
Q&A – Python (Detailed)

1. What are the key features of Python as a programming language?
Python is a high-level, general-purpose programming language.

- **Simple and readable syntax.**
- **Interpreted (no compilation required).**
- **Dynamically typed (no need to declare variable types).**
- **Object-oriented (supports classes and objects).**
- **Cross-platform (runs on Windows, Linux, macOS).**
- **Large standard library and third-party packages.**

---

2. How is Python interpreted and dynamically typed?

- **Interpreted: Code runs line by line using the Python interpreter.**
- **Dynamically typed: Type is decided at runtime.**

```
x = 5     # integer

x = "Hi"   # now string
```

---

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

- **Python 2: Legacy version, print is a statement (print "Hello"), ASCII by default.**
- **Python 3: Current version, print is a function (print("Hello")), Unicode by default, f-strings supported.**

---

4. What is PEP 8 and why is it important?
PEP 8 is the official style guide for Python. It ensures consistency in code (indentation, naming conventions, spacing), making it easier to read and maintain.

---

5. How do you write comments in Python?

- **Single-line: # This is a comment**
- **Multi-line:**

```
"""
This is
a multi-line
comment
"""
```

---

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

- **Numeric: int, float, complex → x = 3.14**
- **Sequence: list, tuple, range → [1,2,3]**
- **Text: str → "Hello"**
- **Set: set, frozenset → {1,2,3}**
- **Mapping: dict → {'a':1}**
- **Boolean: True/False**
- **NoneType: None**

---

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

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

```
lst = [1,2]
lst.append(3)   # works
s = "hi"
s[0] = "H"      # error (immutable)
```

---

8. How is None different from 0 and False?

- **None = no value assigned (type NoneType).**
- **0 = integer zero.**
- **False = boolean false.**
  They are not equal but all are treated as False in conditions.

---

9. What is type casting?
Changing one type to another:

```
int(3.9)    # 3

float(5)    # 5.0

str(123)    # '123'
```

---

10. How do you check the type of a variable?
Use type() function.

```
x = 10

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

---

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

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

---

12. Explain the difference between / and //.

- **/ = floating division (10/3 = 3.33)**
- **// = floor division (10//3 = 3)**

---

13. How does is differ from ==?

- **== checks value equality.**
- **is checks memory identity.**

```
a=[1,2]

b=[1,2]

a==b   # True

a is b # False
```

---

14. What does the % operator do?
It returns the remainder. Example:

```
10 % 3   # 1
```

---

15. Explain operator precedence in Python.
Defines the order in which operations are executed.
Order: Parentheses → Exponent → Multiplication/Division → Addition/Subtraction.

---

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

```
x = 5

if x > 0:

   print("Positive")

elif x == 0:
   print("Zero")
else:
   print("Negative")
```

---

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

- **Nested if: An if inside another if.**
- **elif: Sequential multiple conditions.**

---

18. Can Python have an else without if?

- **No in general syntax.**
- **But loops can have else after them (executed if no break occurs).**

---

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

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

---

20. How does break differ from continue?

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

---

21. What is the use of the pass statement?
pass does nothing. It is a placeholder when code is required syntactically.

---

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

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

---

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

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


print(greet("Tom"))
```

---

24. Function with vs without return value?

- **With return: Gives output.**
- **Without return: Just executes.**

---

25. Explain default arguments.
Predefined values if not passed.

```
def add(x, y=5):
    return x+y


add(3)  # 8
```

****26. Difference between *args and kwargs?**

- ***args: multiple positional args.**
- ****kwargs: multiple keyword args.**

```
def test(*args, **kwargs):
    print(args, kwargs)
```

---

27. Difference between list, tuple, and set?

- **List: ordered, mutable.**
- **Tuple: ordered, immutable.**
- **Set: unordered, unique.**

---

28. How do you add and remove list elements?

- **append(), insert() to add.**
- **remove(), pop() to remove.**

---

29. How do you access dictionary values?

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

print(d['a'])

print(d.get('b'))
```

---

30. How do you merge dictionaries (Python 3.9+)?

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

d2={'b':2}

d3 = d1 | d2   # {'a':1,'b':2}
```

---

31. How do you slice a string?

```
s = "Python"

print(s[0:4])   # Pyth
```

---

32. Difference between .find() and .index()?

- **.find() returns -1 if not found.**

- **.index() raises error.**

---

33. How do you remove whitespace from a string?

- **strip() = both sides.**
- **lstrip() = left.**
- **rstrip() = right.**

---

34. What is string interpolation?
Inserting variables inside strings.

```
name="Tom"

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

---

35. How do you read and write files?

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

f.write("Hello")

f.close()


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

print(f.read())

f.close()
```

---

36. Difference between read(), readline(), readlines()?

- **read() → whole file.**
- **readline() → one line.**
- **readlines() → list of lines.**

---

37. Why is with statement recommended for file handling?
It auto-closes file even if error occurs.

```
with open("test.txt","r") as f:
    print(f.read())
```

---

38. How do you handle exceptions?

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

---

39. Difference between try-except and try-finally?

- **try-except: catches error.**
- **try-finally: ensures cleanup runs regardless.**

---

40. How do you raise custom exception?

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

---

41. How do you import a module?

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

---

42. Difference between import module and from module import function?

- **import math → math.sqrt(4)**

- **from math import sqrt → sqrt(4)**

---

43. How do you install third-party packages?

```
pip install package_name
```

---

44. What is a lambda function?
Anonymous one-line function.

```
square = lambda x: x**2

print(square(5))
```

---

45. Explain list comprehension.
Short way to create lists.

```
[x**2 for x in range(5)]  # [0,1,4,9,16]
```

---

46. Give five built-in functions.
len(), type(), sum(), print(), max()

---

47. Purpose of dir()?
Shows attributes/methods of object.

```
print(dir([]))
```

---

48. How to check Python version?

```
import sys

print(sys.version)
```
