WORKING WITH PYTHON DICITIONARY..!!

Array
It is the collection of elements of a single data type

Example:
PyArray=[1,2,3,4,5,6]
PyArray=["Hello","Hei","PYTHON"]

Linked List :
A linked list is a linear collection of data elements.
32 ==> 15 ==> 36 ==> 56 ==> 24

Advantages over arrays
1) Dynamic size
2) Ease of insertion/deletion

Limitations:
1 Access Very Slow
2 No Index to access
3 We sholud travel sequentially

To avoid the above limitations, Hash table introduced...!!

What is Hash Table?
A hash table (hash map) is a data structure which implements an associative array abstract data type.

Ele % Size=Location (Mod means remainder)
32 % 5=2
15 % 5=0
36 % 5=1
56 % 5=1
24 % 5=4

PYTHON HASHING (HASH TABLES AND HASHLIB)
What is Hash Table?
Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format.
OR
1 Hash Tables or Hashmap
2 It is a set of key-value Pairs
3 No Duplicate keys
4 Also Called as dictionary, map, hash table, associative array
5. In PYTHON use dict keyword or dict() Method.

Components of Hashmap:
Array ==> Data structure used to store the data
Hash function ==> function to convert key into an array index
Collision Handling==> Multiple key value pairs Map the same cell of an Array.

Basic Operations:
I. Search II. Insert III. delete

Hash Function:
index=len(key)-1

Key-Value
Beans-1.85
Corn-2.38
Rice-1.92

A better Hash function in PYTHON
index=sum(ASCII value for each letter in key)%Size

Python Dictionaries
A dictionary is a sequence of items. Each item is a pair made of a key and a value. Dictionaries are not sorted.

Why Dictionary?
1 Dictionaries are Python's most powerful data collection
2 Dictionaries allow us to do fast database-like operations in Python
3 Associative Arrays - PERL / PHP, HashMap - Java, Property Bag - C# / .Net

How to create a dictionary?
Creating a dictionary is as simple as placing items inside curly braces {} separated by comma. An item has a key and the corresponding value expressed as a pair, key: value.

Create a new Dictionary in Python:

```
#Empty dictionary
PyDict = {}
print(type(PyDict))#<class 'dict'>
print(PyDict)#{}

#Dictionary with integer keys
PyDict = {1: "Bigdata", 2: "Hadoop"}
print(type(PyDict))#<class 'dict'>
print(PyDict)#{1:'Bigdata',2:'Hadoop'}

#Dictionary with mixed keys
PyDict={'name': 'NareshIT', 1: [2, 4, 3]}
print(type(PyDict))#<class 'dict'>
print(PyDict)#{'name':'NareshIT',1:[2,4,3]}

Using dict Keyword:
PyDict=dict({1:'Bigdata', 2:'Hadoop'})
print(type(PyDict))#<class 'dict'>

# from sequence having each item as a pair
PyDict=dict([(1,'Bigdata'), (2,'Hadoop')])
print(type(PyDict))#<class 'dict'>

Add key/value to a dictionary in Python
PyDict={1:"Hadoop"}
print(PyDict)#{1: 'Hadoop'}
PyDict[2]="Spark"
print(PyDict)#{1: 'Hadoop', 2: 'Spark'}
PyDict[1]="H2O"
print(PyDict)#{1: 'H2O', 2: 'Spark'}
```

Loop Through a Dictionary
Looping through a dictionary, the return value are the keys of the dictionary.

Example:

```
PyCourses = {1: "PYTHON",2: "ML", 3: "DS" }
print(PyCourses)
for item in PyCourses:
    print(item)
```

Examples:

```
PyCourses = {1: "PYTHON",2: "ML", 3: "DS" }
print(PyCourses)
for item in PyCourses.values():
    print(item)
```

Example:

```
PyCourses = {1: "PYTHON",2: "ML", 3: "DS" }
print(PyCourses)
for item in PyCourses.keys():
    print(item)
```

Example:

```
PyCourses = {1: "PYTHON",2: "ML", 3: "DS" }
print(PyCourses)
for item in PyCourses.items():
    print(item)
```

Python Dictionary Methods

1 clear()	2 copy()
3 fromkeys(seq[, v])	4 get(key[,d])
5 items()	6 keys()
7 pop(key[,d])	8 popitem()
9 update([other])	10 values()

clear()
It is used to clear all keys and values from a dictionary.

Syntax:

```
dict.clear()
```

Example:

```
PyDict={1: 'PYDOOP', 2: 'Spark', 3: 'PYTHON'}
print(PyDict)#{1: 'PYDOOP', 2: 'Spark', 3: 'PYTHON'}
PyDict.clear()
print(PyDict)#{}#empty dictionary
```

copy()
It is used to create duplicate dictionary or shallow dictionary

Syntax:

```
dict.copy()
```

Example:

```
PyDict={1: 'PYDOOP', 2: 'Spark', 3: 'PYTHON'}
print(PyDict)#{1: 'PYDOOP', 2: 'Spark', 3: 'PYTHON'}
PyDict1=PyDict.copy()
print(PyDict1)#{1: 'PYDOOP', 2: 'Spark', 3: 'PYTHON'}
```

The fromkeys() method
It creates a new dictionary from the given sequence of elements with a value provided by the user.

Syntax:

```
dictionary.fromkeys(sequence[, value])
```

fromkeys() Parameters
sequence - sequence of elements which is to be used as keys for the new dictionary
value (Optional) - value which is set to each element of the dictionary-Default-None

Example:
```
keys = {'a', 'e', 'i', 'o', 'u' }
vowels = dict.fromkeys(keys)
print(vowels)
```

Example:
```
keys = {'a', 'e', 'i', 'o', 'u' }
value = 'vowel'
vowels = dict.fromkeys(keys, value)
print(vowels)
```

Example:
```
# this method returns a dictionary with the specified keys and values
# if the value is not specified for the keys, then the default value is
"NONE"
X = ('key1', 'key2', 'key3', 'key4', 'key5')
Y = 1
This_Dict = dict.fromkeys(X, Y)
print(This_Dict)
```

Dictionary get()
It returns the value for the specified key if key is in dictionary.

Syntax:
```
dict.get(key[, value])
```

get() Parameters
key - key to be searched in the dictionary
value (optional) - Value to be returned if the key is not found. The default value is None.

Example:
```
person = {'name': 'NareshIT', 'Exper': 15}
print('Name: ', person.get('name'))
print('Exper: ', person.get('Exper'))
print('Salary: ', person.get('salary'))
print('Salary: ', person.get('salary', '$100000'))
```

Example:
```
PyDict={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
print(PyDict)#{'a': 'BigData', 'b': 'Data Science', 'c': 'PTYHON'}
print(PyDict.get('a'))#BigData
print(PyDict.get('d'))#None
PyDict['d']="Machine Learning"
print(PyDict.get('d'))#Machine Learning
```

Dictionary items()
It returns a view object that displays a list of dictionary's (key, value) tuple pairs.

Syntax:
```
dictionary.items()
```

Example:
```
PyDict={ 'Apple': 5, 'Orange': 4, 'Grapes': 2 }
print(PyDict.items())
```

Dictionary update()
It updates the dictionary with the elements from the another dictionary object or from an iterable of key/value pairs.

Syntax:
dict.update([other])

Example:
```
d = {1: "one", 2: "three"}
d1 = {2: "two"}
d.update(d1)
print(d)
```

Example:
```
# Adding one element / multiple elements to a set
Super_Heroes = {"Superman", "Batman", "Spiderman"}
Super_Heroes.add("Aquaman")   # adding one element to the set
print(Super_Heroes)
Super_Heroes.update(["Wolverine", "Green Arrow", "Flash", "Ironman"])
# adding multiple elements to the set
print(Super_Heroes)
```

Example:
```
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'d':"Machine Learning",'e':"Deep Learning",'f':"Artificial
Intelligence"}
PyDict1.update(PyDict2)
print(PyDict1)# {'a': 'BigData', 'b': 'Data Science', 'c': 'PTYHON', 'd':
'Machine Learning', 'e': 'Deep Learning', 'f': 'Artificial Intelligence'}
```

Dictionary keys()
It returns a view object that displays a list of all the keys in the dictionary

Syntax:
dict.keys()

Example:
```
person = {'name': 'NareshIT', 'age': 15, 'salary': '$35000.0'}
print(person.keys())
empty_dict = {}
print(empty_dict.keys())
```

Example:
```
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'d':"Machine Learning",'e':"Deep Learning",'f':"Artificial
Intelligence"}
print(PyDict1.keys())#dict_keys(['a', 'b', 'c'])
print(PyDict2.keys())#dict_keys(['d', 'e', 'f'])
```

Example:
```
person = {'name': 'NareshIT', 'age': 15, }
print('Before dictionary is updated')
keys = person.keys()
print(keys)
person.update({'salary': '$350000'})
print('\nAfter dictionary is updated')
print(keys)
```

Dictionary values()
The values() method returns a view object that displays a list of all the values in the dictionary.

Syntax:
dictionary.values()

Example:
```
books = { 'BigData': 5, 'Hadoop': 6, 'Data Science': 4 }
print(books.values())
```

Example:
```
books = { 'BigData': 5, 'Hadoop': 6, 'Data Science': 4 }
print(books.values())
print('Original items:', books)
del[books['BigData']]
print('Updated items:', books)
print(books.values())
```

Example:
```
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'d':"Machine Learning",'e':"Deep Learning",'f':"Artificial
Intelligence"}
print(PyDict1.values())#dict_values(['BigData', 'Data Science', 'PTYHON'])
print(PyDict2.values())#dict_values(['Machine Learning', 'Deep Learning',
'Artificial Intelligence'])
```

Dictionary pop()
It removes and returns an element from a dictionary having the given key.

Syntax
```
dictionary.pop(key)
```

pop() Parameters
key - key which is to be searched for removal

Example:
```
books = { 'BigData': 2, 'Hadoop': 3, 'Data Science': 4 }
element = books.pop('BigData')
print('The popped element is:', element)
print('The dictionary is:', books)
```

Example:
```
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'d':"Machine Learning",'e':"Deep Learning",'f':"Artificial
Intelligence"}
print(PyDict1.pop('a'))#BigData
print(PyDict2.pop('f'))#Artificial Intelligence
```

Example:
```
books = { 'BigData': 2, 'Hadoop': 3, 'Data Science': 4 }
element = books.pop('Spark')
#Key Error
```

Dictionary popitem()
It returns and removes an arbitrary element (key, value) pair from the dictionary.

Syntax:
```
dict.popitem()
```

Example:
```
person = {'name': 'NareshIT', 'age': 15, 'salary': '$35000.0'}
result = person.popitem()
print('person = ',person)
print('Return Value = ',result)
```

Example:

```
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'d':"Machine Learning",'e':"Deep Learning",'f':"Artificial
Intelligence"}
print(PyDict1.popitem())#('c', 'PTYHON')
print(PyDict2.popitem())#('f','Artificial Intelligence')
```

Other Dictionary Operations (Dictionary Membership Test)
We can test if a key is in a dictionary or not using the keyword in. Notice that membership test is for keys only, not for values.

Example:

```
squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
print(1 in squares)
print(2 not in squares)
print(49 in squares)
```

Example:

```
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'d':"Machine Learning",'e':"Deep Learning",'f':"Artificial
Intelligence"}
print('a' in PyDict1)#True
print('A' not in PyDict1)#True
print('F' in PyDict2)#False
```

Built-in Functions with Dictionary

Function	Description
all()	Return True if all keys of the dictionary are true (or if the dictionary is empty).
any()	Return True if any key of the dictionary is true. If the dictionary is empty, return False.
len()	Return the length (the number of items) in the dictionary.
sorted()	Return a new sorted list of keys in the dictionary.

len()
It returns the number of items (length) of an object.

Syntax:

```
len(s)
```

Example:

```
fruits = {"mango": 2, "orange": 6}
print("Length:", len(fruits))
```

Python sorted()
The sorted() method sorts the elements of a given iterable in a specific order - Ascending or Descending.

Syntax

```
sorted(iterable[key, reverse])
```

Example:

```
pyDict = {'e': 1, 'a': 2, 'u': 3, 'o': 4, 'i': 5}
print(sorted(pyDict))
print(sorted(pyDict,reverse=True))
```

Example:

```
a = {"bb":"ggg", "aaaa":"d", "ff":"c"}
x = sorted(a,key=len)
print(x)
print(sorted(a.values()))
```

```
print(sorted(a.keys()))
print(sorted(a.items()))
PyDict={3:'e',2:'a',1:'c',7:'b',5:'d'}
PyResult=sorted(PyDict.items(),key=lambda x:x[1])
print(PyResult)
```

setdefault() Method:
It returns the value of the item with the specified key.

Syntax:
dictionary.setdefault(keyname, value)

Parameters:
keyname Required. The keyname of the item you want to return the value from
value   Optional. If the key exist, this parameter has no effect.If the key does not exist, this value becomes the key's value Default value None.

Example:
```
PyCourses = {1: "PYTHON",2: "ML", 3: "DS" }
print(PyCourses)
PyResult= PyCourses .setdefault(1)
print(PyResult)
PyResult= PyCourses .setdefault(4)
print(PyResult)
print(PyCourses)
PyCourses[4]="DL"
print(PyCourses)
```

Example:
```
Cars = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = Cars.setdefault("color", "White")
print(x)
print(Cars)
```

Nested Dictionaries
A dictionary can contain dictionaries, this is called nested dictionaries.

Example:
```
PyDict={'Courses1':
    {"PY":"PYTHON","ML":"MachineLearning"},
    'Courses2':
    {"CY":"CYTHON","DL":"DeepLearning"}
    }
print(PyDict)
```

Example:
```
PyDict={'Courses1':
    {"PY":"PYTHON","ML":"MachineLearning"},
    'Courses2':
    {"CY":"CYTHON","DL":"DeepLearning"}
    }
print(PyDict)
print(PyDict['Courses1'])
print(PyDict['Courses1']['PY'])
```

Example:

```
x={1:"PY"}
y={2:"ML"}
z={3:"DS"}
MyCourses={'First':x,'Second':y,'Third':z}
print(MyCourses)
print(MyCourses['First'])

# How to merge two dictionaries
x = {'a': 1, 'b': 2}
y = {'b': 3, 'c': 4}
z = {**x, **y}
print(z)

Example:
PyDict1={'a':"BigData",'b':"Data Science",'c':"PTYHON"}
PyDict2={'a':"Machine Learning",'b':"Deep Learning",'c':"Artificial
Intelligence"}
PyDict3={**PyDict1,**PyDict2}
print(PyDict3)#{'a': 'Machine Learning', 'b': 'Deep Learning', 'c':
'Artificial Intelligence'}

Dictionary Comprehension:
Comprehension concept applicable for dictionaries also.

Example:
squares={x:x*x for x in range(1,6)}
print(squares)#{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
doubles={x:2*x for x in range(1,6)}
print(doubles)#{1: 2, 2: 4, 3: 6, 4: 8, 5: 10}

Accessing elements in a dictionary
This_Dict = {
    "Brand" : "Ford",
    "Model" : "Mustang",
    "Year" : 1964
}
print(This_Dict["Year"])

# Looping through a dictionary
# while looping through a dictionary, the return value are the keys of the
dictionary, but there are methods to return the values as well
This_Dict = {
    "Brand" : "Ford",
    "Model" : "Mustang",
    "Year" : 1964
}

# Loop for printing keys
for x in This_Dict:
    print(x)

# Loop for printing values
for x in This_Dict:
    print(This_Dict[x])
for x in This_Dict.values():
    print(x)

# Loop for printing for keys and values
for x, y in This_Dict.items():
    print(x, y)
print(This_Dict.get("Brand"))
```

```
# Removing items from a dictionary
This_Dict = {
    "Brand" : "Ford",
    "Model" : "Mustang",
    "Year" : 1964,
    "Color" : "Red",
    "Tyre" : "Tubeless"
}
This_Dict.pop("Model")
print(This_Dict)
This_Dict.popitem()
print(This_Dict)
del This_Dict["Brand"]
print(This_Dict)
```