**Assignment**

**Que-1: What is List? How will you reverse a list?**

**Ans :** In computer programming, a list is a data structure that stores an ordered collection of elements. Each element in a list is associated with an index, which determines its position in the sequence. Lists are commonly used to hold multiple values of the same or different types, and they provide a convenient way to manipulate and organize data.

In many programming languages, including Python, reversing a list means changing the order of its elements so that the first element becomes the last, the second becomes the second-to-last, and so on, until the last element becomes the first. Here's how you can reverse a list in Python:

**python example:**

original\_list = [1, 2, 3, 4, 5]

reversed\_list = original\_list[::-1]

print(reversed\_list)

In this example, the slice notation `[::-1]` creates a new list containing the elements of the `original\_list` in reverse order.

If you want to reverse the list in-place (meaning modify the original list itself without creating a new one), you can use the `reverse()` method:

**Python example:**

original\_list = [1, 2, 3, 4, 5]

original\_list.reverse()

print(original\_list)

**Que-2: Differentiate between append () and extend () methods?**

**Ans :** In Python, the ‘append()’ and ‘extend()’ methods are used to modify lists, but they have different purposes and behaviors. Let's differentiate between the two:

**1. append() Method:**

The ‘append()’ method is used to add a single element to the end of a list. It takes one argument, which is the element to be added. After calling ‘append()’, the length of the list increases by one.

**Python example:**

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list)

Output: [1, 2, 3, 4]

**2. extend() Method:**

The ‘extend()’ method is used to add multiple elements to the end of a list. It takes an iterable (such as a list, tuple, or string) as an argument and adds each element of that iterable to the end of the list. The length of the list increases by the number of elements in the iterable.

**Python example:**

my\_list = [1, 2, 3]

additional\_elements = [4, 5, 6]

my\_list.extend(additional\_elements)

print(my\_list)

Output: [1, 2, 3, 4, 5, 6]

In summary:

- The ‘append()’ method adds a single element (passed as an argument) to the end of the list.

- The ‘extend()’ method adds all elements from an iterable (passed as an argument) to the end of the list.

**Que-3:** **How will you compare two lists?**

**Ans :** You can compare two lists in Python using various methods depending on what you want to achieve. Here are some common approaches:

**1. Element-wise Comparison:**

You can compare lists element by element to check if they are equal. This method checks if both lists have the same elements in the same order.

**Python example:**

list1 = [1, 2, 3]

list2 = [1, 2, 3]

if list1 == list2:

print("The lists are equal.")

else:

print("The lists are not equal.")

**2. Using ‘==’ Operator:**

As shown in the above example, you can directly use the ‘==’ operator to compare two lists. It compares the elements and their order.

**3. Using ‘all()’ Function with ‘zip()’:**

This approach checks if the corresponding elements of both lists are equal using the ‘all()’ function and the ‘zip()’ function.

**Python Example:**

list1 = [1, 2, 3]

list2 = [1, 2, 3]

if all(x == y for x, y in zip(list1, list2)):

print("The lists are equal.")

else:

print("The lists are not equal.")

**4. Comparing Set Representation:**

If you want to check if the lists contain the same elements regardless of their order, you can convert the lists to sets and compare the sets.

**Python example:**

list1 = [1, 2, 3]

list2 = [3, 2, 1]

if set(list1) == set(list2):

print("The lists have the same elements.")

else:

print("The lists do not have the same elements.")

**Que-4:** **What is tuple? Difference between list and tuple.**

**Ans :**

| **Feature** | **Tuple** | **List** |
| --- | --- | --- |
| **Definition** | An ordered collection of elements. | An ordered collection of elements. |
| **Syntax** | Defined using parentheses **( )**. | Defined using square brackets **[ ]**. |
| **Mutability** | Immutable. Elements cannot be changed. | Mutable. Elements can be changed. |
| **Example** | **(1, 2, "hello")** | **[1, 2, "world"]** |
| **Performance** | Generally slightly faster due to immutability. | May have slightly more overhead due to mutability. |
| **Common Use Cases** | Grouping related data together. | Storing dynamic collections, allowing modifications. |
| **Iteration/Access** | Similar to lists. Elements accessed by index. | Similar to tuples. Elements accessed by index. |
| **Adding Elements** | Not applicable. Tuples are immutable. | Elements can be added using methods like **append()**, **extend()**. |
| **Removing Elements** | Not applicable. Tuples are immutable. | Elements can be removed using methods like **remove()**, **pop()**. |
| **Memory Efficiency** | Tends to be more memory-efficient due to immutability. | May consume slightly more memory due to mutability. |

**Que-5: How will you create a dictionary using tuples in python?**

**Ans :** You can create a dictionary using tuples in Python by using a list of tuples as the input to the ‘dict()’ constructor or by using dictionary comprehension. Each tuple within the list should contain two elements: the key and the corresponding value for the dictionary entry.

Here are two ways to create a dictionary using tuples:

**1. Using a List of Tuples and the ‘dict()’ Constructor:**

**Python example:**

tuple\_list = [("apple", 3), ("banana", 5), ("orange", 2)]

my\_dict = dict(tuple\_list)

print(my\_dict)

**Output:**

{'apple': 3, 'banana': 5, 'orange': 2}

**2. Using Dictionary Comprehension:**

**Python example:**

tuple\_list = [("apple", 3), ("banana", 5), ("orange", 2)]

my\_dict = {key: value for key, value in tuple\_list}

print(my\_dict)

**Output:**

{'apple': 3, 'banana': 5, 'orange': 2}

**Que-6: Why Do You Use the Zip () Method in Python?**

**Ans : python zip() method** takes iterable containers and returns a single iterator object, having mapped values from all the containers.

It is used tomap the similar index of multiple containers so that they can be used just using a single entity.

**Que-7: How Many Basic Types Of Functions Are Available In Python?**

**Ans : Python Functions** is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

**There are mainly two types of functions in Pyhton:**

* **Built-in library function:** These are [Standard functions](https://www.geeksforgeeks.org/python-built-in-functions/) in Python that are available to use.
* **User-defined function:** We can create our own functions based on our requirements.