BOSSCODER ACADEMY

MOST ASKED

PYTHON

Interview Questions

TABLE OF CONTENTS

Topic 1

Basic data types in Python:

Q.1 What are the basic data types in Python?

Ans: The basic data types in Python are:

- **Integer:** represents whole numbers.
- **Float:** represents decimal numbers.
- **String:** represents a sequence of characters.
- **Boolean:** represents either True or False.
- **List:** represents an ordered collection of elements.

Q.2 How do you convert a string to an integer in Python?

Ans: You can use the int() function to convert a string to an integer. For example:

```
num_str = "10"
num_int = int(num_str)
```

Q.3 How do you check the data type of a variable in Python?

Ans: You can use the **'type()'** function to check the data type of a variable. For example:

```
num = 10
print(type(num))  # Output: <class 'int'>
```

Q.4 What is the difference between a list and a tuple in Python?

Ans: A list is mutable, which means you can modify its elements, while a tuple is immutable, meaning its elements cannot be changed after creation.

Q.5 How do you create an empty dictionary in Python?

Ans: You can create an empty dictionary using either the curly **braces {}** or the **dict()** function. For example:

```
empty_dict = {}
empty_dict = dict()
```

Topic 2

OOPS concept in Python:

Q.1 What is OOPS and how is it implemented in Python?

Ans: Object-Oriented Programming (OOPS) is a programming paradigm that uses objects to represent real-world entities. In Python, OOPS is implemented through classes and objects. Classes are blueprints for creating objects, and objects are instances of a class.

Q.2 What are the four principles of OOPS?

Ans: The four principles of OOPS are:

- **Encapsulation:** bundling of data and methods that operate on that data within a single unit (class).
- **Inheritance:** ability of a class to inherit properties and methods from its parent class.
- **Polymorphism:** ability of an object to take on different forms or behaviors based on the context.
- **Abstraction:** representing essential features and hiding unnecessary details to simplify the complexity.

Q.3 What is method overloading in Python?

Ans: Method overloading in Python refers to defining multiple methods with the same name but different parameters within a class. However, Python does not support method overloading by default as it does in languages like Java. In Python, you can achieve a similar effect by using default arguments or using variable-length arguments.

Q.4 What is method overriding in Python?

Ans: Method overriding in Python refers to defining a method in a child class that already exists in its parent class with the same name and signature. The method in the child class overrides the method in the parent class, providing a different implementation.

Q.5 What is the difference between a class method and an instance method in Python?

Ans: A class method is a method bound to the class and not the instance of the class. It is defined using the **`@classmethod`** decorator and can access only class-level variables. On the other hand, an instance method is bound to the instance of the class and can access both instance and class-level variables.

Topic 3

String handling functions:

Q.1 How do you concatenate two strings in Python?

Ans: You can concatenate two strings using the **+** operator. For example:

```
str1 = "Hello"
str2 = "World"
result = str1 + str2  # Output: "HelloWorld"
```

Q.2 How do you find the length of a string in Python?

Ans: You can use the **`len()`** function to find the length of a string. For example:

```
str1 = "Hello World"
length = len(str1)  # Output: 11
```

Q.3 How do you convert a string to uppercase in Python?

Ans: You can use the **upper()** method to convert a string to uppercase. For example:

```
str1 = "hello"
uppercase_str = str1.upper()  # Output: "HELLO"
```

Q.4 How do you split a string into a list of substrings in Python?

Ans: You can use the split() method to split a string into a list of substrings based on a delimiter. For example:

```
str1 = "Hello,World"
substrings = str1.split(",")
# Output: ["Hello", "World"]
```

Q.5 How do you check if a string contains a specific substring in Python?

Ans: You can use the **in** keyword to check if a substring is present in a string. For example:

```
str1 = "Hello World"
is_present = "World" in str1  # Output: True
```