Dictionaries in Python

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

A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

Dictionaries are defined using curly braces {}

As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

Ordered mean that element will be kept in same sequence as they were inserted.

In [34]:
```
studentInfo = {
    "SP20M2BB008": "Ghulam Jillani",
    "SP20M2BB030": "Muhammad  Nasir",
    "SP20M2BB032":"Ghulam Murtaza",
    "SP20M2BB033":"Jibran Jaffar",
    "SP20M2BB073":"Mohammad Awais",
    "SP20M2BB079":"Jahantab Siddique"
}
print(studentInfo)
```

```
{'SP20M2BB008': 'Ghulam Jillani', 'SP20M2BB030': 'Muhammad  Nasir', 'SP20M2BB032': 'Ghulam Murtaza', 'SP20M2BB033':
'Jibran Jaffar', 'SP20M2BB073': 'Mohammad Awais', 'SP20M2BB079': 'Jahantab Siddique'}
```

Accessing Element of Dictionary

```
# we can't access elements of dictionary as we access the element of list, because dictionaries don't
work in index, instead as given above we use our custom keys to store data in it. so we can use keys to access
dictionary elements instead of index. we still access element of dictionary using [] operator but this time we will
place key instead of index as,
```

In [35]:
```
print(studentInfo['SP20M2BB030'])
```

```
Muhammad  Nasir
```

```
As shown above we use key to access the actual element this is how we can access element in dictionary.
```

Adding Element in Dictionary

```
we can add element in dictionary using different methods, the one used commonly is,direct assignment as,
dictVar["newKey"] = value
```

In [87]:
```
dictVar = {
    "key1" : 1,
    "key2" : 2
}
dictVar["key3"] = 3
print(dictVar)
```

```
{'key1': 1, 'key2': 2, 'key3': 3}
```

Update method (Add / Replace Values)

we can also add values or replace values using update() method it takes an object of dictionary as well,

In [88]:
```
dictVar = {
    "key1" : 1,
    "key2" : 2
}
updateDict = { #a dict object we will use to update values of dictVar, we can pass this object directly as well
    #in update method
    "key2" : 4,
    "key4" : 5,
    "Key5" : 6
}
dictVar.update(updateDict)
print(dictVar)
```

```
{'key1': 1, 'key2': 4, 'key4': 5, 'Key5': 6}
```

In [89]:
```
#Example 2
myDict = {
    "k1" : "This is val1",
    "K2": "This is val2",
}
print("Before Update:{}".format(myDict))
myDict.update({"k1":"This is first key value"}) #update value of k1
print("After Update:{}".format(myDict))
myDict.update({"k3":"New Val3 Added"}) #here we are adding another key with value using update
print("After 2nd Update:{}".format(myDict))
```

```
Before Update:{'k1': 'This is val1', 'K2': 'This is val2'}
After Update:{'k1': 'This is first key value', 'K2': 'This is val2'}
```

```
After 2nd Update:{'k1': 'This is first key value', 'K2': 'This is val2', 'k3': 'New Val3 Added'}
```

Dictionary Comprehension

Dictionary comprehension in Python is a compact and efficient way to create a new dictionary from an existing iterable. It consists of an expression followed by a for clause, and is written in curly braces {}.
The expression creates key-value pairs for each element in the iterable and adds them to the new dictionary.
In other word we can create a dictionary using list[(t1,v1),(t2,v2)] list of touples which we can get from another dictionary using dic.items() or we can also create our own

In [102]:
```
dic = {
    "name": "Muhammad Nasir",
    "age": 23,
    "city": "Bahawalpur"
}
dic2 = {
    "name": "Muhammad Sabir",
    "age": 23,
    "city": "Kehror Pacca"
}
#providing our own list of tuple
dic3 ={key: value for key, value in [(1,2)]}
print(dic3)
#using list of tuple returned by items() method of dictionary
dic4 ={key: value for key, value in dic.items()}
print(dic4)
```

```
{1: 2}
{'name': 'Muhammad Nasir', 'age': 23, 'city': 'Bahawalpur'}
```

get method

we can also use get method of dictionary class to get the value of provided key as,

In [18]:
```
studentInfo.get("SP20M2BB008")
```

Out[18]:
```
'Ghulam Jillani'
```

Get a list of keys using keys() method

there is a method named keys() of dictionary class which return us an object of class dict_keys which contains all the keys which are itratable, we can use a loop to iterate through each key and can access assosiated element using dict as,

In [40]:
```
myKeys = studentInfo.keys()
print(type(myKeys))
for key in myKeys:
    print(key,studentInfo[key])
```

```
<class 'dict_keys'>
SP20M2BB008 Ghulam Jillani
SP20M2BB030 Muhammad  Nasir
SP20M2BB032 Ghulam Murtaza
SP20M2BB033 Jibran Jaffar
SP20M2BB073 Mohammad Awais
SP20M2BB079 Jahantab Siddique
SP20M2BB027 Muhammad Islam
```

as shown above the type of returned object is dict_keys and we use a for loop to iterate through each key and print the value of key as well as we print key also.

Get All Values

there is a method named values() in dictionary class which we can use to get all values of dictionary in same way as we get the keys in above section

In [42]:
```
myValues = studentInfo.values()
print(type(myValues))
for value in myValues:
    print(value)
studentInfo["SP20M2BB027" ]= "Muhammad Islam"
```

```
<class 'dict_values'>
Ghulam Jillani
Muhammad  Nasir
Ghulam Murtaza
Jibran Jaffar
Mohammad Awais
Jahantab Siddique
Muhammad Islam
```

as shown above the values are printed only because we print only values and the values() method returns the values of dictionary.

Note

the values() or keys() method both does not return a new object, instead they return a reference to the object which is holding either keys or values, which means that if we get key/values using keys()/values() and store that inside a variable, and later on we made a change in our dictionary that change will also reflect in that variable or keys object as.

In [47]:
```
myDict = {
    "key1" : "value1",
}
keys = myDict.keys()
values = myDict.values()
#we stored values and keys inside a variables let print them
print("keys: {}".format(keys))
print("values: {}".format(values))
#lets add an element in dictionary
print("adding value in dictionary")
myDict["key2"] = "value2"
#lets print the keys and values again
print("keys: {}".format(keys))
print("values: {}".format(values))
```

```
keys: dict_keys(['key1'])
values: dict_values(['value1'])
adding value in dictionary
keys: dict_keys(['key1', 'key2'])
values: dict_values(['value1', 'value2'])
```

as shown above when we add value in our dictionary and then print the keys and value object
they are updated according to the dictionary because they are just refering to one of key / value object.
to avoid this behaviour we can use list constructor so that we can also perform list operation on these object as,

In [50]:
```
myDict = {
    "key1" : "value1",
}
keys = list(myDict.keys())
values =list( myDict.values())
#we stored values and keys inside a variables let print them
print("keys: {}".format(keys))
print("values: {}".format(values))
#lets add an element in dictionary
print("adding value in dictionary")
myDict["key2"] = "value2"
#lets print the keys and values again
print("keys: {}".format(keys))
print("values: {}".format(values))
print("length of dictionary: {}".format(type(myDict)))
print(values[-1])
```

```
keys: ['key1']
values: ['value1']
adding value in dictionary
keys: ['key1']
values: ['value1']
length of dictionary: <class 'dict'>
value1
```

When we convert the return dict_keys or dict_values obect into list and store it into a variable
it will not be updated when we add values in dictionary because it is now a completely new object and at end we
used subscript operator [] to get the last value stored in values by passing -1 index in it.

Getting keys and values in single object

In [54]:
```
#The items() method will return each item in a dictionary, as tuples as [(1,10),(2,20)] where 1 and 2 are keys
#with their respective values, it returns a dict_items object not a list object but can be converted into list as
#well
studentInfo = {
    "SP20M2BB008": "Ghulam Jillani",
    "SP20M2BB030": "Muhammad  Nasir",
    "SP20M2BB032":"Ghulam Murtaza",
    "SP20M2BB033":"Jibran Jaffar",
    "SP20M2BB073":"Mohammad Awais",
    "SP20M2BB079":"Jahantab Siddique"
}
items = studentInfo.items()
print(type(items))
#lets convert dict_items object in list as,
myItemsList = list(items)
#print list
print(myItemsList)
```

```
<class 'dict_items'>
[('SP20M2BB008', 'Ghulam Jillani'), ('SP20M2BB030', 'Muhammad  Nasir'), ('SP20M2BB032', 'Ghulam Murtaza'), ('SP20M2
BB033', 'Jibran Jaffar'), ('SP20M2BB073', 'Mohammad Awais'), ('SP20M2BB079', 'Jahantab Siddique')]
```

In [ ]:
```
```

as shown above each record is surrouned in a tule as (key,value) now we can use for loop (we can also use for loop without converting it into list as)

In [55]:
```
for key,value in items:
    print(key,value)
```

```
SP20M2BB008 Ghulam Jillani
SP20M2BB030 Muhammad  Nasir
SP20M2BB032 Ghulam Murtaza
SP20M2BB033 Jibran Jaffar
SP20M2BB073 Mohammad Awais
SP20M2BB079 Jahantab Siddique
```

In [56]:
```
#or we can convert it into list in case we want to perform list specific opeartiong as getting item
#Using index etc as,
list2 = list(items)
```

```
print(list2[-1]) #print last item of list
```

```
('SP20M2BB079', 'Jahantab Siddique')
```

Duplicate Values in Dictionary

```
duplicate values are not allowed in dictionary as we can store duplicate values in list, this to notice here is that
actually duplicate keys are not allowed but we can store duplicate values in different keys but it does not meke any
sense of storing same value in different places,
```

In [10]:
```
duplicateCheck={
    "key1" : "This is key1 Value",
    "key1" : "This is key1 duplicate value", #here we are using key1 again which will override the previous value
    #of key1
    "key2" : "This is key2 Value",
    "key3" : "This is key3 Value",
    "key4" : "This is key4 Value",
}
print(duplicateCheck)
```

```
{'key1': 'This is key1 duplicate value', 'key2': 'This is key2 Value', 'key3': 'This is key3 Value', 'key4': 'This
is key4 Value'}
```

```
Now as shown above when we use key1 two times it basically replace the value stored in "key1" with the
last record whose key is key1, so basically when we use same key for multiple record only the last
record will store in that key, actually it is same as,
x = 4
x = 5
x = 6
x = 7
here our key is x so the last assigned value 7 will be stored and all other will be thrown away.
```

Changeable

```
Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.
```

In [12]:
```
myDict = {
    "name" : "Mohammad Nasir",
    "roll" : "SP20M2BB030",
    "class":"BSCS 6th E"
}
#lets change the value of "class" key
myDict["class"] = "BSCS 7th E"
print(myDict)
```

```
{'name': 'Mohammad Nasir', 'roll': 'SP20M2BB030', 'class': 'BSCS 7th E'}
```

```
as shown above the data inside class key is changed
```

In [13]:
```
#length of dictionary
print(len(myDict))
```

```
3
```

Dict Constructor

```
dcit(key = value,key = value) can be used to create a dictionary object as,
```

In [59]:
```
myDict = dict(name = "Muhammad Nasir", age = 23, role = "Student", occupation = "Flutter Developer")
print(myDict)
```

```
{'name': 'Muhammad Nasir', 'age': 23, 'role': 'Student', 'occupation': 'Flutter Developer'}
```

```
as shown abvoe the dictionary object is printed as we created one using dict() constructor
```

In [60]:
```
print(myDict["name"])
```

```
Muhammad Nasir
```

use of if in

we can use if in to check if the given keys exist in specified dictionary or not, for example

In [61]:
```
if "name" in myDict:
    print("Name exist in dictionary myDict and its value is:{}".format(myDict['name']))
else:
    print("provided key does not exist")
```

```
Name exist in dictionary myDict and its value is:Muhammad Nasir
```

Remove Items from Dictionary

In [105]:
```
#pop() method is used to remove the item from the dictionary, example,
myDict = {
    "name" : "Mohammad Nasir",
    "roll" : "SP20M2BB030",
    "class":"BSCS 6th E"
}
#lets remove "class" from above dictionary
myDict.pop("class") #class key will be removed from dict alon with its value
print(myDict)
```

```
{'name': 'Mohammad Nasir', 'roll': 'SP20M2BB030'}
```

In [106]:
```
#popitem() method will remove last inserted element from dictionary
myDict = {
    "name" : "Mohammad Nasir",
    "roll" : "SP20M2BB030",
    "class":"BSCS 6th E",
    "cgpa" : "3.5"
}
myDict.popitem()
print(myDict)
```

```
{'name': 'Mohammad Nasir', 'roll': 'SP20M2BB030', 'class': 'BSCS 6th E'}
```

In [107]:
```
#The del keyword removes the item with the specified key name:
del myDict['name'] #it will remove name key from dictionary
print(myDict)
```

```
{'roll': 'SP20M2BB030', 'class': 'BSCS 6th E'}
```

In [108]:
```
#we can also delete entire dictionary object using del command as,
myDict = {
    "name" : "Mohammad Nasir",
    "roll" : "SP20M2BB030",
    "class":"BSCS 6th E",
    "cgpa" : "3.5"
}
del myDict
print(myDict) #will generate error because myDict is deleted and does not exist
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[108], line 9
      2 myDict = {
      3     "name" : "Mohammad Nasir",
      4     "roll" : "SP20M2BB030",
      5     "class":"BSCS 6th E",
      6     "cgpa" : "3.5"
      7 }
      8 del myDict
----> 9 print(myDict)

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

In [109]:
```
#The clear() method empties the dictionary. it will remove all elements of dictionary but not delete the
#dictionary object as the del do,
#we can also delete entire dictionary object using del command as,
myDict = {
    "name" : "Mohammad Nasir",
    "roll" : "SP20M2BB030",
    "class":"BSCS 6th E",
    "cgpa" : "3.5"
}
myDict.clear()
print(myDict) #will generate error because myDict is deleted and does not exist
```

```
{}
```

Loop Through Dictionaries

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 [110]:
```
myDict = {
    "name" : "Mohammad Nasir",
    "roll" : "SP20M2BB030",
    "class":"BSCS 6th E",
    "cgpa" : "3.5"
}
for key in myDict:
    print(key)
```

```
name
roll
class
cgpa
```

In [111]:
```
for value in myDict.values():
    print(value)
```

```
Mohammad Nasir
SP20M2BB030
BSCS 6th E
3.5
```

In [112]:
```
for key in myDict:
    print(key,myDict[key])
```

```
name Mohammad Nasir
roll SP20M2BB030
class BSCS 6th E
cgpa 3.5
```

In [114]:
```
for key,value in myDict.items():
    print(key,value)
```

```
name Mohammad Nasir
```

```
roll SP20M2BB030
class BSCS 6th E
cgpa 3.5
```

```
A Dictionary can contain nested dictionaries, lists, tuple and many other data typesk it support almmost every dat
type + user custom data types
```

In [118]:
```
myCompleteDicy = {
    "info" : { #a dictionary in dictionary
        "name" : "Mohammad Nasir",
        "age" :23,
        "mobile" : ["+922012518700","+923183518093"] ,#a list in dictionary
        "gender" : "male",
    },

}
print(myCompleteDicy)
```

```
{'info': {'name': 'Mohammad Nasir', 'age': 23, 'mobile': ['+922012518700', '+923183518093'], 'gender': 'male'}}
```

In [ ]:
```

```