DICTIONARY

A dictionary is a collection which is unordered, changeable and indexed.

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.

1. Why do we use Dictionary?

Let us say we have data in the tabular form as shown below:

Key	*Value*
Name	**Age**
Sam	20
Ram	21
Gauri	30
Maitree	34
John	29

So, with the help of dictionary we can represent the above table as:

Data = {‘Sam’:20, ‘Ram’:21, ‘Gauri’:30, ‘Maitree’:34, ‘John’:29}

2. Creating a Dictionary

2.1 Simple 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.*

```
#empty dictionary
My_dict = { }

#dictionary with integer keys
My_dict = { 1: 'apple', 2: 'ball'}

#dictionary with mixed keys
My_dict = {'name' : 'John', 1: [2, 4, 3]}
```

2.2 Nested Dictionary

- In python, a nested dictionary is a dictionary inside a dictionary.
- It's a collection of dictionaries into one single dictionary.

  Example:

  nested_dict = {'dictA': {'key_1': 'value_1'}, 'dictB': {'key_2': 'value_2'}}

- Here, the ***nested_dict*** is a nested dictionary with the dictionary ***dictA*** and ***dictB***. They are two dictionary each having one *key* and *value*.

- It's not compulsory to have both the values as dictionary in nested dictionary.
- We can also write as,

  Dict = {1: 'Welcome', 2: 'To', 3:{'A' : 'Summer', 'B' : 'Training', 'C' : 'Program'}}

Representation:

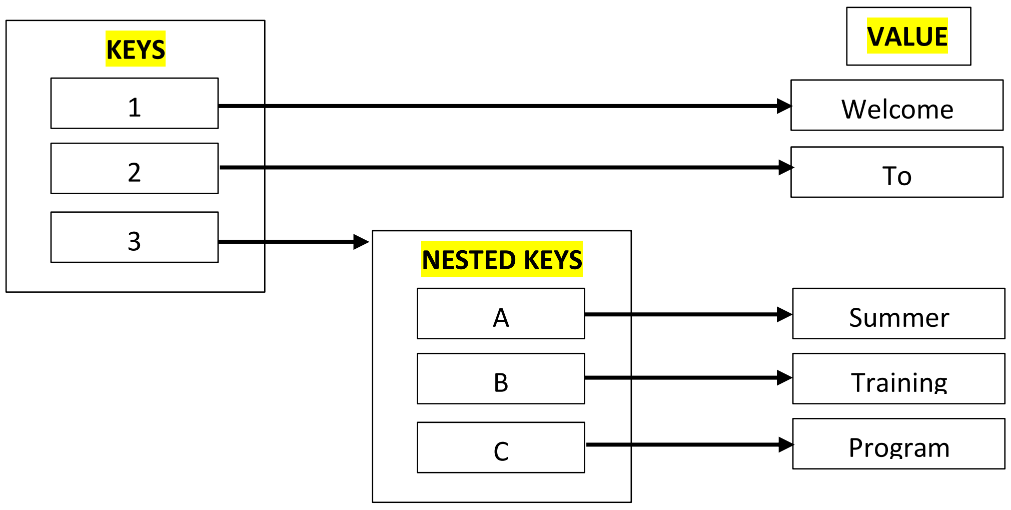

Fig. Schematic of Nested

3. Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value.

3.1 Simple Dictionary

Example:

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print("dict['Name']: ", dict['Name'])
print("dict['Age']: ", dict['Age'])
```

Output:

```
dict['Name']:  Zara
dict['Age']:  7
```

There is also a method called **get()** that will give you the same result:

Example:

```
my_dict = {'name':'Jack', 'age': 26}
print(my_dict['name'])
print(my_dict.get('name'))
```

Output:

```
Jack
Jack
```

3.2 Nested Dictionary

Example:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

print(people[1]['name'])
print(people[1]['age'])
print(people[1]['sex'])
```

Output:

```
John
27
Male
```

**Try to access the data in the 2nd dictionary.*

4. Change or Add Elements in a Dictionary

Values in Dictionary are mutable.

We can add new items or change the value of existing items using assignment operator.

If the key is already present, value gets updated, else a new *key : value* pair is added to the dictionary.

4.1 Simple Dictionary

Example:

```
my_dict = {'name':'Jack', 'age': 26}
print("Original Dict : ", my_dict)

my_dict['age'] = 27
print("Changed : ", my_dict)

my_dict['address'] = 'Downtown'
print("Added : ",my_dict)
```

Output:

```
Original Dict :  {'name': 'Jack', 'age': 26}
Changed :  {'name': 'Jack', 'age': 27}
Added :  {'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```

4.2 Nested dictionary

Example:

```
people = { 1: {'name': 'John', 'age': 27, 'sex': 'Male'},  2: {'name': 'Marie', 'age': 22,
                 'sex': 'Female'} }
people[3] = {}
people[3]['name'] = 'Luna'
people[3]['age'] = '24'
people[3]['sex'] = 'Female'
people[3]['married'] = 'No'
print(people)
```

Output:

```
{1: {'name': 'John', 'age': 27, 'sex': 'Male'},
 2: {'name': 'Marie', 'age': 22, 'sex': 'Female'},
 3: {'name': 'Luna', 'age': '24', 'sex': 'Female', 'married'
: 'No'}}
```

5. Adding another dictionary to the nested dictionary

Example:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},
          3: {'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'}}
people[4] = {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}
print(people)
```

Output:

```
{1: {'name': 'John', 'age': '27', 'sex': 'Male'},
 2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},
 3: {'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'},
 4: {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}}
```

6. Delete or Remove Elements from the dictionary

We can remove the particular item in the dictionary by using the method ***pop()***. This method removes as item with the provided key and returns the value.

The method, ***popitem()*** can be used to remove and return the arbitrary item ***(key, value)*** form the dictionary. All the items can be removes at once using the ***clear()*** method.

We can also use the ***del*** keyword to remove the individual items or the entire dictionary itself.

6.1 Simple Dictionary

Example:

```
squares = {1:1, 2:4, 3:9, 4:16, 5:25}
print("Original Dict : ", squares)
print(squares.pop(4))
print("Pop : ", squares)
print(squares.popitem())
print("Popitem : ",squares)
del squares[5]
print("Delete : ",squares)
squares.clear()
print("Clear : ",squares)

del squares      #deletes the dictionary itself

# Throws Error
# print(squares)
```

Output:

```
Original Dict : {1:1, 2:4, 3:9, 4:16, 5:25}
Pop : {1: 1, 2: 4, 3: 9, 5: 25}
Popitem : {2: 4, 3: 9, 5: 25}
Delete : {2: 4, 3: 9}
Clear : {}
Delete 2 :
```

```
KeyError                                  Traceback (most re
cent call last)
<ipython-input-27-c3dd74e5ca5a> in <module>
      5 print(squares.popitem())
      6 print("Popitem : ",squares)
----> 7 del squares[5]
      8 print("Delete : ",squares)
      9 squares.clear()

KeyError: 5
```

6.2 Nested dictionary

Example:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},
          3: {'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'},
          4: {'name': 'Peter', 'age': '29', 'sex': 'Male', 'married': 'Yes'}}

del people[3], people[4]

print(people)
```

Output:

```
{1: {'name': 'John', 'age': '27', 'sex': 'Male'},
 2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
```

7. Copy A Dictionary

You cannot copy a dictionary simply by typing ***dict2 = dict1***, *because: dict2 will only be a reference to dict1, and changes made in dict1 will automatically also be made in dict2.*

There are ways to make a copy, one way is to use the built-in Dictionary method ***copy()***.

Example:

```
thisdict= {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
        }
mydict = thisdict.copy()
print("Thisdict:",thisdict)
print("mydict :",mydict)
```

Output:

```
Thisdict: {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
mydict : {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

8. Iterating through dictionary

There are multiple ways to iterate over a dictionary in Python.

- Iterate through all keys
- Iterate through all values
- Iterate through all key, value pairs

8.1 Iterate through all keys

The order of states in the below code will change every time because the dictionary doesn't store keys in a particular order.

Example:

```
statesAndCapitals = { 'Gujarat' : 'Gandhinagar',
                      'Maharashtra' : 'Mumbai',
                       'Rajasthan' : 'Jaipur',
                       'Bihar' : 'Patna' }
print('List Of given states:\n')
# Iterating over keys
for key in statesAndCapitals:
    print(key)
```

Output:

```
List Of given states:
Gujarat
Maharashtra
Rajasthan
Bihar
```

Why not the ordered sequence of keys?

8.2 Iterate through all key, value pairs

Example:

```
statesAndCapitals = { 'Gujarat' : 'Gandhinagar',
                      'Maharashtra' : 'Mumbai',
                      'Rajasthan' : 'Jaipur',
                      'Bihar' : 'Patna'
                     }
```

```
print('List Of given states and their capitals:\n')
# Iterating over values
for state, capital in statesAndCapitals.items():
        print(state, ":", capital)
```

Output:

```
List Of given states and their capitals:

Gujarat : Gandhinagar
Maharashtra : Mumbai
Rajasthan : Jaipur
Bihar : Patna
```

8.3 Iterating through Nested Dictionary

Example:

```
people = {1: {'Name': 'John', 'Age': '27', 'Sex': 'Male'},
          2: {'Name': 'Marie', 'Age': '22', 'Sex': 'Female'}}

for p_id, p_info in people.items():
  print("\nPerson ID:", p_id)
  for key in p_info:
    print(key + ':', p_info[key])
```

Output:

```
Person ID: 1
Name: John
Age: 27
Sex: Male

Person ID: 2
Name: Marie
Age: 22
Sex: Female
```

9. Properties of Dictionary Keys

Dictionary value has no restrictions.

They can be any arbitrary python object, either standard objects or user-defined objects.

However, same is not true for the keys.

There are two important points to remember about dictionary keys-

a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins.

Example:

```
dict1 = {"name": "Meena", "Age":5, "name": "Radha"}
print(dict1["name"])
```

Output:

```
Radha
```

b) Keys must be immutable, which means you can use strings, numbers or tuples as dictionary keys but something like [‘key’] is not allowed.

Example:

```
dict1 = {["name"]: "Meena", "Age":5}
print(dict1["name"])
```

When the above code is executed, it gives an error

Output:

```
TypeError                                 Traceback (most
recent call last)
<ipython-input-11-4b8af70cacc5> in <module>
----> 1 dict1 = {["name"]: "Meena", "Age":5}
      2 print(dict1["name"])

TypeError: unhashable type: 'list'
```

10. Python Dictionary Methods

Try all the below commands with dictionary

METHODS	DESCRIPTION
clear()	*The clear() method removes all items from the dictionary.*
copy()	*They copy() method returns a shallow copy of the dictionary.*
fromkeys()	*Create a new dictionary with keys from seq and values set to value.*
get()	*It is a conventional method to access a value for a key.*
items()	*Returns a list of dict's (key, value) tuple pairs*
keys()	*Returns list of dictionary dict's keys*
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.*
setdefault()	*Set dict[key]=default if key is not already in dict*
update()	*Adds dictionary dict2's key-values pairs to dict*
dictionary_name.values()	*returns a list of all the values available in a given dictionary.*

Enhance your learning by watching the video links provided below:

Video Link: https://www.youtube.com/watch?v=rZjhId0VkuY

For more detail (optional): https://www.youtube.com/watch?v=gbMFcxx6jKk&t=1952s

Now that you are clear with the concepts of dictionary, you are ready to solve the Jupyter notebook provided to you.

References:

1. https://www.programiz.com/python-programming/dictionary
2. https://www.tutorialspoint.com/python/python_dictionary.htm
3. https://www.w3schools.com/python/python_dictionaries.asp
4. https://www.geeksforgeeks.org/python-dictionary/
5. https://www.programiz.com/python-programming/nested-dictionary
6. https://www.geeksforgeeks.org/iterate-over-a-dictionary-in-python/