Most Asked Python Interview Questions

1) What is Python, and what are its main features?

Python is a high-level, interpreted programming language that was first released in 1991. It is known for its simplicity, readability, and ease of use, which has made it a popular choice for beginners and experienced developers alike. Some of the main features of Python include:

Easy to Learn: Python has a simple and easy-to-understand syntax, making it easy for beginners to learn and write code.

Interpreted: Python code is executed line by line, which makes it easier to test and debug than compiled languages.

Dynamic Typing: Variables in Python do not need to be declared with their data types, as they are inferred during runtime.

Cross-platform: Python code can run on multiple operating systems, including Windows, macOS, and Linux.

Large Standard Library: Python has a large and comprehensive standard library that includes many useful modules for tasks like web development, data analysis, and machine learning.

2) What is PEP 8, and why is it important?

PEP 8 is a set of guidelines for writing Python code that promotes code readability and consistency. It provides recommendations for naming conventions, code formatting, and style.

Following PEP 8 is important because it helps to make code more readable and maintainable, especially when working in a team. Consistent code style makes it easier for developers to understand and modify code, and reduces the likelihood of errors or bugs.

In summary, PEP 8 is a set of guidelines that promote code readability and consistency, and following it is important for writing clear and maintainable Python code.

3) How do you handle errors and exceptions in Python?

In Python, errors and exceptions are handled using **try-except blocks**.

Here's a general syntax for handling errors and exceptions in Python:

```
try:

# code that may cause an error or exception

except ExceptionType:

# code to handle the error or exception

else:
```

code to execute if no exception is raised

finally:

code to execute regardless of whether an exception is raised or not

Here's how the try-except block works:

The code that may cause an error or exception is enclosed within the try block.

If an exception is raised, the program control jumps to the except block. The ExceptionType can be specific to a particular type of exception, such as ValueError or TypeError, or can be the generic Exception class, which catches all types of exceptions.

If no exception is raised, the code in the else block is executed.

The code in the finally block is executed regardless of whether an exception is raised or not.

3) What is the difference between a list and a tuple in Python?

Mutable vs. Immutable: A list is mutable, which means that you can change its contents by adding, removing, or modifying elements. A tuple, on the other hand, is immutable, which means that you cannot change its contents once it is created.

Syntax: A list is enclosed in square brackets [], while a tuple is enclosed in parentheses ().

Performance: Since tuples are immutable, they can be faster than lists when accessing elements. Lists, on the other hand, are faster when it comes to adding or removing elements.

Use cases: Lists are commonly used when you need to store and manipulate data that may change over time. Tuples are often used to store data that should not be changed, such as coordinates, dates, or configuration values.

Here's an example to illustrate the difference between a list and a tuple:

```
# List example

fruits = ['apple', 'banana', 'orange']

fruits.append('pear')

fruits[1] = 'kiwi'

print(fruits) # Output: ['apple', 'kiwi', 'orange', 'pear']


# Tuple example

coordinates = (10, 20)

print(coordinates[0]) # Output: 10

coordinates[0] = 5 # Raises TypeError: 'tuple' object does not support
item assignment
```

4) What is the difference between Python 2 and Python 3?

Python 2 and Python 3 are two major versions of the Python programming language. Python 2 was released in 2000, and Python 3 was released in 2008. Here are some of the key differences between Python 2 and Python 3:

Print statement: In Python 2, the print statement is written as print "Hello, World!". In Python 3, the print statement is written as print("Hello, World!").

Division operator: In Python 2, the division operator / performs integer division if both operands are integers. In Python 3, the division operator always performs true division, returning a float if necessary. Integer division is performed using the // operator.

String handling: In Python 2, strings are represented as ASCII by default, and Unicode strings are represented using the u prefix. In Python 3, strings are represented as Unicode by default, and ASCII strings are represented using the b prefix.

Range function: In Python 2, the range() function returns a list of integers.

In Python 3, the range() function returns an iterable sequence of integers. If you want a list of integers, you can use the list() function.

Error handling: In Python 2, the except statement can catch multiple exception types using a tuple of exception types. In Python 3, the

In Python 3, the except statement can catch multiple exception types using parentheses and multiple except clauses.

Unicode literals: In Python 2, Unicode literals are written as u"Hello, World!". In Python 3, Unicode literals are written as "Hello, World!" with an optional u prefix.

Function arguments: In Python 2, function arguments are passed using the syntax func(arg1, arg2). In Python 3, function arguments can be passed using keyword arguments, like func(arg1=val1, arg2=val2).

Overall, Python 3 is designed to be more consistent, more intuitive, and more modern than Python 2. While Python 2 is still in use, it is no longer being actively developed or maintained, and most new Python projects are now being developed in Python 3.

4) How do you handle multithreading in Python?

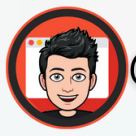
In Python, multithreading can be achieved using the threading module.

Here are the basic steps to handle multithreading in Python:

Import the threading module.

Define a function that will be run in a separate thread. This function should contain the code that you want to execute concurrently.

Create a new thread using the Thread class and pass the function as an argument.



None (NoneType): represents the absence of a value.

These built-in data types are used extensively in Python programming, and there are many built-in functions and operations that can be used with them. In addition, Python allows you to define your own custom data types using classes, which can provide even more flexibility and functionality.

6) What is the use of the pass statement in Python?

In Python, pass is a statement that does nothing. It is used as a placeholder in situations where a statement is required syntactically, but no action is needed.

For example, you might use pass as a placeholder for a function or a class that you plan to implement later:

```
def my_function():

pass # TODO: implement this function later


class MyClass:

pass # TODO: implement this class later
```

In this example, pass is used to indicate that the function or class will be implemented later. Without the pass statement, you would get a syntax error because the function or class definition would be empty.

7) What is the use of the init method in Python?

In Python, __init__ is a special method that is used to initialize objects of a class. This method is called when an object is created from a class and is used to set the initial values of the object's attributes.

The __init__ method takes at least one argument, which is typically named self. self refers to the object that is being created and is used to access the object's attributes and methods. In addition to self, the __init__ method can take any number of additional arguments, which can be used to set the initial values of the object's attributes.

Here's an example of how the __init__ method works:

```
class Person:

def __init__(self, name, age):

self.name = name

self.age = age
```

In this example, we define a Person class with an __init__ method that takes two arguments: name and age. The method sets the object's name and age attributes to the values passed as arguments.

We then create a Person object named person1 with the name "John" and age 30. We can access the object's name and age attributes using dot notation, as shown in the print statements.

PDF Uploaded on Telegram Channel

[Telegram Channel Link in Bio]