Python Collections or Data Structures (Arrays)

- There are four collection data types in the Python programming language:
- List is a collection which is ordered and changeable. Allows duplicate members.[]
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.()
- Set is a collection which is unordered and unindexed. No duplicate members.{}
- Dictionary(associative arrays) is a collection which is unordered, changeable and indexed. No duplicate members.{}

Dictionary

- A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

In [4]:
```
list_a = [11,22,33,44]
set_a = {1,2}
dict = {"key1":"value1","key2":"value2",}
dict
# own index ( keys) - key-value pair data set.
```

Out[4]:
```
{'key1': 'value1', 'key2': 'value2'}
```

In [5]:
```
for k,v in dict.items():
  print('keys : ',k)
  print('values : ',v)
```

```
keys :  key1
values :  value1
keys :  key2
values :  value2
```

In [6]:
```
# items - both keys and values
# keys - only keys
# values - only values
# get()
```

In [7]:
```
dict_a = {'a':10,'b':11,'c':13,'d':True}
dict_a
```

Out[7]:
```
{'a': 10, 'b': 11, 'c': 13, 'd': True}
```

In [11]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2019,"year": 2021}
print(thisdict)
type(thisdict)# not allowing duplicates and returning recently added item
```

```
{'brand': 'Mahindra', 'model': 'XUV300', 'year': 2021}
```

Out[11]:
```
dict
```

In [12]:
```
for a in thisdict:
  print('dict keys are : ',a)
  print('dict values are : ',thisdict[a])
```

```
dict keys are :  brand
dict values are :  Mahindra
dict keys are :  model
dict values are :  XUV300
dict keys are :  year
dict values are :  2021
```

In [18]:
```
for i,j in thisdict.items():
  print("dict values : ",(i,j))
```

```
dict values :  ('brand', 'Mahindra')
dict values :  ('model', 'XUV300')
dict values :  ('year', 2021)
```

- Accessing Items
- You can access the items of a dictionary by referring to its key name, inside square brackets

In [19]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2019}
thisdict['model']
```

Out[19]:
```
'XUV300'
```

In [20]:
```
thisdict.items()

# items() - key-value pair un-packed
# values() - only values
# keys() - only keys
# get(key) - return value
```

Out[20]:
```
dict_items([('brand', 'Mahindra'), ('model', 'XUV300'), ('year', 2019)])
```

- There is also a method called **`get()`** that will give you the same result

In [21]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2019}
print(thisdict["brand"])
```

```
Mahindra
```

- **Change Values**
- You can change the value of a specific item by referring to its key name

In [22]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2019}
thisdict["year"] = "2021"
print('Updated Year key value Is : ',thisdict["year"])
print(thisdict)
```

```
Updated Year key value Is :  2021
{'brand': 'Mahindra', 'model': 'XUV300', 'year': '2021'}
```

In [23]:
```
dicta ={'a':1,'b':2,'c':3}
dir(dicta)
```

Out[23]:
```
['__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__ior__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__ne__',
 '__new__',
 '__or__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__ror__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'clear',
 'copy',
 'fromkeys',
 'get',
 'items',
 'keys',
 'pop',
 'popitem',
 'setdefault',
 'update',
 'values']
```

- **Loop Through a Dictionary**
- You can loop through a dictionary by using a for loop.
- When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

In [24]:
```
thisdict["model"]
```

Out[24]:
```
'XUV300'
```

In [25]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2019}
#Print all key names in the dictionary, one by one:
for a in thisdict:
  print("Dictinary Key Name : ",a)
  print('Dictionary Value  :',thisdict[a])
```

```
Dictinary Key Name :  brand
Dictionary Value  : Mahindra
Dictinary Key Name :  model
Dictionary Value  : XUV300
Dictinary Key Name :  year
Dictionary Value  : 2019
```

In [26]:
```
print(thisdict['year'])
```

```
2019
```

In [27]:
```
#Print all values in the dictionary, one by one
for x in thisdict:
  print('Dictionary key Name: ',x)
  print('Dictionary key values: ',thisdict[x])
```

```
Dictionary key Name:  brand
Dictionary key values:  Mahindra
Dictionary key Name:  model
Dictionary key values:  XUV300
Dictionary key Name:  year
Dictionary key values:  2019
```

- You can also use the values() method to return values of a dictionary

In [29]:
```
for a in thisdict.values():
    print('Keys :',a)
```

```
Keys : Mahindra
Keys : XUV300
Keys : 2019
```

- You can also use the items() method to return keys and values of a dictionary

In [31]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2019}
for a,b in thisdict.items():
    print('Keys :',a)
    print('Values : ',b)
```

```
Keys : brand
Values :  Mahindra
Keys : model
Values :  XUV300
Keys : year
Values :  2019
```

- Getting key names using keys() method

In [32]:
```
for x in thisdict.keys():
    print('Dictionary Key is : ',x)
```

```
Dictionary Key is :  brand
Dictionary Key is :  model
Dictionary Key is :  year
```

- **Check if Key Exists**
- To determine if a specified key is present in a dictionary use the in keyword

In [33]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
y_col="model"
```

```
if  v_col in thisdict:
    print("Yes, 'model' is one of the keys in the thisdict dictionary")
else:
    print("NO Value is not available in Dict")
```

```
Yes, 'model' is one of the keys in the thisdict dictionary
```

- **Dictionary Length**
- To determine how many items (key-value pairs) a dictionary has, use the len() function.

In [34]:
```
print(len(thisdict))
```

```
3
```

- **Adding Items**
- Adding an item to the dictionary is done by using a new index key and assigning a value to it

In [35]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
thisdict["color"] = "red"
thisdict["year"] = 2021
print(thisdict)
```

```
{'brand': 'Mahindra', 'model': 'XUV300', 'year': 2021, 'color': 'red'}
```

- **Removing Items**
- There are several methods to remove items from a dictionary

In [36]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
#The pop() method removes the item with the specified key name
print("Before removing Dict items :",thisdict)
thisdict.pop("year")
print('After removing item : ',thisdict)
```

```
Before removing Dict items : {'brand': 'Mahindra', 'model': 'XUV300', 'year': 2020}
After removing item :  {'brand': 'Mahindra', 'model': 'XUV300'}
```

- The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead)

In [37]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": [2020,2019.2021]}
print("Before removing Dict items :",thisdict)
thisdict.popitem()
print('after removing dit items : ',thisdict)
```

```
Before removing Dict items : {'brand': 'Mahindra', 'model': 'XUV300', 'year': [2020, 201
9.2021]}
after removing dit items :  {'brand': 'Mahindra', 'model': 'XUV300'}
```

- The del keyword removes the item with the specified key name

In [38]:
```
a_list=[1,2,3,4,5,5,6]
print(a_list)
del a_list[2]
print(a_list)
```

```
[1, 2, 3, 4, 5, 5, 6]
[1, 2, 4, 5, 5, 6]
```

In [39]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
print(thisdict)
del thisdict["model"]
print(thisdict)
```

```
{'brand': 'Mahindra', 'model': 'XUV300', 'year': 2020}
{'brand': 'Mahindra', 'year': 2020}
```

- The del keyword can also delete the dictionary completely

In [40]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
del thisdict
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [40], in <cell line: 3>()
      1 thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
      2 del thisdict
----> 3 print(thisdict)

NameError: name 'thisdict' is not defined
```

In [41]:
```
help(thisdict)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [41], in <cell line: 1>()
----> 1 help(thisdict)

NameError: name 'thisdict' is not defined
```

- The clear() method empties the dictionary

In [42]:
```
# len
# del (delete)
# this all are python common functions we can use or apply on any object like vairbales,
```

In [43]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
thisdict.clear()
print(thisdict)
type(thisdict)
```

Out[43]:
```
{}
dict
```

- **Copy a Dictionary**
- You cannot copy a dictionary simply by typing dict2 = dict1, because: dict2 will only be a reference to dict1, and changes made in dict1 will automatically also be made in dict2.

- There are ways to make a copy, one way is to use the built-in Dictionary method copy()

In [44]:
```
thisdict = {"brand": "Mahindra","model": "XUV300","year": 2020}
print(thisdict)
mydict = thisdict.copy()
print(mydict)
type(mydict)
```

```
{'brand': 'Mahindra', 'model': 'XUV300', 'year': 2020}
{'brand': 'Mahindra', 'model': 'XUV300', 'year': 2020}
dict
```

Out[44]:

- Make a copy of a dictionary with the dict() function

In [45]:
```
mydict = dict(thisdict)
print(mydict)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [45], in <cell line: 1>()
----> 1 mydict = dict(thisdict)
      2 print(mydict)

TypeError: 'dict' object is not callable
```

Nested Dictionaries

- A dictionary can also contain many dictionaries, this is called nested dictionaries.

In [46]:
```
myFriends = {"friend1" : {"name" : "Raj","year" : 1981},
             "friend2" : {"name" : "Prasad","year" : [1984,1955]},
             "friend3" : {"name" : "Mahesh","year" : 1985},
             "friend4" : {"name" : "Sridhar","year" : 1986}
            }
print(myFriends['friend2']["year"])
```

```
[1984, 1955]
```

In [47]:
```
myFriends["friend2"]["year"]
```

Out[47]:
```
[1984, 1955]
```