

Module 11

Dictionary

So indexing is possible.

- Dictionary has key value pair form of data
- Dictionaries are ordered (python 3.7 and above)
- Does not support indexing.
- Data can be accessed using Key.

In Python 3.6 and below, Dictionaries are Unordered.

Example

Key	Value
Name	Price
Banana	60
Mango	120
Apple	80
:	:

Creating a dictionary

```
d = {}  
print(type(d))  
<class 'dict'>  
print(d)  
{}
```

①

Two ways of creating an empty dictionary

```
d = dict()  
print(type(d))  
<class 'dict'>  
print(d)  
{}
```

②

```
fruits = {  
    "Banana": 60,  
    "Mango": 120,  
    "Apple": 80  
}  
type(fruits)  
dict
```

①

Two ways of creating a non-empty dictionary.

Key →

Value

fruits.

```
{ "Banana": 60, "Mango": 120, "Apple": 80 }
```

zip (Key, value)

```
name = [ "Orange", "Pine Apple", "Cherry" ]
```

```
prices = [ 120, 100, 130 ]
```

```
fruit1 = dict (zip (name, prices))
```

fruit1

```
{ 'Orange': 120, 'Pine Apple': 100, 'Cherry': 130 }
```

len (fruit1)

3

Accessing elements in Dictionary.

- Dictionaries don't support indexing

fruits.

```
{ "Banana": 60, "Mango": 120, "Apple": 80 }.
```

```
fruits [0]
```

Key Error : 0

```
fruits [ "Apple" ]
```

80

```
fruits [ "Mango" ]
```

120

```
fruits [ "Guava" ]
```

Key Error : 'Guava'

Method 1

```
fruits . get ( "Apple" )
```

80

```
fruits . get ( "Guava" )
```

```
fruits . get ( "Guava", "Not Available" )
```

'Not available'

Method 2 :

Using get () method.
This avoids KeyError while trying to access element which is not available in dictionary.

Updating a dictionary

fruits

```
{ 'Apple' : 120, 'Mango' : 100, 'PineApple' : 90 }
```

fruits ["PineApple"] ← updating the existing value 90

```
fruits [ "PineApple" ] = { "Small" : 90, "Large" : 120 }
```

fruits

```
{ 'Apple' : 120, 'Mango' : 100, 'PineApple' : { 'Small' : 90, 'Large' : 120 } }
```

fruits ["Guava"]

KeyError : 'Guava'

← Updating with new value

```
fruits [ "Guava" ] = 80
```

fruits

```
{ 'Apple' : 120,  
  'Mango' : 100,  
  'PineApple' : { 'Small' : 90, 'Large' : 120 },  
  'Guava' : 80 }
```

```
seasonalFruits = { "Grapes" : 120, "Oranges" : 70, "Berry" : 140 }
```

fruits. update (~~seasonalFruits~~)

fruits

← Update with another dictionary.

```
{ 'Apple' : 120,  
  'Mango' : 100,  
  'PineApple' : { 'Small' : 90, 'Large' : 120 },  
  'Guava' : 80,  
  'Grapes' : 120,  
  'Oranges' : 70,  
  'Berry' : 140 }
```


Deletion operations in Dictionary.

```
fruits = {  
    "Apple" : 120,  
    "Mango" : 100,  
    "Pine Apple" : 90  
}
```

Citizenship check

fruits

```
{ 'Apple' : 120, 'Mango' : 100, 'PineApple' : 90 }
```

"Apple" in fruits

True

"Berry" in fruits

False

dict.pop(key)

```
fruits.pop("Apple")
```

120

fruits

```
{ 'Mango' : 100, 'Pine Apple' : 90 }
```

```
fruits.pop()
```

TypeError: pop expected at least 1 argument, got 0

dict.popitem()

fruits

```
{ 'Apple' : 120, 'Mango' : 100, 'Pine Apple' : 90 }
```

```
fruits.popitem()
```

```
( 'Pine Apple' : 90 )
```

fruits

```
{ 'Apple' : 120, 'Mango' : 100 }
```

```
fruits.popitem()
```

```
( 'Mango' : 100 )
```

fruits

```
{ 'Apple' : 120 }
```

```
fruits.popitem()
```

```
( 'Apple' : 120 )
```

fruits

```
{ }
```

```
fruits.popitem()
```

KeyError: 'popitem(): dictionary is empty'

Delete object

fruits

```
{'Apple': 120, 'Mango': 100, 'PineApple': 90}
```

del fruits

fruits

NameError: name 'fruits' is not defined

X

Iteration in Dictionary

fruits = {

"Apple": 120,

"Mango": 100,

"PineApple": 90,

"Grapes": 120

}

for i in fruits:

print(i)

Apple

Mango

PineApple

Grapes

fruits["Apple"]

120

for i in fruits:

print(fruits[i])

120

100

90

120

for i in fruits:

print(i, fruits[i])

Apple 120

Mango 100

PineApple 90

Grapes 120

①

Using dict.items()

for key, value in fruits.items():
 print(key, value)

Apple 120
Mango 100
Pine Apple 90
Grapes 120

2

fruits.items()

dict_items([('Apple', 120), ('Mango', 100),
('Pine Apple', 90), ('Grapes', 120)])

More Dictionary methods

Keys, Values, items

fruits = {
 'Apple': 120,
 'Mango': 100,
 'PineApple': 90
}

fruits.keys()

dict_keys(['Apple', 'Mango', 'PineApple'])

fruits.values()

dict_values([120, 100, 90])

fruits.items()

dict_items([('Apple': 120), ('Mango': 100), ('PineApple': 90)])

Challenges

- ① Take an input and find the frequency of each letter and return the letter and their frequency.

Example : "Uvaraj Thillai"

"U" : 1

"v" : 1

"a" : 3

"n" : 1

"j" : 1

" " : 1

"T" : 1

"h" : 1

"i" : 2

"l" : 2

FYI

d = { "n" : 1, "n" : 3 }

d

{ 'n' : 3 }

name = "Uvaraj Thillai"

for i in name:

print(i, end = " ")

Uvaraj Thillai

freq = { }

for i in name:

check for presence

If i not in freq:

freq[i] = 1

if present, increment freq by 1.

else:

freq[i] += 1

print(freq)

{ 'U' : 1, 'v' : 1, 'a' : 3, 'n' : 1, 'j' : 1, ' ' : 1, 'T' : 1, 'h' : 1, 'i' : 2, 'l' : 2 }

② What is the output of the following code?

```
dict1 = {'age': 35, 'name': 'abc', 'salary': 45000}
```

```
val = dict1['age']
```

```
if val in dict1:
```

```
    print("This is a member of the dictionary")
```

```
else:
```

```
    print("This is not a member of the dictionary")
```

This is not a member of the dictionary

③ Consider the following dictionary:

```
freshers = {
```

```
    'student 1': {'name': 'Yash', 'salary': 7500},
```

```
    'student 2': {'name': 'Sheet', 'salary': 8000},
```

```
    'student 3': {'name': 'Smit', 'salary': 6500}
```

```
}
```

Which of the following is the correct way of changing the 'salary' of 'Smit' to 8500?

a) `freshers['Smit']['salary'] = 8500`

b) `freshers['student 3']['name' == 'Smit']['salary'] = 8500`

c) `freshers['student 3']['salary'] = 8500` ✓

d) `freshers['student 3']['salary' == 6500] = 8500`

④ What is the output of the following Python code snippet?

```
bbt = {'sheldon': 1, 'Leonard': 2}
```

```
bbt.update({'Penny': 2})
```

```
print(bbt)
```

```
{"sheldon": 1, "Leonard": 2, "Penny": 2}
```


⑤ Sum of values.

Given a dictionary, find the sum of values of every key in the dictionary.

Input Format:

The input contains two lines. The first line has space-separated string values which are the keys of the dictionary. The second line has space-separated integer numbers which are the values of the dictionary.

Output Format:

Print the Sum of the values of the items as an integer.

Sample Input:

```
x y z
25 25 50
```

Sample Output:

```
100
```

```
def returnSum(d):
```

```
    # Calculate the Sum of values in the dictionary
```

```
    total_sum = sum(d.values())
```

```
    return total_sum
```

```
keys = ["x", "y", "z"]
```

```
values = [25, 25, 50]
```

```
# Create dictionary from keys and values
```

```
dictionary = dict(zip(keys, values))
```

```
print(dictionary)
```

```
# output the Sum of values
```

```
print(returnSum(dictionary))
```

```
{'x': 25, 'y': 25, 'z': 50}
```

```
100
```


⑥ Adding Common Keys

Given two dictionaries, WAP for creating a dictionary in such a way that it consists of all the keys that are common in both dictionaries. The values corresponding to the keys in this new dictionary are the sum of the values of those keys in the two dictionaries.

Input format :

The input contains four lines. The first line is space separated string values which are the keys of the first dictionary. The second line is space separated integer numbers which are the values of the first dictionary. The third line is space separated string values which are the keys of the second dictionary. The fourth line is space separated integer numbers which are the values of the second dictionary.

Output format :

Print the resultant dictionary containing added values for common keys.

Sample Input :

a b c
1 2 3
c d e
4 5 6

Sample Output :

{ 'c' : 7 }

```
def commonKey(dict1, dict2):
```

```
    dict3 = {}
```

```
    # Find common keys and sum their values  
    for key in dict1:
```

```
        if key in dict2:
```

```
            dict3[key] = dict1[key] + dict2[key]
```

```
    print(dict3)
```

```
# Input Handling
```

```
keys1 = ['a', 'b', 'c']
```

```
values1 = [1, 2, 3]
```

```
keys2 = ['c', 'd', 'e']
```

```
values2 = [4, 5, 6]
```

```
# Create dictionaries from input
```

```
dict1 = dict(zip(keys1, values1))
```

```
dict2 = dict(zip(keys2, values2))
```

```
# Call the function with the two dictionaries
```

```
commonKey(dict1, dict2)
```

Output

{ 'c' : 7 }