Top 40 Python Interview Questions

1. What are the key features of Python ?

Python is:

- *Interpreted and dynamically typed*
- *Easy to read and write*
- *Open-source and cross-platform*
- *Supports multiple paradigms (OOP, functional, procedural)*
- *Has extensive libraries and frameworks*

2. What is the difference between a list and a tuple ?

- *List : Mutable (can be modified), defined using [].*
- *Tuple : Immutable (cannot be modified), defined using ().*

```
my_list = [1, 2, 3] # Mutable
my_tuple = (1, 2, 3) # Immutable
```

@Tajamulkhann

3. How do you swap two variables in Python without using a third variable ?

Python allows swapping in a single line:

```
a, b = b, a
print(a, b)
```

@Tajamulkhann

4. What is the difference between is and == in Python ?

- *== compares values (checks if two objects have the same value).*
- *is compares identity (checks if two variables point to the same object in memory).*

```
x = [1, 2, 3]
y = [1, 2, 3]
print(x == y) # True (same values)
print(x is y) # False (different objects)
```

@Tajamulkhann

5. What is PEP 8, and why is it important?

- *PEP 8 is the official Python Style Guide that provides best practices for writing clean and readable code.*
- *Example rule:*
    - *Use **4 spaces** per indentation level.*

6. What are mutable and immutable types in Python?

- ***Mutable*** : *Can be changed (Lists, Dictionaries, Sets).*
- ***Immutable*** : *Cannot be changed (Tuples, Strings, Integers).*

```
x = "hello" x[0] = "H"
# Error! Strings are immutable.
```

@Tajamulkhann

7. How does Python handle memory management?

- *Uses reference counting and garbage collection.*
- *Objects with zero references are automatically deleted.*

8. What is the difference between break, continue, and pass?

- *break → Exits the loop completely.*
- *continue → Skips the current iteration and moves to the next one.*
- *pass → Does nothing (used as a placeholder).*

```
for i in range(5):
    if i == 2:
        continue # Skips 2
    print(i)
```

9. What are *args and **kwargs in Python?

args: *Allows a function to take any number of positional arguments.*

****kwargs:*** *Allows a function to take any number of keyword arguments.*

```python
def func(*args, **kwargs):
    print(args, kwargs)

func(1, 2, 3, name="Python", age=30)
```

@Tajamulkhann

10. What is a lambda function?

A lambda function is an anonymous, one-line function

```
add = lambda x, y: x + y

print(add(3, 4)) # Output: 7
```

@Tajamulkhann

11. How do you handle exceptions in Python?

Using try-except:

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Cannot divide by zero!")
```

@Tajamulkhann

12. What is the difference between deepcopy and shallow copy?

- *Shallow copy : Creates a new object but copies references to nested objects.*
- *Deep copy : Recursively copies all objects.*

```
import copy
a = [[1, 2], [3, 4]]
b = copy.deepcopy(a) # Creates a full copy
```

13. How do you generate random numbers in Python?

```
# Using the random module:


import random
print(random.randint(1, 10))
```

@Tajamulkhann

14. How do you reverse a string in Python?

```
# Using slicing:


s = "hello"
print(s[::-1])  # Output: "olleh"
```

@Tajamulkhann

15. How do you remove duplicates from a list?

```
lst = [1, 2, 2, 3, 4, 4]
unique_lst = list(set(lst))
print(unique_lst)  # Output: [1, 2, 3, 4]
```

16. What is the difference between sorted() and .sort()?

```
sorted(lst): # Returns a new sorted list.

lst.sort(): # Sorts the list in place.
```

17. How do you check if a string contains a substring?

```
text = "Hello, World!"

print("World" in text) # Output: True
```

18. How do you read a file in Python?

```
with open("file.txt", "r") as f:
    content = f.read()
```

@Tajamulkhann

19. How do you write to a file in Python?

```
with open("file.txt", "w") as f:
    f.write("Hello, World!")
```

@Tajamulkhann

20. What is the difference between @staticmethod and @classmethod?

```
class MyClass:
    @staticmethod
    def static_method():
        return "Static method"

    @classmethod
    def class_method(cls):
        return "Class method"
```

@staticmethod: *Does not access class attributes.*

@classmethod: *Takes cls as an argument and can modify class attributes.*

21. What is the difference between del, .pop(), and .remove() in lists?

- ***del lst[i]*** → *Removes item at index i without returning it.*
- ***lst.pop(i)*** → *Removes item at index i and returns it.*
- ***lst.remove(x)*** → *Removes the first occurrence of x.*

22. What is a Python generator?

A generator is an iterable that yields values lazily using yield.

```
def my_gen():
    yield 1
    yield 2
    yield 3
gen = my_gen()
print(next(gen))  # Output: 1
```

@Tajamulkhann

23. What is the difference between map(), filter(), and reduce()?

- ***map(func, iterable):*** *Applies func to all elements.*
- ***filter(func, iterable):*** *Returns elements where func is True.*
- ***reduce(func, iterable):*** *Reduces elements to a single value.*

24. What is list comprehension in Python?

A concise and efficient way to create lists.

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

25. What is a Python decorator?

A function that modifies another function without changing its structure.

```
def decorator(func):
    def wrapper():
        print("Before function call")
        func()
        print("After function call")
    return wrapper

@decorator
def say_hello():
    print("Hello!")
say_hello()
```

@Tajamulkhann

26. How do you check for memory usage of an object?

```
# Using sys.getsizeof():


import sys x = [1, 2, 3]
print(sys.getsizeof(x)) #
Output: Size in bytes
```

@Tajamulkhann

27. How do you sort a dictionary by values?

```
data = {"a": 3, "b": 1, "c": 2}
sorted_data = dict(sorted(data.items(), key=lambda item: item[1]))
print(sorted_data) # {'b': 1, 'c': 2, 'a': 3}
```

28. What is monkey patching in Python?

Dynamically modifying a class or module at runtime.

```python
class A:
    def hello(self):
        return "Hello"

def new_hello(self):
    return "Patched Hello"

A.hello = new_hello
print(A().hello())  # "Patched Hello"
```

@Tajamulkhann

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

```
dict1 = {"a": 1}
dict2 = {"b": 2}
merged = dict1 | dict2
print(merged)  # {'a': 1, 'b': 2}
```

30. What is Duck Typing in Python?

Duck typing is a concept in Python where an object's behavior determines its type, rather than its explicit class. If an object implements the necessary methods, it can be used as that type—regardless of its actual class.

```python
class Bird:
    def quack(self):
        return "Quack!"

def make_quack(duck):
    return duck.quack()  # No need to check if it's a 'duck'

print(make_quack(Bird()))  # Output: "Quack!"
```

@Tajamulkhann

31. How do you create a virtual environment?

It ensures that the script runs only when executed directly, not when imported.

```
python -m venv myenv
source myenv/bin/activate  # Mac/Linux
myenv\Scripts\activate  # Windows
```

32. What is the difference between deepcopy() and copy()?

- ***copy()*** → *Creates a shallow copy (references inner objects).*
- ***deepcopy()*** → *Recursively copies all objects.*

33. What is self in Python?

self refers to the instance of a class and allows access to its attributes and methods.

34. What are Python's built-in modules?

Examples:

- *math,*
- *random,*
- *os,*
- *sys,*
- *datetime,*
- *collections,*
- *re,*
- *json.*

35. What is the purpose of enumerate()?

Adds an index while iterating.

```
for i, val in enumerate(["a", "b", "c"]):
    print(i, val)  # 0 a, 1 b, 2 c
```

36. What is zip() used for?

Merging iterables into pairs.

```
names = ["Alice", "Bob"]
ages = [25, 30]
print(list(zip(names, ages)))
# [('Alice', 25), ('Bob', 30)]
```

37. What is a named tuple?

A tuple with named fields.

```
from collections import namedtuple
Person = namedtuple("Person", ["name", "age"])
p = Person("Alice", 30)
print(p.name)  # Alice
```

38. How do you convert JSON to a Python dictionary?

```
import json
data = '{"name": "Alice", "age": 30}'
print(json.loads(data))
# {'name': 'Alice', 'age': 30}
```

39. What is the use of re.match()?

Finds a pattern at the beginning of a string.

```
import re
print(re.match(r"\d+", "123abc"))
# Match object
```

@Tajamulkhann

40. What is dataclass in Python?

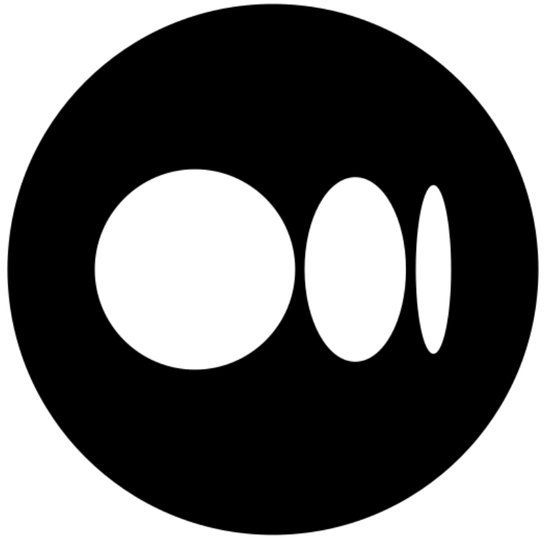
A lightweight class for data storage.

```
from dataclasses import dataclass
@dataclass
class Point:
    x: int
    y: int
```

Free Download

Leave a Testimonial

[Notes & Tips]



[Free Blogs]

[Free Projects]

Click on links to follow!

Found Helpful ?

Repost

Follow for more!