01.Dictionaries

September 17, 2025

0.1 Python Dictionaries

Note Book Owner: Emdadul Hoque

0.1.1 Creating Dictionaries in four different ways

[1]:
```
# Way 01:
d1 = {"name": "Emdadul", "Age": 27}
```

[2]:
```
d1
```

[2]:
```
{'name': 'Emdadul', 'Age': 27}
```

[3]:
```
#way 2:

d2 = {}
d2["name"] = "Emdadul"
d2["age"] = 27
```

[4]:
```
d2
```

[4]:
```
{'name': 'Emdadul', 'age': 27}
```

[8]:
```
#Way 03:

d3 = dict(name= "Emdadul", age= 27)
```

[9]:
```
d3
```

[9]:
```
{'name': 'Emdadul', 'age': 27}
```

[10]:
```
#Way 04:
d4 = dict([('name', 'Emdadul'), ("age", 27)])
```

[63]:
```
d4
```

[63]:
```
{'name': 'Emdadul', 'age': 27}
```

0.2 Access value from a dictionary

[64]:
```
info = dict(name="tareque", age=27, hobby="Cricket")
```

[65]:
```
info
```

[65]:
```
{'name': 'tareque', 'age': 27, 'hobby': 'Cricket'}
```

[67]:
```
#access value using key
info["name"]
```

[67]:
```
'tareque'
```

[68]:
```
#access value using get() methods
info.get("name")
```

[68]:
```
'tareque'
```

0.3 Adding, Replacing, delete Values

[12]:
```
info = {"name": "Tareque", "age": 27}
```

[13]:
```
info["Hobby"] = "Cricket"
```

[14]:
```
info
```

[14]:
```
{'name': 'Tareque', 'age': 27, 'Hobby': 'Cricket'}
```

[15]:
```
#replace old value

info["name"] = "Emdadul"
```

[16]:
```
info
```

[16]:
```
{'name': 'Emdadul', 'age': 27, 'Hobby': 'Cricket'}
```

[19]:
```
#Retrieving Values

print(info["Hobby"])
```

```
Cricket
```

[25]:
```
#formating dictionaries

text = "My name is %(name)s. I am %(age)d years old"%info
#here d use for integer, s use for string and %info use for intigate dictonary
```

[23]:
```
text
```

[23]:
```
'My name is Emdadul. I am 27 years old'
```

[72]:
```python
#deleting items using del operator
info = {"name": "Tareque", "age": 27, "Hobby": "Cricket"}

del info["age"]
```

[73]:
```python
info
```

[73]:
```
{'name': 'Tareque', 'Hobby': 'Cricket'}
```

[74]:
```python
#deleting items using pop method
info.pop("name")
```

[74]:
```
'Tareque'
```

[75]:
```python
info
```

[75]:
```
{'Hobby': 'Cricket'}
```

[29]:
```python
#comparing dictionary
info = {"name": "Tareque", "age": 27}
info1 = {"name": "Tareque", "age": 27}
```

[30]:
```python
info == info1
```

[30]:
```
True
```

[31]:
```python
info != info1
```

[31]:
```
False
```

0.4 dictionary methods

[33]:
```python
#keys() --> Returns a sequemce of keys

info = {"name": "Tareque", "age": 27, "hobby": "Cricket"}
```

[34]:
```python
info.keys()
```

[34]:
```
dict_keys(['name', 'age', 'hobby'])
```

[35]:
```python
#values() --> return a sequence of value

info.values()
```

[35]:
```
dict_values(['Tareque', 27, 'Cricket'])
```

```
[40]: #items --> Returns a sequence of tuples
      info.items()
```

```
[40]: dict_items([('name', 'Tareque'), ('age', 27), ('hobby', 'Cricket')])
```

```
[42]: print(info.items())
```

```
dict_items([('name', 'Tareque'), ('age', 27), ('hobby', 'Cricket')])
```

```
[43]: item = info.items()
```

```
[52]: print(type(item))
      for i in item:
          print(i)
          print(type(i))
```

```
<class 'dict_items'>
('name', 'Tareque')
<class 'tuple'>
('age', 27)
<class 'tuple'>
('hobby', 'Cricket')
<class 'tuple'>
```

```
[54]: #clear()

      info.clear()
```

```
[55]: info
```

```
[55]: {}
```

```
[60]: #get(key) --> return a value for a key

      info = info = {"name": "Tareque", "age": 27, "hobby": "Cricket"}
      info.get("name")
```

```
[60]: 'Tareque'
```

```
[61]: #pop(key) --> Remove a key and returns the value if the key exists

      info.pop("name")
```

```
[61]: 'Tareque'
```

```
[62]: info
```

```
[62]: {'age': 27, 'hobby': 'Cricket'}
```

[48]:
```
#make a dictionary from tuple using zip method
tple = ("name", "age", "hobby")
tple1 = ("tareque", 27, "Cricket")

dic = dict(zip(tple, tple1))
```

[49]:
```
dic
```

[49]:
```
{'name': 'tareque', 'age': 27, 'hobby': 'Cricket'}
```

[50]:
```
lst = ["name", "age", "hobby"]
lst1 = ["Tareque", 27, "Cricket"]

dic = dict(zip(lst, lst1))
```

[51]:
```
dic
```

[51]:
```
{'name': 'Tareque', 'age': 27, 'hobby': 'Cricket'}
```

0.5 Traversing Dictionaries

[82]:
```
info = {"name": "Tareque", "age": 27, "hobby": "Cricket"}
```

[83]:
```
# Way 01
items = info.items()
for key, value in items:
#     key, value = item
    print(key,": ", value)
```

```
name :  Tareque
age :  27
hobby :  Cricket
```

[84]:
```
# Way 02

for key in info:
    print(key, ": ", info[key])
```

```
name :  Tareque
age :  27
hobby :  Cricket
```

0.6 Traversing nested dictionary

[89]:
```
players = {"virat Kohli": {"ODI": 7212, "Test": 3245},
           "Sachin": {"ODI": 18426, "Test": 15921}}
```

[91]:
```
for player_name, player_details in players.items():
    print("player: ", player_name)
    print(" Run Scored in ODI: ", player_details["ODI"])
    print(" Run Scored in Test:", player_details["Test"])
```

```
player:  virat Kohli
 Run Scored in ODI:  7212
 Run Scored in Test: 3245
player:  Sachin
 Run Scored in ODI:  18426
 Run Scored in Test: 15921
```

0.7 Simple programs on Dictionary

Program 01: Write a function histogram that takes string as parameter and generates a frequency of characters contained in it.

s = “AAPPLE”

the porgam should create a dictionary

D = {‘A’: 2, ‘E’: 1, ‘P’: 2, ‘L’: 1}

[97]:
```
def histogram(string):
    D = dict()

    for char in string:
        if char not in D:
            D[char] = 1
        else:
            D[char] = D[char]+1
    return D

s = "AAPPLE"
print(histogram(s))
```

```
{'A': 2, 'P': 2, 'L': 1, 'E': 1}
```

Program 02: Write a program to count the frequency of charcters using the get() method

[100]:
```
def histogram(string):
    D = dict()

    for char in string:
        if char not in D:
            D[char] = 1
        else:
            D[char] = D.get(char)+1
    return D

```

```
s = "AAPPLE"
print(histogram(s))
```

```
{'A': 2, 'P': 2, 'L': 1, 'E': 1}
```

Program 03: Write a program to print and store squares of numbers into a dictionary.

[101]:
```
def sq_of_numbers(n):
    d = dict()
    for i in range(1, n+1):
        if i not in d:
            d[i] = i * i
    return d
print("Square of Number: ")

z = sq_of_numbers(5)
print(z)
```

```
Square of Number:
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Program 04: Write a program to pass a list to a function. Calculate the total number of positive and negative numbers from the list and then display the count in terms of dictionary.

input: L = [1,-2, -3, 4] output: {“Neg”: 2, “Pos”:2}

[104]:
```
def pos_neg_number(lst):
    dic = dict()
    dic["Neg"] = 0
    dic["Pos"] = 0
    for i in lst:
        if i > 0:
            dic["Pos"] += 1
        else:
            dic["Neg"] += 1
    return dic

print(pos_neg_number([1,-2, -3, 4]))
```

```
{'Neg': 2, 'Pos': 2}
```

0.8 Build-in Dictionary Functions and Methods

[118]:
```
dic = {"name": "Emdadul", "Age": 27, "Hobby": "Cricket"}
```

[106]:
```
#len(dict)

len(dic)
```

[106]: 3

[107]:
```
str(dic)
```

[107]:
```
"{'name': 'Emdadul', 'Age': 27, 'Hobby': 'Cricket'}"
```

[110]:
```
# returns a shallow copy of the dictionary. The dictonary returned will not␣
↪have a duplicate copy of dic but will have the same refference
dic1 = dic.copy()
```

[111]:
```
dic1
```

[111]:
```
{'name': 'Emdadul', 'Age': 27, 'Hobby': 'Cricket'}
```

[116]:
```
subject = ["C++", "Python", "JavaScript", "C"]
marks = [10, 20, 30, 40, 50]
dic1.fromkeys(subject, 10)
```

[116]:
```
{'C++': 10, 'Python': 10, 'JavaScript': 10, 'C': 10}
```

[122]:
```
dic = {"name": "Emdadul", "Age": 27, "Hobby": "Cricket"}
dic1 = {"Passion": "Programming", "ocopassion": "Manager"}
```

[123]:
```
dic.update(dic1) # adds the key-value pairs of dict1 to the key-value pairs of␣
↪dic
```

[125]:
```
dic.values()
```

[125]:
```
dict_values(['Emdadul', 27, 'Cricket', 'Programming', 'Manager'])
```

0.9 Difference between a list and a dictionary

There are two main differences between a list and a dictonary

1. First, a list is a ordered set of items, but a dictionary is a data structure that is used for matching one item (key) and another (Value)
2. Second, in list we can use indexing to access a particular item. but, these indexes should be a number. In dictionaries, we can use any(immutable) of value as an index.
3. List are used to look up a value whereas a dictionary is used to take one value and look up another value.

The main advantage of a dictionary is that we don't need to search for value one by one in the entire set of values, we can find a value instrantly

0.10 When to use which Data Structure?

1. Use lists to store a collection of data that does not need random access
2. Use lists if the data has to be modified frequently

3. Use a set if you want to ensure that every element in the data structure must be unique.
4. Use tuples when you want that your data should not be altered.
5. Due to mutability difference, tuples are easier on memory and processor in comparison to lists. This means that you can easily achieve performance optimization by using tuples, wherever possible. Moreover, tuples are best used as heterogeneous collections while lists are best used as homogenoys collections (Where heterogeneous means that the items contained in a tuple may belong to defferent types or concepts)
6. Sets are used to store unordered values and do not have index. unlike tuples and lists, sets can have no duplicate data. However, like list and unlike tuples. we can use the add() function to add an element to set and the update() functions to edit the elements in the set.

```
[ ]:
```