Python basics

1. What is Python? Why is it popular in AI & Data Science?

- Python is a **high-level, Object-oriented, interpreted programming language** known for its simple, readable syntax that emphasizes code clarity and ease of use. Python is called an interpreted language because its code is executed line by line by an interpreter at runtime, rather than being fully compiled into machine code before execution. Python is called object-oriented because it supports classes and objects (A class is a blueprint or template used to create objects). It implements all OOP principles: encapsulation, abstraction, inheritance, and polymorphism.
- Python is **popular in Artificial Intelligence (AI) and Data Science** because it is **simple, powerful, flexible, and supported by a rich ecosystem of libraries** that make complex tasks easier. It is easy to learn and use. **Python is open source and cross platform** (Runs on Windows, Linux, macOS). Python integrates with other technologies. It has powerful libraries for data analysis and machine learning.

2. Difference between list, tuple, set and dictionary

◇ List

- Ordered collection of elements
- Allows **duplicate values**
- **Mutable** (can be changed)
- Indexed (Indexing means accessing an element using its position number)
- Written using []
- Example: my_list = [1, 2, 3, 3]

◇ Tuple

- Ordered collection of elements
- Allows **duplicate values**
- **Immutable** (cannot be changed)
- Indexed (supports slicing)
- Written using ()
- Faster than lists
- Example:

```
my_tuple = (1, 2, 3, 3)
```

◇ Set

- **Unordered** collection of elements
- Does **not allow duplicates**
- Mutable (can add/remove elements)
- No indexing or slicing
- Written using {}
- Useful for removing duplicates
- Example:

```
my_set = {1, 2, 3}
```

◇ Dictionary

- Collection of **key–value pairs**
- Keys must be **unique**
- Mutable
- Accessed using keys, not index
- Written using {key: value}
- Very fast lookup
- Example:

```
my_dict = {"name": "Resya", "age": 22}
```

3. What are variables and data types in Python?

Variable:

- A name that stores a value in memory
- It acts as a container for values
- Example:
  x = 10
  name = "Python"
  pi = 3.14
- Rules:
  1) Must start with a letter (a–z / A–Z) or underscore (_)
  2) Cannot start with a number
  3) Can contain letters, numbers, and underscores
  4) Case-sensitive (Age ≠ age)
  5) Cannot use Python keywords

Data Type:

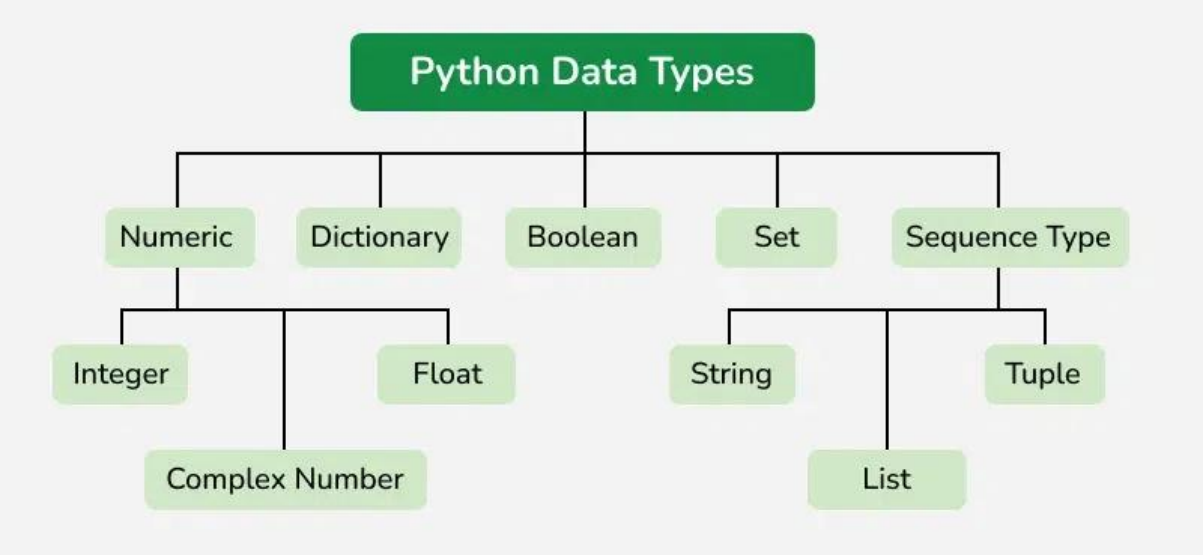
- Specifies the kind of value a variable holds
- Python automatically assigns the data type
- Use type() to check data type
- Example:
  x = 10
  type(x) # int
- Common data types:
  1) **Numeric Data Types**
     int → Whole numbers. a = 5
     float → Decimal numbers. b = 3.14
     complex → Complex numbers. c = 2 + 3j
  2) **Boolean Data Type**
     bool → True or False
  3) **Sequence Data Types**
     list → Ordered, mutable. lst = [1, 2, 3]
     tuple → Ordered, immutable. tup = (1, 2, 3)
     str → Sequence of characters. name = "Resya"

4) Set

set → Unordered, no duplicates.s = {1, 2, 3}

*5) **Dictionary***

dict → Key–value pairs. student = {"name": "Resya", "age": 22}



4. What is the difference between == and =

◇ **= (Assignment Operator)**

- Used to **assign a value** to a variable
- Stores data in a variable
- Does **not** compare values
- Used on the **left side** of a variable

Example:

```
x = 10
name = "Python"
```

- **== (Equality Operator)**

  - Used to **compare two values (**checks whether two values are equal)
  - Returns **True or False**
  - Does **not** assign values
  - Used in **conditions** (if, while)

Example:

```
x = 10
x == 10 # True
x == 5 # False
```

5. Explain if, elif and else

If Statement

An if statement is used when a block of code needs to be executed only if a specific condition evaluates to True. It Used to check a condition. Code runs only if the condition is True.

Syntax:

```
if condition:
    statement(s)
```

Example:

```
age = 18
if age >= 18:
    print("Eligible to vote")
```

Elif (else if) Statement

- Used to **check multiple conditions**
- Checked **only if previous condition is False**
- You can use **multiple elif statements**

Syntax:

```
elif condition:
    statement(s)
```

Example:

```
marks = 75

if marks >= 90:
    print("Grade A")
elif marks >= 60:
    print("Grade B")
```

Else statement

Executes when all above conditions are False. Used as a default block. Comes only once and at the end.

Syntax:

```
else:
   statement(s)
```

Example:

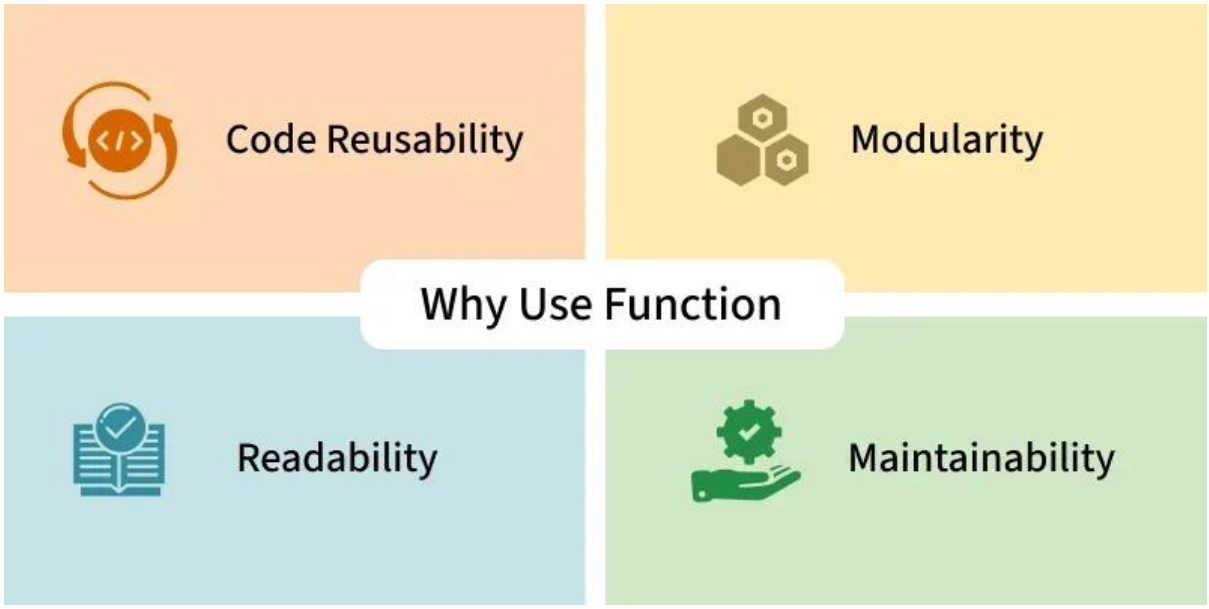
```
marks = 40

if marks >= 90:
   print("Grade A")
elif marks >= 60:
   print("Grade B")
else:
   print("Fail")
```

6. What is a function? Why do we use functions?

Python Functions are a block of statements that does a specific task. The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

Example:

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



Code Reusability: Write once, use many times, Reduces repetition

Modularity: Program divided into logical blocks, each function handles one task

Better Readability: Breaks large programs into smaller parts, Code becomes easy to understand

Improves Maintainability: Easy to update or modify, Changes don't affect whole program

7. Difference between for loop and while loop

Both for loops and while loops are **control flow structures** in programming that allow you to repeatedly execute a block of code.

for Loop

- The for loop is used when you know in advance how many times you want to execute the block of code.
- Iterates over a **sequence** (list, tuple, string, range)
- Loop variable is automatically updated
- Less chance of infinite loop
- Easy to read and write

Example:

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

◇ while Loop

- The while loop is used when you don't know in advance how many times you want to execute the block of code.
- Runs as long as a **condition is True**
- Loop variable must be updated manually
- Higher risk of infinite loop
- More flexible for condition-based loops

Example:

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

8. What is a module? Example

- A Module in python is a file containing definitions and statements. A module can define functions, classes and variables. Modules help organize code into separate files so that programs become easier to maintain and reuse. Instead of writing everything in one place, related functionality can be grouped into its own module and imported whenever needed. That is, A **module** is a Python file that contains reusable code such as functions and variables.

1. Built-in Modules: These come bundled with Python and require no installation - e.g., math, random, os
2. User-Defined Modules: These are modules you create yourself, such as my_module.py

Step 1: Create a file my_module.py

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

def subtract(a, b):
    return a - b
```

Step 2: Use it in another file

```
import my_module

print(my_module.add(10, 5))
print(my_module.subtract(10, 5))
```

9. What is exception handling?

- **Exception handling** is a mechanism to handle **runtime errors**
- Prevents program from crashing suddenly
- Allows normal flow of program even when an error occurs

An **exception** is an error that occurs during program execution. We use exceptional handling to:

- Avoid program termination
- Handle errors gracefully
- Improve program reliability
- Provide user-friendly error messages

Python provides four main keywords for handling exceptions: try, except, else and finally each plays a unique role.

10. What is pip?

- pip stands for Pip Installs Packages
- pip is a Python package manager used to install and manage external libraries
- It is the package manager for Python
- Used to install, update, and remove Python packages
- Downloads packages from PyPI (Python Package Index)