Python Dictionary

Introduction.

Here, we will learn about dictionaries, the operators, and methods used on dictionaries. It is nearly inconceivable to work with python program or scripts, without lists and dictionaries. Python dictionaries can be changed easily at runtime.

Dictionary is like a list but in a general form. It can think like this; a dictionary is a mapping between a set of indexes or keys with a set of values, where each key maps to a value. A combination of a key and its value is called a key-value pair or item.

Suppose we take an example and build a dictionary that maps words from English to French. So the keys and their values will be all strings.

In the Python dictionary, each key is separated by a colon (:) from its values. Commas separate all the items, and the whole dictionary is enclosed within '{' and '}'. In the dictionary, all the keys should have to be unique with data type as strings, tuples, or numbers, and the values can be of any of these three types.

Objectives

Objectives by the end of this topic you should be able to:

- Define a Dictionary In Python
- Accessing Dictionary Values
- Creating a New Dictionary in Python
- Dictionary as Set of Counter
- Updating Dictionary in Python
- Deleting Elements From Dictionary

Learning activities

Learning Activity 9.1: Reading

Read further on dictionaries built-in methods.

Learning Activity 9.2: Journal

Write a Python script to merge two Python dictionaries.

Learning Activity 9.3: Discussion

Write a Python script to print a dictionary where the keys are numbers between 1 and 15 (both included) and the values are square of keys

Assessment

Topic resources

1. The Python Tutorial¶. (n.d.). Retrieved from https://docs.python.org/3/tutorial/index.html
2. Mueller, J. P. (n.d.). *Beginning Programming with Python For Dummies*. S.l.: For Dummies.

3. (n.d.). Python 3.7.4 documentation. Retrieved from https://docs.python.org/3
4. (n.d.). Git Handbook. Retrieved from https://guides.github.com/introduction/git-handbook/
5. Shaw, Z. (2017). *Learn Python 3 the hard way: a very simple introduction to the terrifyingly beautiful world of computers and code*. Boston: Addison-Wesley.
6. Bader, D. (2018). *Python tricks: the book*. Vancouver, BC: Dan Bader.
7. Downey, A. B. (2015). *Think Python*. Sebastopol: OReilly.
8. Ramalho, L. (2016). *Fluent Python:*Beijing: OReilly.

URL Links

https://thomas-cokelaer.info/tutorials/python/dicts.html
https://data-flair.training/blogs/python-dictionary/
https://www.tutorialspoint.com/python/python_dictionary.htm
https://www.w3schools.in/python-tutorial/dictionaries/

Python Dictionary - NOTES

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.
Note – Keys in a dictionary doesn't allows Polymorphism.

Creating a Dictionary

In Python, a Dictionary can be created by placing sequence of elements within curly **{}** braces, separated by 'comma'. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its **Key:value**. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be *immutable*.

Note – Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

```python
# Creating a Dictionary
# with Integer Keys
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print("\nDictionary with the use of Integer Keys: ")
print(Dict)

# Creating a Dictionary
# with Mixed keys
Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}
print("\nDictionary with the use of Mixed Keys: ")
print(Dict)
```

Output:

```
Dictionary with the use of Integer Keys:

{1: 'Geeks', 2: 'For', 3: 'Geeks'}



Dictionary with the use of Mixed Keys:

{1: [1, 2, 3, 4], 'Name': 'Geeks'}
```

Dictionary can also be created by the built-in function **dict().** An empty dictionary can be created by just placing to curly braces**{}**.

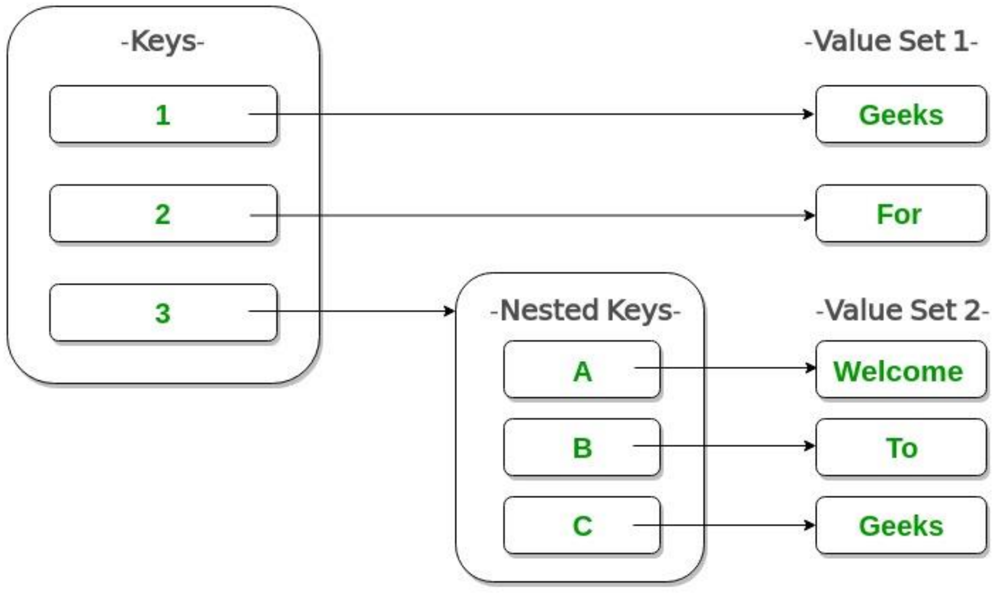
```python
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)

# Creating a Dictionary
# with dict() method
Dict = dict({1: 'Geeks', 2: 'For', 3:'Geeks'})
print("\nDictionary with the use of dict(): ")
print(Dict)

# Creating a Dictionary
# with each item as a Pair
Dict = dict([(1, 'Geeks'), (2, 'For')])
print("\nDictionary with each item as a pair: ")
print(Dict)
```

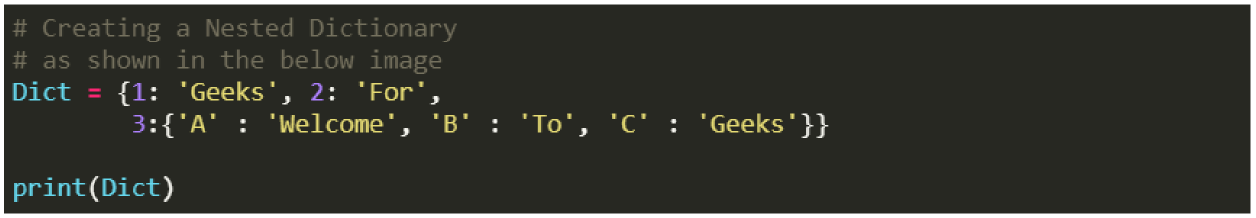
Output:

```
Empty Dictionary:

{}
```

```
Dictionary with the use of dict():
{1: 'Geeks', 2: 'For', 3: 'Geeks'}


Dictionary with each item as a pair:
{1: 'Geeks', 2: 'For'}
```

Nested Dictionary:





```python
# Creating a Nested Dictionary
# as shown in the below image
Dict = {1: 'Geeks', 2: 'For',
        3:{'A' : 'Welcome', 'B' : 'To', 'C' : 'Geeks'}}

print(Dict)
```

Output:

```
{1: 'Geeks', 2: 'For', 3: {'A': 'Welcome', 'B': 'To', 'C': 'Geeks'}}
```

Adding elements to a Dictionary

In Python Dictionary, Addition of elements can be done in multiple ways. One value at a time can be added to a Dictionary by defining value along with the key e.g. Dict[Key] = ‘Value’. Updating an existing value in a Dictionary can be done by using the built-in **update()** method. Nested key values can also be added to an existing Dictionary.

Note- While adding a value, if the key value already exists, the value gets updated otherwise a new Key with the value is added to the Dictionary.

```python
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)

# Adding elements one at a time
Dict[0] = 'Geeks'
Dict[2] = 'For'
Dict[3] = 1
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Adding set of values
# to a single Key
Dict['Value_set'] = 2, 3, 4
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Updating existing Key's Value
Dict[2] = 'Welcome'
print("\nUpdated key value: ")
print(Dict)

# Adding Nested Key value to Dictionary
Dict[5] = {'Nested' :{'1' : 'Life', '2' : 'Geeks'}}
print("\nAdding a Nested Key: ")
print(Dict)
```

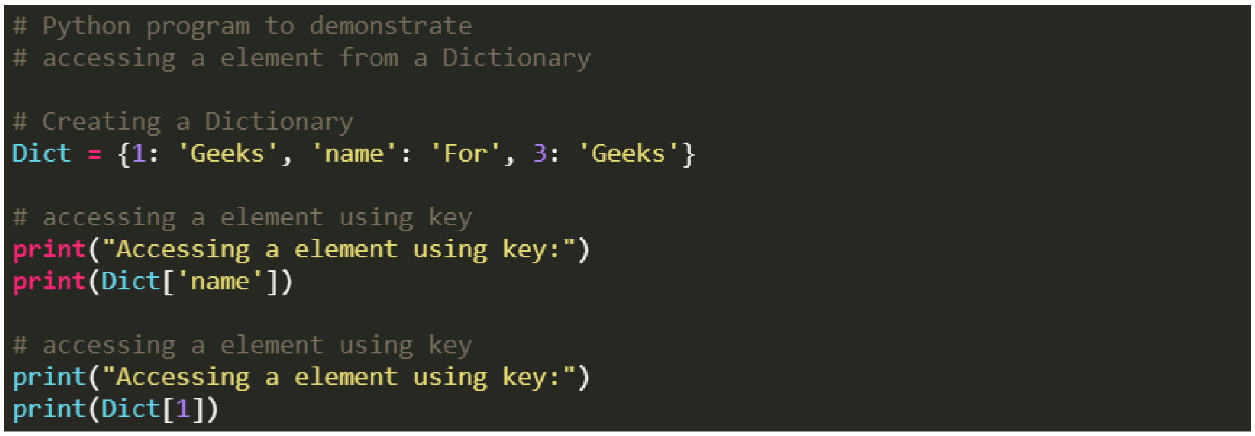
Output:

```
Empty Dictionary:

{}



Dictionary after adding 3 elements:

{0: 'Geeks', 2: 'For', 3: 1}



Dictionary after adding 3 elements:

{0: 'Geeks', 2: 'For', 3: 1, 'Value_set': (2, 3, 4)}
```

```
Updated key value:

{0: 'Geeks', 2: 'Welcome', 3: 1, 'Value_set': (2, 3, 4)}


Adding a Nested Key:

{0: 'Geeks', 2: 'Welcome', 3: 1, 5: {'Nested': {'1': 'Life', '2': 'Geeks'}}, 'Value_set': (2, 3, 4)}
```

Accessing elements from a Dictionary

In order to access the items of a dictionary refer to its key name.Key can be used inside square brackets.



```python
# Python program to demonstrate
# accessing a element from a Dictionary

# Creating a Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# accessing a element using key
print("Accessing a element using key:")
print(Dict['name'])

# accessing a element using key
print("Accessing a element using key:")
print(Dict[1])
```

Output:

```
Accessing a element using key:

For


Accessing a element using key:

Geeks
```

There is also a method called get() that will also help in accessing the element from a dictionary.

```
# Creating a Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# accessing a element using get()
# method
print("Accessing a element using get:")
print(Dict.get(3))
```

Output:

```
Accessing a element using get:

Geeks
```

Accessing element of a nested dictionary

In order to access the value of any key in nested dictionary, use indexing [] syntax.

```
# Creating a Dictionary
Dict = {'Dict1': {1: 'Geeks'},
        'Dict2': {'Name': 'For'}}

# Accessing element using key
print(Dict['Dict1'])
print(Dict['Dict1'][1])
print(Dict['Dict2']['Name'])
```

Output:

```
{1: 'Geeks'}

Geeks

For
```

Removing Elements from Dictionary

Using del keyword

In Python Dictionary, deletion of keys can be done by using the **del** keyword. Using del keyword, specific values from a dictionary as well as whole dictionary can be deleted. Items in a Nested dictionary can also be deleted by using del keyword and providing specific nested key and particular key to be deleted from that nested Dictionary.
Note- del Dict will delete the entire dictionary and hence printing it after deletion will raise an Error.

```python
# Initial Dictionary
Dict = { 5 : 'Welcome', 6 : 'To', 7 : 'Geeks',
        'A' : {1 : 'Geeks', 2 : 'For', 3 : 'Geeks'},
        'B' : {1 : 'Geeks', 2 : 'Life'}}
print("Initial Dictionary: ")
print(Dict)

# Deleting a Key value
del Dict[6]
print("\nDeleting a specific key: ")
print(Dict)

# Deleting a Key from
# Nested Dictionary
del Dict['A'][2]
print("\nDeleting a key from Nested Dictionary: ")
print(Dict)
```

Output:

```
Initial Dictionary:

{'A': {1: 'Geeks', 2: 'For', 3: 'Geeks'}, 'B': {1: 'Geeks', 2: 'Life'}, 5: 'Welcome', 6: 'To', 7: 'Geeks'}



Deleting a specific key:

{'A': {1: 'Geeks', 2: 'For', 3: 'Geeks'}, 'B': {1: 'Geeks', 2: 'Life'}, 5: 'Welcome', 7: 'Geeks'}

Deleting a key from Nested Dictionary:

{'A': {1: 'Geeks', 3: 'Geeks'}, 'B': {1: 'Geeks', 2: 'Life'}, 5: 'Welcome', 7: 'Geeks'}
```

Using pop() method

Pop() method is used to return and delete the value of the key specified.

```python
# Creating a Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# Deleting a key
# using pop() method
pop_ele = Dict.pop(1)
print('\nDictionary after deletion: ' + str(Dict))
print('Value associated to poped key is: ' + str(pop_ele))
```

Output:

```
Dictionary after deletion: {3: 'Geeks', 'name': 'For'}

Value associated to poped key is: Geeks
```

Using popitem() method

The popitem() returns and removes an arbitrary element (key, value) pair from the dictionary.

```
# Creating Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# Deleting an arbitrary key
# using popitem() function
pop_ele = Dict.popitem()
print("\nDictionary after deletion: " + str(Dict))
print("The arbitrary pair returned is: " + str(pop_ele))
```

Output:

```
Dictionary after deletion: {3: 'Geeks', 'name': 'For'}

The arbitrary pair returned is: (1, 'Geeks')
```

Using clear() method

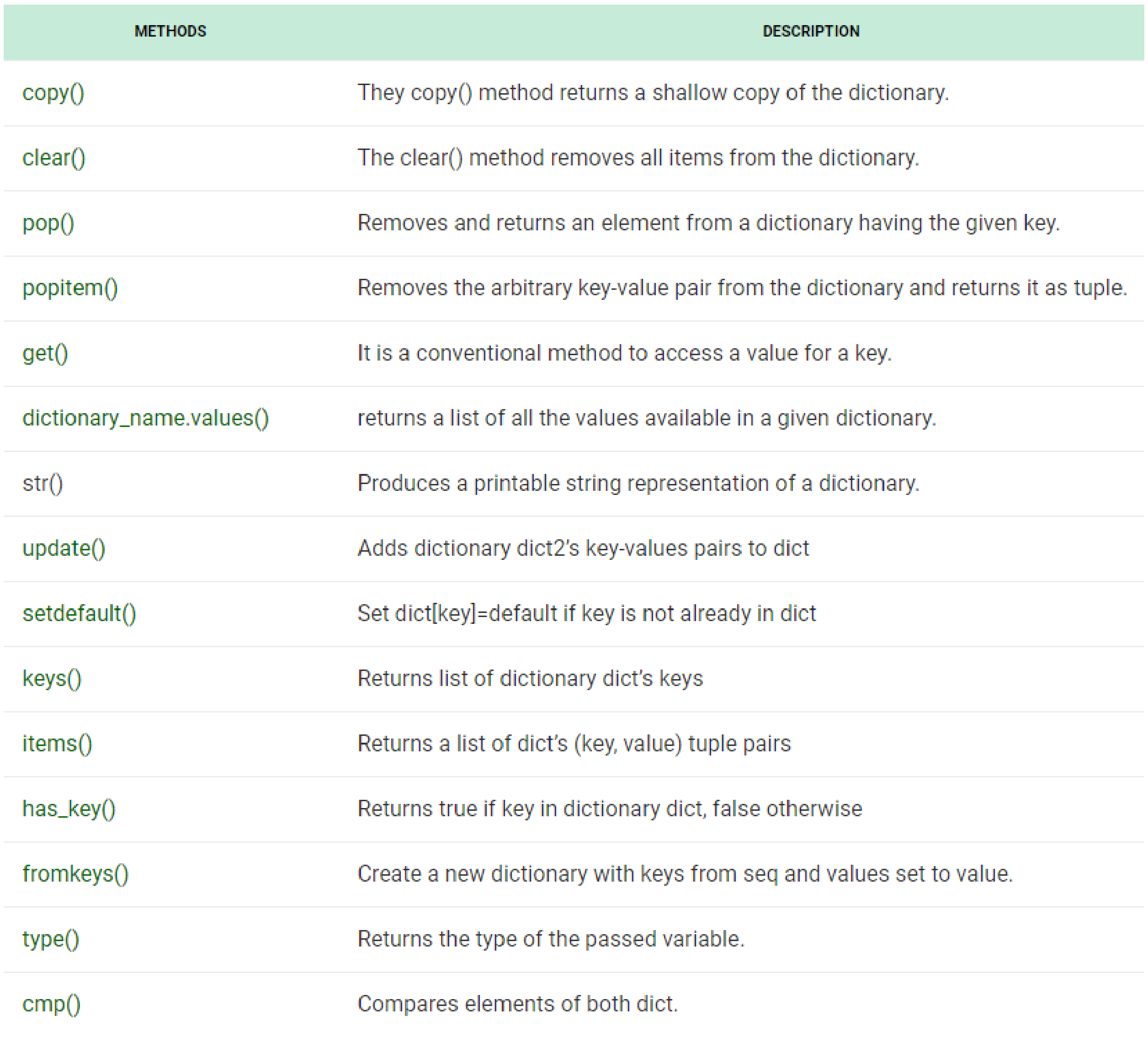
All the items from a dictionary can be deleted at once by using **clear()** method.

```
# Creating a Dictionary
Dict = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}


# Deleting entire Dictionary
Dict.clear()
print("\nDeleting Entire Dictionary: ")
print(Dict)
```

Output:

```
Deleting Entire Dictionary:

{}
```

Dictionary Methods



METHODS	DESCRIPTION
copy()	They copy() method returns a shallow copy of the dictionary.
clear()	The clear() method removes all items from the dictionary.
pop()	Removes and returns an element from a dictionary having the given key.
popitem()	Removes the arbitrary key-value pair from the dictionary and returns it as tuple.
get()	It is a conventional method to access a value for a key.
dictionary_name.values()	returns a list of all the values available in a given dictionary.
str()	Produces a printable string representation of a dictionary.
update()	Adds dictionary dict2's key-values pairs to dict
setdefault()	Set dict[key]=default if key is not already in dict
keys()	Returns list of dictionary dict's keys
items()	Returns a list of dict's (key, value) tuple pairs
has_key()	Returns true if key in dictionary dict, false otherwise
fromkeys()	Create a new dictionary with keys from seq and values set to value.
type()	Returns the type of the passed variable.
cmp()	Compares elements of both dict.

Revision questions

1. Write a Python script to sort (ascending and descending) a dictionary by value. Go to the editor. Click me to see the sample solution

2. Write a Python script to add a key to a dictionary. Go to the editor

Sample Dictionary : {0: 10, 1: 20}
Expected Result : {0: 10, 1: 20, 2: 30} Click me to see the sample solution

3. Write a Python script to concatenate following dictionaries to create a new one. Go to the editor

Sample Dictionary :
dic1={1:10, 2:20}

dic2={3:30, 4:40}
dic3={5:50,6:60}
Expected Result : {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

Click me to see the sample solution

4. Write a Python script to check whether a given key already exists in a dictionary. Go to the editor. Click me to see the sample solution

5. Write a Python program to iterate over dictionaries using for loops. Go to the editor. Click me to see the sample solution