untitled5

September 9, 2024

1 Python List

- Lists are used to store multiple items in a single variable.
- list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [ ] .
- The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.
- A list can contain different data types # Example for list with different data types:

[4]:
```
list = [1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
print(list)
```

```
[1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
```

2 Different types of lists with different datatypes

[5]:
```
list1 = [1, 2, 3, 4, 5]
list2 = [1.1, 2.2, 3.3, 4.4, 5.5]
list3 = ["apple", "banana", "cherry", "date", "elderberry"]
list4 = [True, False, True, False, True]
list5 = [1, "apple", True, 3.14]
list6 = [1 + 2j, 2 + 3j, 3 + 4j, 4 + 5j, 5 + 6j]
list7 = [[1, 2], [3, 4], [5, 6]]
list8 = [51, 72, 67, 84, 96]
list9 = [[1, 2], [3, 4]]
list10 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(list1)
print(list2)
print(list2)
print(list3)
print(list4)
print(list5)
print(list6)
print(list7)
```

```
print(list8)
print(list9)
print(list10)
```

```
[1, 2, 3, 4, 5]
[1.1, 2.2, 3.3, 4.4, 5.5]
[1.1, 2.2, 3.3, 4.4, 5.5]
['apple', 'banana', 'cherry', 'date', 'elderberry']
[True, False, True, False, True]
[1, 'apple', True, 3.14]
[(1+2j), (2+3j), (3+4j), (4+5j), (5+6j)]
[[1, 2], [3, 4], [5, 6]]
[51, 72, 67, 84, 96]
[[1, 2], [3, 4]]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

3 Indexing in python

- list index method is used to find position of element in list Python.
- It returns the position of the first occurrence of that element in the list. If the item is not found in the list, index() function raises a "ValueError" error.

[12]:
```
list = [1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
print(list[0])
print(list[1])
print(list[2])
print(list[3])
print(list[4])
print(list[5])
print(list[6])
```

```
1
3.14
hello
True
[1, 2, 3]
{'key': 'value'}
None
```

4 Slicing of list

- List slicing is the process of accessing a specified portion or subset of a list for some action while leaving the rest of the list alone.

[18]:
```
list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(list[0])
```

```
print(list[-2])
print(list[:6])
print(list[3:6])
print(list[5:])
print(list[:])
```

```
1
9
[1, 2, 3, 4, 5, 6]
[4, 5, 6]
[6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

5 positive indexes

In Python list indexing is a way to access individual elements within a list the Positive indexing refers to accessing elements from the beginning of the list starting with index 0 for the first element. # Negative index Negative indexing in Python allows you to access list elements from the end of the list it starts at -1 for the last element.

[19]:
```
list = [10, 20, 30, 40, 50, 60, 70]
print(list[:4]) # positive slicing
print(list[-4:]) # negative slicing
```

```
[10, 20, 30, 40]
[40, 50, 60, 70]
```

[21]:
```
list = [1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
print(list[:7])
print(list[:-2])
print(list[1:])
print(list[-7:])
```

```
[1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
[1, 3.14, 'hello', True, [1, 2, 3]]
[3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
[1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
```

6 List Operations

Concatenation of list

Concatenation of lists in Python refers to the process of joining two or more lists together to form a new, single list. This is done using the + operator.

Example

[22]:
```
list1 = [1, 2, 3, 4]
list2 = [5, 6, 7, 8, 9]
list3 = list1 + list2
print(list3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[23]:
```
list1 = [10, 20, 30, 40]
list2 = [50, 60, 70, 80, 90]
list1 += list2
print(list1)
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

7 Repetition of Lists

Repetition of lists in Python means creating a new list by repeating the original list a certain number of times. This is done using the * operator.

Example

[27]:
```
lst_1 = [10, 20, 30, 40, 50]
lst_2 = lst_1 * 4
print(lst_2)
```

```
[10, 20, 30, 40, 50, 10, 20, 30, 40, 50, 10, 20, 30, 40, 50, 10, 20, 30, 40, 50]
```

[30]:
```
list_3 = [1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None]
list_4 = list_3 * 3
print(list_4)
```

```
[1, 3.14, 'hello', True, [1, 2, 3], {'key': 'value'}, None, 1, 3.14, 'hello',
True, [1, 2, 3], {'key': 'value'}, None, 1, 3.14, 'hello', True, [1, 2, 3],
{'key': 'value'}, None]
```