

## TYPE-B:DICTIONARY:CH-13

1) Which of the following will result in an error for a given valid dictionary D?

D + 3  
 D \* 3  
 D + {3 : "3"}  
 D.update( {3 : "3"})  
 D.update { {"3" : 3}}  
 D.update("3" : 3)

sol:

D + 3  
 → Error (TypeError)

D \* 3  
 → Error (TypeError)

D + {3 : "3"}  
 → Error (TypeError)

D.update({3 : "3"})  
 → No error (this is valid)

D.update { {"3" : 3} }  
 → Error (SyntaxError)

D.update("3" : 3)  
 → Error (SyntaxError)

2) The following code is giving some error. Find out the error and correct it.

```
d1 = {"a" : 1, 1 : "a", [1, "a"] : "two"}
```

sol:

the error is key is given as mutable element . it can be corrected as `d1 = {"a" : 1, 1 : "a", (1, "a") : "two"}`

3) The following code has two dictionaries with tuples as keys. While one of these dictionaries being successfully created, the other is giving some error. Find out which dictionary will be created successfully and which one will give error and correct it :

```
dict1 = { (1, 2) : [1, 2], (3, 4) : [3, 4]}  

dict2 = { ([1], [2]) : [1,2], ([3], [4]) : [3, 4]}
```

sol:

dict1 created successfully without any errors but the dict2 rises error as the tuple contains mutable type and as a whole it placed as a key. it can be modified as ;  
`dict2 = { (1, 2) : [1,2], (3, 4) : [3, 4]}`

4) Nesting of dictionary allows you to store a dictionary inside another dictionary. Then why is following code raising error ? What can you do to correct it ?

```
d1 = {1 : 10, 2 : 20, 3 : 30}
d2 = {1 : 40, 2 : 50, 3 : 60}
d3 = {1 : 70, 2 : 80, 3 : 90}
d4 = {d1 : "a", d2 : "b", d3 : "c"}
```

sol:

In a dictionary, keys must be hashable (immutable).

Here, d1, d2, and d3 are dictionaries. Dictionaries are mutable and unhashable, so they cannot be used as keys.

Nesting of dictionaries means a dictionary can be stored as a value inside another dictionary, not as a key.

corrected form : d4 = {"a": d1, "b": d2, "c": d3}

5) Why is following code not giving correct output even when 25 is a member of the dictionary?

```
dic1 = {'age': 25, 'name': 'xyz', 'salary': 23450.5}
val = dic1['age']
if val in dic1:
    print("This is member of the dictionary")
else :
    print("This is not a member of the dictionary")
```

sol:

The code gives wrong output because the in operator checks only dictionary keys, not values. Here, val = 25 is a value, not a key, so 25 in dic1 is false.

Correction:

```
if val in dic1.values():
    print("This is member of the dictionary")
```

6) What is the output produced by the following code :

```
d1 = {5 : [6, 7, 8], "a" : (1, 2, 3)}
print(d1.keys())
print(d1.values())
```

sol:

```
dict_keys([5, 'a'])
dict_values([[6, 7, 8], (1, 2, 3)])
```

7) Consider the following code and then answer the questions that follow :

```
myDict = {'a' : 27, 'b' : 43, 'c' : 25, 'd' : 30}
valA = ""
for i in myDict :
    if i > valA :
        valA = i
        valB = myDict[i]
print(valA)          #Line1
print(valB)          #Line2
print(30 in myDict)  #Line3
```

```

myLst = list(myDict.items())
myLst.sort()          #Line4
print(myLst[-1])      #Line5

```

- 1)What output does Line 1 produce ?
- 2)What output does Line 2 produce ?
- 3)What output does Line 3 produce ?
- 4)What output does Line 5 produce ?
- 5)What is the return value from the list sort( ) function (Line 4) ?

sol:

- 1) valA stores the largest key alphabetically.

Keys are: a, b, c, d → largest is 'd'

Output (Line 1): d

- 2) Output of Line 2

valB stores the value corresponding to key 'd'.

Value of 'd' is 30

Output (Line 2): 30

- 3) Output of Line 3

30 in myDict checks whether 30 is a key, not a value.

30 is a value, not a key.

Output (Line 3): False

- 4) Output of Line 5

myDict.items() → [('a',27), ('b',43), ('c',25), ('d',30)]

After sorting, the last element is the largest key-value pair.

Output (Line 5): ('d', 30)

- 5) Return value of list sort() function (Line 4)

The sort() function sorts the list in place and returns None.

Return value: None

- 8) What will be the output produced by following code ?

```

d1 = { 5 : "number", "a" : "string", (1, 2): "tuple" }
print("Dictionary contents")
for x in d1.keys():    # Iterate on a key list
    print (x, ': ', d1[x], end = ' ')
    print (d1[x] * 3)
    print ( )

```

sol:

Dictionary contents

5 : number numbernumbernumber

a : string stringstringstring

(1, 2) : tuple tupletupletuple

- 9)(a) Predict the output:

```
d = dict()
```

```
d['left'] = '<'
d['right'] = '>'
print('{left} and {right} or {right} and {left}')
```

```
sol:
{left} and {right} or {right} and {left}
```

9)(b) Predict the output:

```
d = dict()
d['left'] = '<'
d['right'] = '>'
d['end'] = ''
print(d['left'] and d['right'] or d['right'] and d['left'])
print(d['left'] and d['right'] or d['right'] and d['left'] and d['end'])
print((d['left'] and d['right'] or d['right'] and d['left']) and d['end'])
print("end")
```

```
sol:
>
>
end
```

9)(c) Predict the output:

```
text = "abracadabraaabbccrr"
counts = {}
ct = 0
lst = []
for word in text:
    if word not in lst:
        lst.append(word)
        counts[word] = 0
    ct = ct + 1
    counts[word] = counts[word] + 1
print(counts)
print(lst)
```

```
sol:
{'a': 7, 'b': 4, 'r': 4, 'c': 3, 'd': 1}
['a', 'b', 'r', 'c', 'd']
```

9)(d) Predict the output:

```
list1 = [2, 3, 3, 2, 5, 3, 2, 5, 1, 1]
counts = {}
ct = 0
lst = []
for num in list1:
    if num not in lst:
        lst.append(num)
        counts[num] = 0
    ct = ct+1
```

```
counts[num] = counts[num]+1
print(counts)
for key in counts.keys():
    counts[key] = key * counts[key]
print(counts)
```

```
sol:
{2: 3, 3: 3, 5: 2, 1: 2}
{2: 6, 3: 9, 5: 10, 1: 2}
```

10) 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.  
Perform the following operations on this dictionary :

- 1) Display the keys
- 2) Display the values
- 3) Display the items
- 4) Find the length of the dictionary
- 5) Check if 7 is present or not
- 6) Check if 2 is present or not
- 7) Retrieve the value corresponding to the key 9
- 8) Delete the item from the dictionary corresponding to the key 9

```
sol:
1) print(ODD.keys())
2) print(ODD.values())
3) print(ODD.items())
4) print(len(ODD))
5) print(7 in ODD)
6) print(2 in ODD)
7) print(ODD[9])
8) del ODD[9]
print(ODD)
```

11)(a) Find the errors:

```
text = "abracadbra"
counts = {}
for word in text :
    counts[word] = counts[word] + 1
```

```
sol:
The key word does not exist in the dictionary counts the first time it is encountered, so this causes a KeyError.
```

```
11)(b) my_dict = {}
my_dict[(1,2,4)] = 8
my_dict[[4,2,1]] = 10
print(my_dict)
```

```
sol:
Dictionary keys must be hashable (immutable).
Here, [4,2,1] is a list, and lists are mutable and unhashable, so they cannot be used as dictionary keys.
```

12)(a) Predict the output:

```
fruit = {}
L1 = ['Apple', 'banana', 'apple']
for index in L1 :
    if index in fruit:
        fruit[index] += 1
    else :
        fruit[index] = 1
print(len(fruit))
print(fruit)

sol:
3
{'Apple': 1, 'banana': 1, 'apple': 1}
```

12)(b) Predict the output:

```
arr = {}
arr[1] = 1
arr['1'] = 2
arr[1] += 1
sum = 0
for k in arr:
    sum += arr[k]
print(sum)
```

```
sol:
4
```

13(a) Predict the output

```
a = {(1,2):1,(2,3):2}
print(a[1,2])
```

```
sol:
1
```

13)(b) Predict the output

```
a = {'a':1, 'b':2, 'c':3}
print(a['a','b'])
```

```
sol:
ERROR!
Traceback (most recent call last):
  File "<main.py>", line 2, in <module>
    KeyError: ('a', 'b')
```

```
14)(a) box = {}
      jars = {'Jam' :4}
      crates = {}
      box['biscuit'] = 1
      box['cake'] = 3
```

```
crates['box'] = box
crates['jars'] = jars
print(len(crates[box]))
```

sol:

box is a dictionary, and dictionaries are unhashable, so they cannot be used as keys to access another dictionary.

Here, the keys of crates are strings:  
'box' and 'jars'

But the code is trying to access crates using box (a dictionary), which causes:

TypeError: unhashable type: 'dict'

Corrected code:

```
print(len(crates['box']))
```

```
14)(b) box = {}
    jars = {'Jam':4}
    crates = {}
    box['biscuit'] = 1
    box['cake'] = 3
    crates['box'] = box
    crates['jars'] = jars
    print(len(crates['box']))
```

sol:

2

15) Predict the output :

```
dct = {}
dct[1] = 1
dct['1'] = 2
dct[1.0] = 4
sum = 0
for k in dct:
    print(k, sum)
    sum += dct[k]
print(sum)
```

sol:

1 0

1 4

6

16) Fill in the blanks of the following code so that the values and keys of dictionary d are inverted to create dictionary fd.

```
d = {'a':1, 'b':2, 'c':3}
print(d)
fd = {}
for key, value in d.____():
```

```
fd[____] = ____  
print(fd)
```

sol:

```
items  
value  
key
```