Python Dictionary Manipulation

Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and each element is a combination of “key & key value”.

Note: Dictionaries are Mutable (Changeable)

Important Features of Dictionary

- Each key map to value. So, key and its associated value are known as “key-value pair”or dictionary item.
- Each key of dictionary is separated from its value by colon (:) mark.
- Each dictionary item / element is separated by comma.
- Entire elements of dictionary enclosed in curly braces {}.
- Keys are unique in dictionary while values are not unique.
- The values of dictionary can be of any type but keys must be of an immutable data type(Such as int, float, String, tuple).
- Dictionary is Mutable that mean new item can be add and existing item can be change/delete.

Creation of Dictionary

A Dictionary can be create by placing items (key-value pairs) inside curly braces { } and separated by comma.

Syntax:

Dictionary_name={ key1 : value-1, key2 : value2, key-n : value-n }

Example:

1. days={'S':'Sunday', 'M':'Monday', 'T':'Tuesday'}

Here keys are: 'S', 'M', 'T'

and values are: 'Sunday', 'Monday', 'Tuesday'

2. months={1:"January", 2:"February", 3:"March"}

Here keys are: 1, 2, 3

and values are: 'January', 'February', 'March'

3. temp={1:15, 2:16, 3:10}
4. sale={'Mon':1000, 'Tue':1500, 'Wed':2000}
5. student={'name':['Ajay', 'Vijay'], 'Age':[14,16]}
   Here keys are: 'name', 'Age'
   and values are: 'Ajay', 'Vijay', 14, 16

Creating an Empty Dictionary

1. An empty Dictionary can be created by assigning { } to a variable.

   Di={}

   Here Di is a Dictionary type Variable.

   Print(Di) Output: {}

2. An empty Dictionary can be created by dict() constructor also.

   Di=dict()

   Print(Di) Output: {}

3. Add element to an empty dictionary

   Di[1]=”January” #1 is key and “January” is value

   Di[2]=”February”

   Di[“First”]=”Monday”

   Print(Di)

   Output: {'First': 'Monday', 1: 'January', 2: 'February'}

Special Dictionary Operators

Traversing a Dictionary

A key of dictionary should be used within Square brackets to access elements of the Dictionary.

Syntax:

Dict_Name[key]

Example:

```
days={'S':'Sunday', 'M':'Monday', 'T':'Tuesday'}
print(days['M'])        Output: 'Monday'
print(days['S'])        Output: 'Sunday'
print(days['W'])        Output: KeyError: 'W'
```

Display All Keys of Dictionary using loop

```
days={'S':'Sunday', 'M':'Monday', 'T':'Tuesday'}
for key in days:
        print(key, end=”,”)
# Output:   S, M, T
```

Display All Values of Dictionary using loop

```
for v in days:
    print(days[v], end=”,”)
# Output:   Sunday, Monday, Tuesday
```

Display All Key-Valuepairs of Dictionary using loop

```
for i in days:
    print(i,":",days[i])
```

Output:

```
    S : Sunday
    M : Monday
    T : Tuesday
```

Appending Values in Dictionary

Dictionary is mutable so it support the addition of new element, deletion of existing element, extended with single pair of values, join two dictionary.

1. Add / Extend new element in dictionary.

   Syntax:

   Di_name[key]=new_value

- If key is already existed then its value will be displaced with given new value.
- If key is not existed then the new key will add in dictionary and the new value will be assigned to it.

Example:

```
days={'S':'Sunday', 'M':'Monday', 'T':'Tuesday'}
days['W']="Wednesday"
print(days)
```

Output:

```
{'S':'Sunday', 'M':'Monday', 'T':'Tuesday'], 'W':'Wednesday'}
```

2. Updating element in dictionary by using key

   Syntax:

   Di_name[key=new_value]

   Example:

```
days={'S':'Sunday', 'M':'Monday', 'T':'Tuesday'], 'W':'Wednesday'}
day['S']="Saturday"
print(days)
```

Output:

```
{'S':'Saturday', 'M':'Monday', 'T':'Tuesday'], 'W':'Wednesday'}
```

3. Merge Two dictionaries by using update() method

The update() method can merge the keys and values of one dictionary into another and over write values of same keys.

Syntax:

Di-1_name.update(di-2_name)

Example:

```
d1={1:"Sunday",2:"Monday"}
d2={3:"Tuesday", 4:"Wednesday"}
d1.update(d2)
print(d1)
```

Output: {1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

```
Print(d2)
```

Output: {3: 'Tuesday', 4: 'Wednesday'}

```
d3={2:"Holiday"}
d1.update(d3)    #Value of key 2 will be replaced
print(d1)
```

Output:

{1: 'Sunday', 2: 'Holiday', 3: 'Tuesday', 4: 'Wednesday'}

Membership Operator in Dictionary

Membership operator is Boolean type operator. It used to check whether a key is present in main Dictionary or not. There are two membership operators used in Dictionary.

1. in operator: If key is present in main Dictionary then it will return True otherwise it will return False result.

2. not in operator: If key is not present in main Dictionary then it will return True otherwise it will return False result.

Example:

```
d1={3: 'Tuesday', 4: 'Wednesday'}
Print( 3 in d1)              # Output: True
Print(3 not in d1)           # Output: False
Print(5 in d1)               # Output: False
Print(5 not in d1)           # Output: True
```

Dictionaries are Mutable

Dictionaries are Mutable in nature in Python. Its mean the content of Dictionary can be changed once it is created. In other words, the assignment operator works to change the content of Dictionary.

Example:

```
D={1:"One", 2:"Two"}
D[3]="Three"
Print(D)
```

Output: {1:"One", 2:"Two", 3:"Three"}

```
Lst[2]="Twentry"
Print(D)
```

Output: {1:"One", 2:"Twenty", 3:"Three"}

Dictionaries Built in

Note: All Dictionary methods change the original Dictionary because Dictionaries are Mutable.

1. Update()

The update() method can merge the keys and values of one dictionary into another and over write values of same keys.

Syntax:

```
Di-1_name.update(di-2_name)
```

Example:

```
d1={1:"Sunday",2:"Monday"}
d2={3:"Tuesday", 4:"Wednesday"}
d1.update(d2)
print(d1)
```

Output: {1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

```
Print(d2)
```

Output: {3: 'Tuesday', 4: 'Wednesday'}

```
d3={2:"Holiday"}
d1.update(d3)   #Value of key 2 will be replaced
print(d1)
```

Output:

{1: 'Sunday', 2: 'Holiday', 3: 'Tuesday', 4: 'Wednesday'}

2.len()

len() used to find the number of elements in dictionary or length of dictionary.

Syntax:

len(Dictionary_variable)

Example:

D={1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

Print(len(d))

Output: 4

3.keys()

The keys() return a List of keys from dictionary.

Syntax:

Dictionary_variable.keys()

Example:

D={1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

Print(D.keys())

Output: dict_keys([1, 2, 3, 4])

4.values()

The values() return a List of values from dictionary.

Syntax:

Dictionary_variable.values()

Example:

D={1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

Print(D.values())

Output:

dict_values(['Sunday', 'Monday', 'Tuesday', 'Wednesday'])

5.items()

The items() returns the content of dictionary as a list of tuples having key-value pairs.

Syntax:

Dictionary_variable.items()

Example:

D={1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

Print(D.items())

Output:

dict_items([(1, 'Sunday'), (2, 'Monday'), (3, 'Tuesday'), (4, 'Wednesday')])

6.get()

The get() returns a value for the given key. If key is not available then it returns None.

Syntax:

Di_variable.gey(key, default=None)

Where

Key- It is the key to be to be searched in the dictionary.

Default- It is the value returned in case key does not exist in the dictionary.

Example:

D={1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

D.get(2)

Output: ‘Monday’

D.get(6,”Not Found”)

Output: ‘Not Found’

Deletion Operation on Dictionary

The deletion operation performed by using following ways.

- If index is known- pop(index)
- Remove last element of dictionary- pop()
- Remove All elements of Dictionary- clear()
- Remove more than one but not all element- del keyword
- Remove dictionary variable- del keyword

1. clear()

clear() remove all elements of dictionary and make the dictionary an empty dictionary. This function not take any parameter.

Syntax:

Dictionary_variable.clear()

Example:

D= {1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'Wednesday'}

D.clear()

Output: {}

2. pop()

The pop() function used to remove an element from dictionary whose key is provided as parameter. Key parameter is mandatory.

If index is not provided than pop() will give error.

Note: pop() will show the deleted element.

Syntax:

Dictionary_variable.pop(key)

Example:

```
D1= {1: 'Sunday', 2: 'Holiday', 3: 'Tuesday'}
D1.pop(3)   # 'Tuesday' will remove from dictionary.
D1.pop(5)          # Error
Output: KeyError: 5
```

3. del keyword

The del keyword is used to delete not only dictionary but also any variable in Python from memory.

If we provide a particular key of dictionary then it will delete the particular element from dictionary and the dictionary variable remains in memory.

When we provide a dictionary variable without key then del keyword will delete the complete dictionary variable from memory and dictionary not further allowed to use.

Syntax:

del Dictionary_variable[index] OR

del dictionary_variable

Example:

```
D1={1: 'Sunday', 2: 'Monday', 3: 'Wednesday'}
del D1[2]
# It will remove the element of key-2 and Monday.
```

```
Print(D1)
Output: {1: 'Sunday', 3: 'Wednesday'}
del D1[5]
Output: KeyError: 5
del D1
# It will remove Dictionary variable from memory
print(D1)
```

NameError: name 'd1' is not defined

4. max()

This function will return maximum key from the dictionary.

Syntax:

```
max(Dictionary_Variable)
```

Example:

```
D1={1: 'Sunday', 2: 'Monday', 3: 'Holiday', 4: 'Wednesday', 5: 'Tuesday'}

Print(max(D1))                    # Output:5

days={'S': 'Sunday', 's': 'Saturday', 'M': 'Monday', 'T': 'Thursday'}

Print(max(days )                  # Output:’s’ (Small s)
```

5. min()

This function will return minimum key from the dictionary.

Syntax:

```
max(Dictionary_Variable)
```

Example:

```
D1={1: 'Sunday', 2: 'Monday', 3: 'Holiday', 4: 'Wednesday', 5: 'Tuesday'}
Print(min(D1))                  # Output:1
days={'S': 'Sunday', 's': 'Saturday', 'M': 'Monday', 'T': 'Thursday'}
    Print(min(days )       # Output:'M' (Capital M)
```

6. sum()

This function will return sum of keys from the dictionary.

Syntax:

```
sum(Dictionary_Variable)
```

Example:

```
D1={3: 'Tuesday', 4: 'Wednesday'}
Print(sum(D1))
```

Output: 7

7.copy()

This function used to create a copy of one dictionary into another.

Syntax:

```
New_dic=old_dic.copy()
```

Example:

```
D1={4: 'Wednesday'}
```

D2=D1.copy()

Print(D1) Output: {4: 'Wednesday'}

Print(D2) Output: {4: 'Wednesday'}

D1[1]=”Sunday”

Print(D1) Output: {1: “Sunday”, 4: 'Wednesday'}

Print(D2) Output: {4: 'Wednesday'}

Example:

D1={4: 'Wednesday'}

D2=D1

This will not create new location of D2, but both D1 and D2 will share the common memory location.

Explanation:

Print(D1) Output: {4: 'Wednesday'}

Print(D2) Output: {4: 'Wednesday'}

D1[1]=”Sunday”

Print(D1) Output: {1: “Sunday”, 4: 'Wednesday'}

Print(D2) Output: {1: “Sunday”, 4: 'Wednesday'}

Advance on Dictionary

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

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

Assignments:

1. Write a program to find sum of numeric keys of Dictionary without using sum().
2. WAP to find maximum key in dictionary without max().
3. WAP to find minimum key in dictionary without max().
4. WAP to enter name and marks of 10 student in dictionary where name should be key and marks should be values.