◈ 📌 Python Introduction & Setup (1–10)

1. **What is Python?**

   A high-level, interpreted programming language known for simplicity and readability.

2. **What are the key features of Python?**

   Easy syntax, interpreted, dynamically typed, extensive libraries and an open source.

3. **Is Python compiled or interpreted?**

   Python is interpreted.

4. **What are the main applications of Python?**

   Web development, data science, automation, AI/ML, scripting.

5. **How do you install Python on your system?**

   Download from python.org and run the installer.

6. **What is the difference between Python 2 and Python 3?**

   Python 3 is the current version; it has better Unicode support and print() as a function.

7. **How can you check the installed Python version?**

   Use python --version in command prompt.

8. **What is the role of the `print()` function in Python?**

   To print values or statements in the console.

9. **What is an IDE? Name a few commonly used Python IDEs.**

   Integrated Development Environment (e.g., VS Code, PyCharm, Thonny).

10. **How do you run a Python file from the terminal?**

    Use python filename.py or python3 filename.py.

🔷 📌 Variables in Python (11–20)

11. **What is a variable in Python?**

    Name of the container that holds data value in memory.

12. **How do you declare a variable in Python?**

    x = 10, just assign a value.

13. **Is it necessary to declare the type of a variable in Python?**

    No, Python uses dynamic typing. The type is determined automatically at runtime based on the value assigned.

14. **What are the rules for naming variables in Python?**

    Start with letter/underscore, snake-case with no special characters and is case-sensitive.

15. **What is the difference between global and local variables?**

    Global: defined outside functions.

    Local: defined inside functions.

16. **Can a variable name start with a number in Python? Why or why not?**

    No, not allowed (e.g., 3x=3 is invalid). Variable names must follow certain rules defined by the Python syntax to avoid confusion with numeric literals and maintain clarity in code execution.

17. **What happens if you use a variable without assigning a value?**

    Raises NameError.

18. **How is memory managed for variables in Python?**

    Python manages memory with memory manager automatically which uses reference counting, garbage collection, and internal optimizations like object reuse.

19. **Can Python variable names contain special characters like `$` or `@`?**

    No. $, @ are not allowed.

20. **What is the difference between = and == in Python?**

    = assigns a value, == checks equality.

🔷 📌 Data Types in Python (21–30)

21. **What are the basic data types in Python?**

    int, float, str, bool, list, tuple, set, dict.

22. **What is the difference between `int`, `float`, and `complex`?**

    int: whole numbers, float: decimals, complex: real+imaginary parts.

23. **What is the difference between a list and a tuple?**

    List: mutable, Tuple: immutable.

24. **How is a dictionary different from a list?**

    Dict uses key-value pairs; list uses indexed values.

25. **What is a set and how is it different from a list?**

    Set: unordered, no duplicates; List: ordered, allows duplicates.

26. **What is the difference between mutable and immutable data types?**

    Mutable: can change (list), Immutable: cannot change (tuple, str).

27. **What will `type()` function return if the variable is a string?**

    <class 'str'>

28. **What are Boolean data types?**

    True, False -used for logical decisions.

29. **How do you convert data from one type to another in Python?**

    By using type conversion methods like int(), float(), str() etc.

30. **What does the `len()` function do for different data types?**

    Returns number of items/characters in list, tuple, str, etc.

🔷 📌 Conditional Statements (31–40)

31. **What are conditional statements in Python?**

    Used to execute code blocks based on the condition given.

32. **What is the syntax of an `if` statement in Python?**

    ```
    if condition:
        #code to execute
    ```

33. **What is the difference between `if` and `if-else`?**

    if: one way check, if-else: handles both outcomes.

34. **What is the use of `elif` in Python?**

    For multiple condition checks after if.

35. **Can you use multiple `elif` blocks in a condition?**

    Yes, allowed and common.

36. **What happens if none of the conditions are true in an `if-elif-else` block?**

    Only the else block runs if all are false.

37. **Can we use `if` inside another `if`? Explain with an example.**

    Yes, allowed.

    Example: if a > 0:

    ```
    if b > 0:
        print("Both positive")
    ```

38. **How is indentation important in writing conditionals in Python?**

    Defines block scope. So, there is no need for {} like other languages.

39. **How do you check multiple conditions using `and` / `or`?**

    check multiple conditions using and (all must be true) or or (any one must be true) in a single if statement.

    Example: if a > 5 and b < 10:

    ```
    print("Number is between 6 and 9")
    ```

40. **What is the output of `if ""` or `if 0` in Python? Why?**

    both "" (empty string) and 0 are considered **falsy values** in Python, so the if condition evaluates to False.

◈ 📌 For Loop in Python (41–50)

41. **What is a `for` loop in Python and how is it used?**

    A for loop in Python is used to iterate over a sequence (like a list, string, or range) and execute a block of code for each item**.**

    Example:

```
for item in [1, 2, 3]:
    print(item)
```

42. **What is the syntax of a `for` loop?**

```
for item in sequence:
        #Code to execute
```

43. **How does the `range()` function work with loops?**

range() generates sequence of numbers, used in loops.

44. **Can you loop over strings and lists using `for`? Give examples.**

Yes.

Example:

```
 for char in "Python":
      print(char)
 for item in [1, 2, 3]:
       print(item)
```

45. **What is the use of `break` and `continue` inside a loop?**

break exits loop while continue skips to next iteration.

46. **How do you print only even numbers between 1 and 20 using a loop?**

```
for num in range(2, 21, 2):
          print(num)
```

47. **What is the use of `else` with a `for` loop?**

Runs if loop completes normally (not interrupted by break).

48. **What does `enumerate()` do in a `for` loop?**

Returns index and item.

Example:

```
fruits = ["apple", "banana", "cherry"]
for index, value in enumerate(fruits):
    print(f"{index}: {value}")
```

output:

```
0: apple
1: banana
2: cherry
```

49. **What is a nested loop? Provide an example.**

A loop inside another loop. Example: pattern printing.

50. **Can we use `for` loop with dictionaries? If yes, how?**

Yes.

Example:

```
for k, v in dict.items():
    print(k, v)
```