Python Dictionary Interview Questions

Dictionaries is a datatype in python programming language and also one of the most favorite topics for interviewers. We have shortlisted some of the most important dictionary interview questions which will help you python interviews.

What are dictonaries in Python?

Dictonaries are the datatypes in python in which the dataitems are arranged in the form of keys and values. It is opposite to sequential datatypes, where the data items were arranged by range.

```
#Example of a Dictionary

dict1 = {"name" :  "mradul" , "age" :  23,"city" :  "indore"}

print(dict1)
```

Is python dictionaries mutable?

Yes, dictionaries are mutable you can add, update, delete key, value pairs from it without changing object memory.

For example:

```
>>> d = {1:10,2:20}

>>> id(d)                    #  2189559519040

>>> d[3] = 30

>>> id(d)                    #  2189559519040
```

How to print all keys inside Python Dictionary?

To print all keys inside dictionary you need to call .keys() object inside the dictionary variable.

```
dict1 = {"name":"mradul","age":23,"city":"indore"}

print(dict1.values())
```

Can we use tuple as keys inside python dictionary?

Yes, tuple can we used as key inside python dictionary, only if it contain only string, number or tuple. If a tuple contains any mutable datatype inside it like list, it can not be used as keys.

Can we use lists as keys inside python dictionary?

No, python list can not be used as keys inside python dictionary, as they are mutable in nature.

If we used list objects as keys of dictionary object then it throws exception like TypeError

For example:

```
>>> d = { [ 1 , 2 ] : 10}

TypeError: unhashable type: 'list'
```

What is .item() method inside a dictionary?

.item() is used to arrange the complete dictionary in the form of list with tuples carrying key, value pair.

```
dict1 = {"name" : "mradul" , "age" : 23 , "city" : "indore"}
print(dict1.items())
```

```
# Output

dict_items( [ ('name', 'mradul') , ('age', 23) , ('city', 'indore') ] )
```

How can we loop a dictionary using dict.items() method?

To loop a dictionary you can use the following code.

```
dict1 = {"name" : "mradul" , "age" : 23 , "city" : "indore"}

for k, v in dict1.items():

    print(k, v)
```

What is enumarate function inside a dictionary?
You can using enumerate() function with dictionary to get position index and corresponding index at the same time.

```
for  i  in  enumerate(dict1):
   print(i)
```

```
# output
(0, 'name')
(1, 'age')

(2, 'city')
```

What is zip function in python dictionary and how can you use it? How can combine two dictionaries together?

```
questions = ['name', 'location', 'favorite language']
answers = ['Codersdaily', 'Indore', 'Python']
for q, a in zip(questions, answers):
   print('What is your {0}?  It is {1}.'.format(q, a))
```

```
Output:
```

```
What is your name?  It is Codersdaily.
What is your location?  It is the Indore.

What is your favorite language?  It is Python.
```

Write a code to create a dictionary using dictionary comprehension?

```
>>> {x: x**2 for x in (2, 4, 6)}

{2: 4, 4: 16, 6: 36}
```

What does .pop() method do in dictionary?
pop() method inside dictionary can be used to remove the item based on given key.

```
dict1={"name" : "mradul","age" : 23,"city" :  "indore"}
```

```
print(dict1)
dict1.pop("city")
print(dict1)
```

```
#Output
{'name': 'mradul', 'age': 23, 'city': 'indore'}

{'name': 'mradul', 'age': 23}
```

Note: If given key is not available then it throws exception like KeyError.

```
dict1.pop('salary')
```

```
Output:

KeyError:  'salary'
```

How to merge more than one dictionary

Python dictionary can be merger as {**dict_1, **dict_2, ...,**dict_n}. For Python 3.9+ its can be merged using "|" operator.

```
# Merge Dictionary
dict_1 = {'name': 'Harry', 'age': 27, 'location': 'Helsinki'}

dict_2 = {'job': 'Architect'}

# Merge dict using ** argument

dict_merged_1 = {**dict_1, **dict_2}

print(dict_merged_1)

# Python 3.9 has new feature merge "|"

operatordict_merged_2 = dict_1 | dict_2

print(dict_merged_2)
```

Output

```
{'name': 'Harry', 'age': 27, 'location': 'Helsinki', 'job': 'Architect'}

{'name': 'Harry', 'age': 27, 'location': 'Helsinki', 'job': 'Architect'}
```

Write a logics to find out who are learning a specific course? (return course name as a key and person names as a list of values) ?

input_dict = {"Ramesh" : "Python", "Ravi" : "Django", "Virat":"Django", "Surya" : "Python", "Rohit" : "RESTAPI"}

output_dict = {'Python': ['Ramesh', 'Surya'], 'Django': ['Ravi', 'Virat'], 'RESTAPI': ['Rohit']}

Code:

```
input_dict = {"Ramesh" : "Python", "Ravi" : "Django", "Virat":"Django", "Surya" : "Python", "Rohit" : "RESTAPI"}

output_dict = {}

for k,v in input_dict.items():

    if v not in output_dict:

        output_dict[v] = [k]

    else:

        output_dict[v].append(k)

print( output_dict )
```

How to convert list to a dictionary

So, as you know dictionary has key and value pair but the list does not. So, some cases of converting lists into dictionaries are.

a. In case of one list

Use the list items as keys and then provide values some way.

```
## List to dictionary
pets= ['dog','cat','guinea pig', 'parrot']

# add one value to all
pets_owner = {animal:'Jackson' for animal in pets}

print(pets_owner)
```

Output

```
{'dog': 'Jackson', 'cat': 'Jackson', 'guinea pig': 'Jackson', 'parrot':
'Jackson'}
```

b. For loop and Zip with two lists

```
## List to dictionary
pets= ['dog','cat','guinea pig', 'parrot']

# Adding two list using zip and for loops

numbers =  [2,4,10,2]

pet_number_dict={}

for animal,num in zip(pets,numbers):
    pet_number_dict[animal] = num

print(pet_number_dict)
```

Output

```
{'dog': 2, 'cat': 4, 'guinea pig': 10, 'parrot': 2}
```

c. Zip, Comprehension with two lists

```
## List to dictionary
pets= ['dog','cat','guinea pig', 'parrot']

# Add different values  using zip  and comprehension
pet_number_dict_2 = {animal:num for animal,num in zip(pets,numbers)}

print(pet_number_dict_2)
```

Output

```
{'dog': 2, 'cat': 4, 'guinea pig': 10, 'parrot': 2}
```

What is the difference between duplicating dictionary with and without copy()?

What the question means is dict_2 = dict_1 vs. dict_2 = dict_1.copy(). When you are duplicating a dictionary object without a copy() method, you are not creating a new dictionary but pointing to the same dictionary object. So, when you make changes in the duplicate list it changes the original one too.

```
# Duplicating dict with and without copy
to_buy_list = {
   'eggs': '1 karton',
   'banana': '1 kg',
   'milk': '1 ltr',
   'sugar': '1 kg'
}

print("Original List before {}".format(to_buy_list))

# Lets duplicate without copy
to_buy_list_2 = to_buy_list

# Lets make change to duplicate list
to_buy_list_2['salt'] = '1 kg'

print("Original List after duplication {}".format(to_buy_list))

print("Are the memory address of two dicts same? {}".format(id(to_buy_list)
== id(to_buy_list_2)))
```

Output:

```
Original List before {'eggs': '1 karton', 'banana': '1 kg', 'milk': '1
 ltr', 'sugar': '1 kg'}

Original List after duplication {'eggs': '1 karton', 'banana': '1 kg',
'milk': '1 ltr', 'sugar': '1 kg', 'salt': '1 kg'}
```

Note: **Are the memory address of two dicts same? True**

Now let’s do all that with copy() method.

```
# Duplicating dict with and without copy
to_buy_list = {
'eggs': '1 karton',
'banana': '1 kg',
'milk': '1 ltr',
'sugar': '1 kg'
}

print("Original List before {}".format(to_buy_list))

# Lets duplicate without copy
to_buy_list_2 = to_buy_list.copy()

# Lets make change to duplicate list
to_buy_list_2['salt'] = '1 kg'
```

```
print("Original List after duplication {}".format(to_buy_list))

print("Are the memory address of two dicts same? {}".format(
id(to_buy_list) == id(to_buy_list_2)))
```

Output:

```
Original List before {'eggs': '1 karton', 'banana': '1 kg', 'milk': '1 ltr',
'sugar': '1 kg'}

Original List after duplication {'eggs': '1 karton', 'banana': '1 kg',
'milk': '1 ltr', 'sugar': '1 kg'}

Are the memory address of two dicts same? False
```

Are dictionaries case-sensitive?

Yes, dictionaries are case-sensitive, i.e., the same name of keys, but different cases are treated differently, i.e., ‘apple’ and ‘APPLE’ will be treated as separate keys.

What are different ways of creating a Dictionary?

Three different ways of creating a Dictionary are:

1. Create an empty Dictionary

Dictionary1 = {}

print(Dictionary1)

Output: {}

key1 = 'a'
value1 = 1
Dictionary1[key1] = value1

Output: { 'a' : 1}

2. Create Dictionary using dict() method

```
Dictionary1 = dict({1: 'a', 2: 'b'})

print(Dictionary1)
```

Output: `{1: 'a', 2: 'b'}`

3. Create Dictionary with each item as Pair

```
Dictionary1 = dict([(1,'a'), (2, 'b')])
```

```
print(Dictionary1)
```

Output : {1: 'a', 2: 'b'}

4. Creating Dictionary directly

```
Dictionary1 = {1: 'a', 2: 'b'}
```

Output: `{1: 'a', 2: 'b'}`

What is a Nested Dictionary? How is it created?

A dictionary inside the dictionary is known as a “Nested Dictionary”. For ex-

```
dictionary1 = {
                1 : {'roll': '101', 'name': 'sam'},
                2 : {'roll': '102', 'name': 'ram'}
              }
print(dictionary1)
```

Output: `{1: {'roll': '101', 'name': 'sam'}, 2: {'roll': '102', 'name': 'ram'}}`

The elements of nested dictionary can be accessed using

```
print(dictionary[1]['roll'])
```

Output: `101`

How do you add an element in Dictionary?

Elements in a Dictionary can be added in multiple ways:

1. Adding one pair at a time

```
Dict1 ={}
```

```
Dict1[0] = 'a'
```

```
Dict1[1] = 'b'

print("Dictionary after adding 3 elements: ", Dict1)
```

Output: `{0: 'a', 1: 'b' }`

2. Adding more than one value to a single key

```
Dict1['values'] = 4, 5, 6

print("Dictionary after adding multiple values to a key: ", Dict1)
```

Output: `{0: 'a', 1: 'b', 'values': (4, 5, 6) }`

3. Adding nested key-value pair

```
Dict1['Nested'] = {1: 'Analytics', 2: 'Life'}

print(Dict1)
```

Output:

```
{0: 'a', 1: 'b', 'values': (4, 5, 6), 'Nested': {1: 'Analytics', 2: 'Life'} }
```

Create a dictionary from a given list. For instance- Input : [1, ‘a’, 2, ‘b’, 3, ‘c’] and Output : {1: ‘a’, 2: ‘b’, 3: ‘c’}

```
def Convert_list_dict(dict2):

    x = iter(dict2)

    res_dct1 = dict(zip(x, x))

    return res_dct1

dict1 = [1, 'a', 2, 'b',3, 'c']

print(Convert_list_dict(dict1))
```

Output : {1: 'a', 2: 'b', 3: 'c'}

Create a list of tuples from the dictionary

The list of tuples can be created in following way:

```
dict1 = { 1: 'a', 2: 'b', 3: 'c' }

lst1 = list(dict1.items())

print(lst1)
```

Output: `[(1, 'a'), (2, 'b'), (3, 'c')]`

Create a list from the dictionary.

Suppose the given dictionary is:

```
dict1 = { 1: 'a', 2: 'b', 3: 'c' }
```

A list can be created using the below code:

```
x = list(dict1.keys())

y = list(dict1.values())

for i in y:
      x.append(i)

print(x)
```

Output: `[1, 2, 3, 'a', 'b', 'c']`

How can you delete key-value pair from Dictionary?

Key-value pair can be deleted by using ‘del’ keyword as shown below:

```
del dict1[1]

print(dict1)
```

Output: `{2: 'b', 3: 'c' }`

Is the dictionary mutable?

The term ‘Mutable’ means we can add, remove or update key-value pairs in a dictionary. Yes, the dictionary is mutable. For instance,

```
Dict1 = {1: 'a', 2: 'b', 3: 'c', 4: 'd' }
Dict1[2] = 'h'
print(Dict2)
```

Output:

```
{1: 'a', 2: 'h', 3: 'c', 4: 'd' }
```

Given two lists, create a dictionary from them.

Input: [1, 2, 3, 4, 5], [‘a’, ‘b’, ‘c’, ‘d’, ‘e’]

Output: {1: ‘a’, 2: ‘b’, 3: ‘c’, 4: ‘d’, 5: ‘e’}

Let’s define these two lists as list1 and list2 as follows:

```
list1 = [1, 2, 3, 4, 5]
list2 = ['a', 'b', 'c', 'd', 'e']
dict1 = {}
for i, j in zip(list1, list2):
    dict1[i] = j
print(dict1)
```

Output:

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

Another way of achieving the same output:

```
dict1 = {i:h for i,j in zip(list1, list2)}
print(dict1)
```

Output:

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

Given two lists, create a dictionary from them.

Input: [1, 2, 3, 4, 5], [‘a’, ‘b’, ‘c’, ‘d’, ‘e’] and Output: {1: ‘a’, 2: ‘b’, 3: ‘c’, 4: ‘d’, 5: ‘e’}

Let’s define these two lists as list1 and list2 as follows:

```
list1 = [1, 2, 3, 4, 5]
list2 = ['a', 'b', 'c', 'd', 'e']
dict1 = {}

for i, j in zip(list1, list2):
    dict1[i] = j
```

```
print(dict1)
```

Output: `{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}`

Another way of achieving the same output:

```
dict1 = {i:h for i,j in zip(list1, list2)}

print(dict1)
```

Output: `{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}`

Write a code to sort dictionaries using a key.

Input: { 2 : 'Apple' , 1 : 'Mango' , 3 : 'Orange' , 4 : 'Banana' }

Output:

```
1: Mango
2: Apple
3: Orange
4: Banana
```

Below is the code to sort dictionaries using the key:

```
dict1 = {2: 'Apple', 1:'Mango', 3:'Orange', 4:'Banana'}
print(sorted(dict1.keys()))
for key in sorted(dict1):
      print("Sorted dictionary using key:",(key, color_dict[key]))
```

Output:

```
[1, 2, 3, 4]
1: Mango
2: Apple
3: Orange
4: Banana
```

- Exercise 1: Convert two lists into a dictionary
- Exercise 2: Merge two Python dictionaries into one
- Exercise 3: Print the value of key 'history' from the below dict

- Exercise 4: Initialize dictionary with default values
- Exercise 5: Create a dictionary by extracting the keys from a given dictionary
- Exercise 6: Delete a list of keys from a dictionary
- Exercise 7: Check if a value exists in a dictionary
- Exercise 8: Rename key of a dictionary
- Exercise 9: Get the key of a minimum value from the following dictionary
- Exercise 10: Change value of a key in a nested dictionary