By Awais Manzoor

Python Basics Notes

Data Analyst

Python Basics: Comprehensive Notes

What is Python?

Python is a high-level, interpreted, and general-purpose programming language. It emphasizes code readability with significant indentation.

Key Features of Python:

- **Easy to Learn and Use:** Python has a simple syntax similar to English, making it beginner-friendly.
- **Interpreted Language:** Code is executed line-by-line, enabling easy debugging.
- **Versatile:** Used in web development, data analysis, artificial intelligence, machine learning, automation, and more.
- **Extensive Libraries:** A vast collection of libraries like NumPy, Pandas, Matplotlib, etc., extend its capabilities.

Why Use Python?

- **Cross-platform:** Runs on various operating systems like Windows, macOS, and Linux.
- **Productivity:** High-level language features make development faster.
- **Community Support:** Large and active community ensures help is readily available.
- **Integration:** Easily integrates with C, C++, COM, and .NET components.

Data Types in Python

Python provides built-in data types for various purposes:

1. **Numeric Types:**
    - int — Integer values.
    - float — Decimal values.
    - complex — Complex numbers.
2. **Text Type:**
    - str — Strings (sequence of characters).

3. **Sequence Types:**
    - list — Ordered, mutable collection.
    - tuple — Ordered, immutable collection.
    - range — Sequence of numbers.
4. **Mapping Type:**
    - dict — Key-value pairs.
5. **Set Types:**
    - set — Unordered collection of unique items.
    - frozenset — Immutable version of a set.
6. **Boolean Type:**
    - bool — Represents True or False values.
7. **None Type:**
    - NoneType — Represents the absence of a value.

Variables in Python

Variables are used to store data.

Rules for Naming Variables:

- Must start with a letter or an underscore (_).
- Cannot start with a number.
- Can contain letters, numbers, and underscores.
- Case-sensitive.

Example:

name = "Alice"

age = 25

is_student = True

Lists

Lists are ordered, mutable collections of items.

Example:

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")  # Add an item
fruits.remove("banana")  # Remove an item
```

Common Methods:

- append(), extend(), pop(), sort(), reverse()

Tuples

Tuples are ordered, immutable collections of items.

Example:

```
dimensions = (200, 50, 100)
print(dimensions[0])  # Access an element
```

Dictionary

Dictionaries store data in key-value pairs.

Example:

```
student = {"name": "Alice", "age": 25}
student["grade"] = "A"  # Add a key-value pair
print(student["name"])
```

Set

Sets are unordered collections of unique items.

Example:

```
unique_numbers = {1, 2, 3, 4}
unique_numbers.add(5)
unique_numbers.remove(3)
```

Control Flow: If-Elif-Else

Used for conditional execution.

Example:

```
x = 10
if x > 15:
    print("Greater than 15")
elif x == 10:
    print("Equal to 10")
else:
    print("Less than 15")
```

Loops

For Loop

Used to iterate over sequences.

```
for i in range(5):
    print(i)
```

While Loop

Executes as long as a condition is True.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Functions

Functions are blocks of reusable code.

Syntax:

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


print(greet("Alice"))
```

Object-Oriented Programming (OOP) in Python

OOP organizes code using objects and classes.

Key Concepts:

1. **Abstraction:** Hiding complex implementation details and exposing only the necessary parts.
2. from abc import ABC, abstractmethod
3.
4. class Shape(ABC):
5. @abstractmethod
6. def area(self):
7. pass
8. **Encapsulation:** Bundling data and methods that operate on that data within a class.
9. class Car:
10. def __init__(self, make):
11. self.__make = make # Private attribute
12.
13. def get_make(self):
14. return self.__make
15. **Inheritance:** Deriving new classes from existing ones.
16. class Animal:
17. def speak(self):
18. print("Animal speaks")
19.
20. class Dog(Animal):
21. def speak(self):
22. print("Dog barks")
23. **Polymorphism:** Objects can take on multiple forms.
24. def make_sound(animal):
25. animal.speak()
26.

27. make sound(Dog())

File Handling

Used to read and write files.

Reading Files:

```
with open("example.txt", "r") as file:
    content = file. Read()
    print(content)
```

Writing Files:

```
with open("example.txt", "w") as file:
    file.write("Hello, World!")
```

This guide covers Python basics to advanced concepts. Let me know if you need deeper explanations or examples for any topic!