COMPUTER SCIENCE

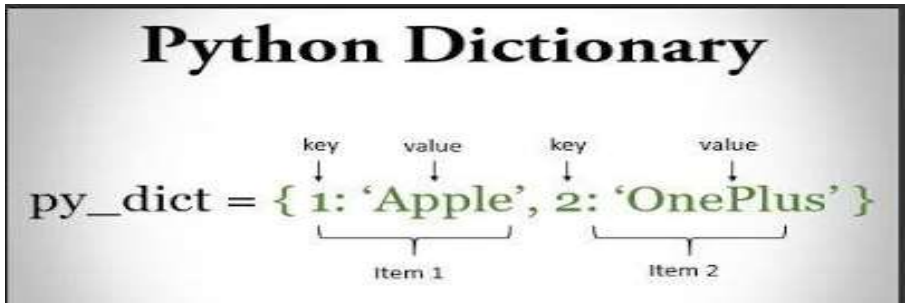
CLASS 11 – STUDY MATERIAL

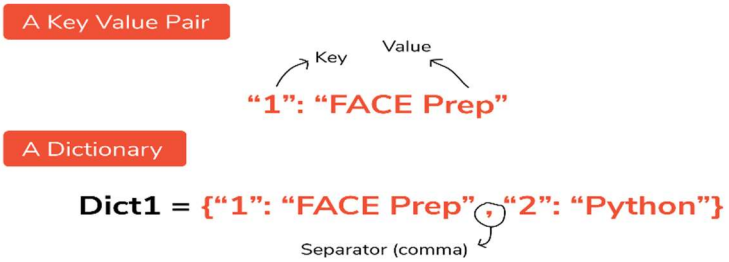
DICTIONARIES

INTRODUCTION TO DICTIONARIES

It is another collection in Python but with different in way of storing and accessing. Other collection like list, tuple, strings are having an index associated with every element but Python Dictionary have a "**key"** associated with **every element.** That's why python dictionaries are known as **KEY:VALUE** pairs.
Like with English dictionary we search any word for meaning associated with it, similarly in Python we search for **"key"** to get its associated value rather than searching for an index.

The data type dictionary falls under **mapping**.
It is a mapping between a set of keys and a set of values.
The key-value pair is called an **item**.
A key is separated from its value by a colon (:) and consecutive items are separated by commas.
Items in dictionaries are unordered, so we may not get back the data in the same order in which we had entered the data initially in the dictionary.





CREATING AND ACCESSING DICTIONARY

Creating a Dictionary

Syntax to create dictionary:

dictionary_name = {key1: value, key2: value}

Example

```
>>> emp = {"empno":1,"name":"Shahrukh","fee":1500000}
```

Here

Keys are: “empno”, “name” and “fee”

Values are: 1, “Shahrukh”, 1500000

Note:

1) Dictionary elements must be between curly brackets
2) Each value must be paired with key element
3) Each key-value pair must be separated by comma(,)

To create a dictionary, the items entered are separated by commas and enclosed in curly braces. Each item is a key value pair, separated through colon (:). The keys in the dictionary must be unique and should be of any immutable data type i.e. number, string or tuple. The values can be repeated and can be of any data type.

```
#dict1 is an empty dictionary
>>> dict1 = {}
>>> dict1
{}
#dict3 is the dictionary that maps names of #the students to marks in percentage
>>> dict3 = {'Mohan':95,'Ram':89,'Suhel':92, 'Sangeeta':85}
>>> dict3
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}
```

It is enclosed in curly braces {} and each item is separated from other item by a comma(,). Within each item, key and value are separated by a colon (:).Passing value in dictionary at declaration is dictionary initialization.

e.g.

dict = {‘Subject': ‘Informatic Practices', 'Class': ‘11'}

Creating A Dictionary

Dict1 = {} # empty dictionary

DaysInMonth = {"Jan":31,"Feb":28,"Mar":31,"Apr":31

"May":31,"Jun":30,"Jul":31,"Aug":31

"Sep":30,"Oct":31,"Nov":30,"Dec":31}

Note: Keys of dictionary must of immutable type such as:

- A python string
- A number
- A tuple(containing only immutable entries)
- If we try to give mutable type as key, python will give an error
- >>>dict2 = {[2, 3]:”abc”} #Error

Accessing Items in a Dictionary

Items of a sequence (string, list and tuple) are accessed using a technique called indexing. The items of a dictionary are accessed via the keys rather than via their relative positions or indices. Each key serves as the index and maps to a value.

>>> dict3 = {'Mohan':95,'Ram':89,'Suhel':92, 'Sangeeta':85}

>>> dict3 ['Ram']

89

>>> dict3 ['Sangeeta']

85

#using unspecified key

>>> dict3 ['Shyam']

KeyError: 'Shyam'

dict = {'Subject': 'Computer', 'Class': 11}

print (dict)

print ("Subject: ", dict['Subject'])

print ("Class: ", dict.get('Class'))

OUTPUT

{'Class': '11', 'Subject': Computer'}

('Subject : ', 'Computer')

(‘Class' :11)

Iterating / Traversing through A Dictionary

Python allows to apply “for” loop to traverse every element of dictionary based on their “key”. For loop will get every key of dictionary and we can access every element based on their key.

mydict={'empno':1,'name':'Shivam','dept':'sales','salary':25000}

for key in mydict:

print(key,'=',mydict[key])

Following example will show how dictionary items can be accessed through loop.

e.g.

```
dict = {'Subject': 'Computer Science', 'Class': 11}
for i in dict:
        print(dict[i])
```

OUTPUT

11

Computer Science

We can access each item of the dictionary or traverse a dictionary using for loop.

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92,'Sangeeta':85}
```

Method 1:

```
>>> for key in dict1:
                print(key,':',dict1[key])
```

Mohan: 95

Ram: 89

Suhel: 92

Sangeeta: 85

Method 2:

```
>>> for key, value in dict1.items():
        print(key,':',value)
```

Mohan: 95

Ram: 89

Suhel: 92

Sangeeta: 85

Accessing keys and values simultaneously

```
>>> mydict={'empno':1,'name':'Shivam','dept':'sales','salary':25000}
>>>mydict.keys()
dict_keys(['empno', 'name', 'dept', 'salary'])
>>>mydict.values()
dict_values([1, 'Shivam', 'sales', 25000])
```

We can convert the sequence returned by keys() and values() by using list() as shown below:

```
>>> list(mydict.keys())
['empno', 'name', 'dept', 'salary']
>>> list(mydict.values())  [1, 'Shivam', 'sales', 25000]
```

Updating/Manipulating Dictionary Elements –MUTABILITY OF A DICTIONARY

We can change/modify the individual element of dictionary and add/insert new elements.

e.g.

```
dict = {'Subject': 'Computer Science', 'Class': 11}
```

```
dict['Subject']='computer science'
print(dict)
OUTPUT
{'Class': 11, 'Subject': 'computer science'}

>>> d = {'Subject': 'Computer Science', 'Class': 11}
>>> d['marks']=90
>>> d
{'Subject': 'Computer Science', 'Class': 11, 'marks': 90}
```

Characteristics of a Dictionary

Unordered set

- A dictionary is a unordered set of key:value pair

Not a sequence

- Unlike a string, tuple, and list, a dictionary is not a sequence because it is unordered set of elements. The sequences are indexed by a range of ordinal numbers. Hence they are ordered but a dictionary is an unordered collection

Indexed by Keys, Not Numbers

- Dictionaries are indexed by keys. Keys are immutable type

Keys must be unique

- Each key within dictionary must be unique. However two unique keys can have same values.

  >>> data={1:100, 2:200,3:300,4:200}

Mutable

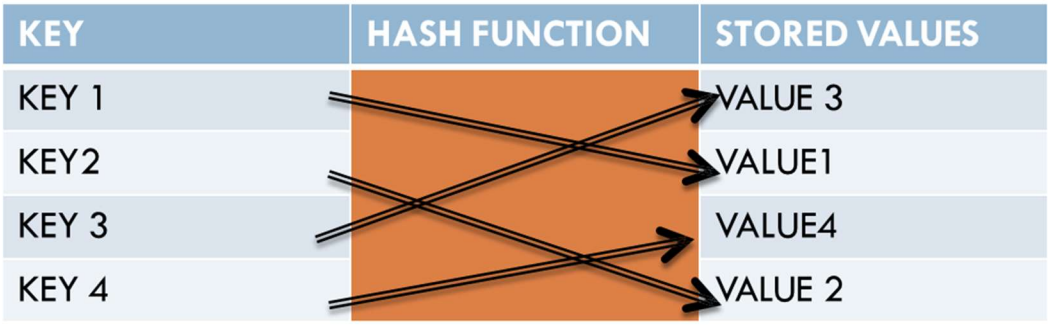
- Like lists, dictionary are also mutable. We can change the value of a certain “key” in place

  Data[3]=400
- >>>Data
- So, to change value of dictionary the format is :

- DictionaryName[“key” / key]=new_value
- You can not only change but you can add new key:value pair :

Internally stored as Mappings

Internally, the key:value pair are associated with one another with some internal function(called hash function). This way of linking is called mapping



Deleting Dictionary Elements

del, pop() and clear() statement are used to remove elements from the dictionary.

del e.g.

```
dict = {'Subject': 'Computer Science', 'Class': 11}
print('before del', dict)
del dict['Class'] # delete single element
print('after item delete', dict)
del dict #delete whole dictionary
print('after dictionary delete', dict)
```

Output

('before del', {'Class': 11, 'Subject': 'Computer Science'}) ('after item delete', {'Subject': 'Computer Science'})

('after dictionary delete', <type 'dict'>)

Deleting elements from Dictionary

Del dictionaryName[“Key”]

```
>>> D1 = {1:10,2:20,3:30,4:40}
```

>>> del D1[2]

>>> D1 1:10,2:20,4:40

If you try to remove the item whose key does not exists, the python runtime error occurs.

del D1[5] #Error

del D1 #deleted the entire dictionary from memory

pop() elements from Dictionary

dictionaryName.Pop([“Key”])

>>> D1 = {1:10,2:20,3:30,4:40}

>>> D1.pop(2) 1:10,2:20,4:40

Note: if key passed to pop() doesn’t exists then python will raise an exception.

Pop() function allows us to customized the error message displayed by use of wrong key

>>> d1

{'a': 'apple', 'b': 'ball', 'c': 'caterpillar', 'd': 'dog'}

>>>d1.pop(“a”)

>>> d1.pop(“d”,”Not found”)

Not found

Checking the existence of key (membership operators)

We can check the existence of key in dictionary using “in” and “not in”.

>>>alpha={"a":"apple","b":"boy","c":"cat","d":"dog"}

>>> 'a' in alpha

True

>>>"e" in alpha False

>>>"e" not in alpha

True

If you pass “value” of dictionary to search using “in” it will return False

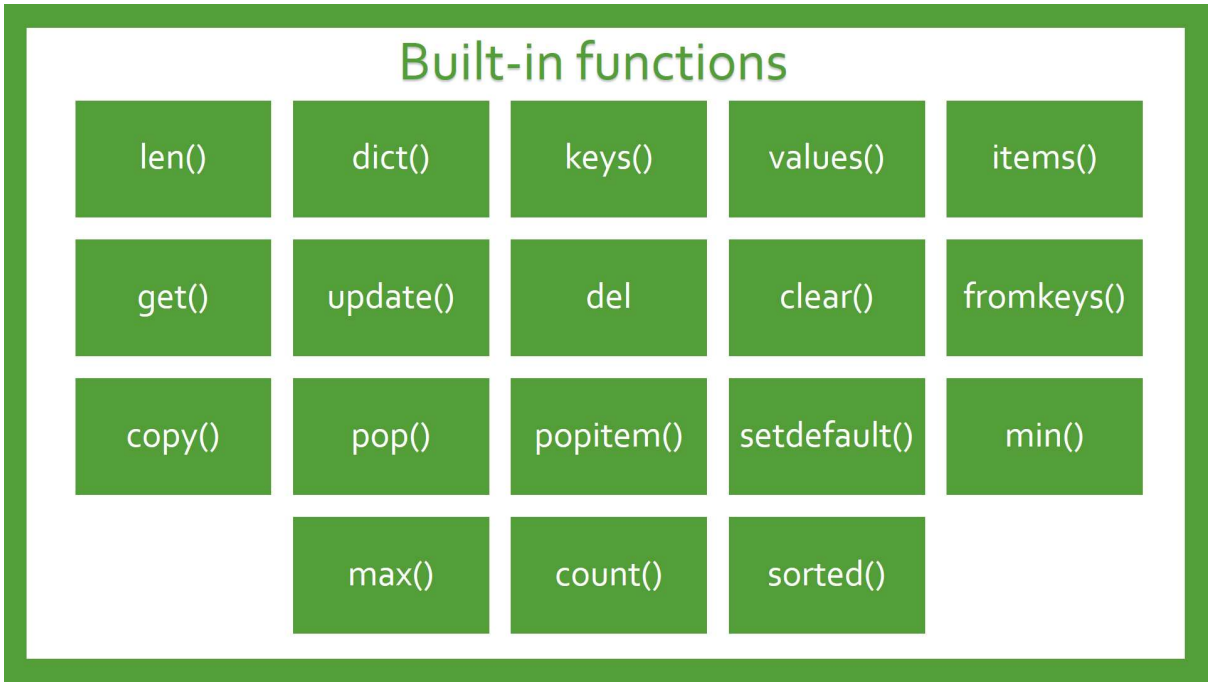
>>>"apple" in alpha

 False

To search for a value we have to search in dict.values()

>>>"apple" in alpha.values()

Built-in Dictionary Functions



Python provides many functions to work on dictionaries.

len()

Returns the length or number of key: value pairs of the dictionary passed as the argument

>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}

>>> len(dict1)

4

dict()

Creates a dictionary from a sequence of key-value pairs

```
pair1 = [('Mohan',95),('Ram',89), ('Suhel',92),('Sangeeta',85)]
>>> pair1

[('Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)]
>>> dict1 = dict(pair1)
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}
```

keys()

Returns a list of keys in the dictionary

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict1.keys()
dict_keys(['Mohan', 'Ram', 'Suhel', 'Sangeeta'])
```

values()

Returns a list of values in the dictionary

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict1.values()
dict_values([95, 89, 92, 85])
```

items()

Returns a list of tuples (key — value) pair

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict1.items()
dict_items([( 'Mohan', 95), ('Ram', 89), ('Suhel', 92), ('Sangeeta', 85)])
```

get()

Returns the value corresponding to the key passed as the argument

If the key is not present in the dictionary it will return None

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
```

```
>>> dict1.get('Sangeeta')
85
>>> dict1.get('Sohan')
```

update()

appends the key-value pair of the dictionary passed as the argument to the key-value pair of the given dictionary

```
>>> dict1 = {'Mohan':95, 'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict2 = {'Sohan':79,'Geeta':89}
>>> dict1.update(dict2)
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Sohan': 79, 'Geeta': 89}
>>> dict2
{'Sohan': 79, 'Geeta': 89}
```

clear()

Deletes or clear all the items of the dictionary

```
>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> dict1.clear()
>>> dict1
{ }
```

del

Deletes the item with the given key

To delete the dictionary from the memory we write:

del Dict_name

```
>>> dict1 = {'Mohan':95,'Ram':89, 'Suhel':92, 'Sangeeta':85}
>>> del dict1['Ram']
>>> dict1
{'Mohan':95,'Suhel':92, 'Sangeeta': 85}
>>> dict1
NameError: name 'dict1' is not defined
```

DICTIONARIES

min()

min(dictionary_name)

Returns the minimum of a **key** in a dictionary.

max()

Returns the maximum of a **key** in a dictionary.

max(dictionary_name)

```
>>> vowels
{'e': None, 'a': None, 'i': None, 'u': None, 'o': None}
>>> min(vowels)
'a'
>>> max(vowels)
'u'
```

Alphabets on the basis of ASCII values.

```
>>> d
{1: 34, 2: 55, 3: 78, 4: 12}
>>> min(d)
1
>>> max(d)
4
```

DICTIONARIES

sorted()

sorted(dictionary_name)

The sorted function returns a sorted sequence of the **keys** in a dictionary.

The sorting is in **ascending** order and does not modify the original dictionary.

Alphabets on the basis of ASCII values.

```
>>> d={1:34,2:55,3:78,4:12}
>>> sorted(d)
[1, 2, 3, 4]
>>> car={'B':'BMW','M':'merc','R':'rolls royce','A':'audi'}
>>> sorted(car)
['A', 'B', 'M', 'R']
```

DICTIONARIES

dictionary_name.copy()

copy()

The copy() method returns a copy of the specified dictionary.

d={1:34,2:55,3:78,4:12}

```
>>> newcopy=d.copy()
>>> newcopy
{1: 34, 2: 55, 3: 78, 4: 12}
```

DICTIONARIES

setdefault()

Returns the value of the item with the specified key. If the key does not exist: insert the key, with the specified value.

dictionary.setdefault(*keyname, value*)

Second parameter is optional.

Inserting the value if key does not exists.

```
>>> car = {"brand": "Ford","model": "Mustang","year": 1964}
>>> x = car.setdefault("color", "White")
>>> x
'White'
>>> car
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'}
```

Fetching the value if key exists.

```
>>> car
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'}
>>> x = car.setdefault("model", "Bronco")
>>> x
'Mustang'
```

DICTIONARIES

fromkeys()

fromkeys(keys,value)

It created a new dictionary from the given sequence of elements with a value provided by the user.

```
>>> k={'a','e','i','o','u'}
>>> vowels=dict.fromkeys(k,'vowel')
>>> vowels
{'e': 'vowel', 'a': 'vowel', 'i': 'vowel', 'u': 'vowel', 'o': 'vowel'}
```

Datatype set

When the value is not specified, None is assigned to each key.

```
>>> vowels=dict.fromkeys(k)
>>> vowels
{'e': None, 'a': None, 'i': None, 'u': None, 'o': None}
```

DICTIONARIES

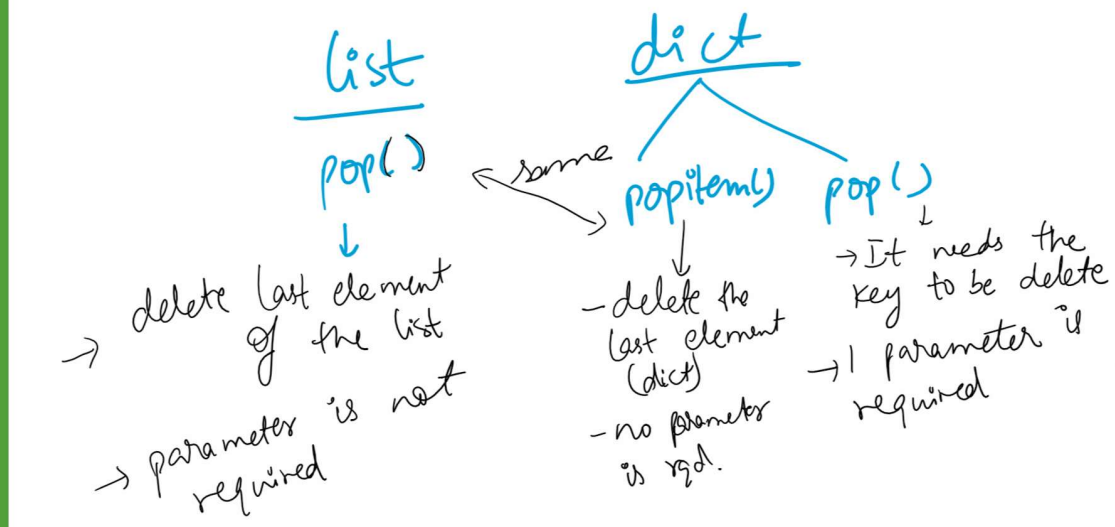
popitem() **Dict.popitem()**

It removes the last inserted item in the dictionary.
It returns the key-value pair that has been removed from the dictionary.

```
>>> car.popitem()
('A', 'audi')
>>> car
{'B': 'BMW', 'M': 'merc', 'R': 'rolls royce'}
```

Pop() requires the key that needs to be deleted and returns the value corresponding to the key that has been deleted.

```
>>> car.pop('M')
'merc'
>>> car
{'B': 'BMW', 'R': 'rolls royce'}
```



Syntax: list.pop() diction.popitem() d.pop(key)

Manipulating Dictionaries

The following examples show the application of those manipulation methods on dictionaries.

(a) Create a dictionary 'ODD' of odd numbers between 1 and 10, where the key is the decimal number and the value is the corresponding number in words.

```
>>> ODD = {1:'One',3:'Three',5:'Five',7:'Seven',9:'Nine'}
>>> ODD
```

{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven', 9: 'Nine’}

(b) Display the keys in dictionary ‘ODD’.

>>> ODD.keys()

dict_keys([1, 3, 5, 7, 9])

Check if 2 is present or not in dictionary ‘ODD’

>>> 2 in ODD

False

(c) Display the values in dictionary ‘ODD’.

>>>ODD. Values()

dict_values(['One', 'Three', 'Five', 'Seven', 'Nine’])

(d) Display the items from dictionary ‘ODD’

>>>ODD. Items()

dict_items([(1, 'One'), (3, 'Three'), (5, 'Five'), (7, 'Seven'), (9, 'Nine’)])

(e) Find the length of the dictionary ‘ODD’.

>>> len(ODD)

5

(f) Check if 7 is present or not in dictionary ‘ODD’

>>> 7 in ODD

True

(h) Retrieve the value corresponding to the key 9

>>> ODD.get(9)

'Nine'

(i) Delete the item from the dictionary, corresponding to the key 9. ‘ODD’

>>> del ODD[9]

>>> ODD

{1: 'One', 3: 'Three', 5: 'Five', 7: 'Seven'}

PROGRAMS

Write a program to enter names of employees and their salaries as input and store them in a dictionary. Here n is to input by the user.

```
#Program 4-4
#Program to create a dictionary which stores names of employees
#and their salary
num = int(input("Enter the number of employees whose data to be stored: "))
count = 1
employee = dict() #create an empty dictionary
for count in range (n):
        name = input("Enter the name of the Employee: ")
        salary = int(input("Enter the salary: "))
        employee[name] = salary
        print("\n\nEMPLOYEE_NAME\tSALARY")
for k in employee:
        print(k,'\t\t',employee[k])
```

Write a program to count the number of times a character appears in a givenstring

```
#Program 4-5
#Count the number of times a character appears in a given string
st = input("Enter a string: ")
dic = {} #creates an empty dictionary
for ch in st:
        if ch in dic: #if next character is already in dic
                dic[ch] += 1
        else:
                dic[ch] = 1 #if ch appears for the first time
for key in dic:
        print(key,':',dic[key])
```

Write a program to convert a number entered by the user into its corresponding number in words. for example if the input is 876 then the output should be ‘Eight Seven Six’.

```
num = input("Enter any number: ") #number is stored as string
#numberNames is a dictionary of digits and corresponding number
#names
numberNames = {0:'Zero',1:'One',2:'Two',3:'Three',4:'Four',\
5:'Five',6:'Six',7:'Seven',8:'Eight',9:'Nine'}
result = ''
for ch in num:
    key = int(ch) #converts character to integer
    value = numberNames[key]
    result = result + ' ' + value

print("The number is:",num)
print("The number Name is:”, result)
```

Suggested programs:

1. count the number of times a character appears in a given string using a dictionary
2. create a dictionary with names of employees, their salary and access them