MIT World Peace University

Python Programming
Second Year B. Tech, Semester 4

Different Operations on the Dictionary and Tuple Data Structures

Assignment No. 5

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

February 28, 2023

Contents

1 Aim

Write a python program to create, append and remove etc. operation on Dictionary and Tuple.

2 Objectives

1. To learn and implement Dictionary and Tuple Data Structure.

3 Theory

3.1 Different Operations performed on Dictionaries

Following are the different operations performed on Dictionaries:

- ***Creating a Dictionary***
- ***Accessing elements from a Dictionary***
- ***Changing and Adding Dictionary elements***
- ***Removing elements from a Dictionary***

3.1.1 Creating a Dictionary

A dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a 'comma'.

```
>>> d={1:'a',2:'b'}
>>> d
{1: 'a', 2: 'b'}
```

3.1.2 Accessing elements from a Dictionary

While indexing is used with other container types to access values, a Dictionary uses keys. Key can be used either inside square brackets [] or with the get() method. If we use the square brackets [], then a KeyError is raised in case a key is not found in the dictionary. On the other hand, the get() method returns None if the key is not found.

```
>>> d={1:'a',2:'b'}
>>> d[1]
'a'
>>> d.get(1)
'a'
```

3.1.3 Changing and Adding Dictionary elements

In Python, Dictionary are mutable. It means that we can change the content of a dictionary any time. To add a new item to the dictionary, we can use the familiar square brackets along with the new key.

```
>>> d={1:'a',2:'b'}
>>> d[3]='c'
>>> d
{1: 'a', 2: 'b', 3: 'c'}
```

3.1.4 Removing elements from a Dictionary

There are various methods to remove items from a dictionary:

- ***Using pop() method***
- ***Using popitem() method***
- ***Using del keyword***
- ***Using clear() method***

3.2 Different Operations performed on Tuples

Following are the different operations performed on Tuples:

- ***Creating a Tuple***
- ***Accessing elements from a Tuple***
- ***Changing and Adding Tuple elements***
- ***Removing elements from a Tuple***

3.2.1 Creating a Tuple

A tuple is a collection which is ordered and unchangeable. In Python tuples are written with round brackets.

```
>>> t = ("apple", "banana", "cherry")
>>> t
('apple', 'banana', 'cherry')
```

3.2.2 Accessing elements from a Tuple

You can access tuple items by referring to the index number, inside square brackets.

```
>>> t = ("apple", "banana", "cherry")
>>> t[1]
'banana'
```

3.2.3 Changing and Adding Tuple elements

Tuples are unchangeable, so you cannot add items to it after it has been created.

```
>>> t = ("apple", "banana", "cherry")
>>> t[3] = "orange"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

3.2.4 Removing elements from a Tuple

Tuples are unchangeable, so you cannot remove items from it, but you can delete the tuple completely.

```
>>> t = ("apple", "banana", "cherry")
>>> del t
```

4 Input and Output

4.1 Input

Different Dictionary and Tuple Data Structure and different operations

4.2 Output

Display Different operation performed on Dictionary and Tuple Data Structure

5 Code

5.0.1 Write a python program to create, append and remove etc. operation on Dictionary and Tuple.

5.0.2 Creating a tuple

[3]:
```
my_tuple = (1, 2, 3, 4, 4, 4, 5, 6, 7, 8, 9, 10)
print(my_tuple)
print("Getting an element from a tuple: ", my_tuple[3])
print("Index of the first occurrence of 5 is: ", my_tuple.index(5))
```

5.0.3 Appending to a tuple

[29]:
```
# my_tuple.append(4) # this wont work as tuples are immutable
my_tuple = my_tuple + (1, "added element")
my_tuple
```

[29]:
```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 1, 'added element', 1, 'added element')
```

5.0.4 Difference between tuple and list

[52]:
```
# lists are mutable
my_list = [1, 2, 3, 4, 5]
my_tuple = tuple(my_list)

# Deleting is allowed even tho you cant remove single elements.
del(my_list)
# tuples are not mutable
# my_tuple[0] = 2 # not allowed

# trying to sort a list
```

```python
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print("Before sorting: ", my_list)
print("After sorting: ", sorted(my_list))

try:
    my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 'hi']
    print("Before sorting: ", my_list)
    print("After sorting: ", sorted(my_list))
except TypeError as e:
    print("You need all elements of the same type to sort a list. ")

# same goes with tuples
```

```
Before sorting:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
After sorting:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Before sorting:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 'hi']
You need all elements of the same type to sort a list.
```

5.0.5 Some Properties of tuple

[10]:
```python
weird_tuple = (4)
print(type(weird_tuple))

# You need comma to make it a tuple
weird_tuple = (4,)
print(type(weird_tuple))

weird_tuple = ((4,),)
print(type(weird_tuple))
```

```
<class 'int'>
<class 'tuple'>
<class 'tuple'>
```

5.0.6 You can change mutable elements inside a tuple

[19]:
```python
weird_tuple = (4, 5, 6, 7, [1, 2, 3], "hello")
weird_tuple[4][1] = 10 # allowed
try:
    weird_tuple[4] = 10 # not allowed
except TypeError as e:
    print("Thats not allowed")
try:
    weird_tuple[5][1] = '3' # not allowed
except TypeError as e:
    print("thats also not possible as strings are still immutable")
```

```
Thats not allowed
thats also not possible as strings are still immutable
```

5.0.7 Printing everything about the tuple

```
[23]: for i in weird_tuple:
          print(i)
```

```
4
5
6
7
[1, 10, 3]
hello
```

5.0.8 Some functions on tuples

```
[80]: print(my_tuple.count(4))
      print(my_tuple.index(4))
```

```
1
3
```

5.1 Dictionaries

```
[96]: example_dictionary = {
          'a' : 1,
          'b' : 2,
          'c' : 3,
          'd' : 4,
          'e' : 5,
      }

      example_dictionary
```

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

5.1.1 Different ways to create a dictionary

```
[97]: alphabets = ['a', 'b', 'c', 'd', 'e']
      alphabet_dictionary = {_ + 1: i for _, i in enumerate(alphabets)}
      alphabet_dictionary
```

```
[97]: {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

5.1.2 Accessing elements from a dictionary

[98]:
```
# usually
print(alphabet_dictionary[1])

# or the get method
print(alphabet_dictionary.get(5))
```

```
a
e
```

5.1.3 Adding items to a dictionary

[99]:
```
alphabet_dictionary[6] = 'f'
alphabet_dictionary
```

[99]:
```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
```

5.1.4 Some functions of a dictionary

[100]:
```
alphabet_dictionary[6] = 'f'

# deleting an element
print(alphabet_dictionary.pop(6))
print(alphabet_dictionary)

print(alphabet_dictionary.popitem())
print(alphabet_dictionary)

keys = alphabet_dictionary.keys()
print(keys)
print(type(keys))

values = alphabet_dictionary.values()
print(values)
print(type(values))

items = alphabet_dictionary.items()
print(items)
print(type(items))
```

```
f
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
(5, 'e')
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
dict_keys([1, 2, 3, 4])
<class 'dict_keys'>
dict_values(['a', 'b', 'c', 'd'])
```

```
<class 'dict_values'>
dict_items([(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')])
<class 'dict_items'>
```

5.2 Assignment things

[126]:
```
my_dict = {1: "Ramesh", 2: "Suresh", 3: "Rajesh", 4: "Rakesh", 5: "Mahesh", 6:␣
 ↪"Ganesh"}
for key, val in my_dict.items():
    print(key, val)
```

```
1 Ramesh
2 Suresh
3 Rajesh
4 Rakesh
5 Mahesh
6 Ganesh
```

[127]:
```
# to remove items from a dictionary
my_dict.popitem()
print(my_dict)
my_dict[7] = 'Paresh'
my_dict
```

```
{1: 'Ramesh', 2: 'Suresh', 3: 'Rajesh', 4: 'Rakesh', 5: 'Mahesh'}
```

[127]:
```
{1: 'Ramesh', 2: 'Suresh', 3: 'Rajesh', 4: 'Rakesh', 5: 'Mahesh', 7: 'Paresh'}
```

5.3 2. dict from list

[128]:
```
list1 = ['name', 'panel', 'rollno']
list2 = ['Ramesh', 'A', 1]
my_dict = {i: j for i, j in zip(list1, list2)}
my_dict
```

[128]:
```
{'name': 'Ramesh', 'panel': 'A', 'rollno': 1}
```

5.4 3. sort elements

[129]:
```
sorted_dictionary = {i: my_dict.get(i) for i in sorted(my_dict)}
sorted_dictionary
```

[129]:
```
{'name': 'Ramesh', 'panel': 'A', 'rollno': 1}
```

5.5 4. Make 1 list of keys and other list of values

[132]:
```
print(my_dict)
list_keys = list(my_dict.keys())
list_values = list(my_dict.values())

print(list_keys)
print(list_values)
```

```
{'name': 'Ramesh', 'panel': 'A', 'rollno': 1}
['name', 'panel', 'rollno']
['Ramesh', 'A', 1]
```

5.6 5. Find the mean value of the dictionary

[133]:
```
marks = {
    'marks1' : 90,
    'marks2' : 80,
    'marks3' : 80,
    'marks4' : 80,
    'marks5' : 80,
}
mean_value = sum(marks.values()) / len(marks)
print(mean_value)
```

```
82.0
```

5.7 6. Perform the following operations on the dictionary

[135]:
```
my_dict = {'name': ['Yash', 'Neal', 'Dev'], 'Roll': [1, 12, 3], 'Marks': [90,␣
  ↪80, 70]}
print("Name is: ", my_dict['name'][2])
print("Roll is: ", my_dict['Roll'][1])
print("Highest Marks are: ", max(my_dict['Marks']))
```

```
Name is:  Dev
Roll is:  12
Highest Marks are:  90
```

6 Conclusion

The List data structure in python was studied in detail. The different operations that can be performed on a list were also studied. The code was written and tested to check the output. The differences between lists and tuples were also studied.

7 FAQ

1. **What will be the output of the following code snippet?**

```
a=[1,2,3,4,5,6,7,8,9]
print(a[::2])

> [1, 3, 5, 7, 9]
```

2. **What will be the output of the following code snippet?**

```
l = [1, 2, 3]
init_tuple = ('Python',) * (l.__len__() - l[::-1][0])
print(init_tuple)

> ()
```

3. **State the difference between list and dictionary.**

   (a) List is mutable whereas dictionary is immutable.

   (b) List is ordered whereas dictionary is unordered.

   (c) List is indexed whereas dictionary is not indexed.

   (d) List is iterable whereas dictionary is not iterable.

   (e) List is a data structure that stores a sequence of values whereas dictionary is a data structure that stores a sequence of key-value pairs.