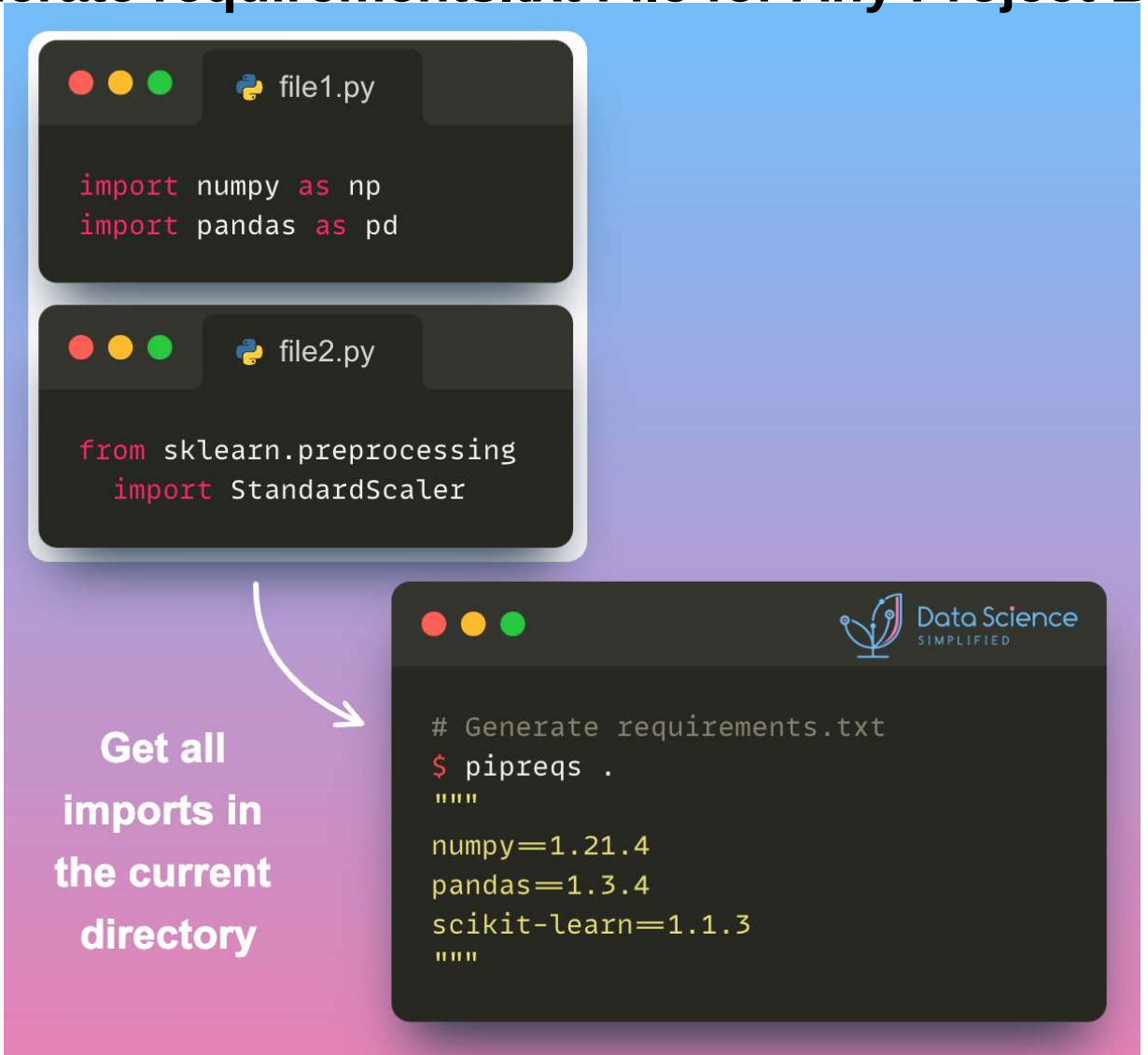


datefinder: Automatically Find Dates and Time in a Python String

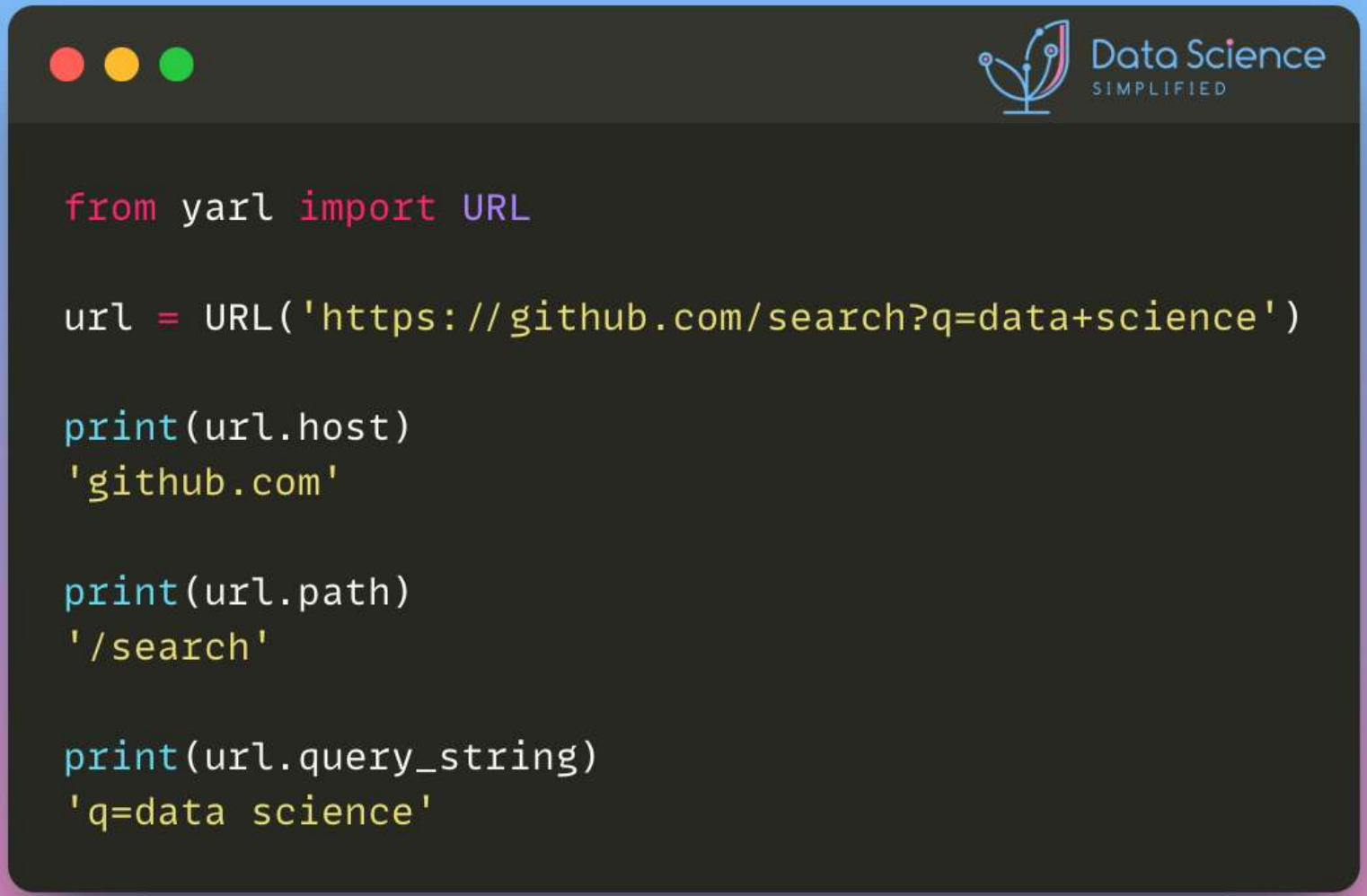


pathlib: A Simpler Way to Handle Python Files

pipreqs: Generate requirements.txt File for Any Project Based on Imports



yarl: Create and Extract Elements From a URL Using Python

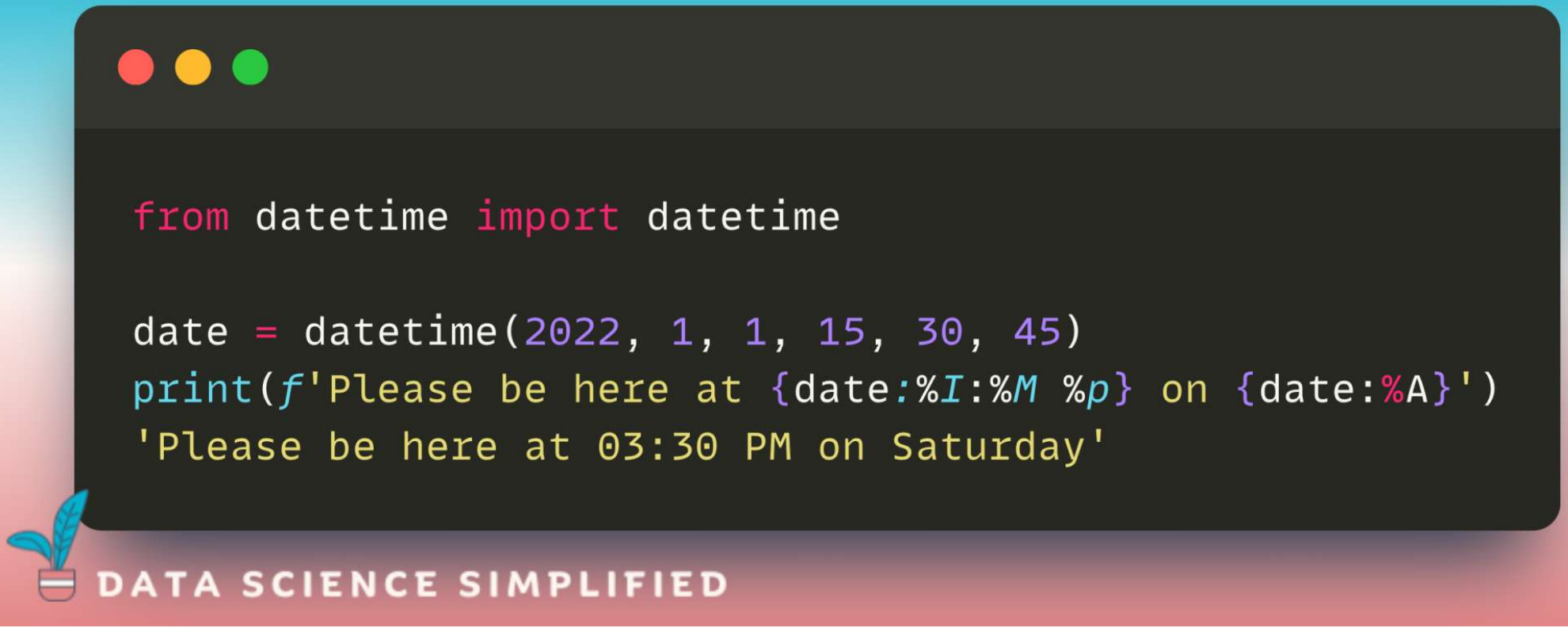


Read HTML Tables Using Pandas

```
import pandas as pd

df = pd.read_html("https://en.wikipedia.org/wiki/Poverty")
df[1]
```

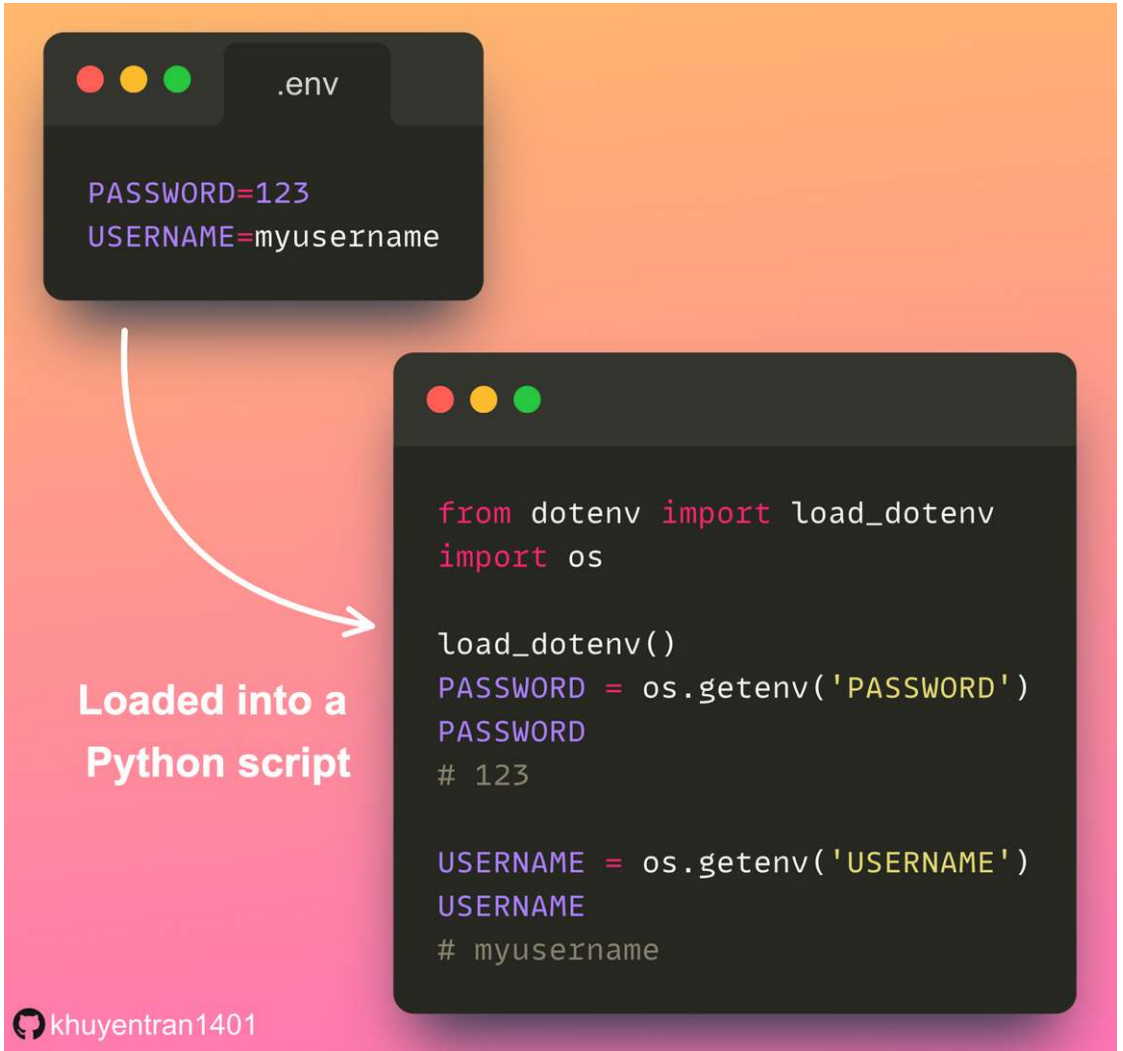
Region	$1 per day			$1.25 per day[94]		$1.90 per day[95]				
Region	1990	2002	2004	1981	2008	1981	1990	2000	2010	2015	2018
East Asia and Pacific	15.4%	12.3%	9.1%	77.2%	14.3%	80.2%	60.9%	34.8%	10.8%	2.1%	1.2%
Europe and Central Asia	3.6%	1.3%	1.0%	1.9%	0.5%	—	—	7.3%	2.4%	1.5%	1.1%
Latin America and the Caribbean	9.6%	9.1%	8.6%	11.9%	6.5%	13.7%	15.5%	12.7%	6%	3.7%	3.7%
Middle East and North Africa	2.1%	1.7%	1.5%	9.6%	2.7%	—	6.5%	3.5%	2%	4.3%	7%
South Asia	35.0%	33.4%	30.8%	61.1%	36%	58%	49.1%	—	26%	—	—
Sub-Saharan Africa	46.1%	42.6%	41.1%	51.5%	47.5%	—	54.9%	58.4%	46.6%	42.3%	40.4%
World	—	—	—	52.2%	22.4%	42.7%	36.2%	27.8%	16%	10.1%	—

F-strings: Making Date and Time Display a Breeze



Improve Code Readability with Enums

Store Sensitive Information Securely in Python with .env Files

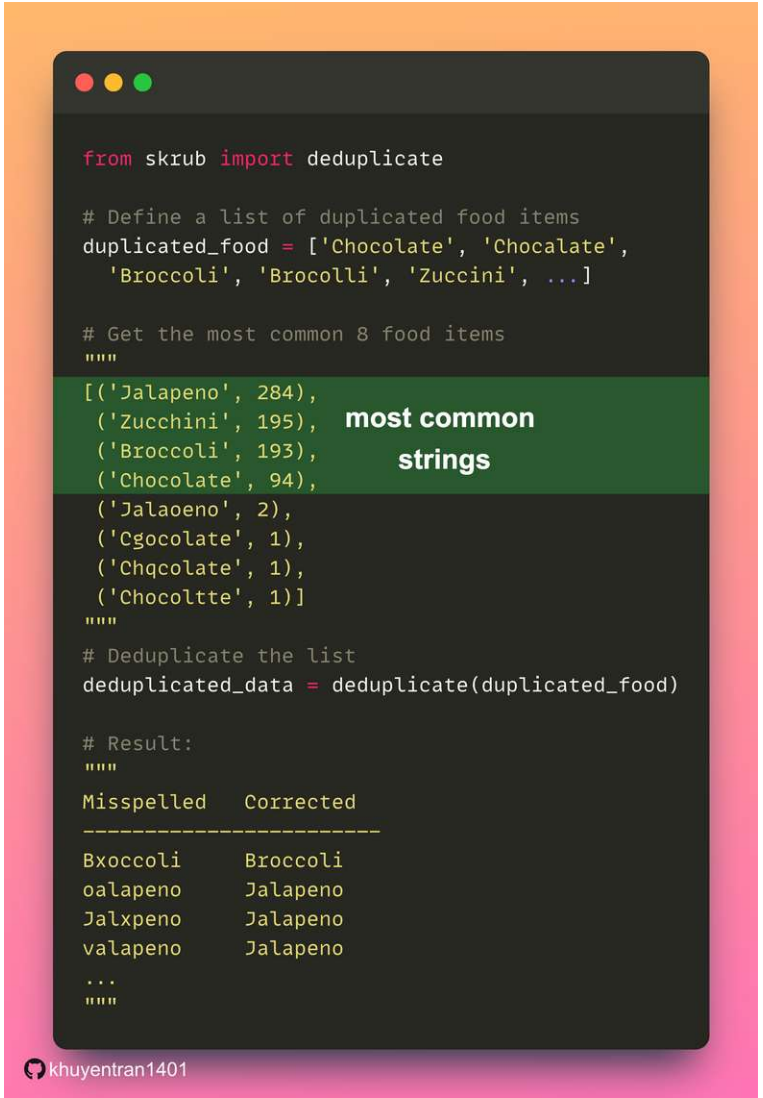


Rocketry: Modern Scheduling Library for Python

```python
from rocketry.conds import (
    daily, hourly, weekly, time_of_day, time_of_week,
)
from pathlib import Path

@app.cond()
def file_exists(file):
    return Path(file).exists()

@app.task(daily.after("08:00") & file_exists("myfile.csv"))
def do_work():
    ...

@app.task(hourly & time_of_day.between("22:00", "06:00"))
def do_hourly_at_night():
    ...

@app.task((weekly.on("Mon") | weekly.on("Sat")))
def do_twice_a_week():
    ...
```

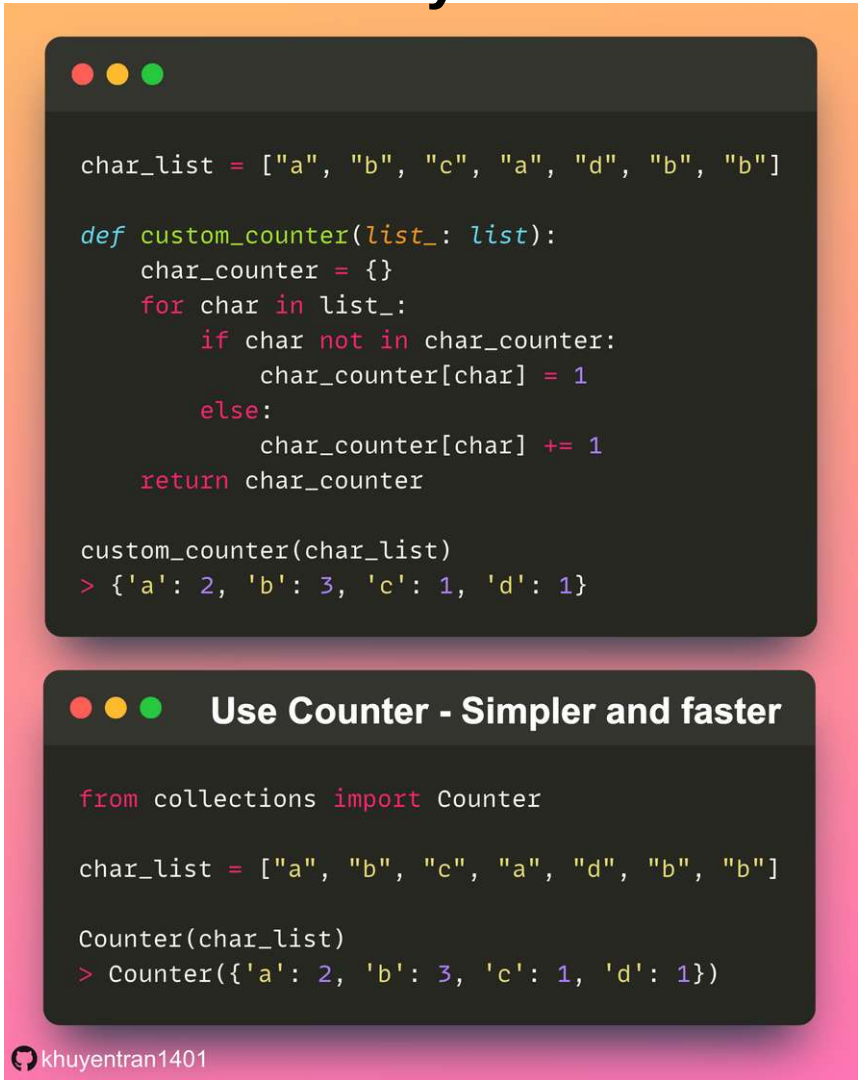
khuyentran1401

How to Handle Misspellings in Real-World Datasets

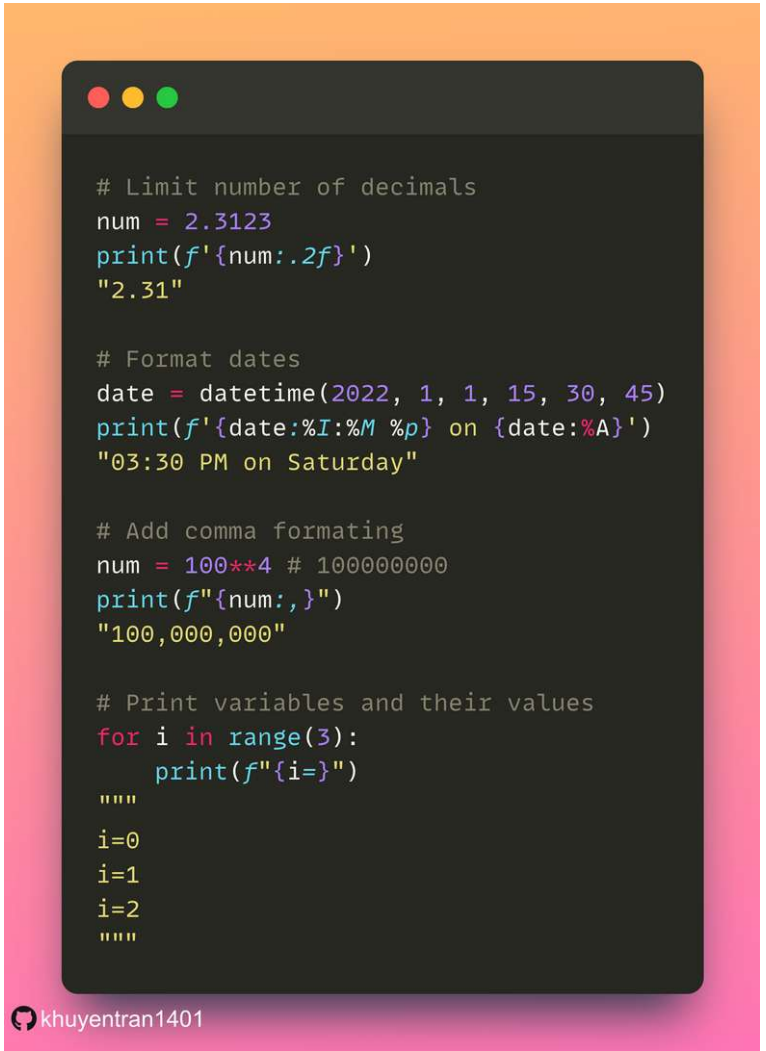


Faker: Create Fake Data in One Line of Code

The Most Efficient Way to Count Items in a List



Increase Code Readability and Output Presentation with Python's f-Strings



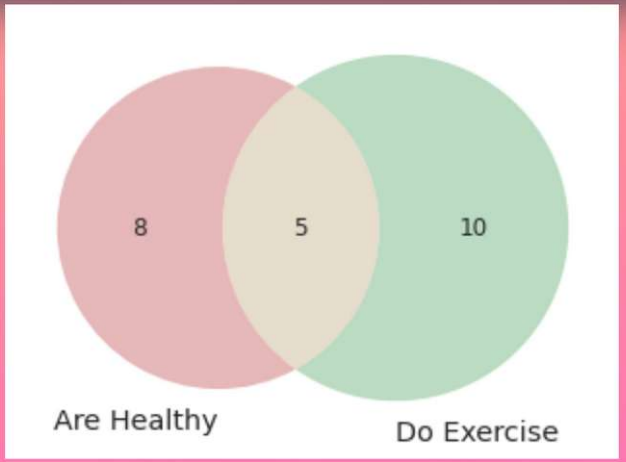

```python
# Limit number of decimals
num = 2.3123
print(f'{num:.2f}')
"2.31"

# Format dates
date = datetime(2022, 1, 1, 15, 30, 45)
print(f'{date:%I:%M %p} on {date:%A}')
"03:30 PM on Saturday"

# Add comma formating
num = 100**4 # 100000000
print(f"{num:,}")
"100,000,000"

# Print variables and their values
for i in range(3):
    print(f"{i=}")
"""
i=0
i=1
i=2
"""
```

tqdm: Add Progress Bar to Your Pandas Apply

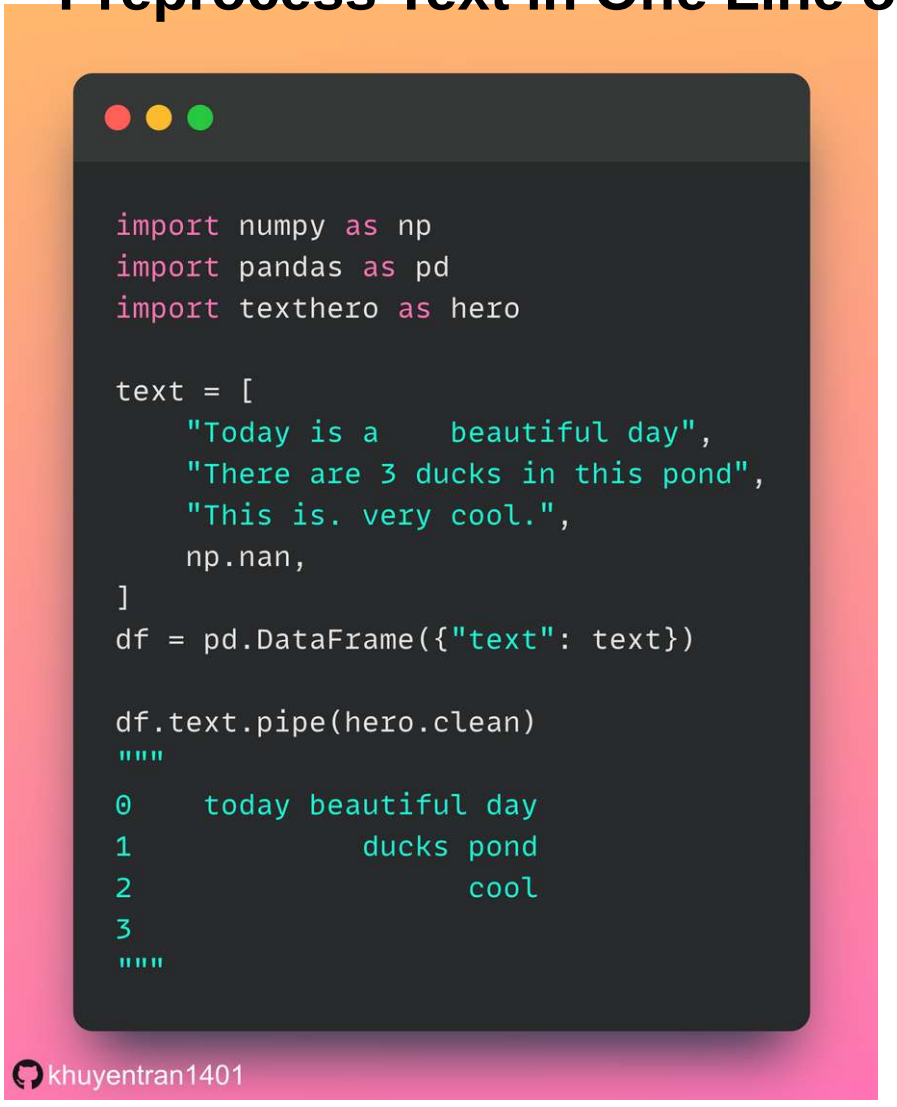
Create a Venn Diagram Using Python

Khuyen Tran
khuyentran1401

```python
import matplotlib.pyplot as plt
from matplotlib_venn import venn2

venn2(
  subsets = (8, 10, 5),
  set_labels = ('Are Healthy', 'Do Exercise')
)

plt.show()
```



Preprocess Text in One Line of Code with Texthero



Processing text in a DataFrame often involves writing lengthy code. Texthero simplifies this by enabling one-line preprocessing, including:

filling missing values

converting upper case to lower case

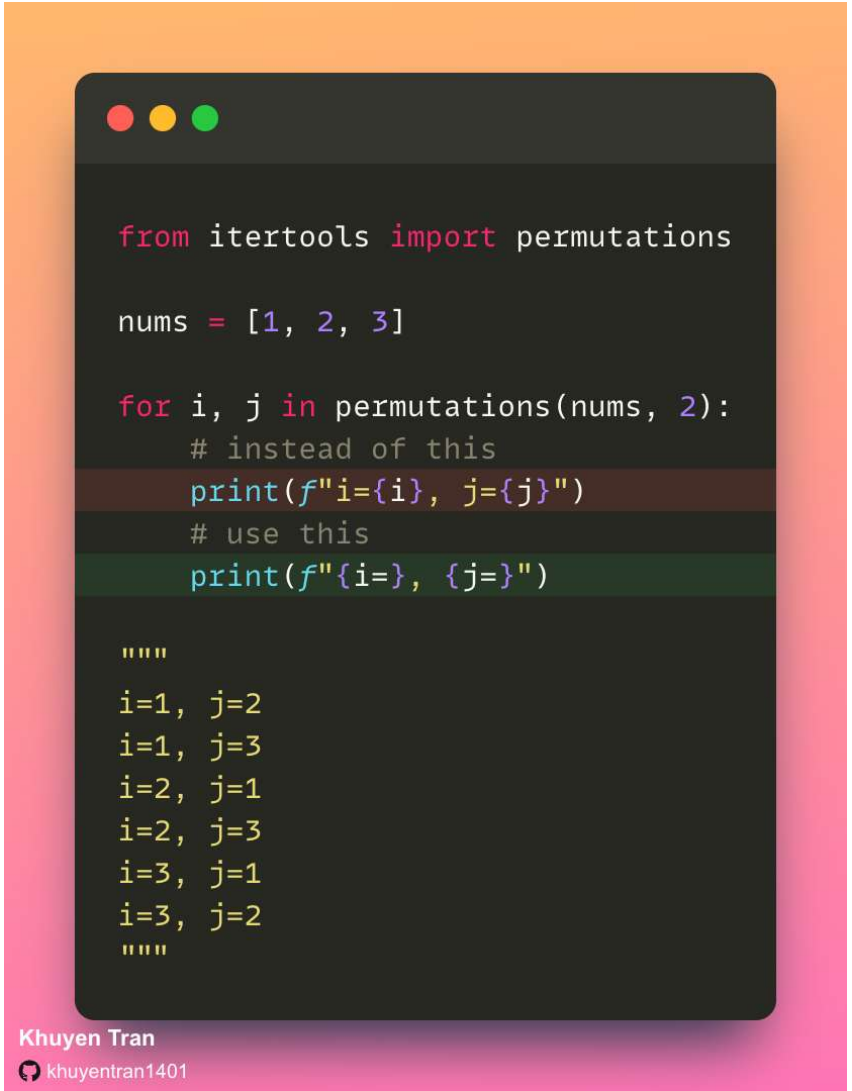
removing digits

removing punctuation

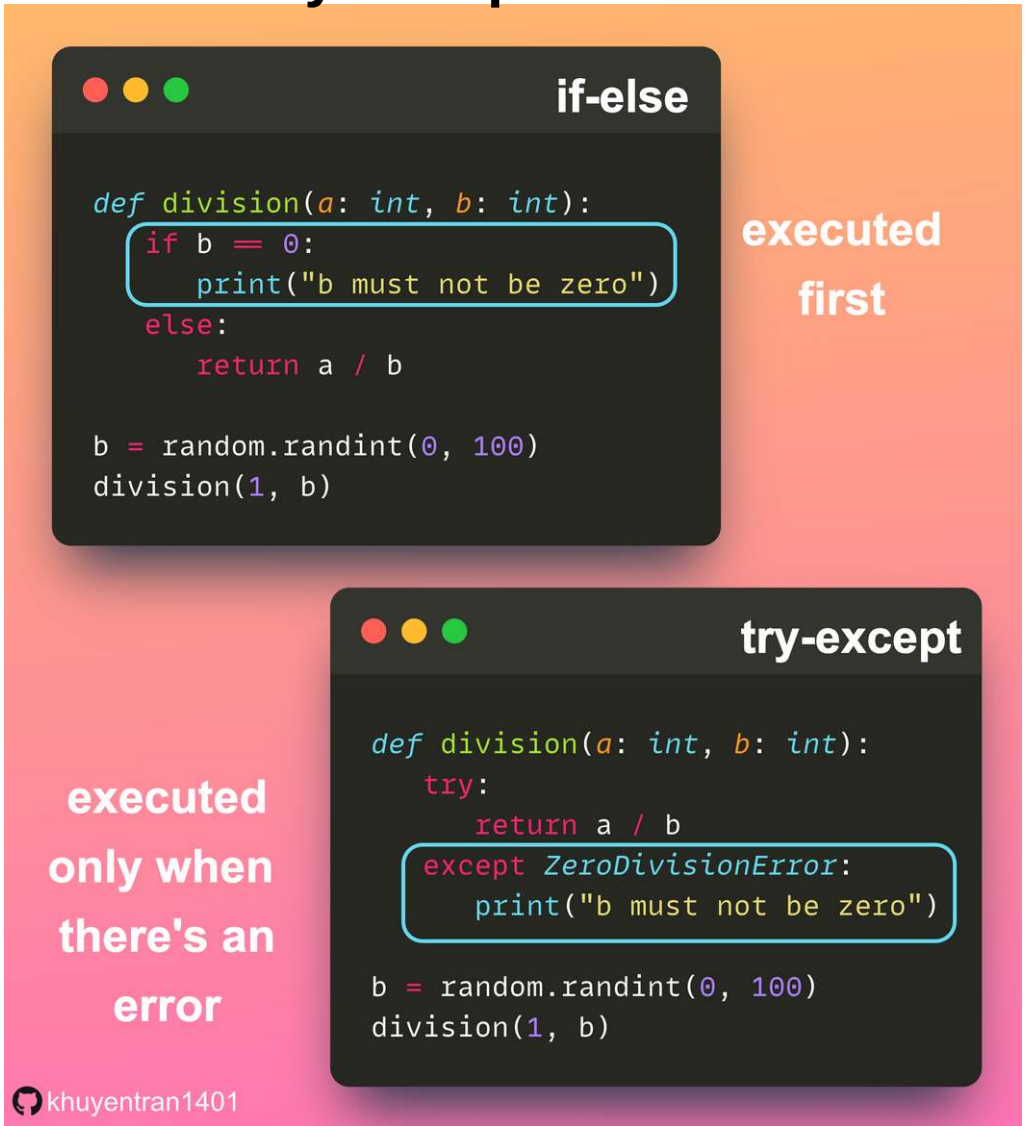
removing stopwords

removing whitespace

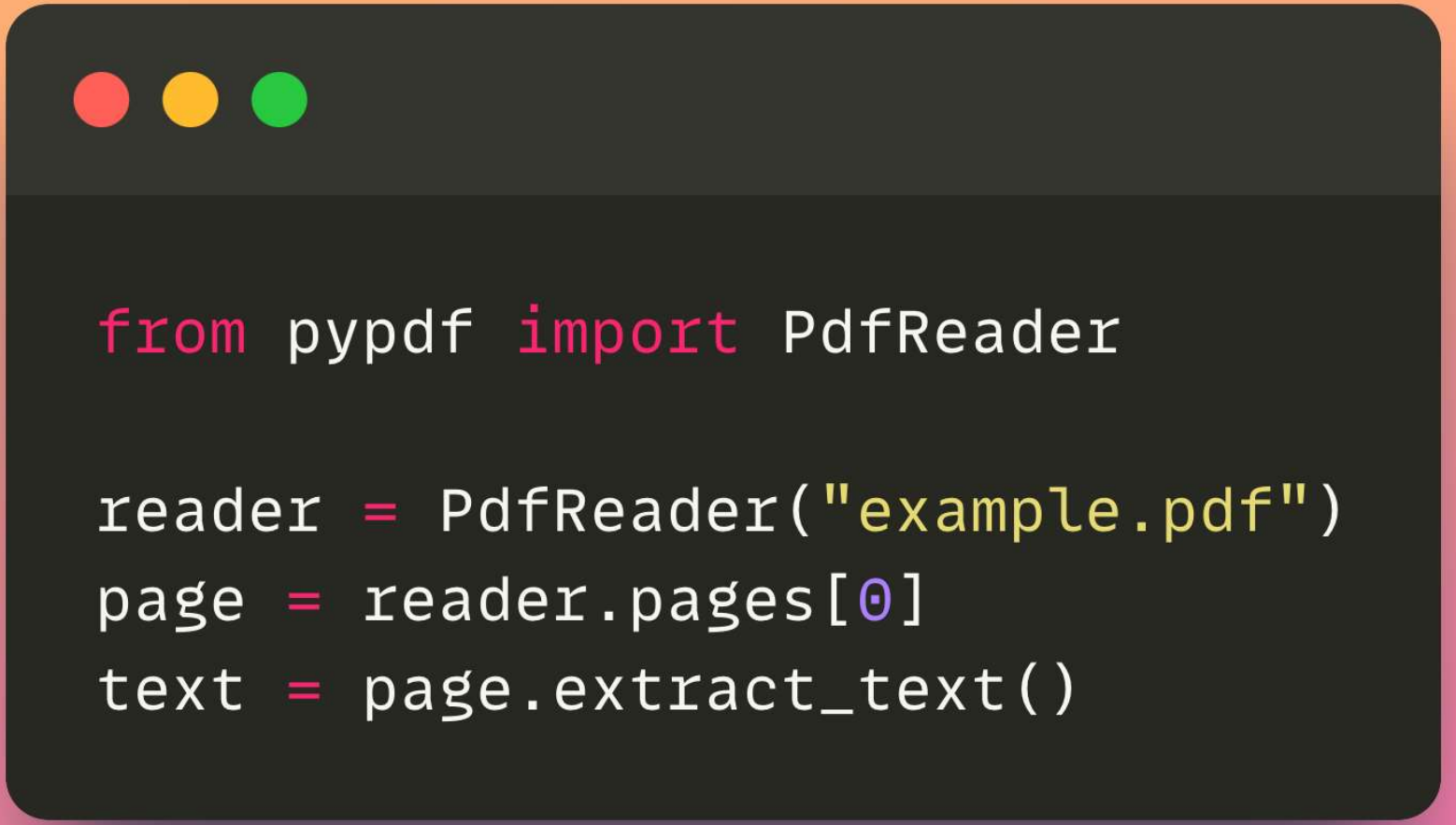
Debug Your Python Code with an Equal Sign in an f-String



try-except vs if-else



Supercharge PDF Text Extraction in Python with pypdf



```python
from pypdf import PdfReader

reader = PdfReader("example.pdf")
page = reader.pages[0]
text = page.extract_text()
```

khuyentran1401

Polars vs. Pandas for CSV Loading and Filtering

```python
import pandas as pd

df = pd.read_csv("data.csv")
df[
  (df["type"] == "heliport") &
  (df["continent"] == "EU")
]
```

143 ms ± 8.3 ms

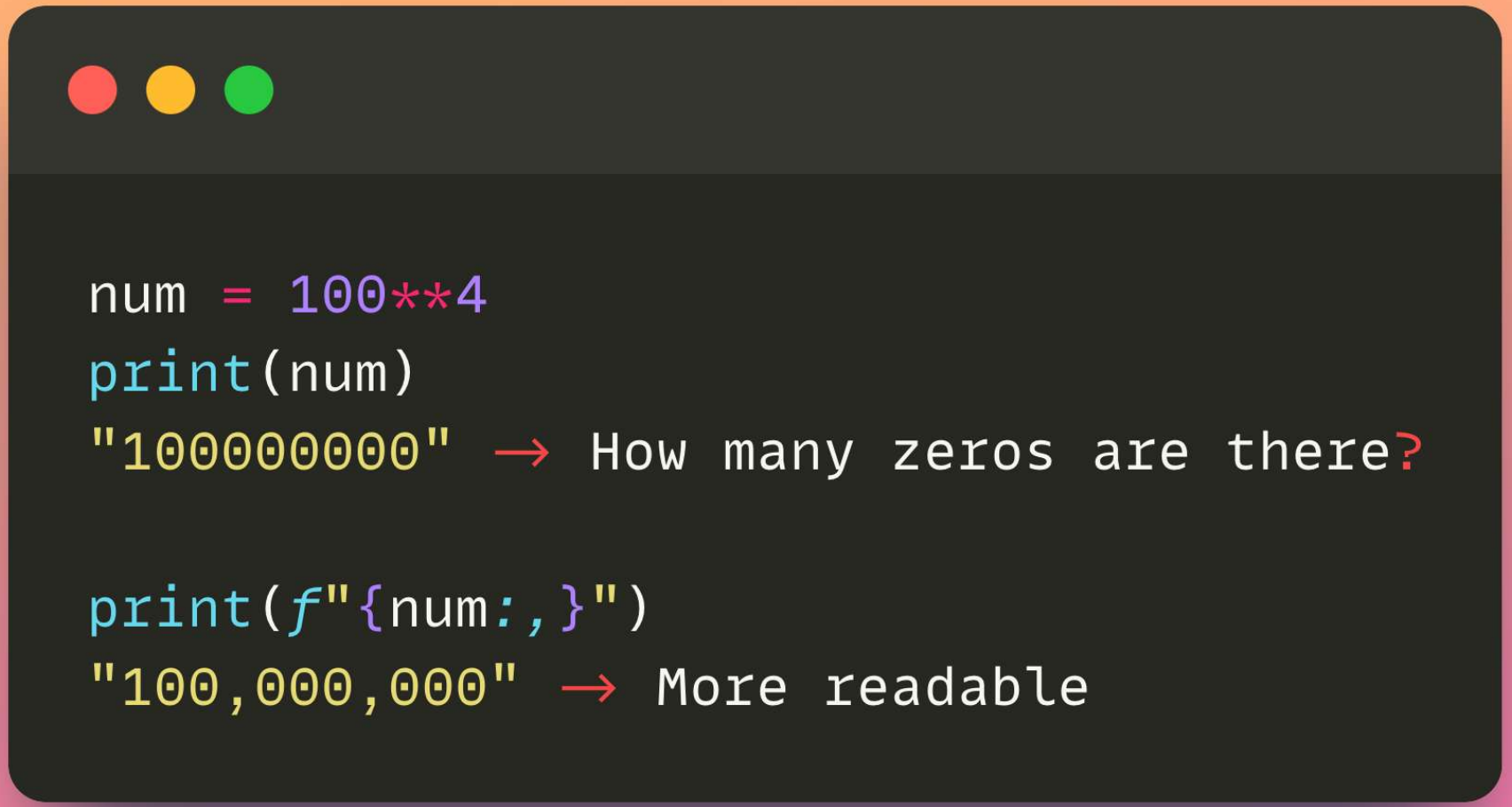
57k rows

```python
import polars as pl

pl.scan_csv("data.csv").filter(
  (pl.col("type") == "heliport") &
  (pl.col("continent") == "EU")
).collect()
```

5.6 ms ± 594 µs **25.5 times faster**

khuyentran1401

Enhance Number Readability with f-Strings and Comma Formatting



khuyentran1401

Decorator in Python

```python
import time

def time_func(func):
  def wrapper(*args, **kwargs):
    start = time.time()
    func(*args, **kwargs)
    end = time.time()
    print(f'Elapsed time: {(end - start)*1000:.3f}ms')
  return wrapper

@time_func
def add(num1: int, num2: int):
    print(f"Add {num1} and {num2}")
    return num1 + num2

@time_func
def multiply(num1: int, num2: int):
    print(f"Multiply {num1} and {num2}")
    return num1 * num2

add(1, 2)
multiply(1, 2)
"""
Add 1 and 2
Elapsed time: 1.006ms
Multiply 1 and 2
Elapsed time: 0.027ms"""
```

khuyentran1401

Polars: Speed Up Data Processing 12x with Lazy Execution

pandas

```
import pandas as pd

pd_df = pd.DataFrame(data)

%%timeit
pd_df[
  (pd_df['Cat1'] == 'a') &
  (pd_df['Cat2'] == 'b') &
  (pd_df['Num1'] ≥ 70)
]
```

```
706 ms ± 75.4 ms per loop
```

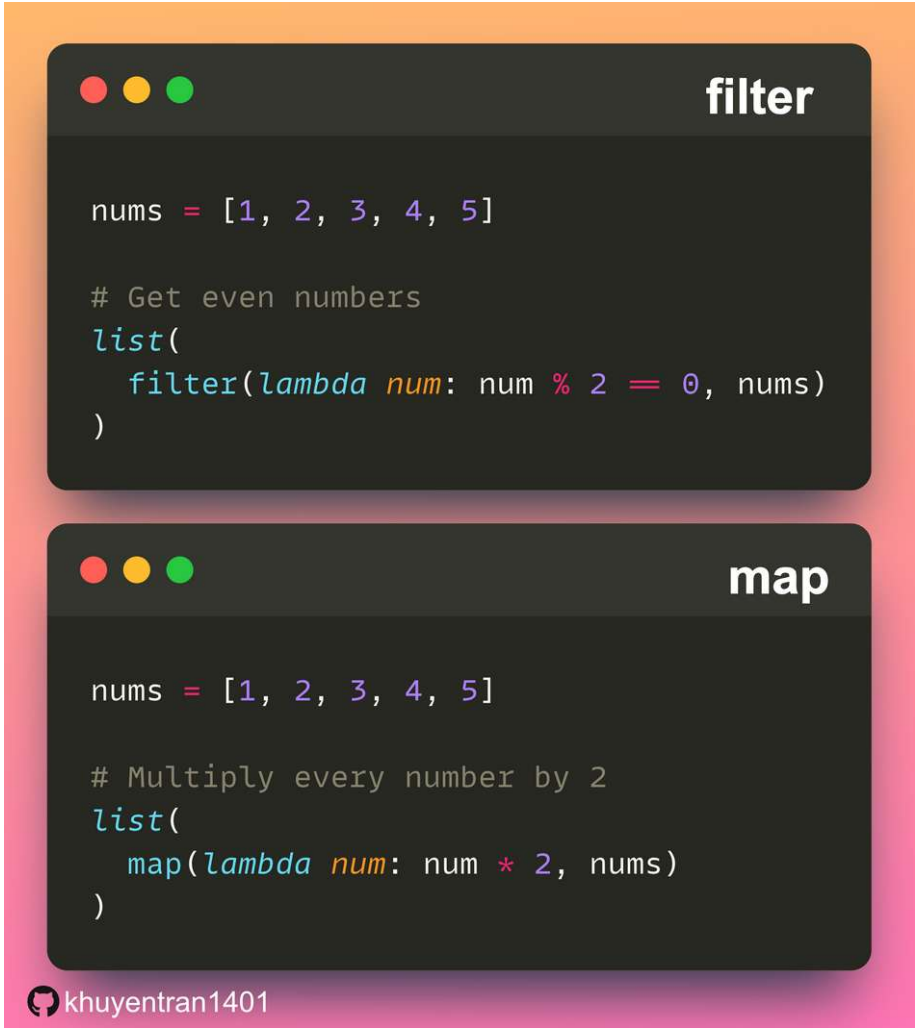
polars

```
import polars as pl

pl_df = pl.DataFrame(data)

%%timeit
pl_df.lazy().filter(
  (pl.col('Cat1') == 'a') &
  (pl.col('Cat2') == 'b') & (
  pl.col('Num1') ≥ 70)
).collect()
```

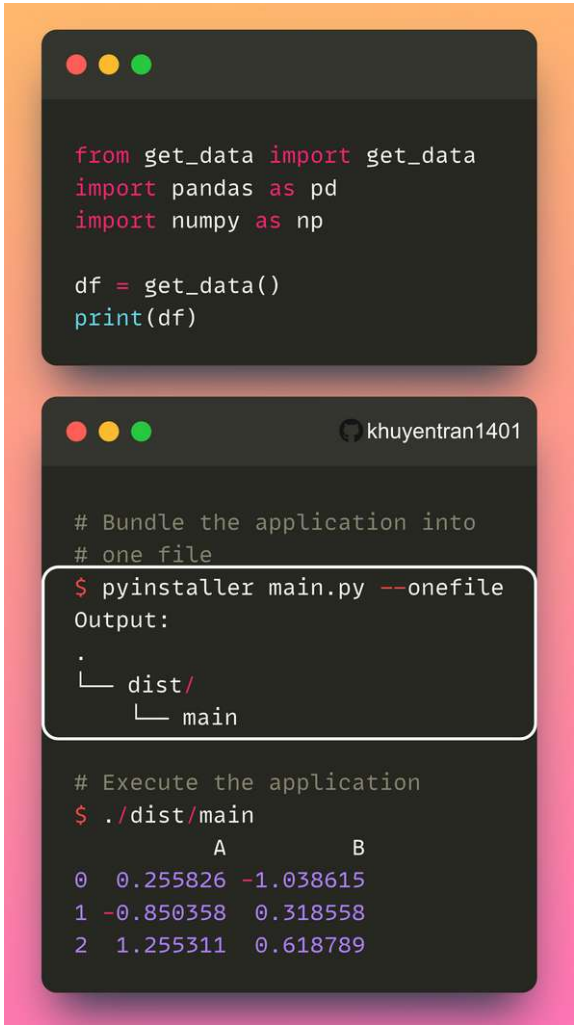
```
58.1 ms ± 428 µs per loop
```

khuyentran1401

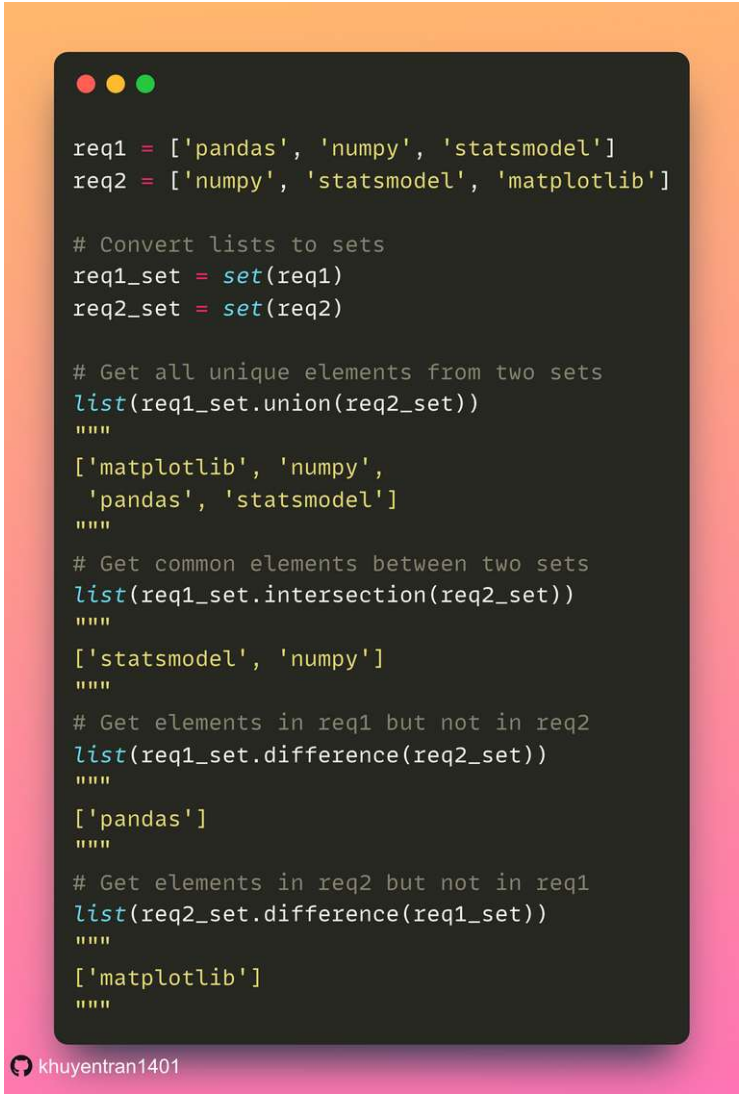
Python’s Powerful Iterable Manipulation: Map and Filter



PyInstaller: Bundle a Python Application Into a Single Executable

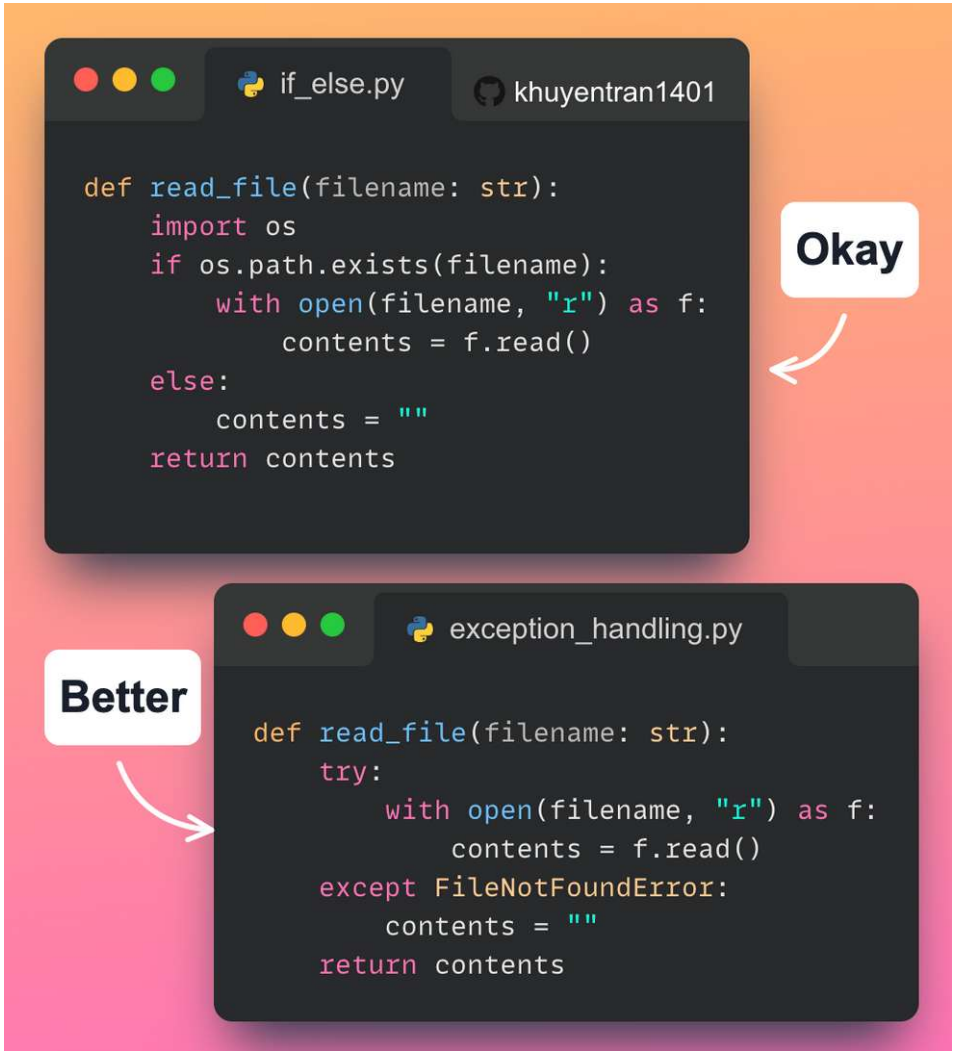


Perform Set Operations on a Python List



Python Fire: Generate a CLI for Any Python Objects in Two Lines of Code

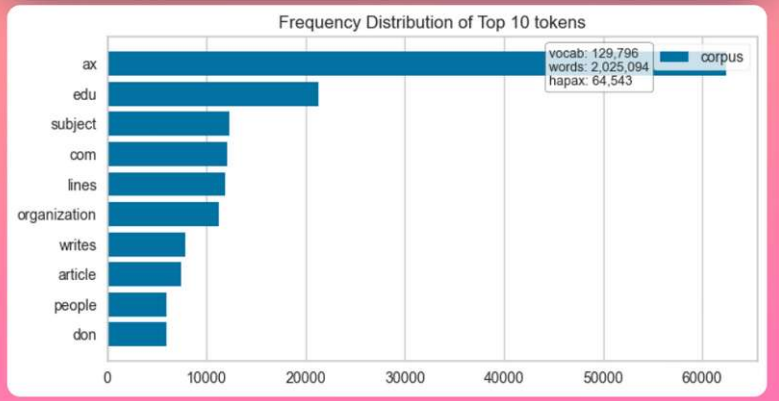
Exception Handling vs. If-Else Statements: Which is Better for Error Handling?



Visualize the Frequency Tokens in a Text Corpora

Khuyen Tran
khuyentran1401

```python
from sklearn.feature_extraction.text
  import CountVectorizer
from yellowbrick.text import FreqDistVisualizer

# Convert corpora to a matrix of token counts
vec = CountVectorizer(stop_words='english')
docs = vec.fit_transform(data)
feat = vectorizer.get_feature_names()

# Plot a token frequency distribution
vis = FreqDistVisualizer(features=feat)
vis.fit(docs)
vis.show()
```



Geopy: Extract Location Based on Python String

```
>>> from geopy.geocoders import Nominatim
>>> geolocator = Nominatim(user_agent='find_location')
>>> location = geolocator.geocode('30 North Circle Drive, Edwardsville, IL')

>>> print(location.address)
Circle Drive, Edwardsville, Madison County, Illinois, 62026, United States of
America

>>> print((location.latitude, location.longitude))
(38.796956, -89.999882)
```

Concurrently execute tasks on separate CPUs

```python
from joblib import Parallel, delayed
import multiprocessing

def add_three(num: int):
    return num + 3

num_cores = multiprocessing.cpu_count()
results = Parallel(n_jobs=num_cores)(delayed(add_three)(i) for i in range(10))

# Output: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

Enumerate


#Instead of this
'''
>>> for i in range(len(arr)):
print(i, arr[i])
'''
#Use this
>>> arr = ['a','b','c','d','e']
>>> for i, val in enumerate(arr):
print(i,val)
0 a
1 b
2 c
3 d
4 e

Colorama: Produce a colored terminal text in Python

```
>>> from colorama import Fore, Back, Style
>>> print(Fore.RED)
>>> print("I'm red")
I'm red
>>> print(Fore.YELLOW)
>>> print("I'm yellow")
I'm yellow
>>> print(Back.BLUE)
>>> print("I'm yellow with a blue background")
I'm yellow with a blue background #Will have a blue background on your terminal
>>> print(Style.RESET_ALL)
>>> print("I'm normal now")
I'm normal now
```

Filter Rows only if Column Contains Values from another List

```
>>> import pandas as pd
>>> df = pd.DataFrame({'a':[1,2,3], 'b': [4,5,6]})
>>> df
   a  b
0  1  4
1  2  5
2  3  6
>>> l = [1,2,6,7]
>>> df.a.isin(l)
0     True
1     True
2    False
Name: a, dtype: bool
>>> df = df[df.a.isin(l)]
>>> df
   a  b
0  1  4
1  2  5
```

Pyfiglet: Make large and unique letters out of ordinary text in Python

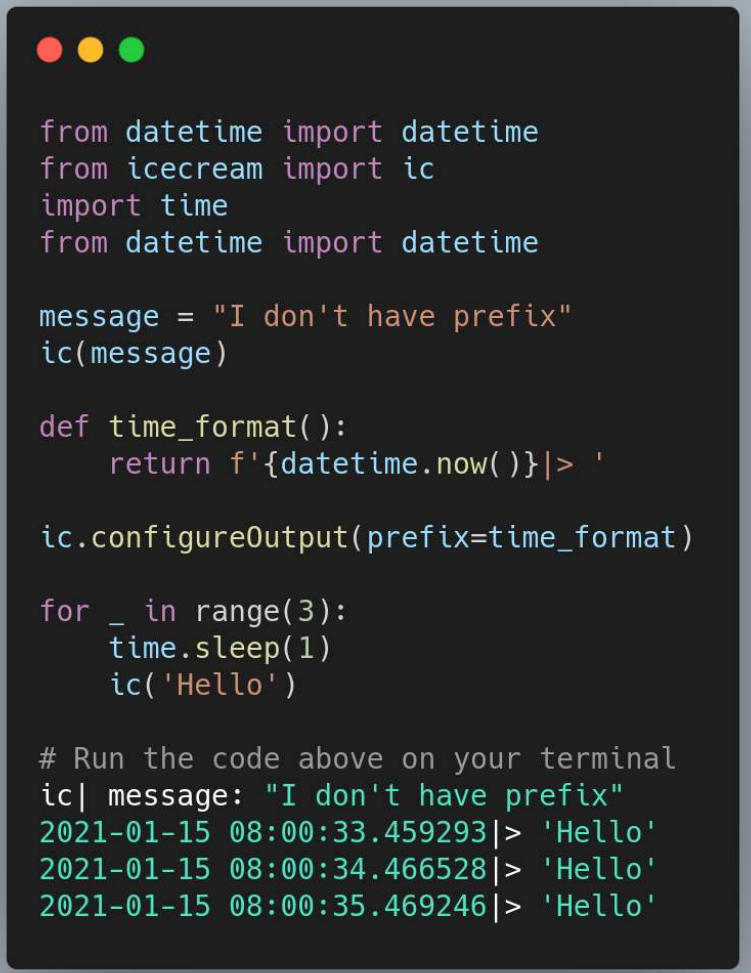
argparse: Python Library to Parse Arguments from Command Line

```python
# test.py
import argparse

parser = argparse.ArgumentParser()

# Add optional argument
parser.add_argument("-p", "--Parameter",
                    default=4, # default value
                    type=int,  # data type
                    help = "Choose the parameter") # description

# Read arguments from command line
args = parser.parse_args()
print("Your chosen parameter is % s" % args.Parameter)

'''
# On your terminal
$ python test.py # use default value
Your chosen parameter is 4

$ python test.py -p 5 # change default value
Your chosen parameter is 5

$ python test.py -p 'hello'
usage: test.py [-h] [-p PARAMETER]
test.py: error: argument -p/--Parameter: invalid int value: 'hello'
'''
```

Pathlib: Iterate Over All Files that End with '.csv' in a Directory

```
>>> from pathlib import Path

>>> directory_name = 'data'

# Loop files in a directory
>>> pathlist = Path(directory_name).rglob('*.csv')
>>> for path in pathlist:
...     print(str(path))
...
data/data1.csv
data/data2.csv
data/data3.csv
```

Icecream: Adding a Datetime Stamp to Python print



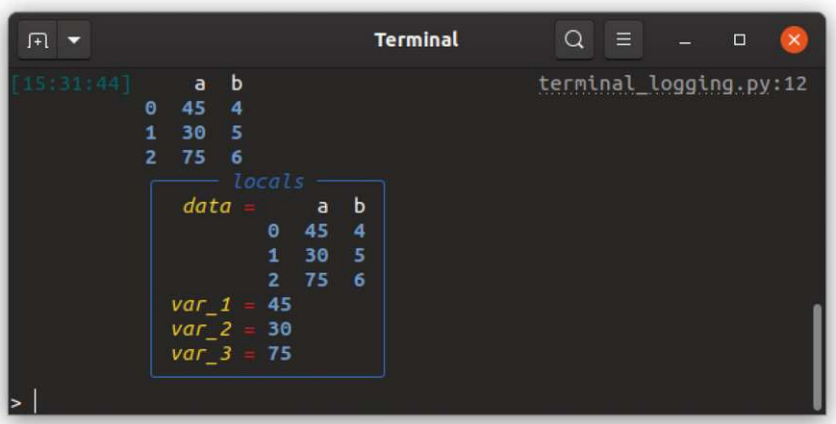
```python
from datetime import datetime
from icecream import ic
import time
from datetime import datetime

message = "I don't have prefix"
ic(message)

def time_format():
    return f'{datetime.now()}|> '

ic.configureOutput(prefix=time_format)

for _ in range(3):
    time.sleep(1)
    ic('Hello')

# Run the code above on your terminal
ic| message: "I don't have prefix"
2021-01-15 08:00:33.459293|> 'Hello'
2021-01-15 08:00:34.466528|> 'Hello'
2021-01-15 08:00:35.469246|> 'Hello'
```

Schedule: Schedule your Python Functions to Run At a Specific Time

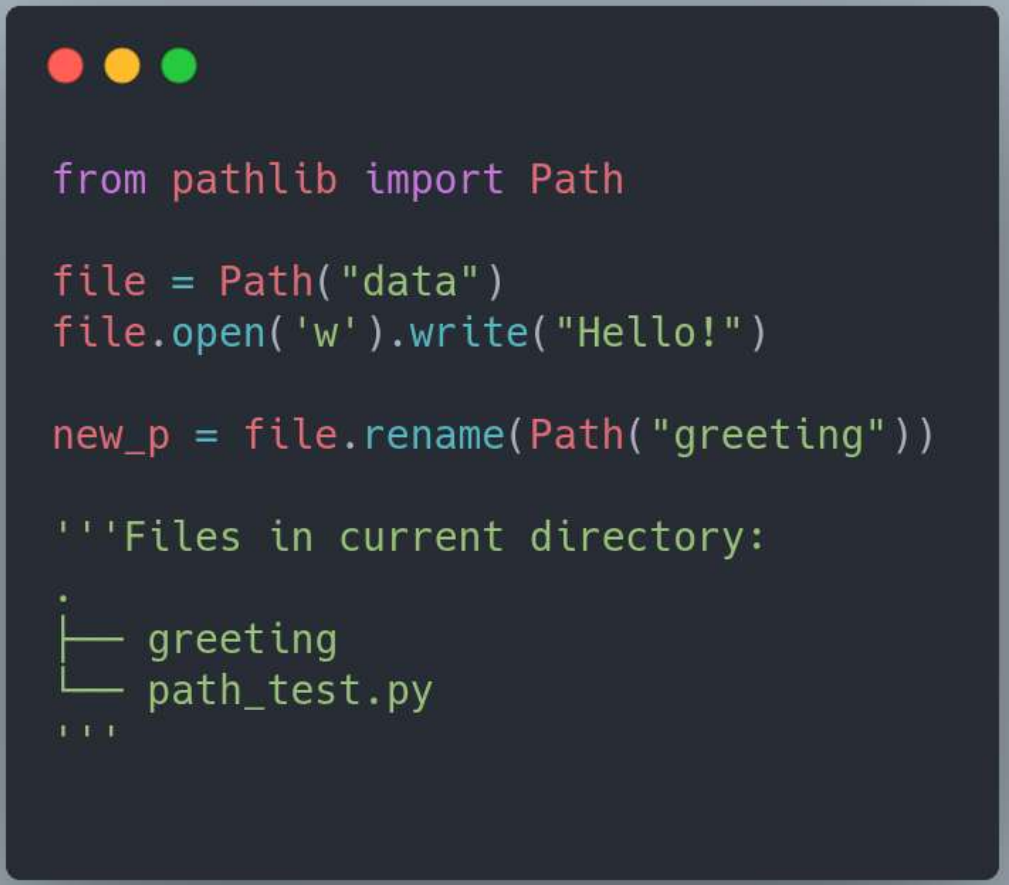
```python
import schedule
import time

def get_incoming_data():
    print("Get incoming data")

def train_model():
    print("Retraining model")

schedule.every().day.at("10:30").do(get_incoming_data)
schedule.every().wednesday.at("8:00").do(train_model)

while True:
    schedule.run_pending()
    time.sleep(1)
```

Rich's Console: Debug your Python Function in One Line of Code

```python
from rich.console import Console
import pandas as pd

console = Console()

data = pd.DataFrame({'a': [1,2,3], 'b': [4,5,6]})
def edit_data(data):
    var_1 = 45
    var_2 = 30
    var_3 = var_1 + var_2
    data['a'] = [var_1,var_2,var_3]
    console.log(data, log_locals=True)

edit_data(data)
```



pathlib: Create, Write, and Rename Files in One Line of Code



```python
from pathlib import Path

file = Path("data")
file.open('w').write("Hello!")

new_p = file.rename(Path("greeting"))

'''Files in current directory:
.
├── greeting
└── path_test.py
'''
```

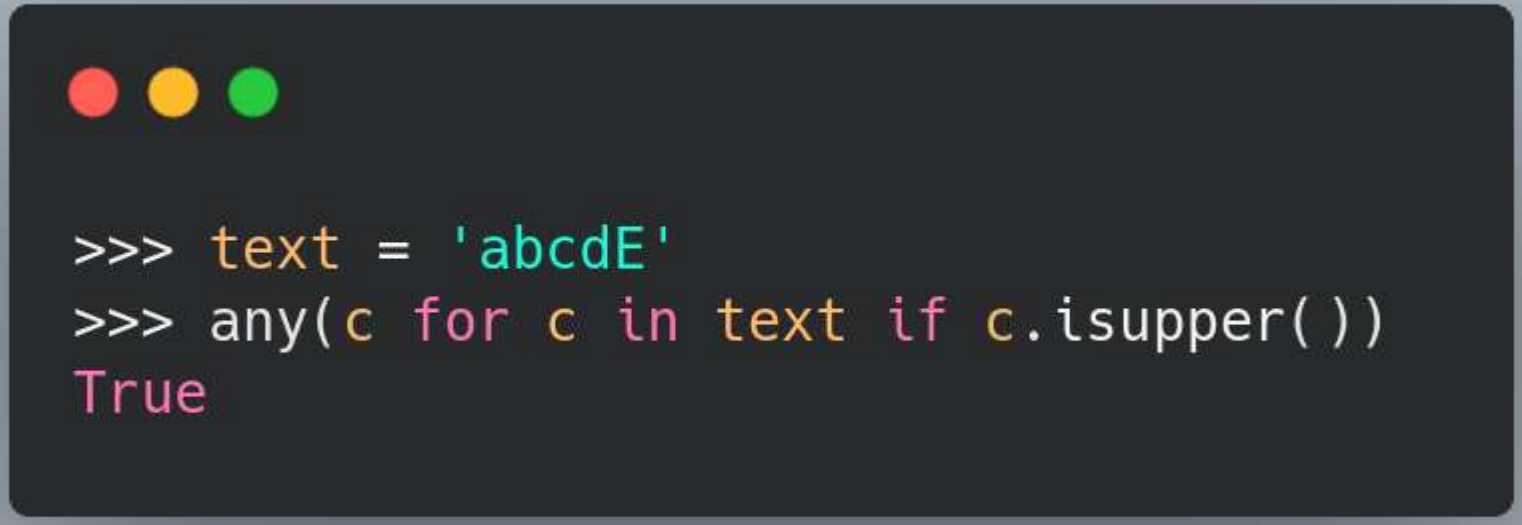
join method: Turn an Iterable into a Python String

```
fruits = ['apples', 'oranges', 'grapes']

fruits_str = ', '.join(fruits)

print(f"Today, I need to get some {fruits_str}
in the grocery store")

"""Output:
Today, I need to get some apples, oranges,
grapes in the grocery store
"""
```

Shutil: Move Files in Python

```
"""Current directory before moving

.
├── dir1
│   └── example.txt
├── dir2
└── move_files.py
"""
#--------------------------------------#

# move_files.py
import shutil

shutil.move('dir1/example.txt', 'dir2')


#--------------------------------------#
"""Current directory after moving
.
├── dir1
├── dir2
│   └── example.txt
└── move_files.py
"""
```

any: Check if Any Element of an Iterable is True



snoop : Smart Print to Debug your Python Function

String find: Find the Index of a Substring in a Python String

```
>>> sentence = "Today is is Saturday"

# Find the index of first occurrence of the substring
>>> sentence.find("day")
2

# Start searching for the substring at index 3
>>> sentence.find("day", 3)
17

>>> sentence.find('nice')
-1 # No substring is found
```

How to Improve the Readability of your JSON file using Indent

```
>>> import json
>>> pet = dict(
...     kind="dog",
...     name= "Bim Bim",
...     age=7,
...     favorite_food='yogurt'
... )

# Without identation
>>> print(json.dumps(pet))
{"kind": "dog", "name": "Bim Bim", "age": 7,
"favorite_food": "yogurt"}

# With indentation
>>> print(json.dumps(pet, indent=4))
{
    "kind": "dog",
    "name": "Bim Bim",
    "age": 7,
    "favorite_food": "yogurt"
}
```

rich.inspect: Produce a Beautiful Report on any Python Object

```
from rich import inspect

inspect('hello', methods=True)
```

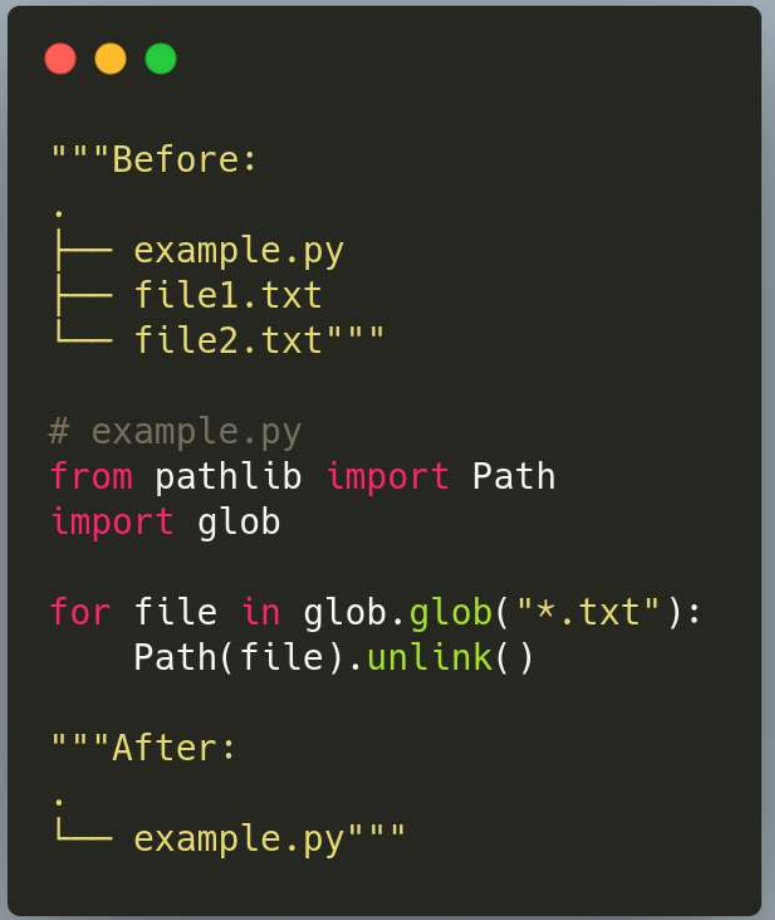
Hyper

```
 ──────────────────────────────── <class 'str'> ─────────────────────────────────
 str(object='') -> str
 str(bytes_or_buffer[, encoding[, errors]]) -> str

 ┌──────────────────────────────────────────────────────────────────────────────┐
 │ 'hello'                                                                      │
 └──────────────────────────────────────────────────────────────────────────────┘

   capitalize = def capitalize(): Return a capitalized version of the
                string.
     casefold = def casefold(): Return a version of the string suitable
                for caseless comparisons.
       center = def center(width, fillchar=' ', /): Return a centered
                string of length width.
        count = def count(...) S.count(sub[, start[, end]]) -> int
       encode = def encode(encoding='utf-8', errors='strict'): Encode
                the string using the codec registered for encoding.
     endswith = def endswith(...) S.endswith(suffix[, start[, end]]) ->
                bool
   expandtabs = def expandtabs(tabsize=8): Return a copy where all tab
                characters are expanded using spaces.
         find = def find(...) S.find(sub[, start[, end]]) -> int
       format = def format(...) S.format(*args, **kwargs) -> str
```

How to Execute Shell Commands in a Python Script

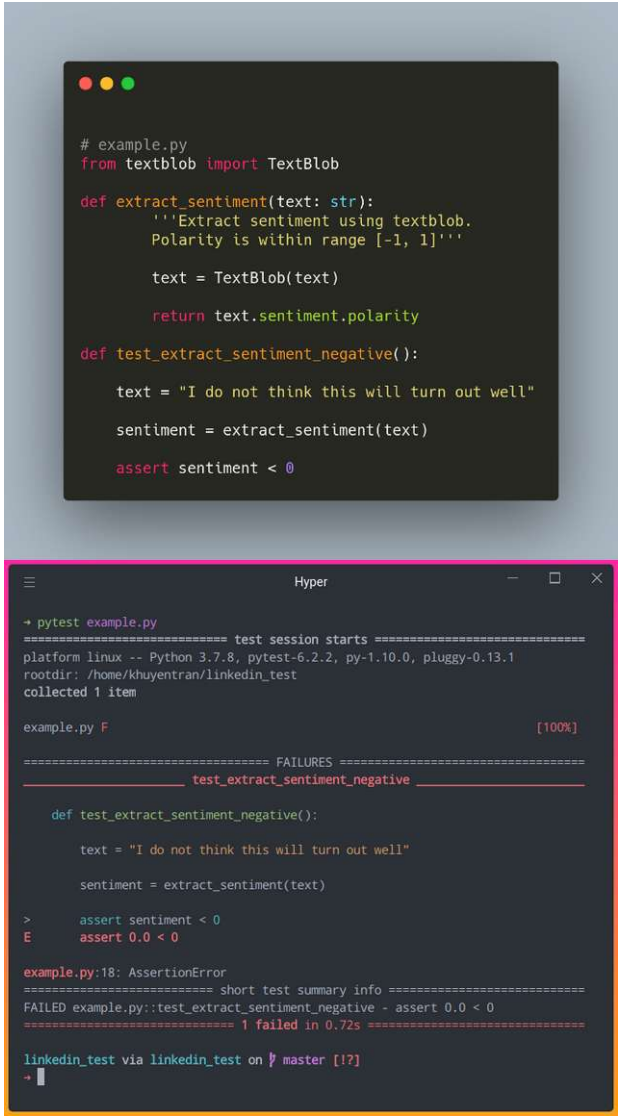
```python
import os
os.system("echo Files in the current directory are:")
os.system("ls")

"""Terminal output
Files in the current directory are:
example.py          test.py
"""
```

glob and Path.unlink: Remove All Flles that End with “.txt”



```
"""Before:
.
├── example.py
├── file1.txt
└── file2.txt"""

# example.py
from pathlib import Path
import glob

for file in glob.glob("*.txt"):
    Path(file).unlink()

"""After:
.
└── example.py"""
```

pytest: Don’t Assume. Test your Python Code



re.sub: Replace One String with Another String Using Regular Expression

```
>>> import re

>>> text = "Today is 3/7/2021"

>>> match_pattern = r'(\d+)/(\d+)/(\d+)'

>>> print(re.sub(match_pattern, 'Sunday', text))
Today is Sunday

>>> print(re.sub(match_pattern, r'\3-\1-\2', text))
Today is 2021-3-7
```

python-dotenv: How to Load the Secret Information from .env File

```
# .env

USERNAME=my_user_name
PASSWORD=secret_password
```

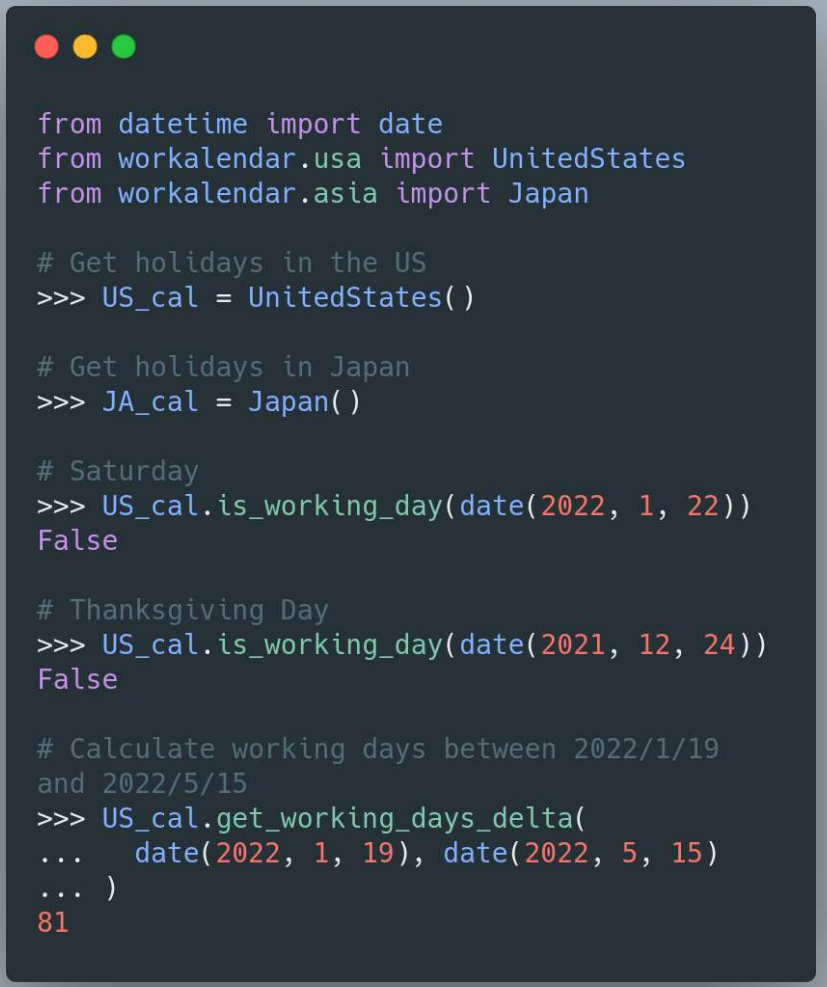
```
"""Files in current directory
.
├── .env
└── example.py"""

# example.py
from dotenv import load_dotenv
import os

load_dotenv()
PASSWORD = os.getenv('PASSWORD')
print(PASSWORD)
# Output: secret_password
```

Typer: Build a Command-Line Interface in a Few Lines of Code

```python
# main.py
import typer

def process_data(data: str, version: int):
    print(f'Processing {data},'
          f'version {version}')

if __name__ == '__main__':
    typer.run(process_data)
```

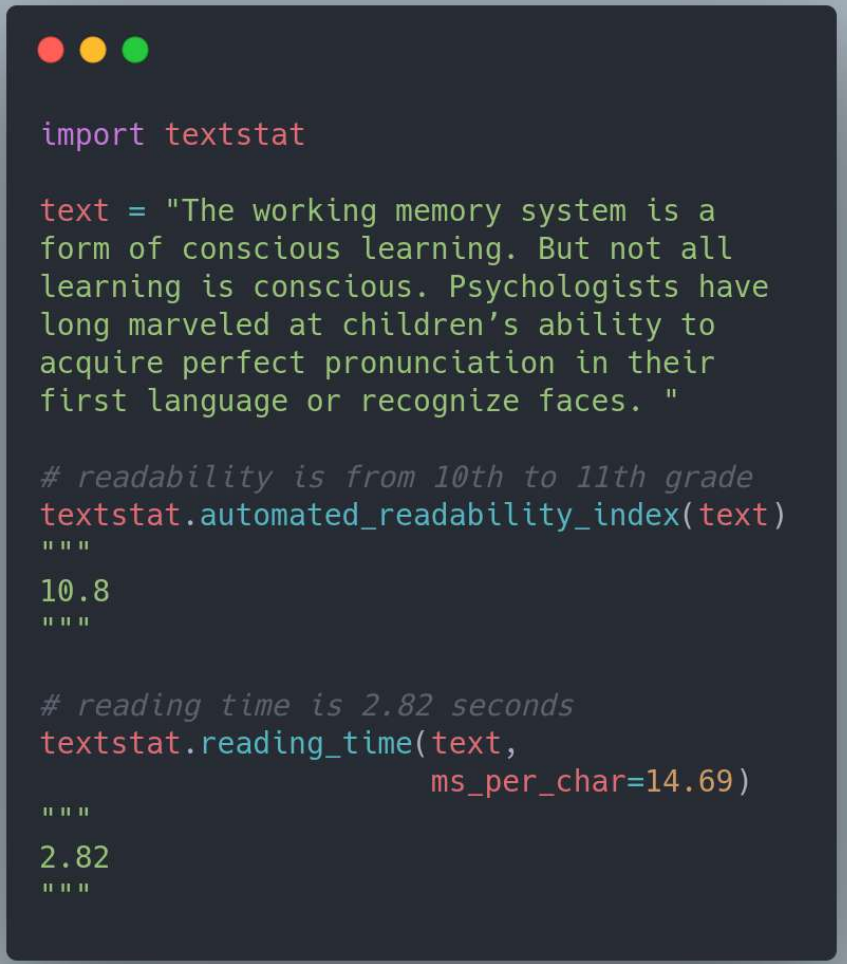
```
# On your terminal

$ python main.py data version_1
Usage: main.py [OPTIONS] DATA VERSION
Try 'main.py --help' for help.

Error: Invalid value for 'VERSION':
version_1 is not a valid integer

$ python main.py data 1
Processing data,version 1
```

Workalendar: Handle Working-Day Computation in Python



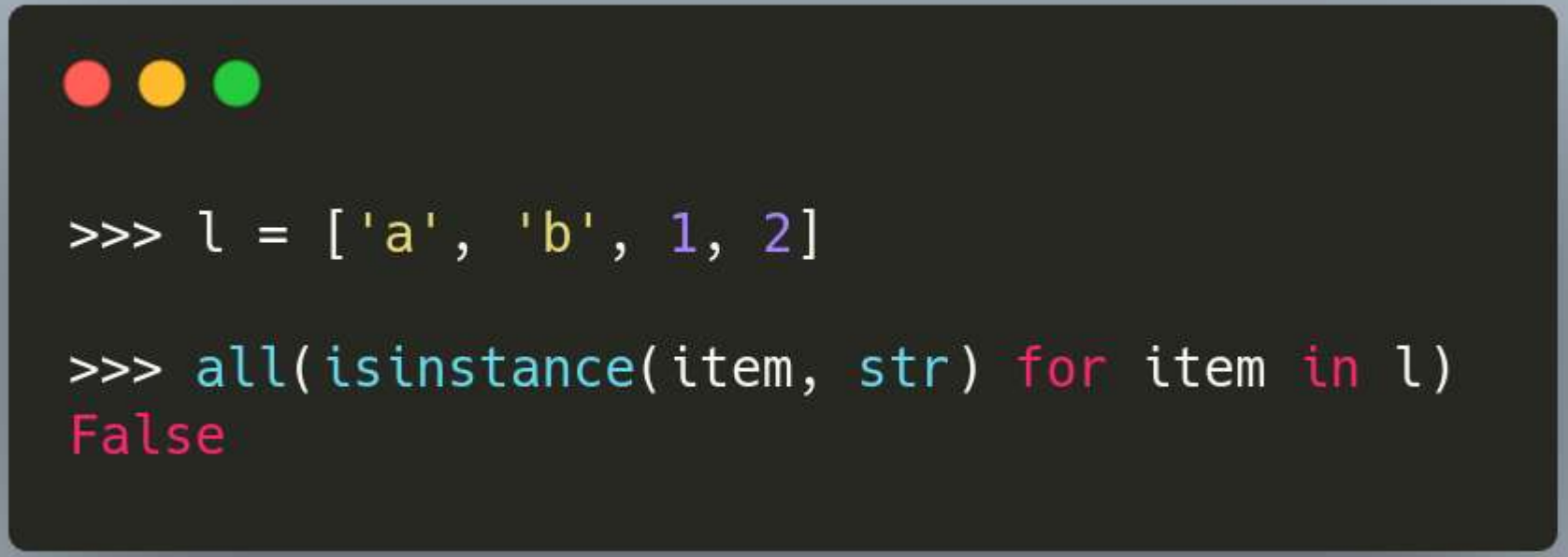
```
from datetime import date
from workalendar.usa import UnitedStates
from workalendar.asia import Japan

# Get holidays in the US
>>> US_cal = UnitedStates()

# Get holidays in Japan
>>> JA_cal = Japan()

# Saturday
>>> US_cal.is_working_day(date(2022, 1, 22))
False

# Thanksgiving Day
>>> US_cal.is_working_day(date(2021, 12, 24))
False

# Calculate working days between 2022/1/19
and 2022/5/15
>>> US_cal.get_working_days_delta(
...   date(2022, 1, 19), date(2022, 5, 15)
... )
81
```

textstat: Calculate Statistics From Text



```
import textstat

text = "The working memory system is a
form of conscious learning. But not all
learning is conscious. Psychologists have
long marveled at children's ability to
acquire perfect pronunciation in their
first language or recognize faces. "

# readability is from 10th to 11th grade
textstat.automated_readability_index(text)
"""
10.8
"""

# reading time is 2.82 seconds
textstat.reading_time(text,
                      ms_per_char=14.69)
"""
2.82
"""
```

all: Check if All Elements of an Interable Are Strings



```
>>> l = ['a', 'b', 1, 2]

>>> all(isinstance(item, str) for item in l)
False
```

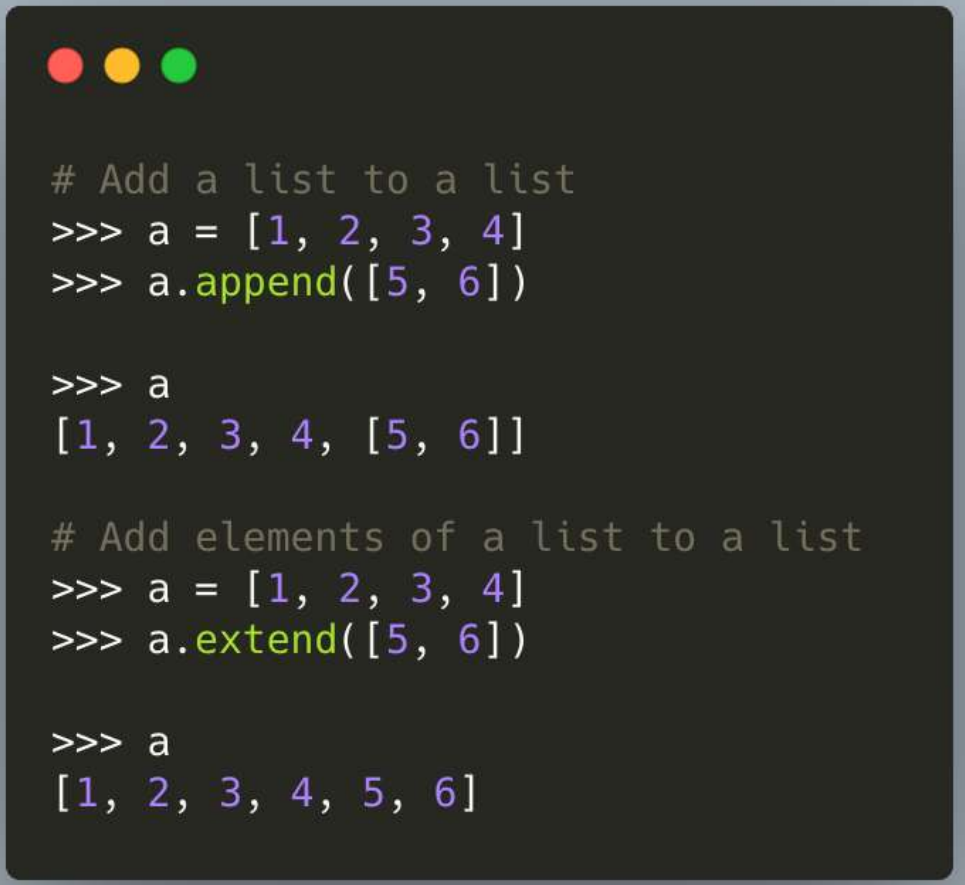
Set Difference: Find the Difference Between 2 Sets

```
>>> a = [1, 2, 3, 4]
>>> b = [1, 3, 4, 5, 6]

# Find elements in a but not in b
>>> diff = set(a).difference(set(b))
>>> list(diff)
[2]

# Find elements in b but not in a
>>> diff = set(b).difference(set(a))
>>> list(diff)
[5, 6]
```

tqdm.set_description: Set a Description for Your Progress Bar

```python
from tqdm import tqdm
from time import sleep

fruits = tqdm(['apple', 'orange', 'grape'])
for fruit in fruits:
    sleep(0.3)
    fruits.set_description(f"Picking {fruit}")

"""
Picking apple:  33%|██████            | 1/3
[00:05<00:10,  5.00s/it]
Picking orange:  67%|████████████      | 2/3
[00:10<00:05,  5.00s/it]
Picking grape: 100%|██████████████████| 3/3
[00:00<00:00,  3.27it/s]
"""
```

Difference between list append and list extend



Create a New Directory and File. os or pathlib?

os_example.py

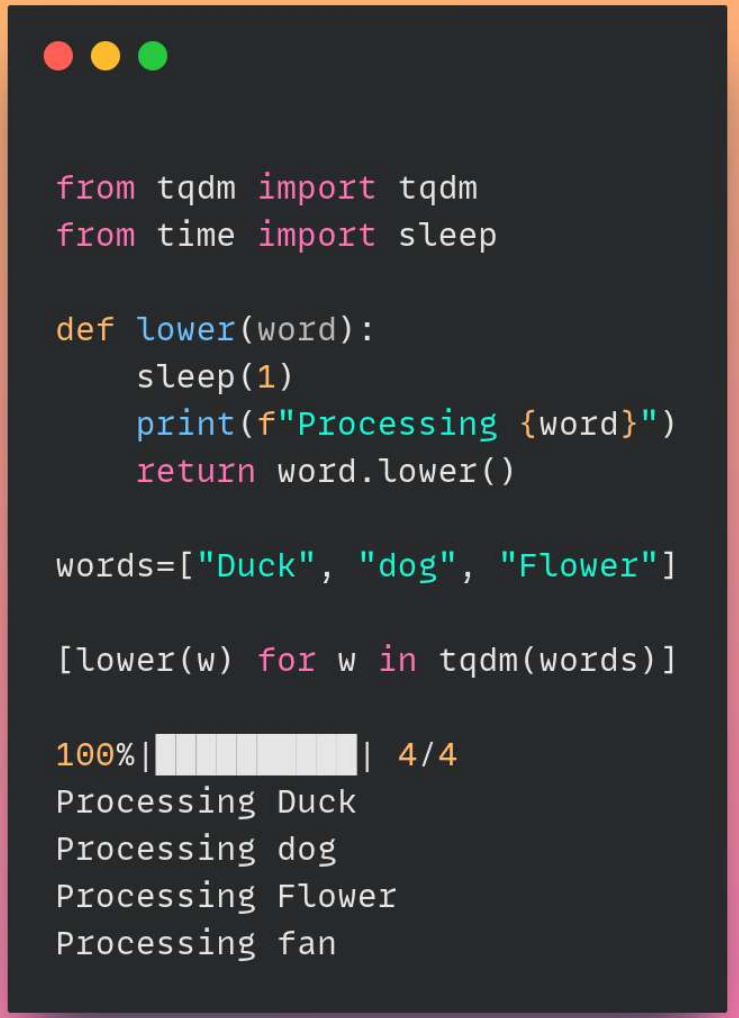
```python
import os

path = 'new'
file = 'new_file.txt'

# Create a new directory
if not os.path.exists(path):
    os.makedirs(path)

# Create new file inside new directory
with open(os.path.join(path, file), 'wb'):
    pass
```

pathlib_example.py

```python
from pathlib import Path

# Create a new directory
folder = Path('new')
folder.mkdir(exist_ok=True)

# Create new file inside new directory
file = folder / 'new_file.txt'
file.touch()
```

Khuyen Tran
khuyentran1401

Add Progress Bar to Your List Comprehension



Khuyen Tran
khuyentran1401

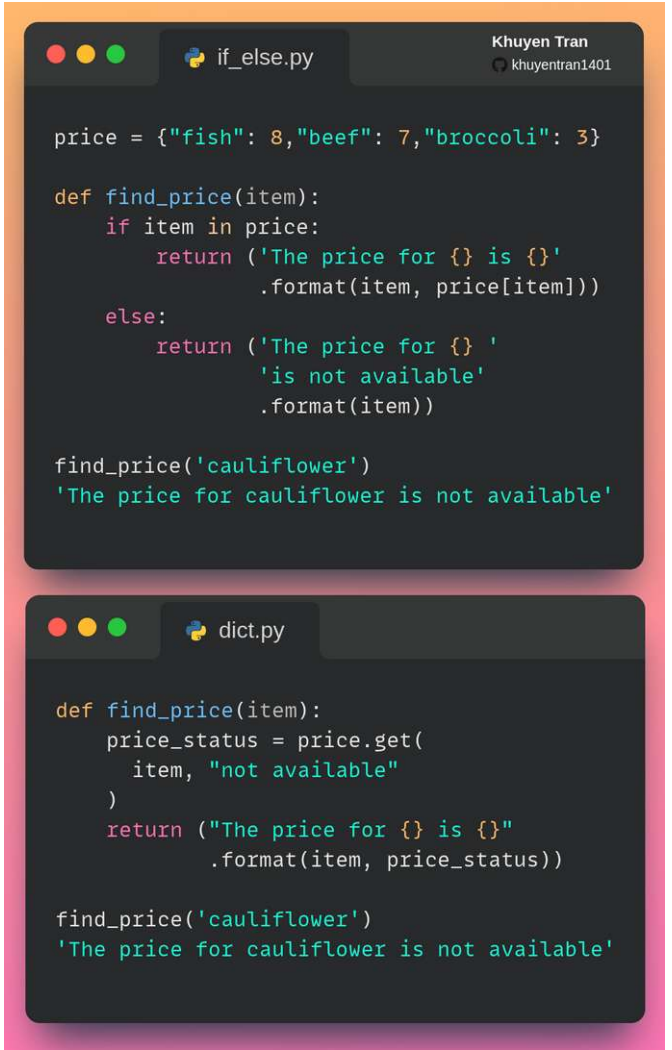
Control the Number of Printed Decimals with f-String

```python
num = 2.3123

print(f'{num:.1f}') # Limit to 1 decimal
2.3

print(f'{num:.2f}') # Limit to 2 decimals
2.31
```

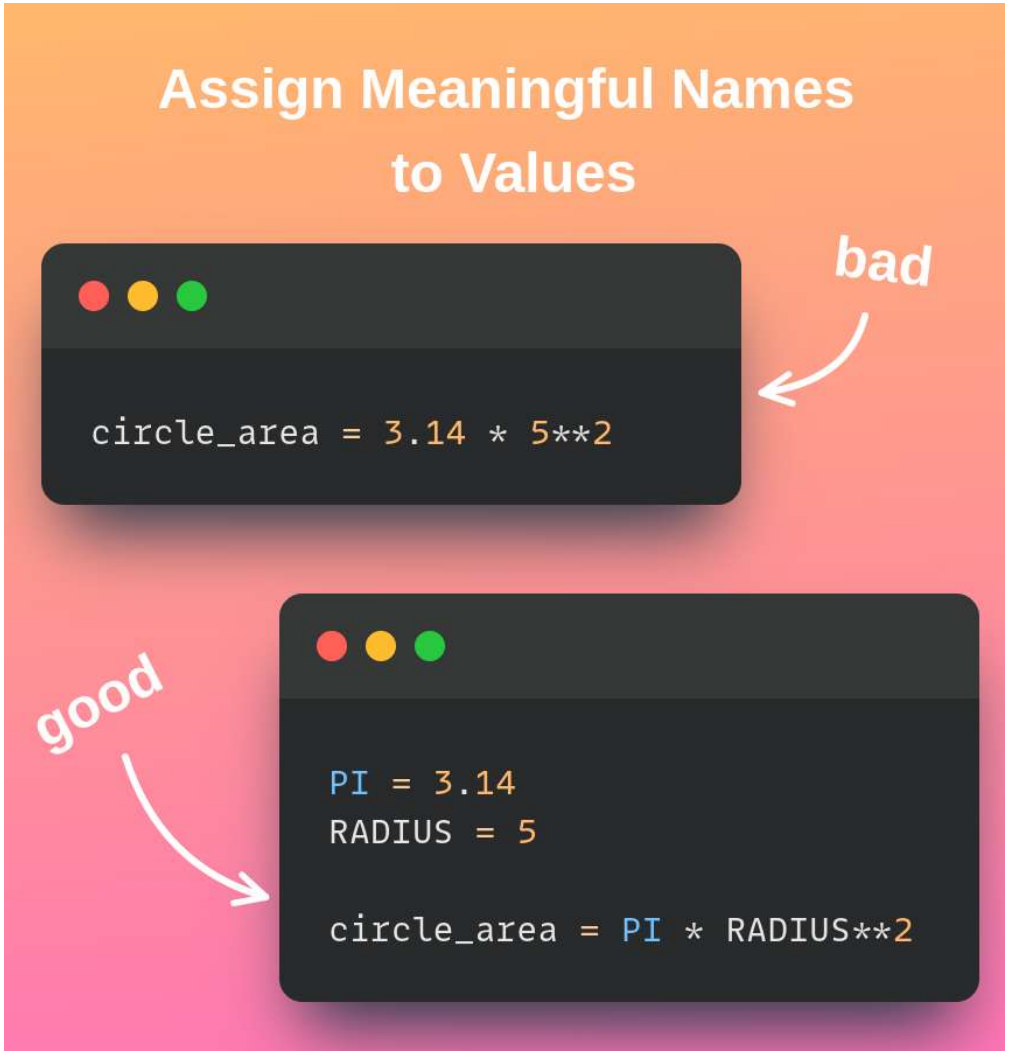
Khuyen Tran

khuyentran1401

Dictionary as an Alternative to If-Else



Assign Names to Values



Multiline Strings

parathenses.py

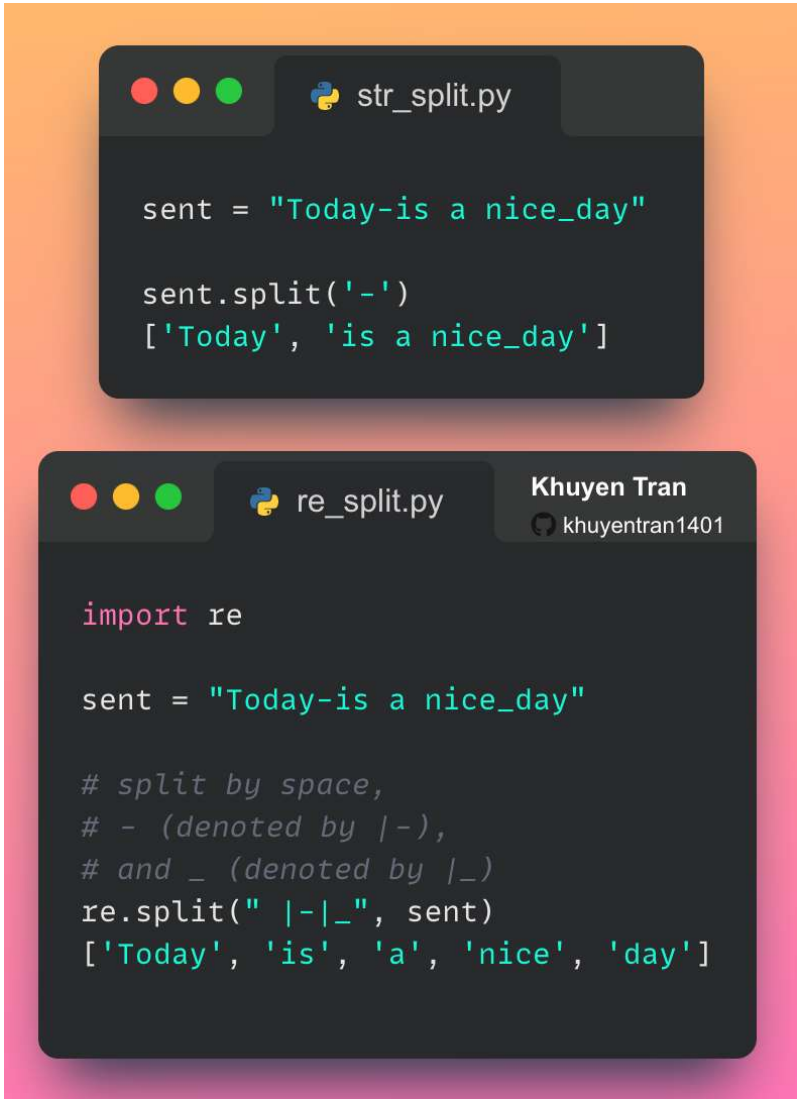
```python
text1 = (
    "This is a very "
    "long sentence "
    "that is made up."
)
text1
"""
This is a very long sentence that
is made up.
"""
```

backslash.py

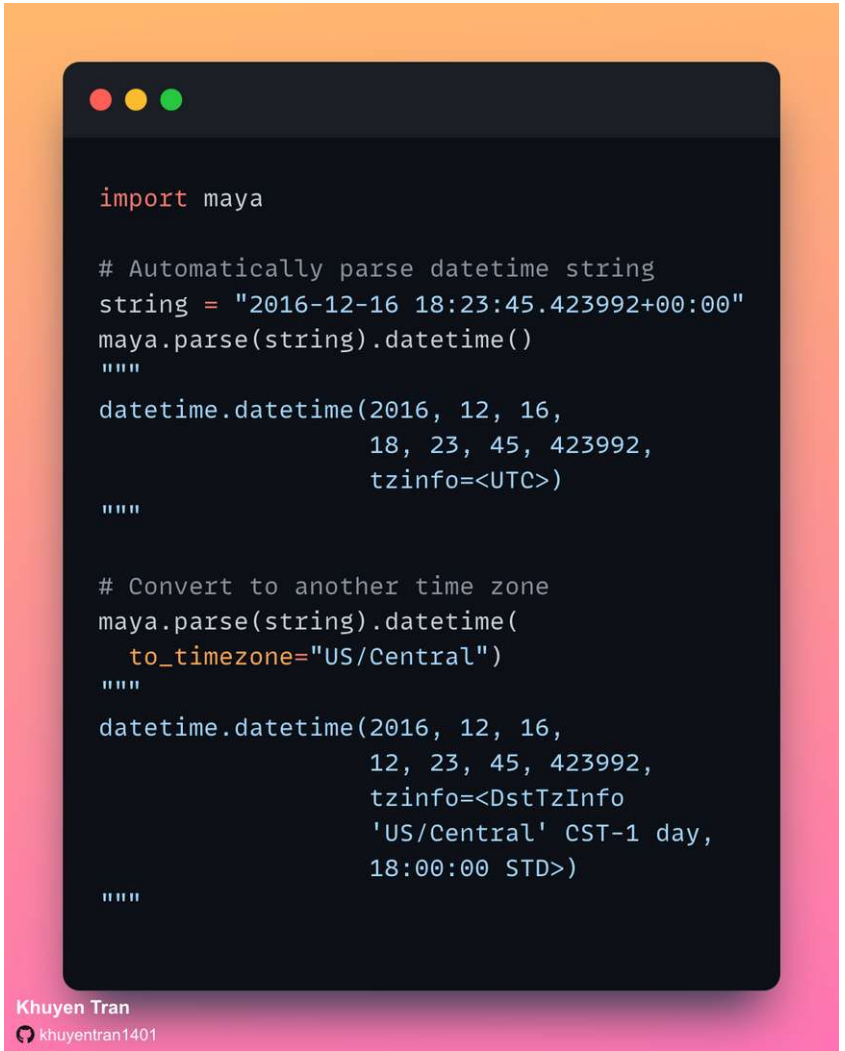
```python
text2 = "This is a very "\
    "long sentence "\
    "that is made up."
text2
"""
This is a very long sentence that
is made up.
"""
```

Khuyen Tran
khuyentran1401

Split a String by Multiple Characters



Maya: Convert the string to datetime automatically



If you want to convert a string type to a datetime type, the common way is to use strptime(date_string, format). But it is quite inconvenient to specify the structure of your datetime string, such as %Y-%m-%d %H:%M:%S.

To convert the string to datetime automatically, use maya. You just need to parse the string, and maya will figure out the structure of your string.

Expand an Equation Using Python

```python
from sympy import symbols, expands
x, y = symbols("x y")
expr = x * (3 * x + y)
expand(expr)
```

$$x(3x + y) = 3x^2 + xy$$

Khuyen Tran

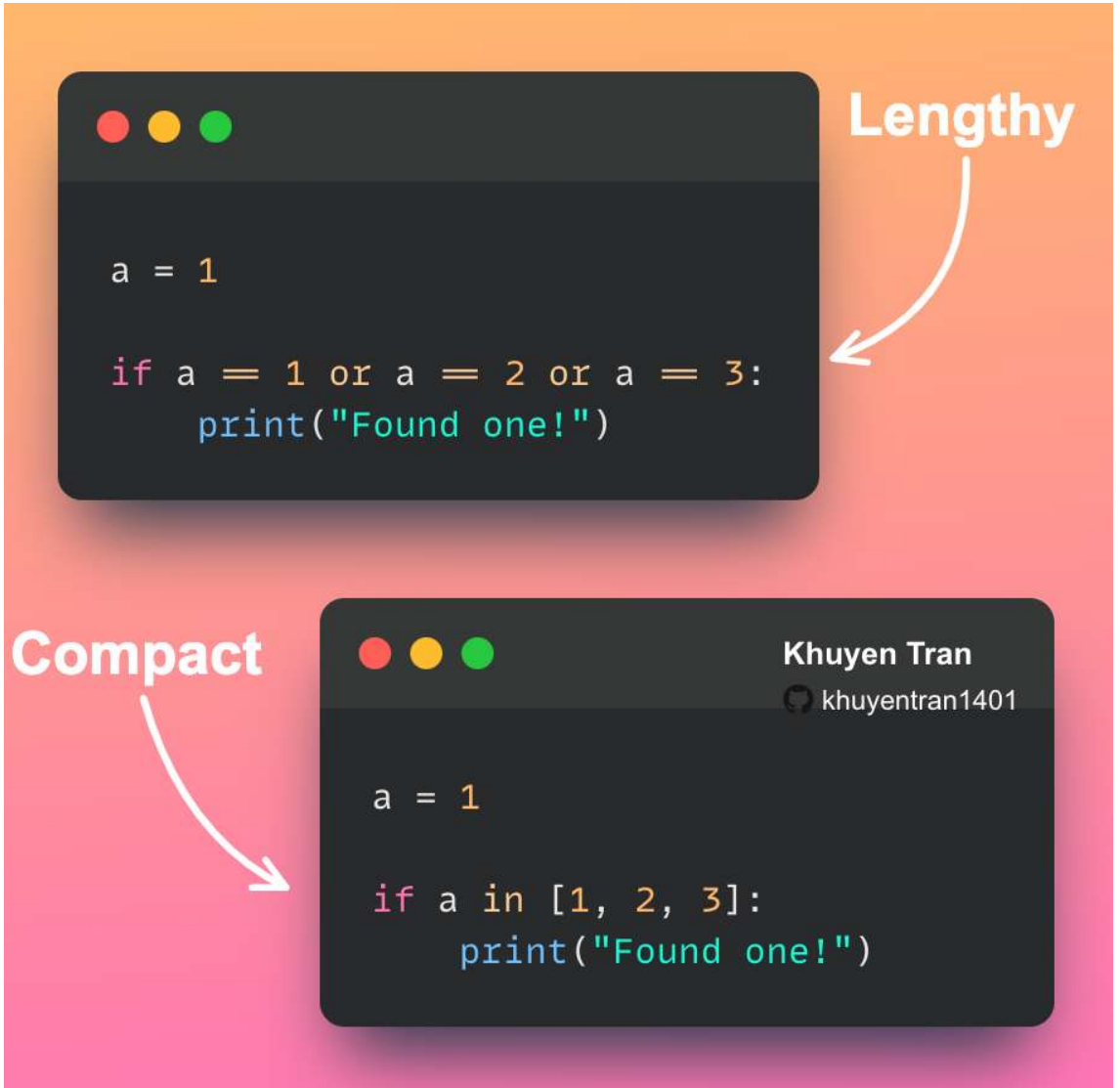
khuyentran1401

Get the Parent of the Current Path with pathlib

```python
from pathlib import Path

path = Path.cwd()

print(f'Current path: {path}')
print(f'Parent: {path.parent}')
print(f'Grandparent: {path.parent.parent}')

"""
Current path: /Users/khuyen/book/Chapter2
Parent: /Users/khuyen/book
Grandparent: /Users/khuyen
"""
```

Khuyen Tran
khuyentran1401

Don’t Use Multiple OR Operators. Use in Instead



Generate a Tree View with rich

Pad a String With Zero Using f-String

```
for hour in range(8, 12):
    print(f'It is {hour:02} AM! Wake up!')
"""
It is 08 AM! Wake up!
It is 09 AM! Wake up!
It is 10 AM! Wake up!
It is 11 AM! Wake up!
"""
```

khuyentran1401

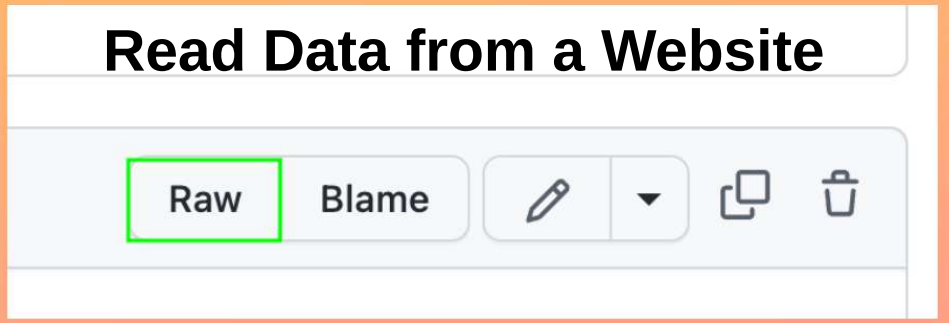
gdown: Download a File from Google Drive in Python

```
import gdown

# Format of url:
# https://drive.google.com/uc?id=YOURFILEID
url = "https://drive.google.com/uc?id=\
       1jI1cmxqnwsmC\-vbl8dNY6b4aNBtBbKy3"

output = "Twitter.zip"

gdown.download(url, output)
```

khuyentran1401

Use Comparison and Arithmetic Operators on Dates in Python

Read Data from a Website

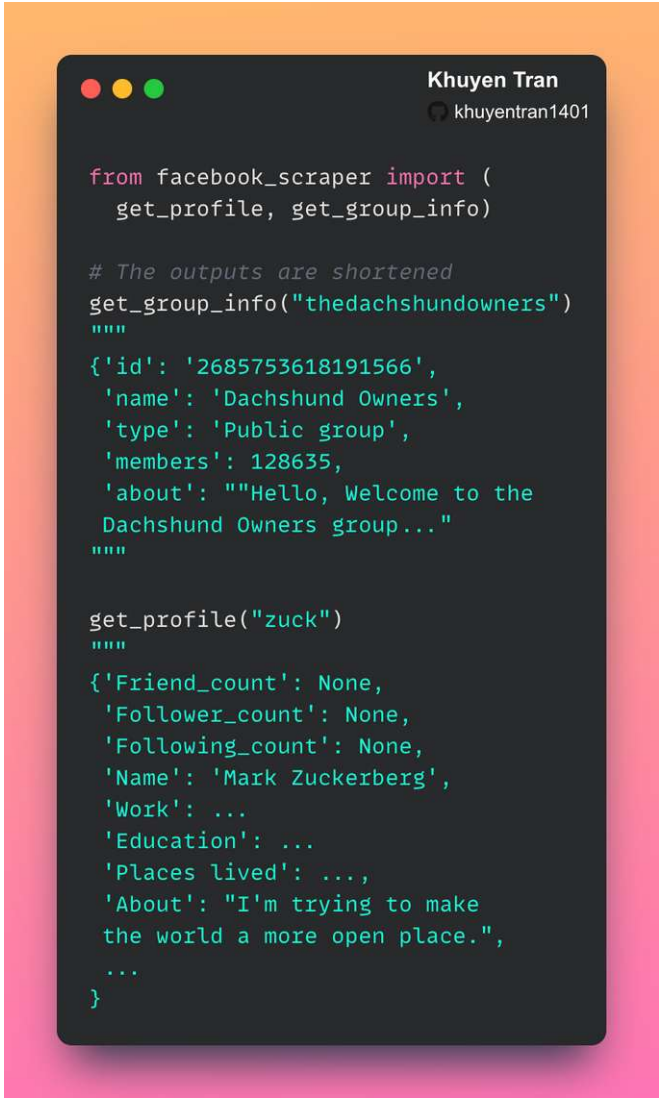


Khuyen Tran
khuyentran1401

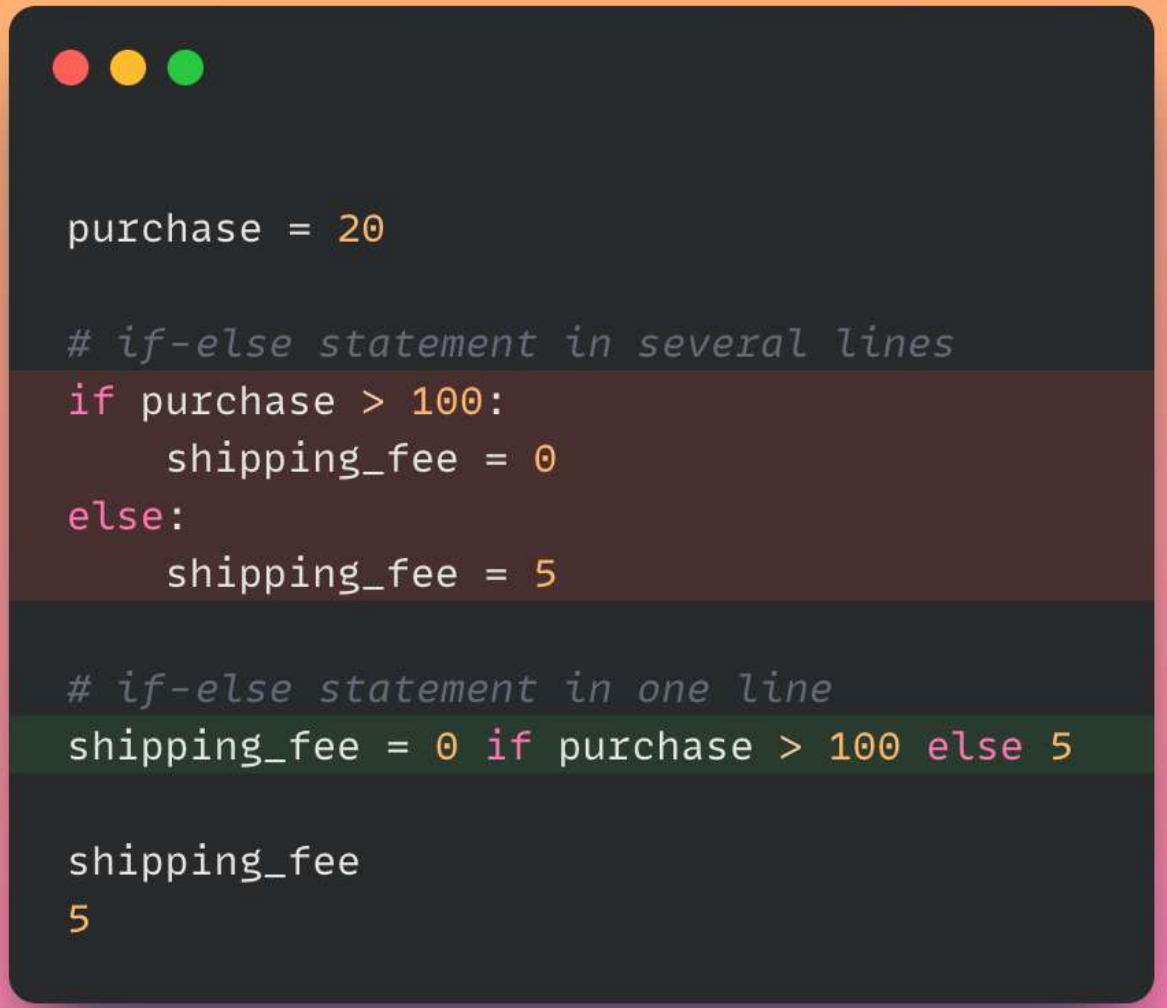
```
import pandas as pd

df = pd.read_csv("https://raw.githubusercontent."
"com/mwaskom/seaborn-data/master/exercise.csv")

df.head(3)
"""
   id     diet  pulse    time  kind
0   1  low fat     85   1 min  rest
1   1  low fat     85  15 min  rest
2   1  low fat     88  30 min  rest
"""
```

Dictdiffer: Find the Differences Between Two Dictionaries

Scrape Facebook Public Pages Without an API Key



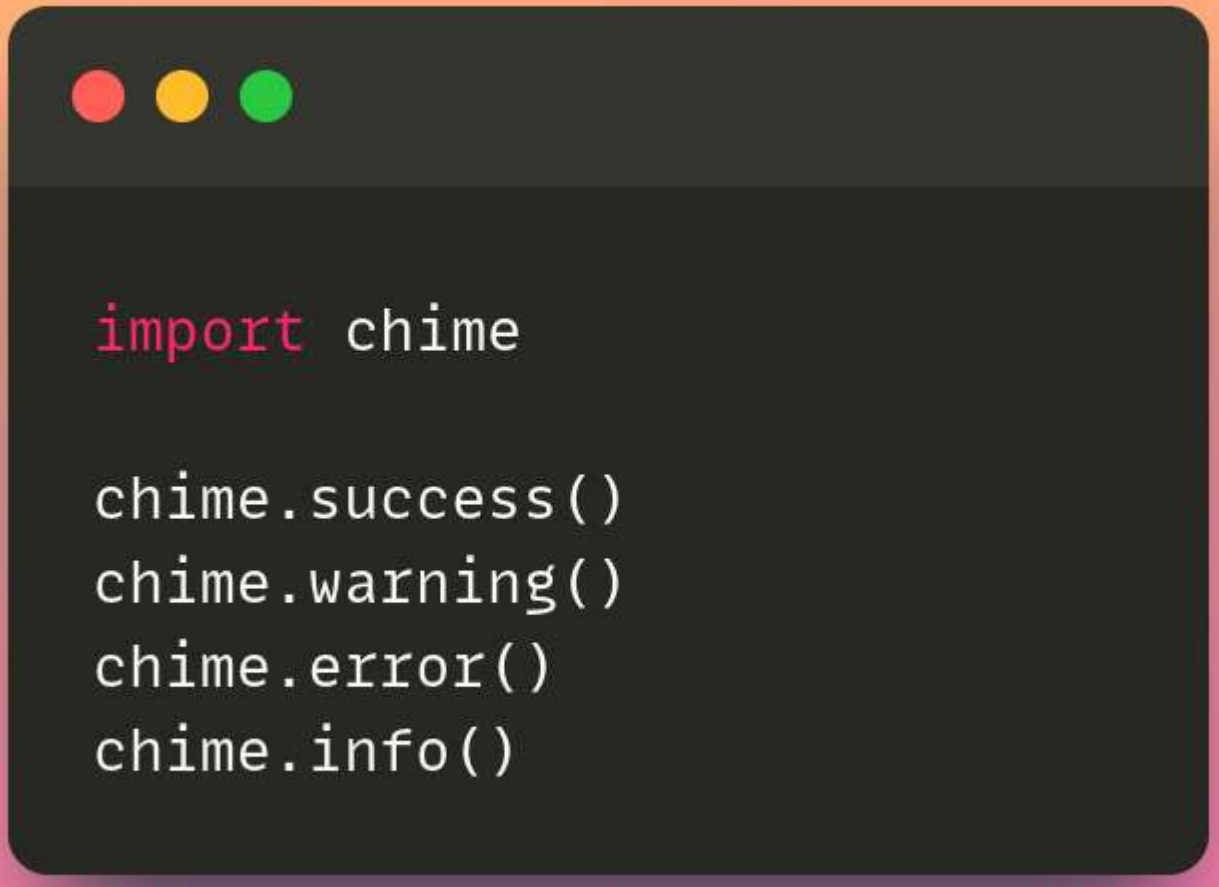
Condense an If-Else Statement into One Line



```
purchase = 20

# if-else statement in several lines
if purchase > 100:
    shipping_fee = 0
else:
    shipping_fee = 5

# if-else statement in one line
shipping_fee = 0 if purchase > 100 else 5

shipping_fee
5
```

khuyentran1401

Effortlessly Create Sound Notifications in Python



Khuyen Tran

khuyentran1401

Python Switch Statement

< Python 3.10

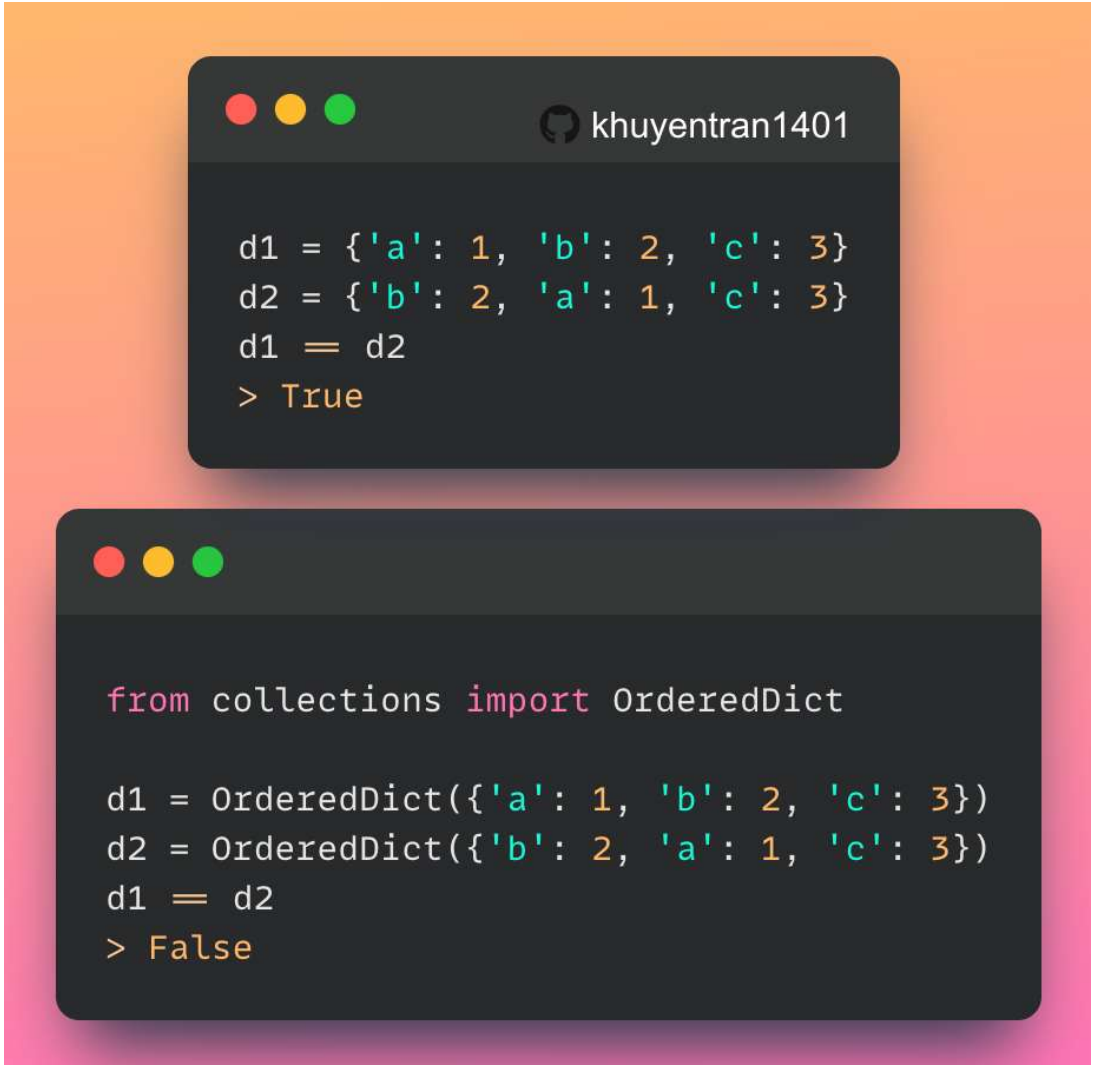
```python
def get_price(food: str):
    if food == "apple":
        return 4
    elif food == "orange":
        return 3
    elif food == "grape":
        return 5
    else:
        return "Unknown"
```

```python
def get_price(food: str):
  match food:
    # if food == "apple"
    case "apple":
      return 4
    case "orange":
      return 3
    case "grape":
      return 5
    case _:  # else
      return "Unknown"
```

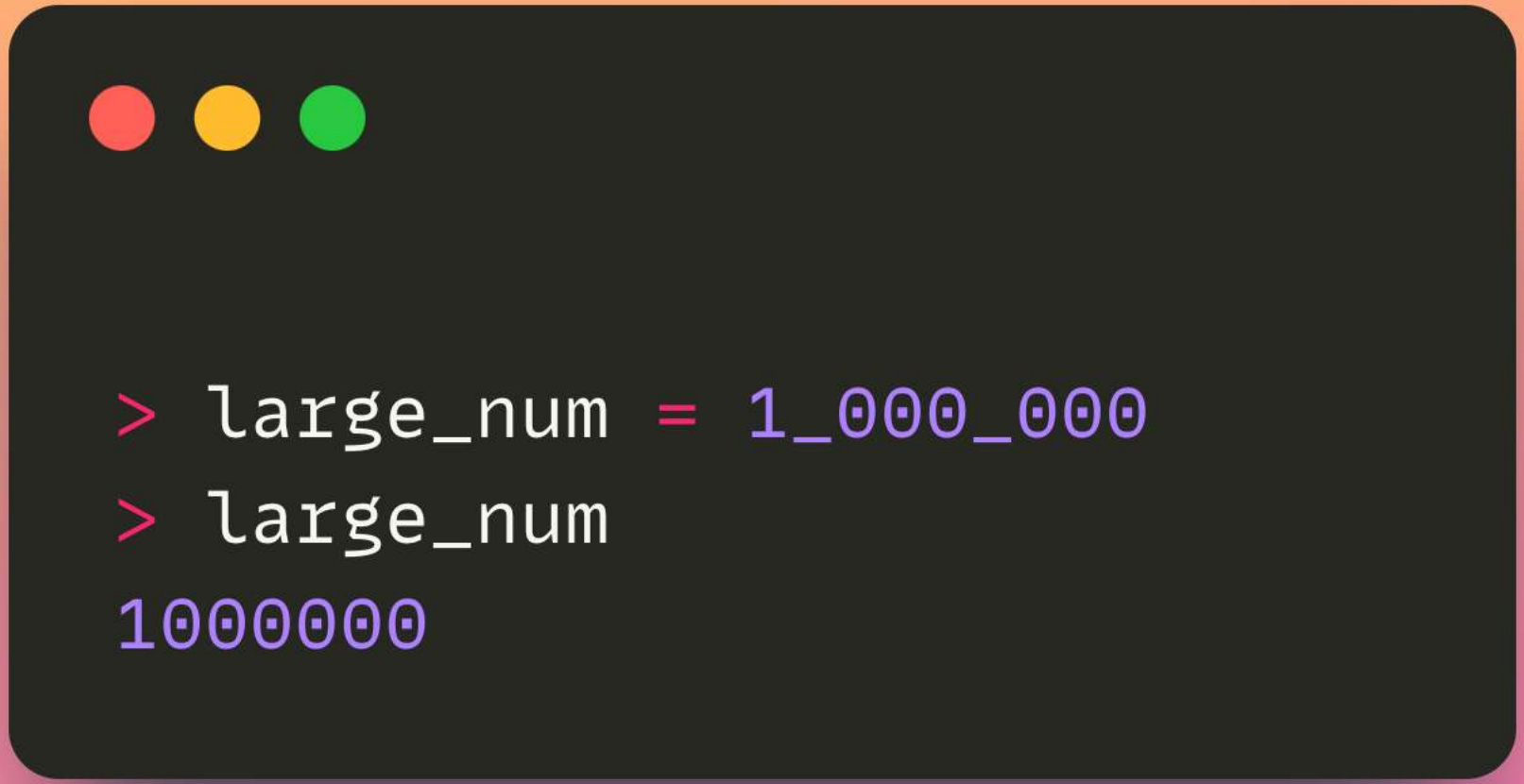
>= Python 3.10

khuyentran1401

OrderedDict: Create an Ordered Python Dictionary



Use Underscores to Format Large Numbers in Python



khuyentran1401

heartrate — Visualize the Execution of a Python Program in Real-Time