DICTIONARY

Ordered from Py3.7 onwards, Changeable, No Duplicate Items (Duplicate values will overwrite existing values), Any Data Types

In [1]:

```
car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.values()    # The view object contains the values of the dictionary, as a list. The view object will reflect
print(x)
for i in x:
    print(i)
y = list(x)
print(type(y))
print(y)
print(car)
car["year"] = 2018  # if the key is not present then it will create new key and assign the value to it
print(x)
print(car)
```

```
dict_values(['Ford', 'Mustang', 1964])
Ford
Mustang
1964
<class 'list'>
['Ford', 'Mustang', 1964]
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
dict_values(['Ford', 'Mustang', 2018])
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

In [2]:

```
thisdict =      {"brand": "Ford", "model": "Mustang", "year": 1964}
x = thisdict["model"]   # if the key is not present then, it will throw an error. Use get funtion to handle error
print(x)
```

```
Mustang
```

In [3]:

```
car = {"brand": "Ford","model": "Mustang","year": 1964,"owner": None}
x = car.get("model")    # dictionary.get(keyname, [value]) . A value to return if the specified key does not exist.D
print(x)

x = car.get("price")
print(x)

x = car.get("price", 15000)
print(x)

x = car.get("owner", "Sharan")
print(x)
```

```
Mustang
None
15000
None
```

In [4]:

```
car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.keys()  # The view object will reflect any changes done to the dictionary
print(x)
car["color"] = "white"
print(x)
```

```
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])
```

In [5]:

```
car = {"brand": "Ford", "model": "Mustang",  "year": 1964}
x = car.copy()
print(x)
print(car)

y = dict(car)
print(y)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

In [6]:

```
x = ('key1', 'key2', 'key3')    # could be any iterable
y = 0   # this value of y can be any object, and its value is for all the keys as their values
thisdict = dict.fromkeys(x, y)  # dict.fromkeys(keys, [value]) . returns a dictionary with the specified keys and th
print(thisdict)
thisdict = dict.fromkeys(x)
print(thisdict)
print(x)
```

```
{'key1': 0, 'key2': 0, 'key3': 0}
{'key1': None, 'key2': None, 'key3': None}
('key1', 'key2', 'key3')
```

In [7]:

```
car = {"brand": "Ford", "model": "Mustang",  "year": 1964}
car.update({"color": "White"})  # updates or inserts if not exists, the specified items to the dictionary. The speci
print(car)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'}
```

In [8]:

```
car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.setdefault("model", "Bronco")   # dictionary.setdefault(keyname, [value]) . for value, If the key exist, thi
print(x)
print(car)
x = car.setdefault("color", "white")
print(x)
print(car)
```

```
Mustang
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
white
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'white'}
```

In [9]:

```
 car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.items() # The items() method returns a view object. The view object contains the key-value pairs of the dict
print(x)
print(type(x))  # view object
car["year"] = 2018
print(x)
```

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
<class 'dict_items'>
```

```
dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 2018)])
```

In [10]:

```
car = {"brand": "Ford", "model": "Mustang",  "year": 1964}
car.clear() # removes all the elements from dictionary
print(car)
```

```
{}
```

In [11]:

```
car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.pop("model")
print(car)
print(x)    # value of removed item key.

# dictionary.pop(keyname, [defaultvalue]) . defaultvalue is a value to return if the specified key do not exist.If t
x = car.pop("model", "not exists")  # error line
print(x)
```

```
{'brand': 'Ford', 'year': 1964}
Mustang
not exists
```

In [12]:

```
car = {"brand": "Ford","model": "Mustang","year": 1964}
x = car.popitem()   # removes the item that was last inserted into the dictionary. In versions before 3.7, the popit
print(x)
print(car)
```

```
('year', 1964)
{'brand': 'Ford', 'model': 'Mustang'}
```

In [13]:

```
thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}
del thisdict["model"]
print(thisdict)

del thisdict
try:
    print(thisdict)
except:
    print("this will cause an error because \"thisdict\" no longer exists. ")
```

```
{'brand': 'Ford', 'year': 1964}
```

```
for x, y in thisdict.items():

for x in thisdict.keys():

for x in thisdict.values():

for x in thisdict:
```

In [14]:

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
print(myfamily)

child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
print(myfamily)
```

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', '
year': 2011}}
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007}, 'child3': {'name': 'Linus', '
year': 2011}}
```