1- List

Python lists are a versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Lists are used to store multiple items in a single variable

In [1]:
```
# empty list
First_list = []
```

In [2]:
```
First_list2 = list()
```

list contains a list of elements, such as strings, integers, objects or a mixture of types.

1.1- Create List

In [7]:
```
my_list = [1, 2, 3]
my_list
```

Out[7]:
```
[1, 2, 3]
```

In [8]:
```
my_list2 = ["a", "b", "c"]
my_list2
```

Out[8]:
```
['a', 'b', 'c']
```

In [9]:
```
my_list3 = ["a", 1, "Python", 5]
my_list3
```

Out[9]:
```
['a', 1, 'Python', 5]
```

1.2- Nested List

In [10]:
```
my_list = [1, 2, 3]
my_list2 = ["a", "b", "c"]
my_list3 = ["a", 1, "Python", 5]
my_nested_list = [my_list, my_list2, my_list3]
print(my_nested_list)
```

```
[[1, 2, 3], ['a', 'b', 'c'], ['a', 1, 'Python', 5]]
```

1.3- List Extend Method

In [12]:
```
world_list = []
one_list = [4, 5]
world_list.extend(one_list)
world_list
```

Out[12]:
```
[4, 5]
```

In [13]:
```
# Add two  lists
my_list = [1, 2, 3]
my_list2 = ["a", "b", "c"]
world_list_1 = my_list + my_list2
world_list_1
```

Out[13]:
```
[1, 2, 3, 'a', 'b', 'c']
```

1.4- Sorting out a list

In [14]:
```
a_list = [34, 23, 67, 100, 88, 2]
a_list.sort()
a_list
```

Out[14]:
```
[2, 23, 34, 67, 88, 100]
```

In [15]:
```
# A none list
a_list = [34, 23, 67, 100, 88, 2]
sorted_list = a_list.sort()
sorted_list
print(sorted_list)
```

```
None
```

However, when you call the sort() method on a list, it sorts the list in place. So if we try to assign the result to another variable, then we'll find out that we'll get a None object, which is like a Null in other languages. Thus when we want to sort something, just remember that we sort them in-place and we cannot assign it to a different variable

1.5- Slicing a list

In [17]:
```
a_list = [34, 23, 67, 100, 88, 2]
a_list[0:3]
```

Out[17]:
```
[34, 23, 67]
```

2- Tuples

Python Tuple is a collection of objects separated by commas. In some ways, a tuple is similar to a Python list in terms of indexing, nested objects, and repetition but the main difference between both is Python tuple is immutable😊

In [18]:
```
m_tuple=(1,2,3,4,5)
m_tuple[0:4]
```

Out[18]:
```
(1, 2, 3, 4)
```

In [19]:
```
# an empty
another_tuple=tuple()
```

2.1- Tuple casting

In [21]:
```
my_tuple=tuple([1,2,3])
my_tuple
```

Out[21]:
```
(1, 2, 3)
```

3- Dictionaries

Dictionaries are used to store data values in key:value pairs.

In [22]:
```
# create empty dict
my_dict = {}
```

In [23]:
```
another_dict = dict()
```

In [24]:
```
# dict with values
other_dict = {"one":1, "two":2, "three":3}
other_dict
```

Out[24]:
```
{'one': 1, 'two': 2, 'three': 3}
```

3.1- Accessing a value within a dictionary

In [25]:
```
my_dict = {"name":"ali", "address":"Uk 123"}
my_dict["name"]
```

Out[25]:
```
'ali'
```

In [29]:
```
 # Adding true and false statement
```

In [27]:
```
"name" in my_dict
```

Out[27]:
```
True
```

In [28]:
```
"state" in my_dict
```

Out[28]:
```
False
```

In [30]:
```
# Checking all the keys in a dictionary
my_dict.keys()
```

Out[30]:
```
dict_keys(['name', 'address'])
```

In [31]:
```
# Checking values in dict
my_dict.values()
```

Out[31]:
```
dict_values(['ali', 'Uk 123'])
```

In [32]:
```
# The importance of in
"name" in my_dict
```

Out[32]:
```
True
```

In [33]:
```
"name" in my_dict.keys()
```

Out[33]:
```
True
```

In [ ]: