PYTHON BASICS QUESTIONS

1. **What is Python, and why is it popular?**
   Answer:
   Python is a high-level, interpreted programming language created by **Guido van Rossum** and released in **1991**. It is known for its **simple syntax**, which is easy to read and write, making it ideal for beginners as well as professionals.
   Reasons why Python is popular:
   1. **Simple and Readable Syntax:**
      Python's code is clean and easy to understand, which reduces development time.
   2. **Versatile:**
      It can be used for **web development, data science, artificial intelligence, machine learning, automation, game development**, and more.
   3. **Large Community Support:**
      Python has a vast and active community. You can easily find libraries, frameworks, and help from forums like Stack Overflow.
   4. **Rich Libraries and Frameworks:**
      Python comes with powerful libraries such as **NumPy, Pandas, TensorFlow, Django, Flask**, etc., which simplify complex tasks.
   5. **Cross-platform:**
      Python runs on Windows, Linux, macOS, and many other platforms without changing the code.
   6. **Beginner-Friendly:**
      Because of its simplicity and readability, Python is widely taught in schools, colleges, and bootcamps.
   7. **Open Source:**
      Python is free to use and distribute, which makes it widely adopted in both academia and industry.

2. **What is an interpreter in Python?**
   Answer:
   An **interpreter in Python** is a program that reads and **executes Python code line by line**. Unlike a compiler, which converts the entire code into machine language before running, the interpreter **executes each line one at a time**, making it easier to find and fix errors quickly.

   When you run a Python program, the interpreter translates the high-level Python code into low-level machine instructions that the computer can understand and execute.

3. **What are pre-defined keywords in Python?**
   Answer:

Pre-defined keywords in Python are **special reserved words** that have **specific meanings** in the language. These keywords are used to define the **syntax and structure** of Python programs. You **cannot use them as variable names, function names, or identifiers**.

Examples of Python Keywords:

- if, else, elif – used for conditional statements
- for, while – used for loops
- def, return – used to define functions
- class – used to define a class
- import, from – used to import modules
- True, False, None – special constants
- try, except, finally – used for error handling

Total Keywords:

As of Python 3.10+, there are **35 pre-defined keywords** (may change slightly in future versions).

4. **Can keywords be used as variable names in Python?**

Answer:

No, keywords **cannot be used as variable names** in Python.

Python keywords are **reserved words** that have **special meaning** in the language's syntax. If you try to use a keyword as a variable name, Python will show a **syntax error**.

5. **What is mutability in Python?**

Answer:

Mutability in Python refers to whether an object's **value can be changed** after it is created.

- If an object **can be changed**, it is called **mutable**.
- If an object **cannot be changed**, it is called **immutable**.

✅ **Mutable Objects:**

These can be modified after creation.

Examples:

- list
- dict
- set

❌ **Immutable Objects:**

These cannot be changed after creation.

Examples:

- int

- float
- str
- tuple

6. **Why are lists mutable, but tuples are immutable?**
   Answer:
   In Python, **lists are mutable** because they are designed to **store data that can change**. You can add, remove, or modify elements in a list after it is created.

   On the other hand, **tuples are immutable** because they are meant to store **fixed, constant data**. Once a tuple is created, its elements **cannot be changed, added, or removed**.

7. **What is the difference between "==" and "is" operators in Python**
   Answer:
   In Python, both == and is are used for comparison, but they have **different meanings:**

✅ **== (Equality Operator):**

- Checks **whether the values of two objects are equal.**
- It **compares the content** of the objects.

Example:
a = [1, 2, 3]
b = [1, 2, 3]
print(a == b) # ✅ Output: True (values are same

✅ **is (Identity Operator):**

- Checks **whether two objects refer to the same memory location.**
- It **compares the identity** of the objects.

Example:
a = [1, 2, 3]
b = [1, 2, 3]
print(a is b) # ❌ Output: False (different memory locations)

✅ **Summary Table:**

OPERATOR	MEANING	COMPARES
==	Equality	Values

IS	Identity (same memory?)	Memory Address

8. **What are logical operators in Python?**

   Answer:

   Logical operators in Python are used to **combine multiple conditions** (Boolean expressions). They return **True or False** based on the logic applied.

   ✅ **Python has 3 logical operators:**

Operator	**Description**	**Example**
and	Returns True if **both** conditions are True	x > 5 and x < 10
or	Returns True if **at least one** condition is True	x > 5 or x < 3
not	**Reverses** the result (True ⇄ False)	not(x > 5)

Examples:

```
x = 7

# AND operator
print(x > 5 and x < 10)   # True

# OR operator
print(x > 10 or x == 7)   # True

# NOT operator
print(not(x > 10))        # True
```

9. **What is type casting in Python?**

   Answer:

   Type casting in Python means **converting one data type into another**. It is used when you need to perform operations between different types or display data in a specific format.

   ✅ **Types of Type Casting:**

   1. **Implicit Type Casting**
      - Done **automatically** by Python.

- Converts smaller type to a larger type (e.g., int to float)
  Example:
  ```
  a = 5
  b = 2.0
  c = a + b
  print(c)      # Output: 7.0 (int converted to float automatically)
  ```

Explicit Type Casting

- Done **manually** using functions like int(), float(), str(), etc.

  Example:

  ```
  x = "10"
  y = int(x)     # Converting string to integer
  print(y + 5)   # Output: 15
  ```

✅ **Common Type Casting Functions:**

- int() → Converts to integer
- float() → Converts to float
- str() → Converts to string
- bool() → Converts to boolean

10. What is the difference between implicit and explicit type casting in Python?
Answer:

Type casting is the process of converting one data type to another in Python. It is of two types: **implicit** and **explicit**.

✅ **1. Implicit Type Casting:**

- **Automatically** done by Python.
- Happens **without user intervention**.
- Python **converts smaller data types to larger data types** to avoid data loss.

Example

```
a = 5      # int

b = 2.5     # float

c = a + b   # int + float = float
```

```
print(c)    # Output: 7.5
```

✅ **2. Explicit Type Casting:**

- **Manually** done by the programmer.
- Requires the use of **casting functions** like int(), float(), str(), etc.
- Used when Python **does not automatically convert** the type.
- **Example:**

```
x = "10"
y = int(x)      # Converting string to integer
print(y + 5)    # Output: 15
```

✅ **Summary Table:**

Feature	**Implicit Casting**	**Explicit Casting**
Who performs it?	Python (automatically)	Programmer (manually)
Use of functions	No	Yes (e.g., int(), float())
Risk of error	Very low	Possible if data is invalid

11. What is the purpose of conditional statements in Python?

Answer:
The purpose of **conditional statements** in Python is to **make decisions in a program** based on certain conditions.
They allow the program to **execute different blocks of code** depending on whether a condition is **True or False**.

✅ **Why use conditional statements?**
- To **control the flow** of the program
- To **perform different actions** based on user input or data
- To make the program more **interactive and intelligent**

✅ **Common conditional statements in Python:**
- if
- if...else
- if...elif...else

✅ **Example:**

```
age = 18
```

```
if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

🟢 Output: You are eligible to vote.

In this example, the condition age >= 18 decides which block of code will run.

12. How does the elif statement work in Python?

Answer:
The elif statement in Python stands for **"else if"**. It is used **after an if statement** to check **multiple conditions** one by one.
When the if condition is **False**, Python checks the next elif condition. If that is also False, it moves to the next elif, and finally to the else block (if present).

✅ **Syntax:**

```
if condition1:
    # Executes if condition1 is True
elif condition2:
    # Executes if condition2 is True
elif condition3:
    # Executes if condition3 is True
else:
    # Executes if none of the above conditions are True
```

✅ **Example:**

```
marks = 85

if marks >= 90:
    print("Grade: A")
elif marks >= 80:
    print("Grade: B")
elif marks >= 70:
    print("Grade: C")
else:
    print("Grade: D")
```

🟢 **Output:** Grade: B
Python will stop checking further conditions once it finds the first True one.

13. What is the difference between for and while loops in Python?

Answer:

In Python, both for and while loops are used to **repeat a block of code**, but they work in **different ways**.

✅ **1. for Loop:**

- Used when you **know in advance** how many times you want to loop.
- Iterates over a **sequence** (like list, range, string, etc.)

Example:

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

🟢 Output: 0 1 2 3 4

✅ **2. while Loop:**

- Used when you want to **repeat code until a condition becomes False**.
- You **don't always know** how many times it will run.

Example:

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

🟢 Output: 0 1 2 3 4

✅ **Summary Table:**

Feature	**for loop**	**while loop**
Use Case	When loop count is known	When loop count is unknown
Condition Check	Automatically done in sequence	Must be given by the programmer
Common Use	Iterating over list, range, string	Running code until a condition fails

14. Describe a scenario where a while loop is more suitable than a for loop.

Answer:

A while loop is more suitable when you **don't know in advance how many times the loop should run**, and the loop needs to continue **until a certain condition becomes false.**

✅ **Example Scenario: User Login System**

Suppose you are building a login system where a user has to enter the correct password. You don't know how many attempts the user will take, so a while loop is a better choice.

Example Code:

```
correct_password = "yash123"
user_input = ""

while user_input != correct_password:
    user_input = input("Enter password: ")

print("Access granted!")
```

In this case:

- The loop keeps asking the user until the correct password is entered.
- Since the number of attempts is **unknown**, using a for loop is not suitable.

✅ **Summary:**
Use a while loop when the repetition depends on a condition, not on a fixed number of times.