1. What are the key features of Python as a programming language?

- Easy to learn and read (simple syntax)
- Interpreted language
- Dynamically typed
- Cross-platform
- Large standard library
- Supports OOP and functional programming

2. How is Python interpreted and dynamically typed?

- **Interpreted**: Code runs line by line (no need to compile).
- **Dynamically typed**: No need to declare variable types, Python infers them at runtime.

3. Explain the difference between Python 2 and Python 3.

- **Python 2**: Print is a statement (print "Hello"), range() returns list.
- **Python 3**: Print is a function (print("Hello")), range() returns iterator.
  👉 Python 2 is obsolete, Python 3 is standard.

4. What is PEP 8 and why is it important?

PEP 8 is the official Python style guide. It ensures readability and consistency across code.

5. How do you write comments in Python?

- **Single-line**: # This is a comment
- **Multi-line**: Triple quotes (""" comment """)

6. What are Python's built-in data types? Give examples.

- **Numeric**: int, float, complex
- **Sequence**: list, tuple, range

- **Text**: str
- **Set types**: set, frozenset
- **Mapping**: dict
- **Boolean**: bool
- **Special**: NoneType

7. What is the difference between mutable and immutable types? Provide examples.

- **Mutable**: Can be changed after creation → list, dict, set
- **Immutable**: Cannot be changed → int, float, tuple, str

8. How is None different from 0 and False?

- **None**: Absence of value (null).
- **0**: Numeric zero.
- **False**: Boolean false.
  👉 They are not equal (None != 0 != False).

9. What is type casting? Give examples using int(), float(), and str().

Changing one type to another.

```
int("10")    # 10

float("3.5")  # 3.5

str(100)     # "100"
```

10. How do you check the type of a variable?

Using type().

```
x = 5

print(type(x))  # <class 'int'>
```

11. What are the different types of operators in Python?

- Arithmetic (+, -, *, /, //, %, **)
- Comparison (==, !=, <, >, <=, >=)
- Logical (and, or, not)
- Assignment (=, +=, -=)
- Identity (is, is not)
- Membership (in, not in)
- Bitwise (&, |, ^, ~, <<, >>)

12. Explain the difference between / and //.

- / → Division with float result (5/2 = 2.5)
- // → Floor division (5//2 = 2)

13. How does the is operator differ from ==?

- == → Checks **value equality**.
- is → Checks **object identity** (memory location).

14. What does the % operator do?

It gives the remainder of division.
👉 Example: 10 % 3 = 1

15. Explain operator precedence in Python.

Order in which operators are evaluated:
() > ** > *, /, %, // > +, - > comparison > logical

16. How do you write an if-elif-else statement? Give an example.

x = 10

if x > 15:

```
    print("Big")
elif x > 5:
    print("Medium")
else:
    print("Small")
```

17. What is the difference between nested if and multiple elif conditions?

- **Nested if**: One if inside another.
- **Elif**: Sequential conditions checked one by one.

18. Can Python have an else without if? Explain.

Yes, but only with loops (while or for).
Runs if the loop completes without break.

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

- **for**: Iterates over a sequence.
- **while**: Runs until a condition becomes false.

20. How does break differ from continue?

- **break**: Exits the loop completely.
- **continue**: Skips current iteration, continues loop.

21. What is the use of the pass statement?

It does nothing—used as a placeholder.

```
if True:
    pass  # To be implemented later
```

22. How do you use a for loop with the range() function?

```
for i in range(5):
    print(i)  # 0,1,2,3,4
```

23. How do you define and call a function in Python?

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


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

24. What is the difference between a function with and without a return value?

- **With return**: Sends a value back.
- **Without return**: Performs an action but returns None.

25. Explain default arguments in Python functions.

Default values assigned if no argument is passed.

```
def greet(name="Guest"):
    print("Hello", name)


greet()  # Hello Guest
```

26. What is the difference between *args and **kwargs?

- *args: Passes variable-length positional arguments (tuple).
- **kwargs: Passes variable-length keyword arguments (dict).

27. Explain the difference between a list, tuple, and set.

- **List**: Ordered, mutable, allows duplicates.
- **Tuple**: Ordered, immutable.
- **Set**: Unordered, mutable, unique elements only.

28. How do you add and remove elements from a list?

```
lst = [1,2,3]

lst.append(4)      # Add

lst.remove(2)      # Remove by value

lst.pop()          # Remove last
```

29. How do you access dictionary values?

```
d = {"a": 1, "b": 2}

print(d["a"])      # 1

print(d.get("b"))   # 2
```

30. How do you merge two dictionaries in Python 3.9+?

```
d1 = {"a": 1}

d2 = {"b": 2}

d3 = d1 | d2
```

31. How do you slice a string in Python?

```
s = "Python"

print(s[0:4])  # Pyth

print(s[::-1]) # nohtyP
```

32. What is the difference between .find() and .index()?

- .find() → Returns -1 if not found.
- .index() → Raises error if not found.

33. How do you remove whitespace from a string?

```
s = " hello "
```

```
print(s.strip())   # "hello"
```

34. What is string interpolation in Python? Give examples using f-strings.

String interpolation = embedding variables into strings.

```
name = "Alice"

print(f"Hello {name}")
```

35. How do you read and write files in Python?

```
f = open("test.txt", "w")

f.write("Hello")

f.close()


f = open("test.txt", "r")

print(f.read())

f.close()
```

36. What is the difference between read(), readline(), and readlines()?

- read() → Reads entire file.
- readline() → Reads one line.
- readlines() → Reads all lines into a list.

37. Why is the with statement recommended for file handling?

It automatically closes the file after use.

```
with open("file.txt") as f:

   data = f.read()
```

38. How do you handle exceptions in Python?

```
try:
   x = 1 / 0
except ZeroDivisionError:
   print("Error!")
```

39. What is the difference between try-except and try-finally?

- **try-except**: Handles errors.
- **try-finally**: Ensures code in finally always runs.

40. How do you raise a custom exception?

```
raise ValueError("Invalid input")
```

41. How do you import a module in Python?

```
import math
print(math.sqrt(16))
```

42. What is the difference between import module and from module import function?

- import math → Access with math.sqrt().
- from math import sqrt → Access directly sqrt().

43. How do you install third-party packages in Python?

Using **pip**:

```
pip install numpy
```

44. What is a lambda function?

An anonymous, one-line function.

```
square = lambda x: x*x
print(square(5))
```

45. Explain list comprehension with an example.

A short way to create lists.

```
squares = [x*x for x in range(5)]
```

46. What are Python’s built-in functions? Give five examples.

len(), sum(), max(), min(), sorted().

47. What is the purpose of the dir() function?

Lists all attributes and methods of an object.

48. How do you check Python’s version from within a script?

```
import sys

print(sys.version)
```