Basic Python interview questions that are commonly asked during technical interviews:

1. What is Python?

- Python is a high-level, interpreted programming language known for its simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

2. What are the key features of Python?

- Easy to learn and read

- Interpreted language

- Dynamically typed

- Extensive standard library

- Supports exception handling

- Automatic memory management (garbage collection)

- Supports multiple programming paradigms

3. What are Python's data types?

- Python has several built-in data types:

- Numeric types: `int`, `float`, `complex`

- Sequence types: `list`, `tuple`, `range`

- Text type: `str`

- Mapping type: `dict`

- Set types: `set`, `frozenset`

- Boolean type: `bool`

- Binary types: `bytes`, `bytearray`, `memoryview`

4. What is the difference between `list` and `tuple`?

- List: Mutable, can be modified after creation. Syntax: `[1, 2, 3]`

- Tuple: Immutable, cannot be modified after creation. Syntax: `(1, 2, 3)`

5. What is a dictionary in Python?

- A dictionary is a collection of key-value pairs, where each key is unique. It is mutable and indexed by keys. Syntax: `{'key1': 'value1', 'key2': 'value2'}`

6. What is the difference between `==` and `is` in Python?

- `==` checks for value equality.

- `is` checks for identity (i.e., whether two variables point to the same object in memory).

7. What are Python decorators?

- Decorators are functions that modify the behavior of other functions or methods. They are often used for logging, access control, memoization, etc. Example:

```python

def my\_decorator(func):

def wrapper():

print("Something before the function.")

func()

print("Something after the function.")

return wrapper

@my\_decorator

def say\_hello():

print("Hello!")

```

8. What is the difference between `deepcopy` and `shallow copy`?

- Shallow Copy: Creates a new object but inserts references to the original objects. Use `copy.copy()`.

- Deep Copy: Creates a new object and recursively copies all objects found in the original. Use `copy.deepcopy()`.

9. What is a lambda function?

- A lambda function is an anonymous function defined with the `lambda` keyword. It can have any number of arguments but only one expression. Example:

```python

add = lambda x, y: x + y

print(add(2, 3)) Output: 5

```

10. What is the purpose of `\_\_init\_\_` in Python?

- `\_\_init\_\_` is a special method in Python classes, known as the constructor. It is automatically called when a new instance of the class is created. It is used to initialize the object's attributes.

11. What is the difference between `append()` and `extend()` in lists?

- `append()`: Adds a single element to the end of the list.

- `extend()`: Adds multiple elements (from an iterable) to the end of the list.

12. What is the Global Interpreter Lock (GIL)?

- The GIL is a mutex that prevents multiple native threads from executing Python bytecode simultaneously. This can be a bottleneck in CPU-bound multi-threaded programs.

13. How do you handle exceptions in Python?

- Exceptions are handled using `try`, `except`, `else`, and `finally` blocks. Example:

```python

try:

result = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

else:

print("Division successful!")

finally:

print("This will always execute.")

```

14. What is the difference between `range()` and `xrange()`?

- In Python 2, `range()` returns a list, while `xrange()` returns an iterator (more memory-efficient).

- In Python 3, `xrange()` is removed, and `range()` behaves like `xrange()` from Python 2.

15. What is a generator in Python?

- A generator is a function that yields values one at a time using the `yield` keyword. It is memory-efficient and allows for lazy evaluation. Example:

```python

def my\_generator():

yield 1

yield 2

yield 3

for value in my\_generator():

print(value)

```

16. What is the difference between `args` and `kwargs`?

- `args`: Used to pass a variable number of non-keyword arguments to a function.

- `kwargs`: Used to pass a variable number of keyword arguments to a function.

17. What is the difference between `break` and `continue`?

- `break`: Exits the loop immediately.

- `continue`: Skips the rest of the code inside the loop for the current iteration and moves to the next iteration.

18. What is the difference between `import module` and `from module import `?

- `import module`: Imports the entire module, and you need to use `module.function()` to access its functions.

- `from module import `: Imports all functions and variables directly into the namespace, which can lead to name conflicts.

19. What is the purpose of `if \_\_name\_\_ == "\_\_main\_\_":`?

- This construct allows a Python file to be used as both a reusable module and a standalone script. Code inside this block will only execute if the file is run directly, not when imported as a module.

20. What is the difference between `str` and `repr`?

- `str`: Returns a user-friendly string representation of an object (used by `print()`).

- `repr`: Returns an unambiguous string representation of an object, often used for debugging.

---

These questions cover fundamental Python concepts and are a good starting point for preparing for a Python interview. Make sure to practice coding examples and understand the underlying principles!