LECTURE NINE

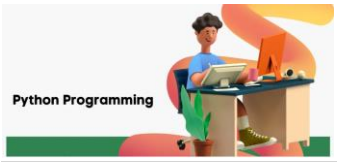
Dictionary

- Python dictionary is an unordered collection of items.
- Creating a dictionary is as simple as placing items inside curly braces {} separated by commas. An item has a key and a corresponding value that is expressed as a pair (key: value).
- Values in dictionary can be of any data type and can repeat
- keys in dictionary must be of immutable type and must be unique.

List Vs Set Vs Dictionary Vs Tuple

Lists	Sets	Dictionaries	Tuples
List = [10, 12, 15]	Set = {1, 23, 34} Print(set) -> {1, 23,24} Set = {1, 1} print(set) -> {1}	Dict = {"Ram": 26, "mary": 24}	Words = ("spam", "egss") Or Words = "spam", "eggs"
Access: print(list[0])	Print(set). Set elements can't be indexed.	print(dict["ram"])	Print(words[0])
Can contains duplicate elements	Can't contain duplicate elements. Faster compared to Lists	Can't contain duplicate keys, but can contain duplicate values	Can contains duplicate elements. Faster compared to Lists
List[0] = 100	set.add(7)	Dict["Ram"] = 27	Words[0] = "care" ->TypeError
Mutable	Mutable	Mutable	Immutable - Values can't be changed once assigned
List = []	Set = set()	Dict = {}	Words = ()
Slicing can be done print(list[1:2]) -> [12]	Slicing: Not done.	Slicing: Not done	Slicing can also be done on tuples
Usage: Use lists if you have a collection of data that doesn't need random access. Use lists when you need a simple, iterable collection that is modified frequently.	Usage: - Membership testing and the elimination of duplicate entries. - when you need uniqueness for the elements.	Usage: - When you need a logical association b/w key:value pair. - when you need fast lookup for your data, based on a custom key. - when your data is being constantly modified.	Usage: Use tuples when your data cannot change. A tuple is used in combination with a dictionary, for example, a tuple might represent a key, because its immutable.

6/25/2016 Rajkumar Rampalli - Python 15



- indexing is dictionary uses keys. Keys can be used either inside square brackets [] or with the get() method. If we use the square brackets [], 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.

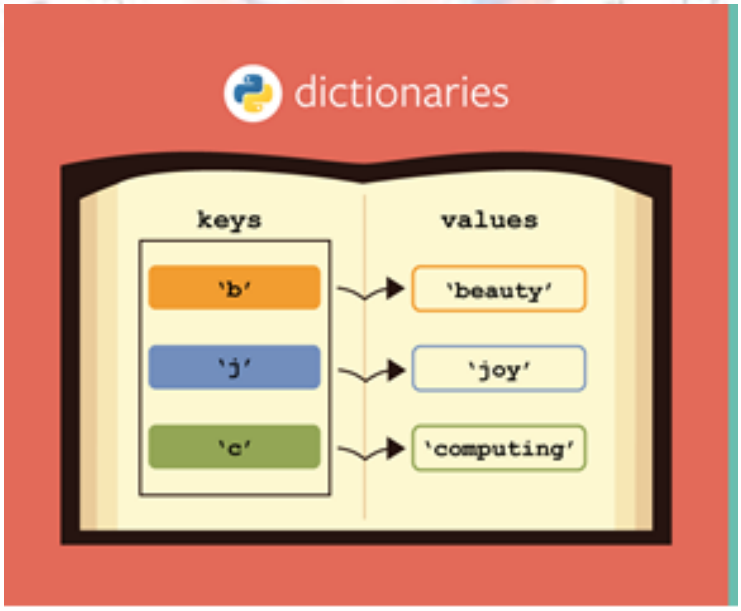
Example: Create dictionary, Indexing

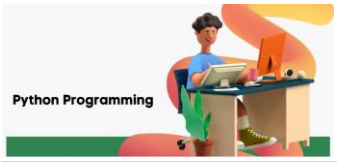
```
# empty dictionary
my_dict = {}

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

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

#Indxing
print(my_dict.get('name'))
print(my_dict['name'])
```





Operations and Functions on Dictionary

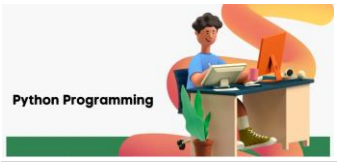
- python len function: it returns the number of items (length) in an object. use the **len**()to get the length of the given dictionary.
- Changing and Adding Dictionary elements, we can add new items or change the value of existing items using an **assignment operat**or.
- Removing elements from Dictionary, we can remove a particular item in a dictionary by using the **pop()** method. This method removes an item with the provided key and returns the value. We can also use the **del** keyword to remove individual items or the entire dictionary itself

Example: len, Changing, Adding and Removing

```
f = {'name':'john','age':20 }
print(len(f))
# Changing and adding Dictionary Elements
my_dict = {'name': 'Jack', 'age': 26}
# update value
my_dict['age'] = 27
print(my_dict)
# add item
my_dict['address'] = 'Iraq'
print(my_dict)
# Removing elements from a dictionary
print(my_dict.pop('age'))
del my_dict['name']
print(my_dict)
```

Output

```
2
{'name': 'Jack', 'age': 27}
{'name': 'Jack', 'age': 27, 'address': 'Iraq'}
27
{'address': 'Iraq'}
```



Common Python Dictionary Methods

There are numerous methods available with the dictionary object, some of the commonly used methods are:

Method	Description
clear()	Removes all items from the dictionary.
items()	Returns a view object that displays a list of dictionary's (key, value) tuple pairs.
keys()	Returns a view object that displays a list of all the keys in the dictionary
values()	Returns a view object that displays a list of all the values in the dictionary.

Example: Common Python Dictionary Methods

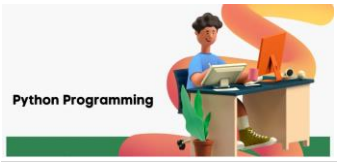
```
sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }

print(sales. items() )
print(sales. keys ())
print(sales.values())

for item in sales.items():
    print(item)
```

Output

```
dict_items([('apple', 2), ('orange', 3), ('grapes', 4)])
dict_keys(['apple', 'orange', 'grapes'])
dict_values([2, 3, 4])
('apple', 2)
('orange', 3)
('grapes', 4)
```



Example (1): Write Python Program to Merge Two Dictionaries

```
dict_1 = {1: 'a', 2: 'b'}
dict_2 = {2: 'c', 4: 'd'}
print({**dict_1, **dict_2})
print(dict_1|dict_2)        #  It works for python 3.9 and above
```

Example (2): Write Python Program to Check if a Key is Already Present in a Dictionary

```
my_dict = {1: 'a', 2: 'b', 3: 'c'}
x=int(input("key"))
if x in my_dict:
    print("present")
 else:
     print("not present")
```

Example(3): Write Python Program to Convert Two Lists Into a Dictionary

```
index = [1, 2, 3]
languages = ['python', 'c', 'c++']

dictionary = dict(zip(index, languages))
print(dictionary)

#zip() function takes iterables, aggregates them in a tuple, and returns it
```

Example (4): Write Python Program to read a Dictionary from user

```
n = int(input("enter a n value:"))
d = {}

for i in range(n):
    keys = input() # here i have taken keys as strings
    values = int(input()) # here i have taken values as integers
    d[keys] = values
print(d)
```