**Python Dictionary🡪**

**1.Dictionary :**  
(i) A Dictionary is a mutable object in python , means the values of the dictionary can be changed

(ii)A Dictionary is a datatype that contains homogenous (same type) data normally as a value

(iii)A Dictionary is an unordered set of elements

(iv)Each element of a Dictionary is a pair of **key:value**, all the elements are enclosed within { }

(v)Each key in the Dictionary must be immutable object and unique, no duplicate key a Dictionary can contain

(vi)Each value in the Dictionary is a mutable object and same/different value a Dictionary can contain

(vii)A Dictionary is accessed by key, not by index number

(viii)Each value of a Dictionary can be accessed by accessing its corresponding key

(ix) keys( ) and values( ) methods are two popular methods of Dictionary those are frequently used

(x)**Example :**

(a) dt1 = { “a”:1 , “e”:2 , “i”:3 , “o”:4 , “u”:5 } --🡪vowel dictionary

(b) dt2 = { 1:”a” , 2:”e” , 3:”i” , 4:”o” , 5:”u” }

(c) dt3 = { 1:’Monday’ , 2:’Tuesday’ , 3:’Wednesday’ , 4:’Thursday’ , 5:’Friday’ , 6:’Saturday’ , 7:’Sunday’ }

**Practical🡪**

>>> dt1={"a":1,"e":2,"i":3,"o":4,"u":5} #declaring a dictionary (vowel dictionary)

>>>

>>> #accessing each element value of the dictionary dt1 by corresponding key

>>>

>>> print(dt1["a"]) # displaying value of the key – “a”

1

>>> print(dt1["b"])

Traceback (most recent call last):

File "<pyshell#6>", line 1, in <module>

print(dt1["b"])

KeyError: 'b'

>>> print(dt1["e"]) # displaying value of the key – “e”

2

>>> print(dt1["i"]) # displaying value of the key – “i”

3

>>> print(dt1["o"]) # displaying value of the key – “o”

4

>>> print(dt1["u"]) # displaying value of the key – “u”

5

2. **Data type used as key in the Dictionary :**

Each key in the dictionary must be immutable and unique. A dictionary cannot contain duplicate keys.

So only the following immutable objects/data types can be used as a key in the dictionary:

(i) Python string

(ii) a number

(iii) a tuple

**(i)Using python string as key in dictionary:**

>>> dt1={"Monday":1,"Tuesday":2,"Wednesday":3}

>>> print(dt1["Monday"])

1

>>> print(dt1["Tuesday"])

2

>>> print(dt1["Wednesday"])

3

>>>

**(ii)Using number as key in dictionary:**

>>> dt2={1:"a",2:"e",3:"i",4:"o",5:"u"}

>>>

>>> print(dt2[1])

a

>>> print(dt2[2])

e

>>> print(dt2[3])

i

>>> print(dt2[4])

o

>>> print(dt2[5])

u

>>>

**(iii)Using python tuple as key in dictionary:**

>>> dt3={(80,95):35,(70,80):45,(60,70):64}

>>>

>>> print(dt3[(80,95)])

35

>>> print(dt3[(70,80)])

45

>>> print(dt3[(60,70)])

64

>>>

**3.Use of keys( ) and values( ) – methods over a dictionary:**

>>> dt1={1:"a",2:"e",3:"i",4:"o",5:"u"}

>>>

**(i)Use of keys( ) – method - <dictionary name>.keys() 🡪**

>>> dt1.keys() # use-type-1

dict\_keys([1, 2, 3, 4, 5])

>>> print(dt1.keys()) #use-type-2

dict\_keys([1, 2, 3, 4, 5])

>>>

**(ii)Use of values( ) – method - <dictionary name>.values() 🡪**

>>> dt1.values() #use-type-1

dict\_values(['a', 'e', 'i', 'o', 'u'])

>>> print(dt1.values()) #use-type-2

dict\_values(['a', 'e', 'i', 'o', 'u'])

**4.Modifying, Changing and Expanding a Dictionary:**

**Declaring a dictionary🡪**

>>> Employee={'name':'John','salary':14500,'age':27}

>>> print(Employee)

{'name': 'John', 'salary': 14500, 'age': 27}

**(a)Modifying/Changing Dictionary value🡪**

**(i)Modifying name(value of key-‘name’)**

>>> Employee['name']='Arnold'

>>> print(Employee)

{'name': 'Arnold', 'salary': 14500, 'age': 27}

**(ii)Modifying salary(value of key-‘salary’)**

>>> Employee['salary']=15200

>>> print(Employee)

{'name': 'Arnold', 'salary': 15200, 'age': 27}

**(iii)Modifying age(value of key-‘age’)**

>>> Employee['age']=28

>>> print(Employee)

{'name': 'Arnold', 'salary': 15200, 'age': 28}

>>>

**(b)Expanding dictionary / adding elelmnts in dictionary(appending key:value pair in the dictionary):**

**(i)Expanding – key:value – ‘dept’:’production’🡪**

>>> Employee['dept']='production'

>>> print(Employee)

{'name': 'Arnold', 'salary': 15200, 'age': 28, 'dept': 'production'}

**(i)Expanding – key:value – ‘doj’:’25/6/2018’🡪**

>>> Employee['doj']='25/6/2018'

>>> print(Employee)

{'name': 'Arnold', 'salary': 15200, 'age': 28, 'dept': 'production', 'doj': '25/6/2018'}

5.keys( ) – keys( ) – is a built-in/system defined function of a dictionary that returns a list of keys of the dictionary.

The output of the keys( ) – function can be used within for loop to access each key of the dictionary one

after another.

Example –

>>> Employee= {'name': 'Arnold', 'salary': 15200, 'age': 28, 'dept': 'production', 'doj': '17/10/2018'}

(i)

>>> print(Employee.keys())

dict\_keys(['name', 'salary', 'age', 'dept', 'doj']) # keys( ) function returns a list of all keys from dictionary

(ii)

>>> L=Employee.keys() # storing the list of keys returned by keys( ) – function into L

>>> print(L)

dict\_keys(['name', 'salary', 'age', 'dept', 'doj']) # keys( ) function returns a list of all keys from dictionary

6.values( ) – values( ) – is a built-in/system defined function of a dictionary that returns a list of values of the

dictionary. The output of the valuess( ) – function can be used within for loop to access each value of

the dictionary one after another.

Example –

>>> Employee= {'name': 'Arnold', 'salary': 15200, 'age': 28, 'dept': 'production', 'doj': '17/10/2018'}

(i)

>>> print(Employee.values())

dict\_values(['Arnold', 15200, 28, 'production', '17/10/2018']) # values( ) function returns a list of all values from

dictionary

(ii)

>>> Lst=Employee.values() # storing the list of values returned by values( ) – function into Lst

>>> print(Lst)

dict\_values(['Arnold', 15200, 28, 'production', '17/10/2018']) # values( ) function returns a list of all values from

dictionary

Q.1. Print the keys and values of the following dictionary using for loop and list concept :

Employee= {'name': 'Arnold', 'salary': 15200, 'age': 28, 'dept': 'production', 'doj': '17/10/2018'}

Ans -

Employee= {'name': 'Arnold', 'salary': 15200, 'age': 28, 'dept': 'production', 'doj': '17/10/2018'}

Lkey=Employee.keys() #Lkey=['name', 'salary', 'age', 'dept', 'doj']

Lvalue=Employee.values() #Lvalue=['Arnold', 15200, 28, 'production', '17/10/2018']

print("Displaying all keys: ")

for k in Lkey: #Lkey=['name', 'salary', 'age', 'dept', 'doj'] , k---->'name', 'salary' , 'age' , 'dept' , 'doj'

print(k)

print("Displaying all values: “)

for v in Lvalue: ") #Lvalue=['Arnold', 15200, 28, 'production', '17/10/2018'] , v----> 'Arnold' , 15200 , 28 ,

print(v) 'production' , '17/10/2018'

**4.Multiple ways/methods to create a dictionary:**

Method-1 : Declaring/Initializing a dictionary:

>>> Employee={'name':'John','salary':12600,'age':26,'dept':'Production'}

>>> print(Employee)

{'name': 'John', 'salary': 12600, 'age': 26, 'dept': 'Production'}

Method-2 : Creating Empty dictionary:

**(i) by using { } – braces :**

>>> Student={}

>>> print(Student)

{}

**(ii)by using dict( ) – function :**

>>> Student=dict()

>>> print(Student)

{}

Method-3 : Creating a data dictionary usinf **dict( ) function** in **4 ways** :

**Way-1 : using dict( ) function , key without quotes , =(equalto) symbol instead of : (colon)🡪**

>>> Student=dict(name='John',cls=12,sec='A',marks=85)

>>> print(Student)

{'name': 'John', 'cls': 12, 'sec': 'A', 'marks': 85}

**Way-2 : using dict( ) function and conventional way of key:value method 🡪**

>>> Employees=dict=({'name':'John','salary':12600,'age':26,'dept':'Production'})

>>> print(Employees)

{'name': 'John', 'salary': 12600, 'age': 26, 'dept': 'Production'}

**Way-3 : using dict( ) function with zip( ) function 🡪**

**(set of keys and set of values are given separately) 🡪**

>>> Employee = dict( zip( ('name', 'salary', 'age'), ('John', 14500, 27) ) )

>>> print(Employee)

{'name': 'John', 'salary': 14500, 'age': 27}

**Way-4 : using dict( ) function and [ ] and ,(comma) between key and value 🡪[key,value] 🡪**

>>> Worker=dict([['name','John'],['salary',14700],['age',28]])

>>> print(Worker)

{'name': 'John', 'salary': 14700, 'age': 28}

**5.Deleting elements(key & value) from dictionary 🡪**

>>> Employee={'name':'John','salary':12600,'age':26,'dept':'Production'}

>>> del Employee['dept']

>>> print(Employee)

{'name': 'John', 'salary': 12600, 'age': 26}

**6.Checking the existence of a key in a dictionary🡪**

Using the membership operator **‘in’** and **‘not in’** 🡪

>>> Employee={'name':'John','salary':12600,'age':26,'dept':'Production'}

>>>

**>>> 'name' in Employee**

**True**

>>>

**>>> 'salary' in Employee**

**True**

>>>

**>>> 'dept' not in Employee**

**False**

>>>

**>>> 'location' not in Employee**

**True**

**7.Checking the existence of a value in a dictionary🡪**

>>> Employee={'name':'John','salary':12600,'age':26,'dept':'Production'}

**>>> Employee.values()**

**dict\_values(['John', 12600, 26, 'Production'])**

>>> 'John' in Employee.values()

True

>>> 12600 in Employee.values()

True

>>>

>>> 15300 in Employee.values()

False

>>>

>>> 'John' not in Employee.values()

False

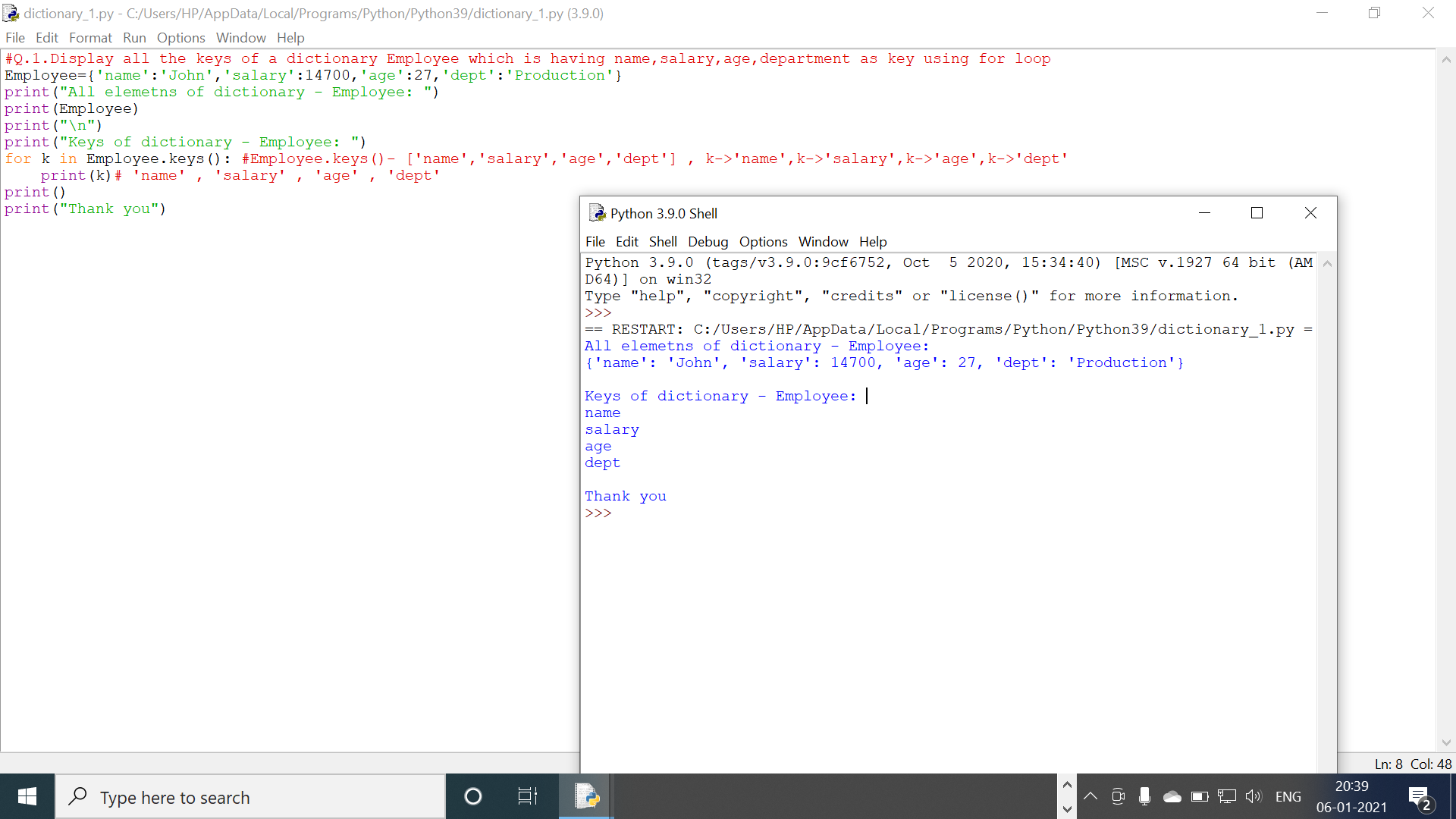
>>>

>>> 'Arnold' not in Employee.values()

True

Dictionary Program

Program-1



Program-2

