Lists

A list in Python is used to store the sequence of various types of data. Python lists are mutable type its mean we can modify its element after it created. However, Python consists of six data-types that are capable to store the sequences, but the most common and reliable type is the list.

A list can be defined as a collection of values or items of different types. The items in the list are separated with the comma (,) and enclosed with the square brackets [].

Ex:

1. L1 = ["John", 102, "USA"]
2. L2 = [1, 2, 3, 4, 5, 6]

Characteristics of Lists

The list has the following characteristics:

* The lists are ordered.
* The element of the list can access by index.
* The lists are the mutable type.
* The lists are mutable types.
* A list can store the number of various elements.

## **List indexing and splitting**

The indexing is processed in the same way as it happens with the strings. The elements of the list can be accessed by using the slice operator [].

The index starts from 0 and goes to length - 1. The first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index, and so on.

1. list = [1,2,3,4,5,6,7]
2. **print**(list[0])
3. **print**(list[1])
4. # Slicing the elements
5. **print**(list[0:6])
6. # By default the index value is 0 so its starts from the 0th element and go for index -1.
7. **print**(list[:])
8. **print**(list[2:5])
9. **print**(list[1:6:2])

## **Updating List values**

Lists are the most versatile data structures in Python since they are mutable, and their values can be updated by using the slice and assignment operator.

1. list = [1, 2, 3, 4, 5, 6]
2. **print**(list)
3. # It will assign value to the value to the second index
4. list[2] = 10
5. **print**(list)
6. # Adding multiple-element
7. list[1:3] = [89, 78]
8. **print**(list)

## **Adding elements to the list**

Python provides append() function which is used to add an element to the list. However, the append() function can only add value to the end of the list.

1. l =[]
2. #Number of elements will be entered by the user
3. n = int(input("Enter the number of elements in the list:"))
4. # for loop to take the input
5. **for** i **in** range(0,n):
6. # The input is taken from the user and added to the list as the item
7. l.append(input("Enter the item:"))
8. **print**("printing the list items..")
9. # traversal loop to print the list items
10. **for** i **in** l:
11. **print**(i, end = "  ")

## **Basic List Operations**

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| len([1, 2, 3]) | 3 | Length |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
| ['Hi!'] \* 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
| 3 in [1, 2, 3] | True | Membership |
| for x in [1, 2, 3]: print x, | 1 2 3 | Iteration |

# **Python Tuple**

Python Tuple is used to store the sequence of immutable Python objects. The tuple is similar to lists since the value of the items stored in the list can be changed, whereas the tuple is immutable, and the value of the items stored in the tuple cannot be changed.

## **Creating a tuple**

A tuple can be written as the collection of comma-separated (,) values enclosed with the small () brackets. The parentheses are optional but it is good practice to use. A tuple can be defined as follows.

1. T1 = (101, "Peter", 22)
2. print(type(T1))

## **Tuple indexing and slicing**

The indexing and slicing in the tuple are similar to lists. The indexing in the tuple starts from 0 and goes to length(tuple) - 1.

The items in the tuple can be accessed by using the index [] operator. Python also allows us to use the colon operator to access multiple items in the tuple.

1. tup = (1,2,3,4,5,6,7)
2. print(tup[0])
3. print(tup[1])
4. tuple = (1,2,3,4,5,6,7)
5. #element 1 to end
6. print(tuple[1:])
7. #element 0 to 3 element
8. print(tuple[:4])
9. #element 1 to 4 element
10. print(tuple[1:5])
11. # element 0 to 6 and take step of 2
12. print(tuple[0:6:2])

## **List vs. Tuple**

|  |  |  |
| --- | --- | --- |
| **SN** | **List** | **Tuple** |
| 1 | The literal syntax of list is shown by the []. | The literal syntax of the tuple is shown by the (). |
| 2 | The List is mutable. | The tuple is immutable. |
| 3 | The List has the a variable length. | The tuple has the fixed length. |
| 4 | The list provides more functionality than a tuple. | The tuple provides less functionality than the list. |
| 5 | The list is used in the scenario in which we need to store the simple collections with no constraints where the value of the items can be changed. | The tuple is used in the cases where we need to store the read-only collections i.e., the value of the items cannot be changed. It can be used as the key inside the dictionary. |
| 6 | The lists are less memory efficient than a tuple. | The tuples are more memory efficient because of its immutability. |

# **Python Dictionary**

Python Dictionary is used to store the data in a key-value pair format. The dictionary is the data type in Python, which can simulate the real-life data arrangement where some specific value exists for some particular key. It is the mutable data-structure. The dictionary is defined into element Keys and values.

* Keys must be a single element
* Value can be any type such as list, tuple, integer, etc.

In other words, we can say that a dictionary is the collection of key-value pairs where the value can be any Python object. In contrast, the keys are the immutable Python object, i.e., Numbers, string, or tuple.

## **Creating the dictionary**

The dictionary can be created by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:)

1. # Creating an empty Dictionary
2. Dict = {}
3. **print**("Empty Dictionary: ")
4. **print**(Dict)
6. # Creating a Dictionary
7. # with dict() method
8. Dict = dict({1: 'Java', 2: 'T', 3:'Point'})
9. **print**("\nCreate Dictionary by using  dict(): ")
10. **print**(Dict)
12. # Creating a Dictionary
13. # with each item as a Pair
14. Dict = dict([(1, 'Devansh'), (2, 'Sharma')])
15. **print**("\nDictionary with each item as a pair: ")
16. **print**(Dict)

## **Properties of Dictionary keys**

1. In the dictionary, we cannot store multiple values for the same keys. If we pass more than one value for a single key, then the value which is last assigned is considered as the value of the key.

2. In python, the key cannot be any mutable object. We can use numbers, strings, or tuples as the key, but we cannot use any mutable object like the list as the key in the dictionary.

## **Accessing the dictionary values**

Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}

**print**(type(Employee))

**print**(Employee["Name"])

**print**(Employee["Age"])

## **Adding dictionary values**

The dictionary is a mutable data type, and its values can be updated by using the specific keys. The value can be updated along with key **Dict[key] = value**. The update() method is also used to update an existing value

1. Dict = {}
2. **print**(Dict)
3. # Adding elements to dictionary one at a time
4. Dict[0] = 'Peter'
5. Dict[2] = 'Joseph'
6. Dict[3] = 'Ricky'
7. **print**(Dict)