Python Basics Questions

1. What is Python, and why is it popular?

Python is a high-level, interpreted programming language known for its simple syntax, readability, and versatility.
Popularity reasons:

- Easy to learn and write
- Extensive libraries (e.g., NumPy, Pandas, TensorFlow)
- Large community and support
- Widely used in web development, data science, automation, AI, etc.

2. What is an interpreter in Python?

An **interpreter** in Python is a program that reads and executes Python code line by line. It translates Python code into machine code at runtime, making it easier to debug and test.

3. What are pre-defined keywords in Python?

Keywords are reserved words that have special meaning in Python, such as if, for, while, True, False, None, def, etc.
They are part of the Python syntax and cannot be used for anything else.

4. Can keywords be used as variable names?

No, **keywords cannot be used as variable names**. Doing so will result in a **SyntaxError** because Python reserves them for specific language constructs.

5. What is mutability in Python?

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

- **Mutable** objects: Can be changed (e.g., list, dict, set)
- **Immutable** objects: Cannot be changed (e.g., tuple, str, int)

6. Why are lists mutable, but tuples are immutable?

- **Lists** are designed for dynamic data—items can be added, removed, or changed.
- **Tuples** are meant to represent fixed collections, improving performance and ensuring data consistency.

7. What is the difference between == and is operators in Python?

- == checks **value equality**: whether two variables have the same value.
- is checks **identity**: whether two variables point to the **same object** in memory.

8. What are logical operators in Python?

Python has three **logical operators**:

- and – True if both conditions are True
- or – True if at least one condition is True
- not – Inverts the truth value

9. What is type casting in Python?

Type casting means converting one data type to another.
Example: int("5") converts the string "5" to an integer.

10. What is the difference between implicit and explicit type casting?

- **Implicit casting**: Done automatically by Python (e.g., 3 + 4.0 → result is 7.0)
- **Explicit casting**: Done manually using functions like int(), float(), str()

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

Conditional statements (if, elif, else) let you **control the flow** of your program by executing different blocks of code based on conditions.

12. How does the elif statement work?

elif (short for **else if**) allows multiple conditions to be checked in sequence. Once a true condition is found, the rest are skipped.

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

- for loop: Iterates over a sequence (e.g., list, range)
- while loop: Repeats **as long as a condition is true**

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

Use a while loop when:

- You **don’t know beforehand** how many times to repeat.
- You’re waiting for a **condition to become False**, like in user input or sensor readings.

Practical Questions

1. Print "Hello, World!"

```
print("Hello, World!")
```

2. Display your name and age

```
name = "Ajaz"
age = 21
print("Name:", name)
print("Age:", age)
```

3. Print all pre-defined Python keywords

```
help('keywords')
```

4. Check if a word is a Python keyword

```
import keyword
word = input("Enter a word: ")
if keyword.iskeyword(word):
    print(f"'{word}' is a Python keyword.")
else:
    print(f"'{word}' is not a Python keyword.")
```

5. List vs Tuple mutability

```
# List is mutable
my_list = [1, 2, 3]
my_list[0] = 100
```

```
print("Modified list:", my_list)

# Tuple is immutable
my_tuple = (1, 2, 3)
# my_tuple[0] = 100  # This will cause a TypeError
print("Tuple cannot be changed:", my_tuple)
```

6. Mutable vs Immutable argument behavior

```
def change_list(lst):
    lst.append(100)  # modifies original list
def change_number(num):
    num += 10  # doesn't change original number
my_list = [1, 2, 3]
my_num = 5
change_list(my_list)
change_number(my_num)
print("After function call - List:", my_list)   # List changed
print("After function call - Number:", my_num)  # Number unchanged
```

---

7. Basic arithmetic operations

```
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))
print("Sum:", a + b)
print("Difference:", a - b)
print("Product:", a * b)
print("Quotient:", a / b if b != 0 else "Division by zero error")
```

8. Demonstrate logical operators

```
x = 10
y = 5
```

```
print(x > 0 and y > 0)  # True
print(x < 0 or y > 0)   # True
print(not(x == y))      # True
```

9. Convert user input to int, float, and bool

```
data = input("Enter something: ")
int_data = int(data)
float_data = float(data)
bool_data = bool(data)
print("As integer:", int_data)
print("As float:", float_data)
print("As boolean:", bool_data)
```

10. Type casting list elements

```
str_list = ["1", "2", "3"]
int_list = [int(x) for x in str_list]
print("Original:", str_list)
print("Converted to integers:", int_list)
```

11. Check if number is positive, negative, or zero

```
num = float(input("Enter a number: "))
if num > 0:
   print("Positive number")
elif num < 0:
   print("Negative number")
else:
   print("Zero")
```

12. Print numbers 1 to 10 using for loop

```
for i in range(1, 11):
   print(i)
```

13. Sum of even numbers from 1 to 50

```
even_sum = 0
for i in range(1, 51):
    if i % 2 == 0:
        even_sum += i
print("Sum of even numbers between 1 and 50:", even_sum)
```

14. Reverse a string using while loop

```
s = input("Enter a string: ")
reversed_str = ""
i = len(s) - 1
while i >= 0:
    reversed_str += s[i]
    i -= 1
print("Reversed string:", reversed_str)
```

15. Factorial using while loop

```
num = int(input("Enter a number: "))
factorial = 1
i = 1
while i <= num:
    factorial *= i
    i += 1

print("Factorial of", num, "is", factorial)
```