Python Basics

1. Variables

Variables are named containers used to store data in memory. They act as references to objects and can hold different types of values. Python is dynamically typed, so you don’t need to declare the type explicitly.

```
name = "Alice" # String

age = 25 # Integer

height = 5.6 # Float

print(name, age, height) # Output: Alice 25 5.6
```

2. Data Types

Python has several built-in data types to represent different kinds of data. The most common ones are:

- **Integer (int)**: Whole numbers (e.g., 5, -10).
- **Float (float)**: Decimal numbers (e.g., 3.14, -0.5).
- **String (str)**: Sequence of characters (e.g., "Hello", 'Python').
- **Boolean (bool)**: Logical values (True, False).
- **NoneType (None)**: Represents the absence of a value.

```
integer_num = 42 # int

float_num = 3.14159 # float

text = "Hello, Python!" # str

is_active = True # bool

nothing = None # NoneType

print(type(integer_num), type(float_num), type(text), type(is_active), type(nothing))
\
# Output: <class 'int'> <class 'float'> <class 'str'> <class 'bool'> <class 'NoneType'>
```

3. Lists

Lists are ordered, mutable (changeable) collections of items, which can hold elements of different data types. They are defined using square brackets [ ].

- **Key Features**:
  - Can be modified (add, remove, update elements).
  - Supports indexing and slicing.
  - Can contain duplicates.

```
# Creating a list

fruits = ["apple", "banana", "cherry", 42]

# Accessing elements
print(fruits[1])
# Output: banana

# Modifying list
fruits.append("orange")
# Add item fruits[2] = "grape"
# Update item
print(fruits)
# Output: ['apple', 'banana', 'grape', 42, 'orange']

# List slicing
print(fruits[1:3])
# Output: ['banana', 'grape']
```

4. Dictionaries

Dictionaries are ordered, mutable collections of key-value pairs. They are defined using curly braces {} and use keys to access values.

- **Key Features**:
  - Keys must be unique and immutable (e.g., strings, numbers, tuples).
  - Values can be of any data type.
  - Fast lookup by key.

```
# Creating a dictionary
person = {"name": "Alice", "age": 25, "city": "New York"}

# Accessing values
 print(person["name"])
# Output: Alice

# Adding/Updating key-value pairs
person["job"] = "Engineer"

# Add new key-value
person["age"] = 26

# Update existing value
print(person)
# Output: {'name': 'Alice', 'age': 26, 'city': 'New York', 'job': 'Engineer'}
```

```
# Accessing keys and values
print(person.keys())
# Output: dict_keys(['name', 'age', 'city', 'job'])

print(person.values())
# Output: dict_values(['Alice', 26, 'New York', 'Engineer'])
```

5. Tuples

Tuples are ordered, immutable collections of items, defined using parentheses (). Once created, their elements cannot be changed.

- **Key Features**:
  - Immutable (cannot modify after creation).
  - Faster than lists due to immutability.
  - Can contain duplicates.

```
# Creating a tuple
coordinates = (10, 20, 30)

# Accessing elements
print(coordinates[1])
# Output: 20

# Slicing
print(coordinates[0:2])
# Output: (10, 20)
```

```
# Tuples are immutable # coordinates[1] = 50
# This will raise a TypeError

# Tuple with mixed types
mixed = (1, "hello", 3.14)
print(mixed)
# Output: (1, 'hello', 3.14)
```

6. Sets

Sets are unordered, mutable collections of unique elements, defined using curly braces {} or the set() function.

- **Key Features**:
  - No duplicates allowed.
  - Unordered, so no indexing or slicing.
  - Useful for mathematical operations like union, intersection, and difference.

```
# Creating a set
numbers = {1, 2, 3, 4, 4}

# Duplicates are removed
print(numbers)

# Output: {1, 2, 3, 4}
```

```python
# Adding elements
numbers.add(5)
print(numbers)
# Output: {1, 2, 3, 4, 5}
# Set operations
set_a = {1, 2, 3}
set_b = {3, 4, 5}
print(set_a.union(set_b))
# Output: {1, 2, 3, 4, 5}
print(set_a.intersection(set_b))
# Output: {3}
print(set_a.difference(set_b))
# Output: {1, 2}
```

What is Streamlit?

- **Definition**: Streamlit is an open-source Python framework for building web apps by writing Python scripts. It turns data scripts into shareable web applications with minimal effort, focusing on simplicity and interactivity.
- **Use Cases**:
  - Data dashboards (e.g., visualizing sales data).
  - Machine learning model demos (e.g., interactive model predictions).
  - Prototyping tools for data analysis or chatbots.
  - Sharing Python-based projects (e.g., displaying lists or dictionaries from your earlier questions).
- **Key Advantage**: No need to write HTML, CSS, or JavaScript—Streamlit handles the frontend, letting you focus on Python logic.

Key Features of Streamlit

1. **Simple Syntax**: Write Python code, and Streamlit renders it as a web app with widgets (buttons, sliders, text inputs, etc.).
2. **Reactive Execution**: The app updates automatically when user inputs change, without needing manual refresh.
3. **Widgets**: Interactive components like buttons, sliders, file uploaders, and text inputs for user interaction.
4. **Data Visualization**: Seamless integration with libraries like Matplotlib, Plotly, and Altair for charts and graphs.

Key Streamlit Commands

Here’s a quick reference for common Streamlit functions:

- st.title("Text"): Add a title.
- st.write("Text"): Display text or objects.
- st.button("Label"): Create a clickable button.
- st.text_input("Label"): Create a text input field.
- st.number_input("Label"): Create a numeric input field.
- st.pyplot(fig): Display a Matplotlib figure.
- st.table(df): Display a pandas DataFrame as a table.