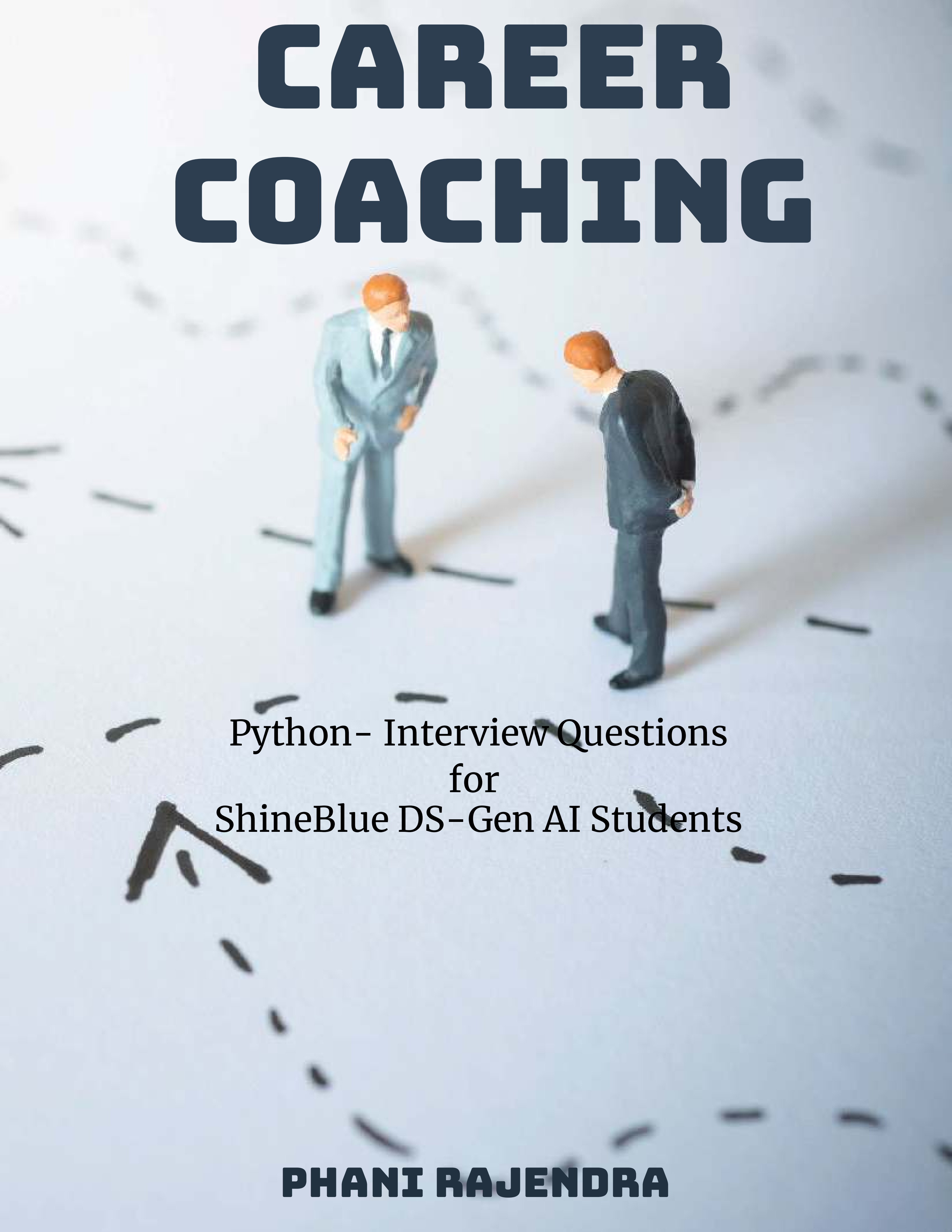
CAREER COACHING



Python- Interview Questions
for
ShineBlue DS-Gen AI Students

PHANI RAJENDRA

Python Interview Questions

1. What is Python ? Were do we use it ?

Python is a widely-used general-purpose, high-level programming language. Python is mainly used for the following

1. Web Development

• Backend development using **Django** and **Flask**

• API development using **FastAPI** and **Flask-RESTful**

• Web scraping using **Beautiful Soup** and **Scrapy**

2. Data Science & Analytics

• Data analysis using **pandas** and **NumPy**

• Data visualization with **Matplotlib** and **Seaborn**

• Statistical analysis using **SciPy**

3. Machine Learning & AI

• ML model development using **Scikit-learn**

• Deep learning with **TensorFlow**, **PyTorch**, and **Keras**

• Natural Language Processing (NLP) with **spaCy** and **NLTK**

• Computer vision using **OpenCV**

4. Automation & Scripting

• Task automation with **PyAutoGUI** and **Selenium**

• Web automation using **Selenium**

• File handling automation with **os** and **shutil**

5. Cybersecurity & Ethical Hacking

• Network security using **Scapy**

• Penetration testing with **Metasploit (Python scripts)**

• Reverse engineering and malware analysis

6. Embedded Systems & IoT

• Microcontroller programming with **MicroPython**

• IoT applications using **Raspberry Pi**

7. Game Development

• Developing games with **Pygame**

• AI-based gaming with **Unity (Python scripts)**

8. Finance & FinTech

• Stock market analysis using **pandas**

• Cryptocurrency trading bots with **ccxt**

• Risk assessment and fraud detection

9. DevOps & Cloud Computing

• Cloud automation using **AWS SDK (boto3)**

• Continuous integration/deployment (CI/CD) with **Jenkins & Ansible**

• Kubernetes automation with **k8s-client**

10. Big Data & Distributed Computing

• Handling large-scale data with **Dask**

• Parallel computing with **Apache Spark (PySpark)**

11. Robotics & AI-powered Systems

• Controlling robots using **ROS (Robot Operating System)**

• Autonomous vehicles with **Deep Learning & OpenCV**

12. Education & Research

• Teaching programming concepts in schools and universities

• Research in bioinformatics, physics, and mathematics

1. Is Python a complier or an interpreter?

Actually, Python is a partially compiled language and partially interpreted language. The compilation part is done first when we execute our code and this will generate byte code internally.

This byte code gets converted by the Python virtual machine(p.v.m) according to the underlying platform(machine and operating system)

The following is the process that happens in the while processing a program internally.

1. **Execution Process**:

• Python code is first **compiled into an intermediate bytecode (.pyc files)**.

• This bytecode is then **interpreted by the Python Virtual Machine (PVM)** line by line.

• This process makes Python highly portable across different operating systems.

2. **No Explicit Compilation Step**:

• Unlike C, C++, or Java, Python does not require manual compilation before execution.

• The Python interpreter automatically converts code into machine-executable instructions at runtime.

3. **Dynamic Typing**:

• Since Python is interpreted, it supports **dynamic typing** (variables don’t need explicit type declarations).

4. **Slower Execution Compared to Compiled Languages**:

• Because Python executes code line by line (interpreted mode), it is generally **slower than compiled languages** like C or Java.

• However, this makes it **more flexible and easy to debug**.

Can Python Be Compiled?

• While Python is primarily interpreted, there are **ways to compile it** into machine code:

• **Cython** (Compiles Python to C for performance optimization)

• **PyInstaller** (Bundles Python code into a standalone executable)

• **Nuitka** (Compiles Python into C++ and then compiles it to machine code)

• **JIT Compilation** (Just-In-Time compilation using PyPy)

So, Python is **interpreted by default**, but with tools like Cython or PyPy, it can be compiled for performance improvements.

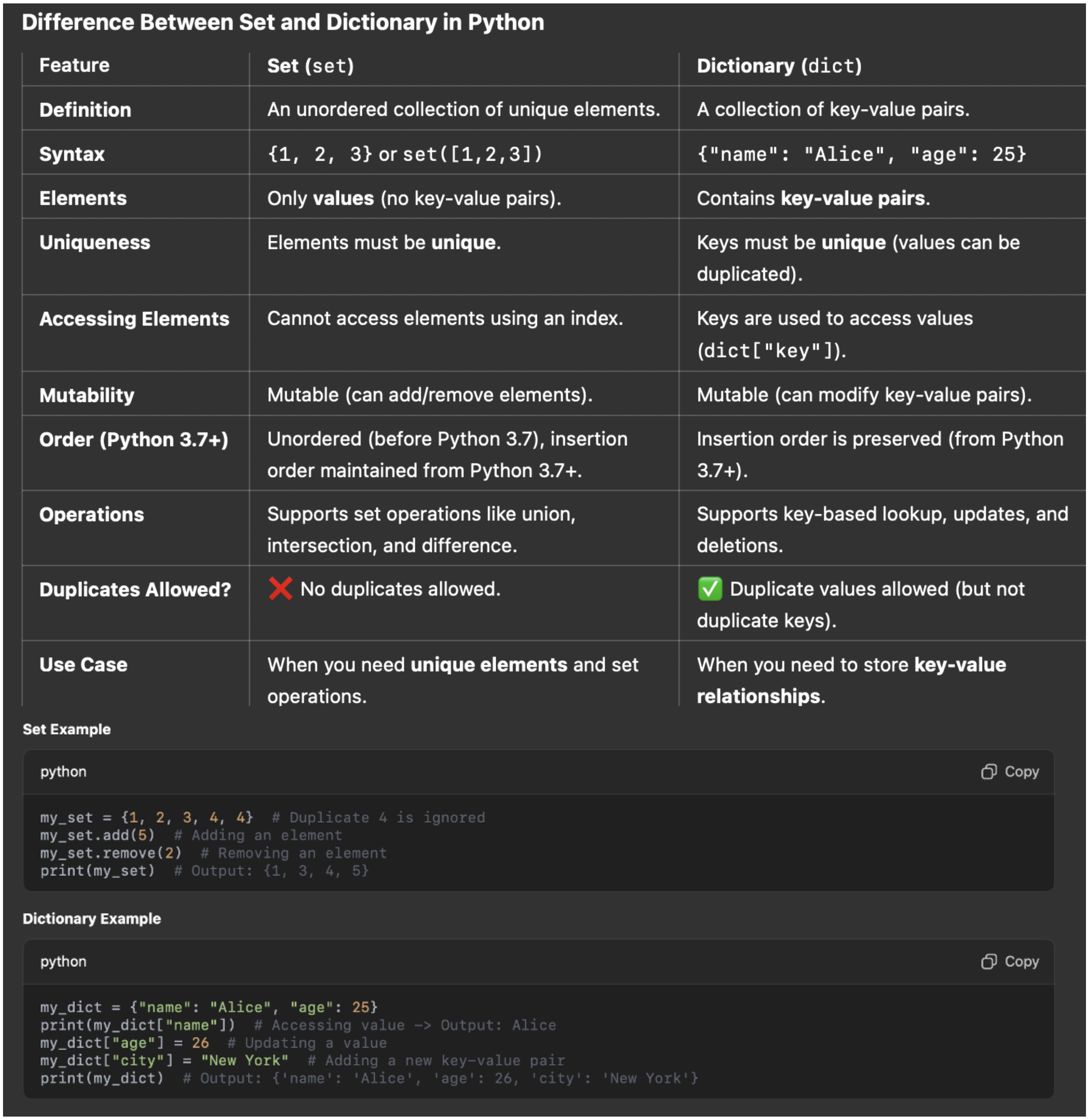
1. How do we create comments in python ?

Comments in python can be created using a ‘#’ symbol and multiline comments can be made using ‘Doc String’ or six(6) single quotes between paragraphs.

1. What are differences between dictionary and set Datatypes ?

Dictionaries and Sets datatype serve different purposes in python. Dictionary uses set of flower brackets and sets uses parenthesis

Here More detailed information about the differences between both of them



Difference Between Set and Dictionary in Python

Feature	Set (`set`)	Dictionary (`dict`)
Definition	An unordered collection of unique elements.	A collection of key-value pairs.
Syntax	`{1, 2, 3}` or `set([1,2,3])`	`{"name": "Alice", "age": 25}`
Elements	Only **values** (no key-value pairs).	Contains **key-value pairs**.
Uniqueness	Elements must be **unique**.	Keys must be **unique** (values can be duplicated).
Accessing Elements	Cannot access elements using an index.	Keys are used to access values (`dict["key"]`).
Mutability	Mutable (can add/remove elements).	Mutable (can modify key-value pairs).
Order (Python 3.7+)	Unordered (before Python 3.7), insertion order maintained from Python 3.7+.	Insertion order is preserved (from Python 3.7+).
Operations	Supports set operations like union, intersection, and difference.	Supports key-based lookup, updates, and deletions.
Duplicates Allowed?	❌ No duplicates allowed.	✅ Duplicate values allowed (but not duplicate keys).
Use Case	When you need **unique elements** and set operations.	When you need to store **key-value relationships**.

Set Example

python Copy

```python
my_set = {1, 2, 3, 4, 4}  # Duplicate 4 is ignored
my_set.add(5)  # Adding an element
my_set.remove(2)  # Removing an element
print(my_set)  # Output: {1, 3, 4, 5}
```

Dictionary Example

python Copy

```python
my_dict = {"name": "Alice", "age": 25}
print(my_dict["name"])  # Accessing value -> Output: Alice
my_dict["age"] = 26  # Updating a value
my_dict["city"] = "New York"  # Adding a new key-value pair
print(my_dict)  # Output: {'name': 'Alice', 'age': 26, 'city': 'New York'}
```

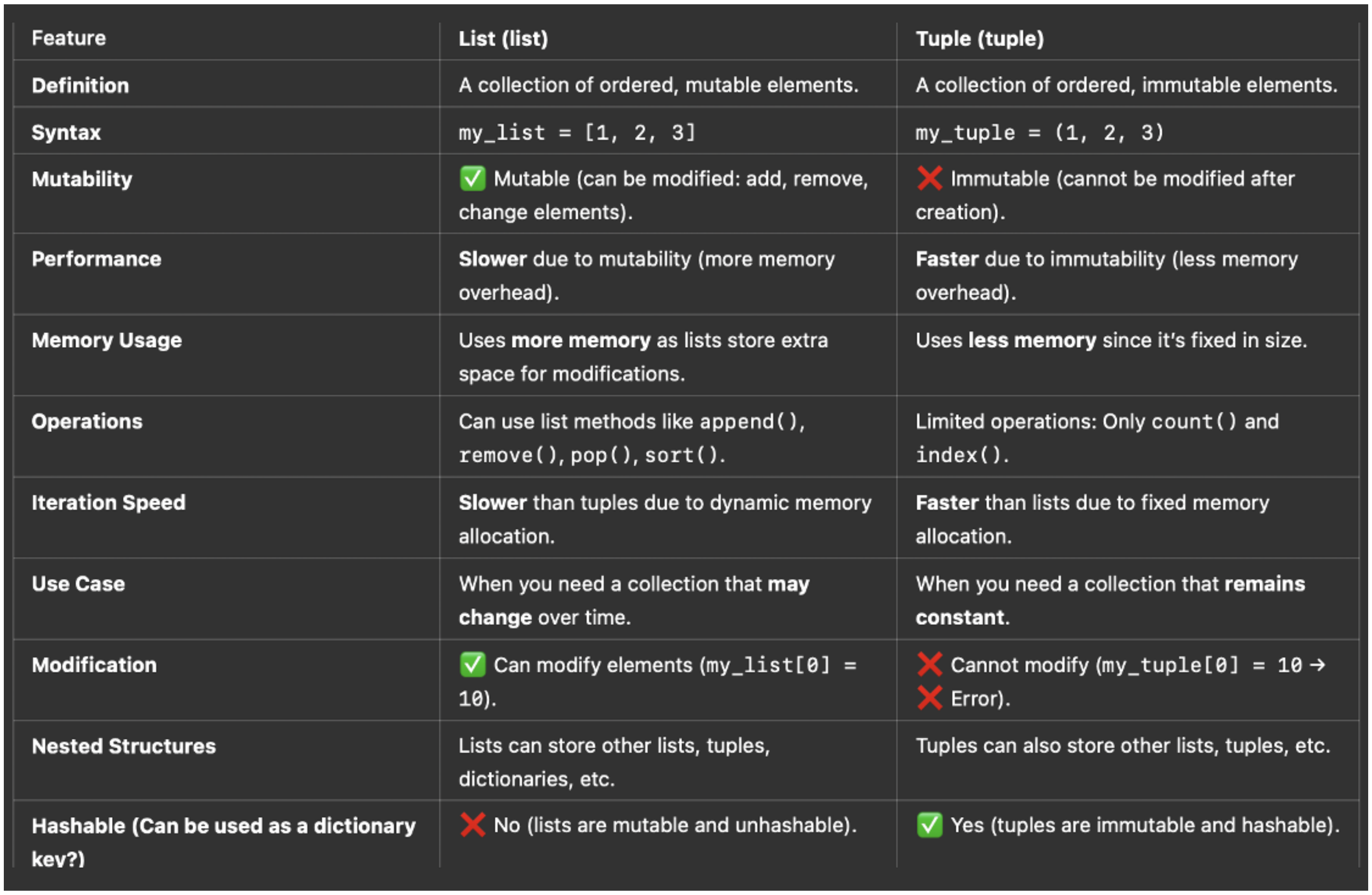
1. How is exceptions are handled in python ?

Python basically uses there types of exception handling methods known as i)try block ii) except block III) finally block .

1. What are the main points to remember while using Python ?

We should remember that Python is case sensitive and space sensitive and indentation matters a lot in Python.

1. What are differences between Python List DataType and Tuple DataTypes ?



Feature	List (list)	Tuple (tuple)
Definition	A collection of ordered, mutable elements.	A collection of ordered, immutable elements.
Syntax	`my_list = [1, 2, 3]`	`my_tuple = (1, 2, 3)`
Mutability	✅ Mutable (can be modified: add, remove, change elements).	❌ Immutable (cannot be modified after creation).
Performance	**Slower** due to mutability (more memory overhead).	**Faster** due to immutability (less memory overhead).
Memory Usage	Uses **more memory** as lists store extra space for modifications.	Uses **less memory** since it's fixed in size.
Operations	Can use list methods like `append()`, `remove()`, `pop()`, `sort()`.	Limited operations: Only `count()` and `index()`.
Iteration Speed	**Slower** than tuples due to dynamic memory allocation.	**Faster** than lists due to fixed memory allocation.
Use Case	When you need a collection that **may change** over time.	When you need a collection that **remains constant**.
Modification	✅ Can modify elements (`my_list[0] = 10`).	❌ Cannot modify (`my_tuple[0] = 10` → ❌ Error).
Nested Structures	Lists can store other lists, tuples, dictionaries, etc.	Tuples can also store other lists, tuples, etc.
Hashable (Can be used as a dictionary key?)	❌ No (lists are mutable and unhashable).	✅ Yes (tuples are immutable and hashable).

8 . What is slicing in Python ?

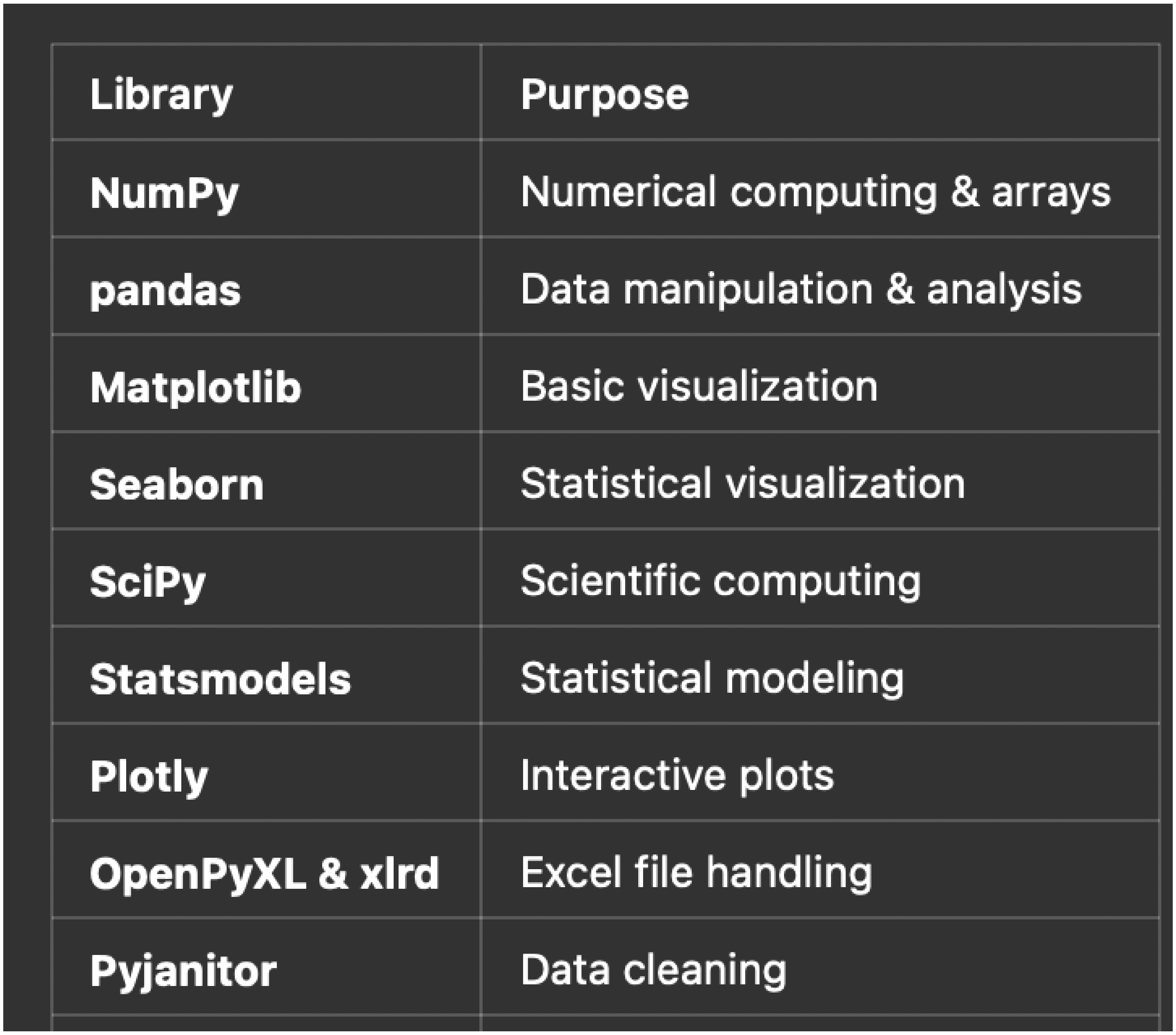
Python Slicing is a string operation for extracting a part of the string, or some part of a list. With this operator, one can specify where to start the slicing, where to end, and specify the step. List slicing returns a new list from the existing list.

9 . What is pip in python and why do we use it ?

PIP is an acronym for Python Installer Package which provides a seamless interface to install various Python modules

10. List the popular Python libraries used in Data Analysis ?

Here are list of Python Libraries used for Data Analysis.



Library	Purpose
NumPy	Numerical computing & arrays
pandas	Data manipulation & analysis
Matplotlib	Basic visualization
Seaborn	Statistical visualization
SciPy	Scientific computing
Statsmodels	Statistical modeling
Plotly	Interactive plots
OpenPyXL & xlrd	Excel file handling
Pyjanitor	Data cleaning

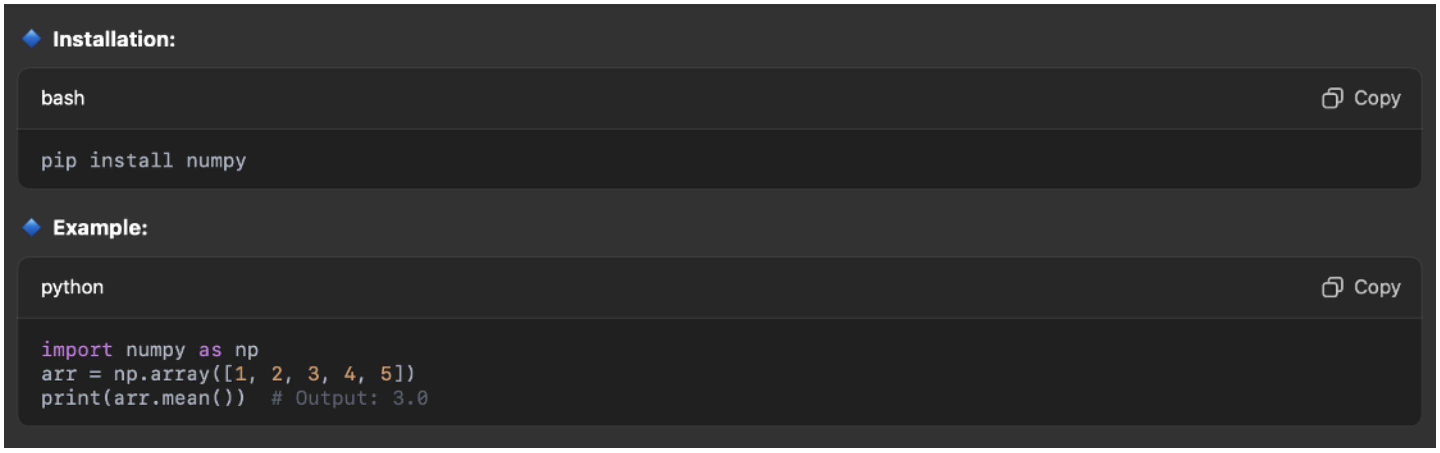
1. NumPy (Numerical Python): Purpose: Handling large, multi-dimensional arrays and matrices, along with mathematical functions.

🔹 **Key Features:**

• Fast array operations (ndarray)

• Linear algebra, Fourier transforms, and random number generation

• Element-wise operations on arrays



🔹 **Installation:**

bash

```bash
pip install numpy
```

🔹 **Example:**

python

```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr.mean())  # Output: 3.0
```

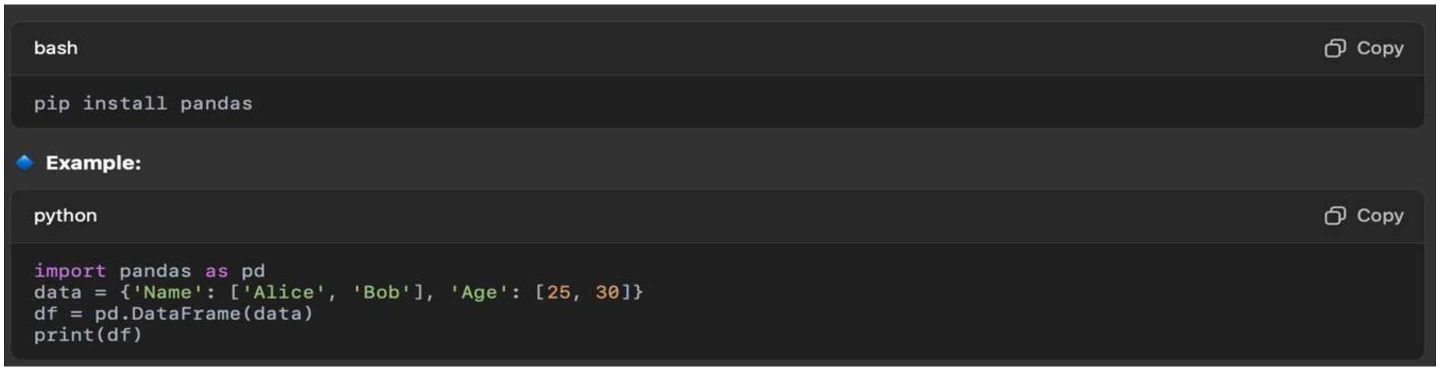
2. Pandas (Data Analysis Library): Purpose: Data manipulation, cleaning, and analysis using **DataFrame** and **Series** structures.

🔹 **Key Features:**

• Data cleaning, filtering, merging, and aggregation

• Handling missing values

• Time-series analysis



```bash
pip install pandas
```

🔹 **Example:**

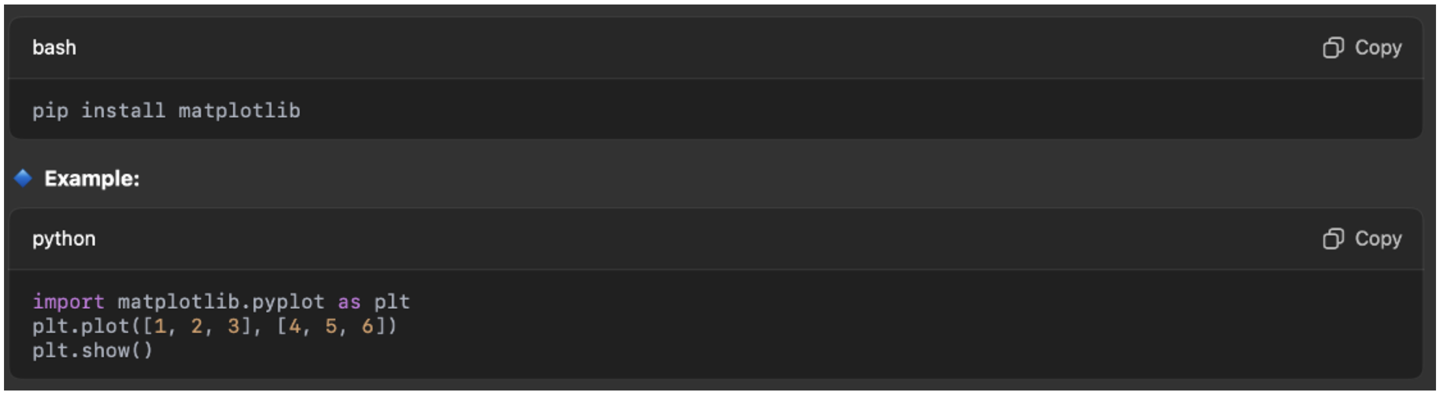
```python
import pandas as pd
data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]}
df = pd.DataFrame(data)
print(df)
```

3. Matplotlib (Data Visualization) Purpose: Creating static, animated, and interactive visualizations.

🔹 **Key Features:**

• Line charts, bar plots, histograms, scatter plots

• Customizable styles and annotations



```bash
pip install matplotlib
```

🔹 **Example:**

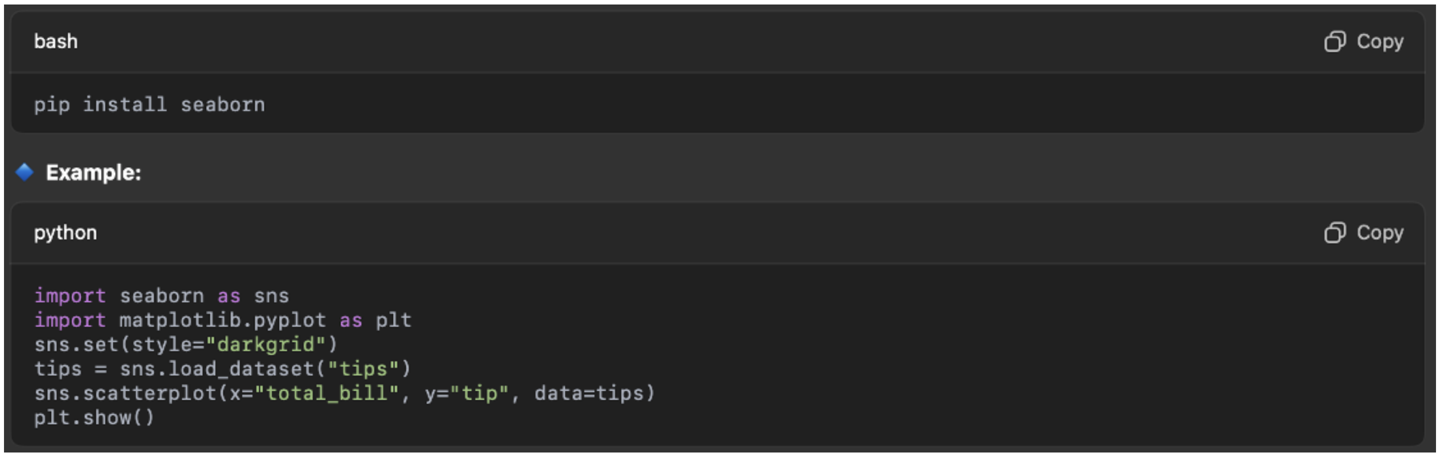
```python
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.show()
```

4. Seaborn (Statistical Data Visualization) Purpose: High-level API for beautiful statistical visualizations.

🔹 **Key Features:**

• Built-in themes for visually appealing charts

• Supports complex visualizations like violin plots, pair plots


bash Copy

```bash
pip install seaborn
```

🔹 **Example:**

python Copy

```python
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="darkgrid")
tips = sns.load_dataset("tips")
sns.scatterplot(x="total_bill", y="tip", data=tips)
plt.show()
```

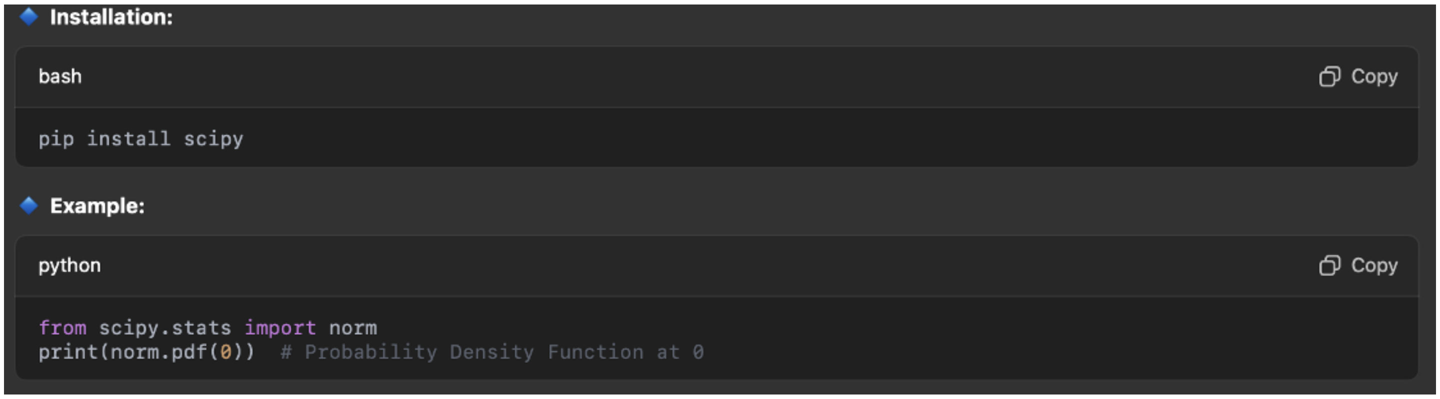
5. SciPy (Scientific Computing) Purpose: Advanced scientific computations like optimization, integration, signal processing.

🔹 **Key Features:**

• Statistical functions, linear algebra

• Optimization and interpolation

• Image processing


🔹 **Installation:**

bash Copy

```bash
pip install scipy
```

🔹 **Example:**

python Copy

```python
from scipy.stats import norm
print(norm.pdf(0))  # Probability Density Function at 0
```

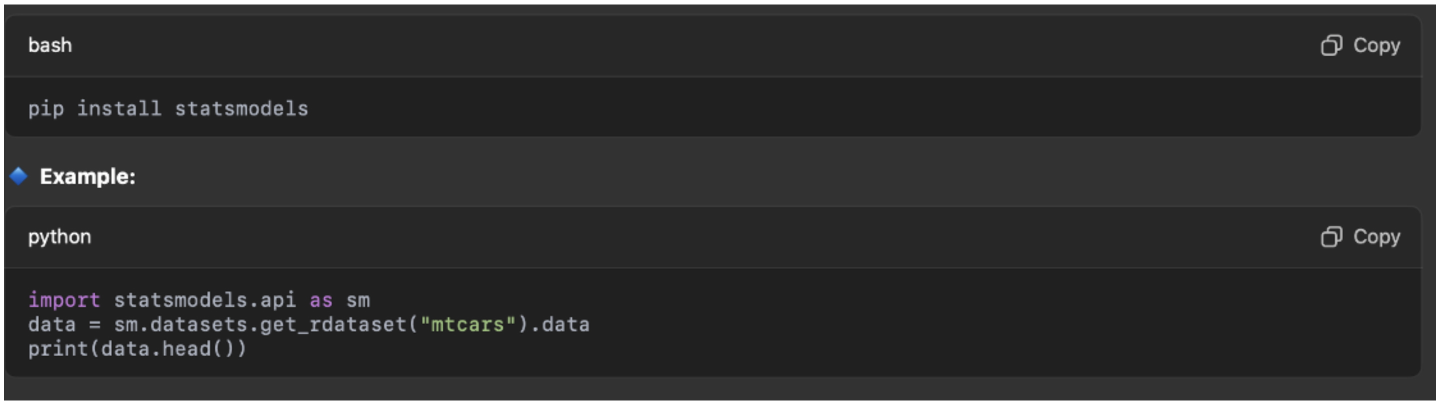
6. Statsmodels (Statistical Analysis)Purpose: Estimation of statistical models, hypothesis testing.

🔹 **Key Features:**

• Linear and logistic regression

• Time-series analysis

• Hypothesis testing



```bash
pip install statsmodels
```

🔹 **Example:**

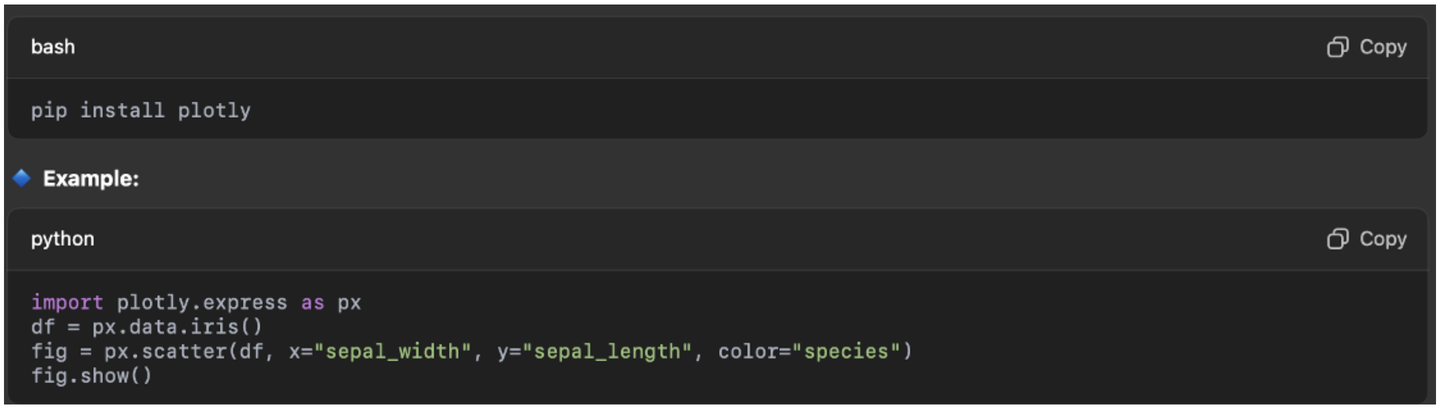
```python
import statsmodels.api as sm
data = sm.datasets.get_rdataset("mtcars").data
print(data.head())
```

7. Plotly (Interactive Visualization)Purpose: Interactive, web-based data visualization.

🔹 **Key Features:**

• Zooming, hovering, exporting charts

• Supports 3D plots, heatmaps

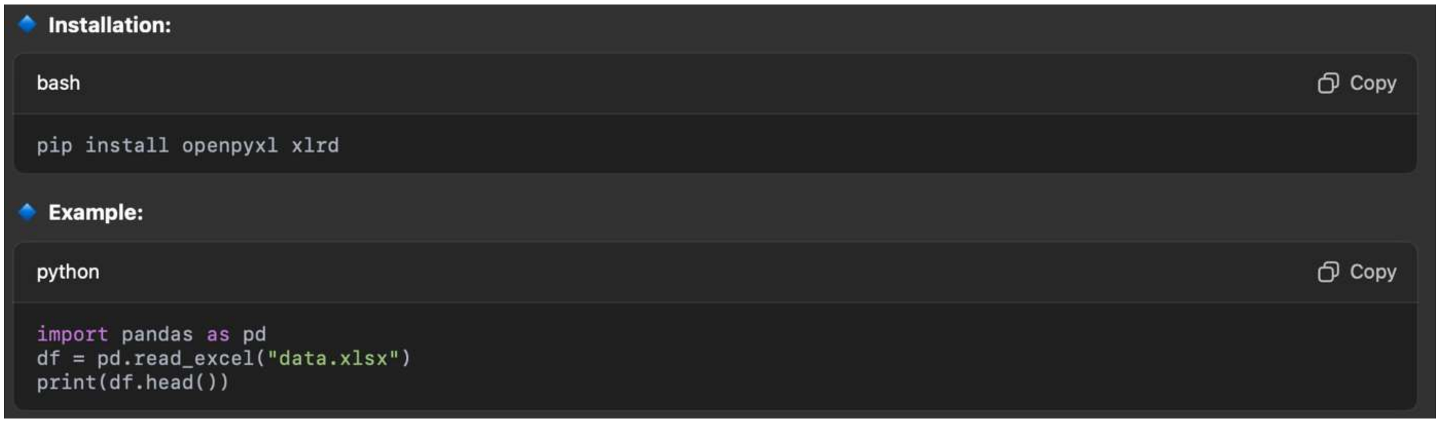


```bash
pip install plotly
```

🔹 **Example:**

```python
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species")
fig.show()
```

8. OpenPyXL & xlrd (Excel Handling) Purpose: Reading/writing Excel files.



🔹 **Installation:**

bash

```bash
pip install openpyxl xlrd
```

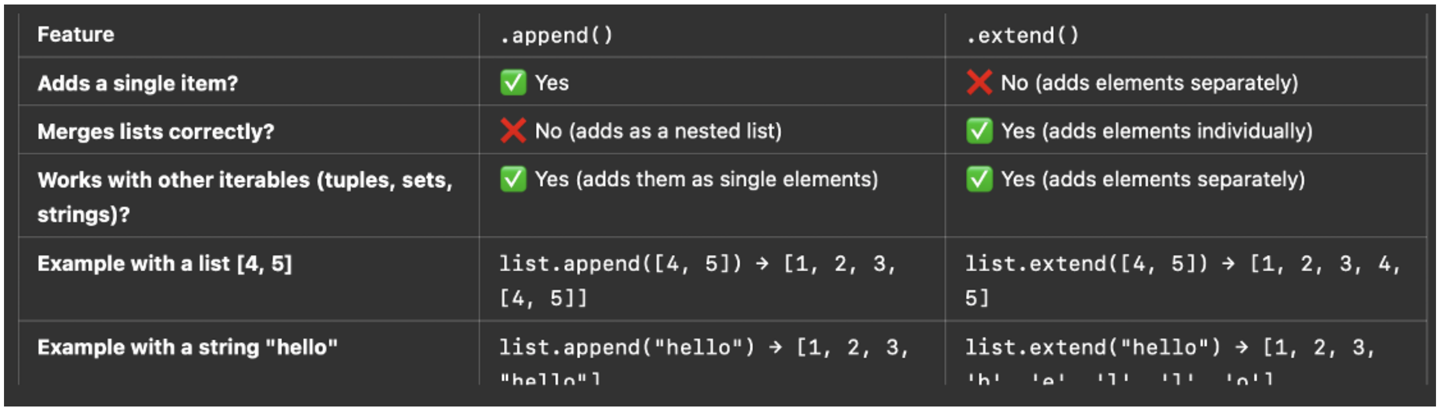
🔹 **Example:**

python

```python
import pandas as pd
df = pd.read_excel("data.xlsx")
print(df.head())
```

Explain differences between .append() and .extend() methods ?

The functionality is same but the way they operate differs. Here are some of the important differences listed as below.



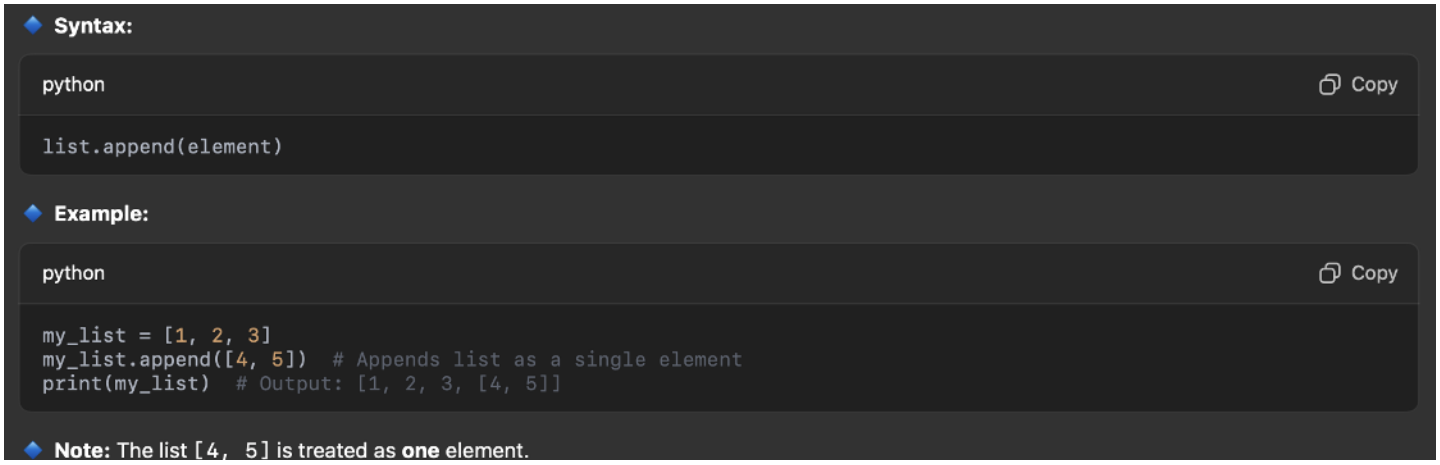
Feature	`.append()`	`.extend()`
Adds a single item?	✅ Yes	❌ No (adds elements separately)
Merges lists correctly?	❌ No (adds as a nested list)	✅ Yes (adds elements individually)
Works with other iterables (tuples, sets, strings)?	✅ Yes (adds them as single elements)	✅ Yes (adds elements separately)
Example with a list [4, 5]	`list.append([4, 5]) → [1, 2, 3, [4, 5]]`	`list.extend([4, 5]) → [1, 2, 3, 4, 5]`
Example with a string "hello"	`list.append("hello") → [1, 2, 3, "hello"]`	`list.extend("hello") → [1, 2, 3, 'h', 'e', 'l', 'l', 'o']`

.append() Method Purpose: Adds a **single element** (or an object) to the end of the list.

🔹 **Key Characteristics:**

• Takes **one argument** and **adds it as a single item**.

• If you pass a list, the **entire list is added as a single element**, **not** individual elements.



🔹 **Syntax:**

```python
list.append(element)
```

🔹 **Example:**

```python
my_list = [1, 2, 3]
my_list.append([4, 5])  # Appends list as a single element
print(my_list)  # Output: [1, 2, 3, [4, 5]]
```

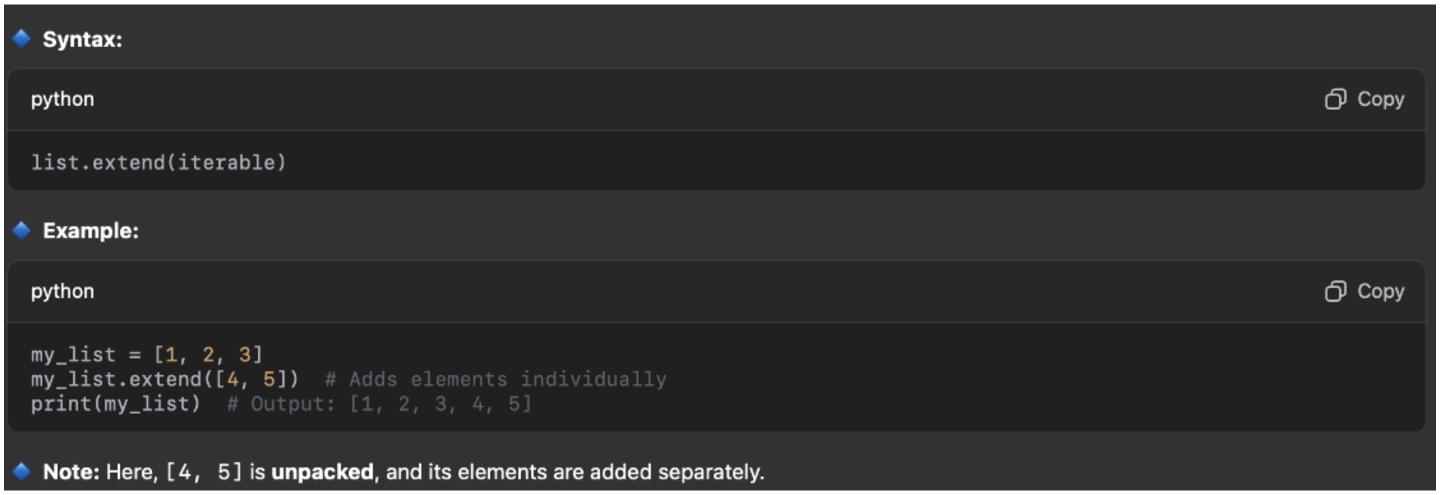
🔹 **Note:** The list `[4, 5]` is treated as **one** element.

.extend() Method -Purpose: Extends a list by **adding elements from an iterable (e.g., list, tuple, string)**.

🔹 **Key Characteristics:**

• Iterates through the argument and **adds each element individually**.

• Useful when you want to merge two lists



🔹 **Syntax:**

```python
list.extend(iterable)
```

🔹 **Example:**

```python
my_list = [1, 2, 3]
my_list.extend([4, 5])  # Adds elements individually
print(my_list)  # Output: [1, 2, 3, 4, 5]
```

🔹 **Note:** Here, `[4, 5]` is **unpacked**, and its elements are added separately.

When to Use .append() vs .extend()

✅ **Use .append()** when:

• You want to add a **single object** (like a number, list, string, dictionary) without unpacking.

• You need to maintain the original structure of the added element.

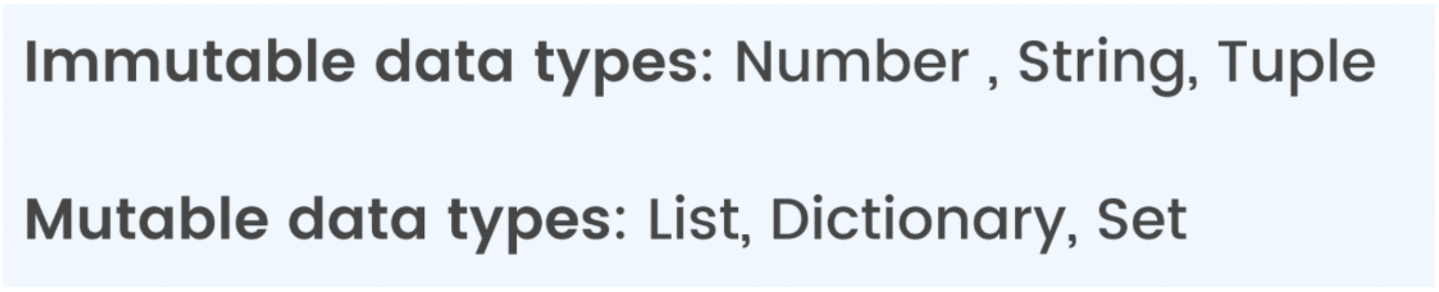
✅ **Use .extend()** when:

• You need to **add multiple elements from an iterable** individually.

• You want to **merge** two lists instead of nesting one inside the other.

What are the common datatypes of Python ?

The following are the datatypes in Python



Immutable data types: Number , String, Tuple

Mutable data types: List, Dictionary, Set

Wish you all
the best .