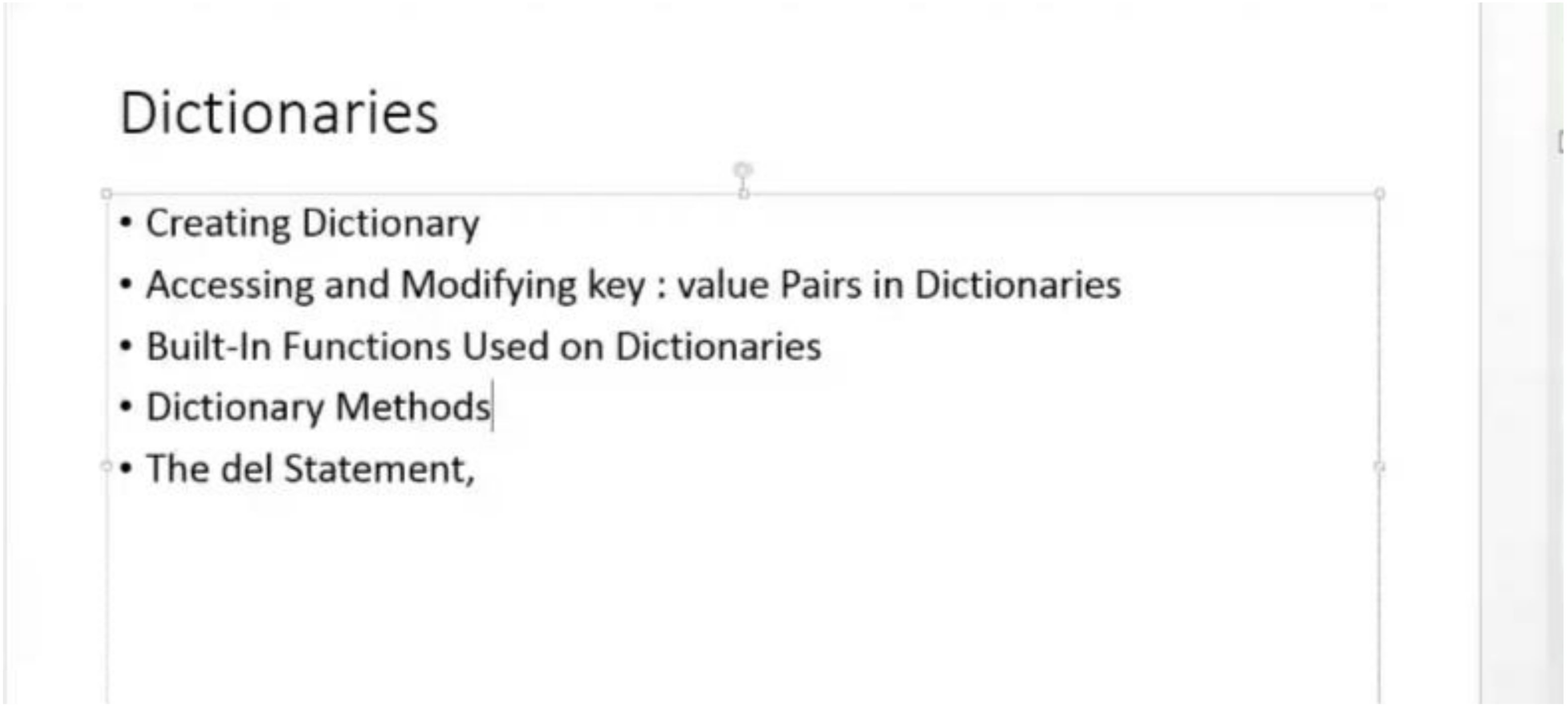
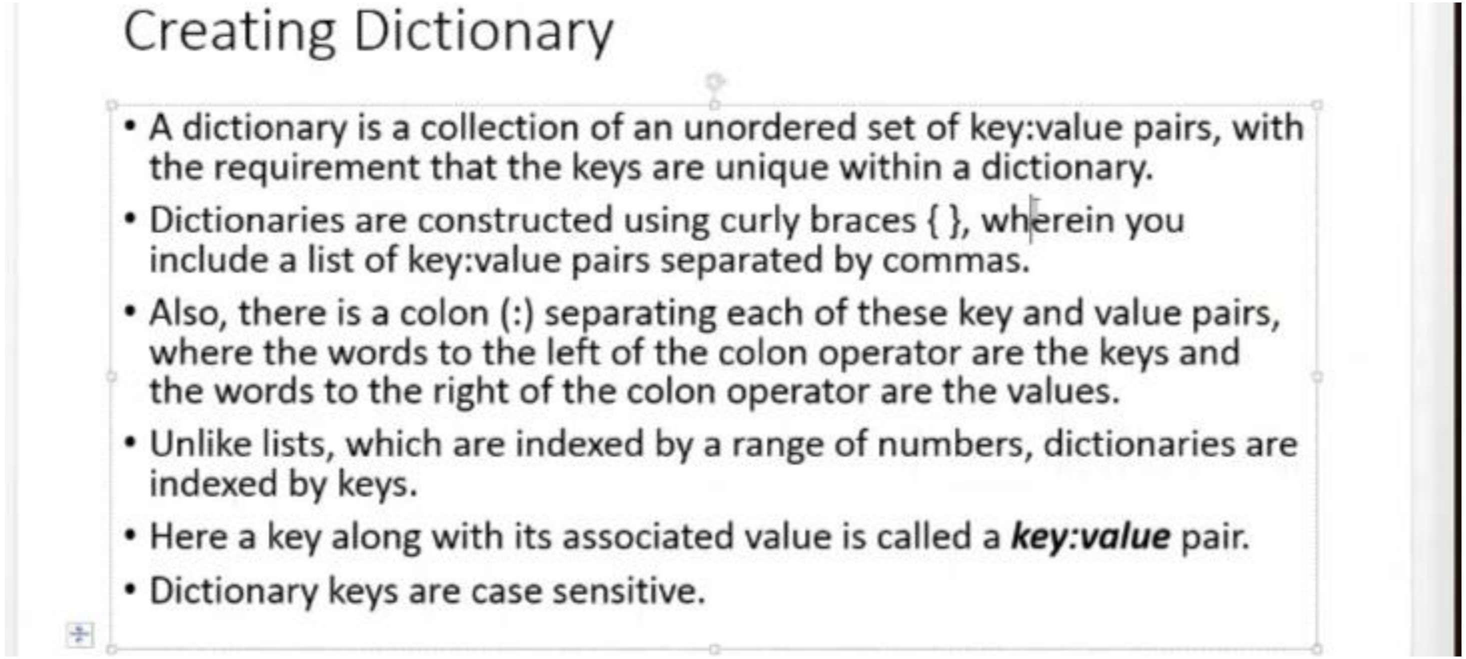


Dictionaries

- Creating Dictionary
- Accessing and Modifying key : value Pairs in Dictionaries
- Built-In Functions Used on Dictionaries
- Dictionary Methods
- The del Statement,



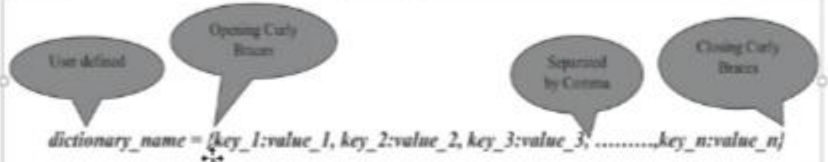
Creating Dictionary

- A dictionary is a collection of an unordered set of key:value pairs, with the requirement that the keys are unique within a dictionary.
- Dictionaries are constructed using curly braces { }, wherein you include a list of key:value pairs separated by commas.
- Also, there is a colon (:) separating each of these key and value pairs, where the words to the left of the colon operator are the keys and the words to the right of the colon operator are the values.
- Unlike lists, which are indexed by a range of numbers, dictionaries are indexed by keys.
- Here a key along with its associated value is called a ***key:value*** pair.
- Dictionary keys are case sensitive.

- Dictionary keys are immutable type and can be either a string or a number.
- Since lists can be modified in place using index assignments, slice assignments, or methods like *append()* and *extend(),* you cannot use lists as keys. Duplicate keys are not allowed in the dictionary.

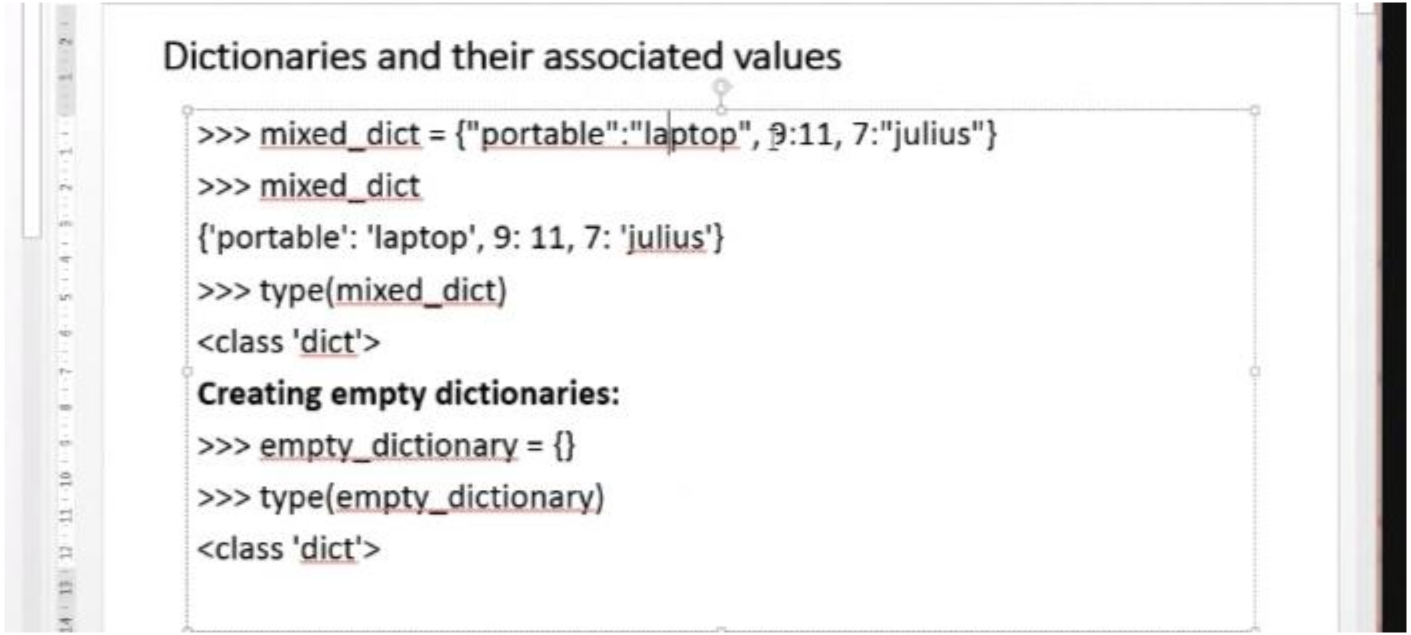
Creating dictionaries

- The syntax for creating a dictionary is



dictionary_name = {key_1:value_1, key_2:value_2, key_3:value_3, ...,key_n:value_n}

```
fish = {"g": "goldfish", "s":"shark", "n": "needlefish", "b":"barramundi","m":"mackerel"}
```

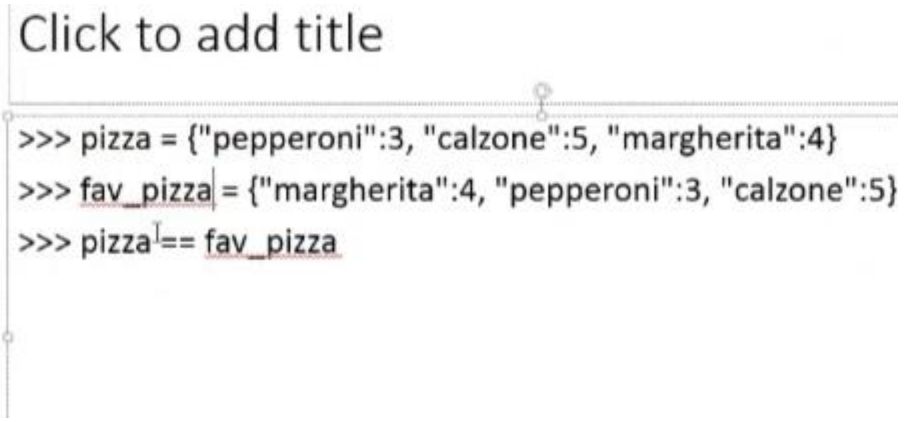


Dictionaries and their associated values

```
>>> mixed_dict = {"portable":"laptop", 9:11, 7:"julius"}
>>> mixed_dict
{'portable': 'laptop', 9: 11, 7: 'julius'}
>>> type(mixed_dict)
<class 'dict'>
```

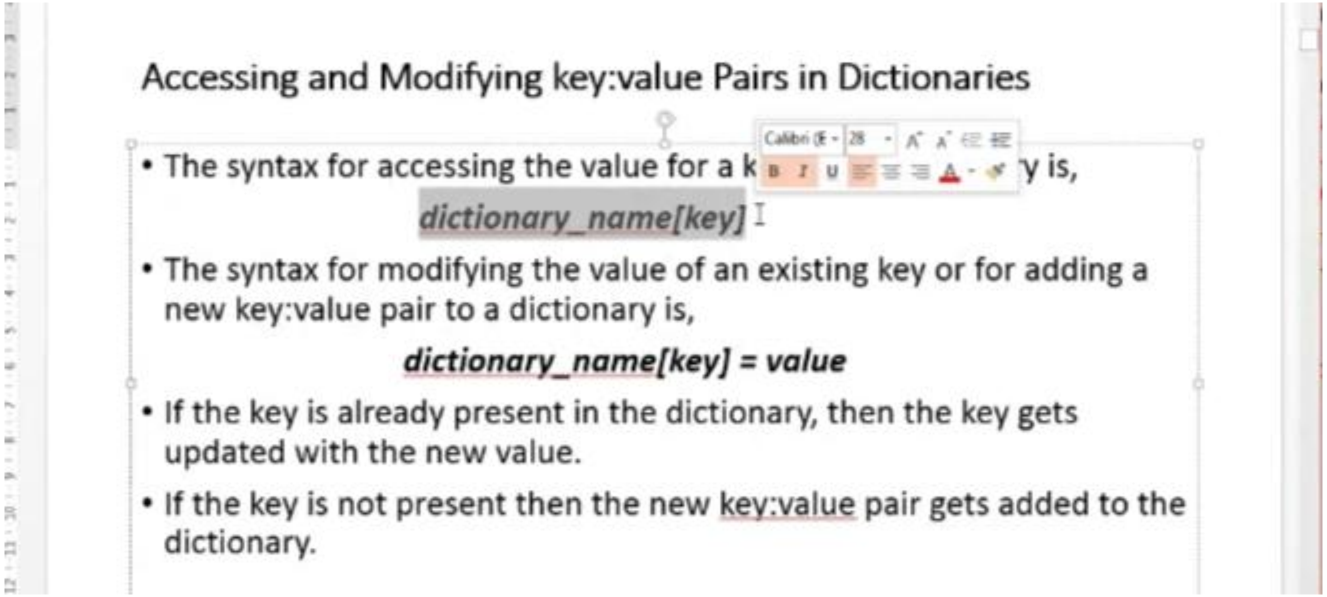
Creating empty dictionaries:

```
>>> empty_dictionary = {}
>>> type(empty_dictionary)
<class 'dict'>
```



Click to add title

```
>>> pizza = {"pepperoni":3, "calzone":5, "margherita":4}
>>> fav_pizza = {"margherita":4, "pepperoni":3, "calzone":5}
>>> pizza== fav_pizza
```



Accessing and Modifying key:value Pairs in Dictionaries

- The syntax for accessing the value for a k y is,

  dictionary_name[key]

- The syntax for modifying the value of an existing key or for adding a new key:value pair to a dictionary is,

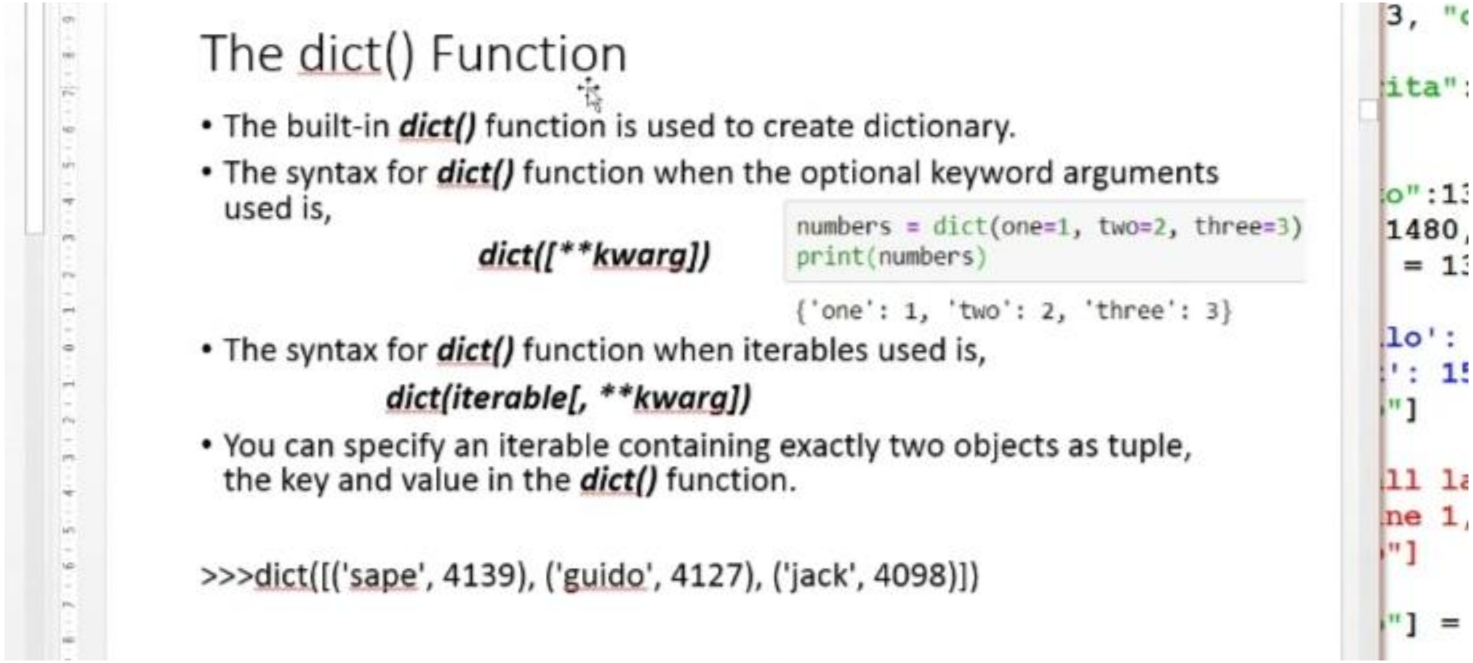
  dictionary_name[key] = value

- If the key is already present in the dictionary, then the key gets updated with the new value.
- If the key is not present then the new key:value pair gets added to the dictionary.

Check for the presence of a key in the dictionary

- You can check for the presence of a key in the dictionary using in and not in membership operators.
- It returns either a Boolean True or False value.

```
>>> clothes = {"rainy":"raincoats", "summer":"tees",
"winter":"sweaters"}
>>> "spring" in clothes
>>> "spring" not in clothes
```



The dict() Function

- The built-in ***dict()*** function is used to create dictionary.
- The syntax for ***dict()*** function when the optional keyword arguments used is,

  dict([**kwarg])

```
numbers = dict(one=1, two=2, three=3)
print(numbers)
```

```
{'one': 1, 'two': 2, 'three': 3}
```

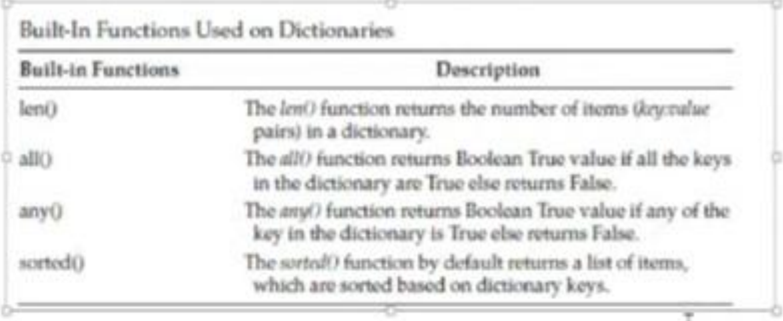
- The syntax for ***dict()*** function when iterables used is,

  dict(iterable[, **kwarg])

- You can specify an iterable containing exactly two objects as tuple, the key and value in the ***dict()*** function.

```
>>>dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
```

Built-in functions used on dictionaries

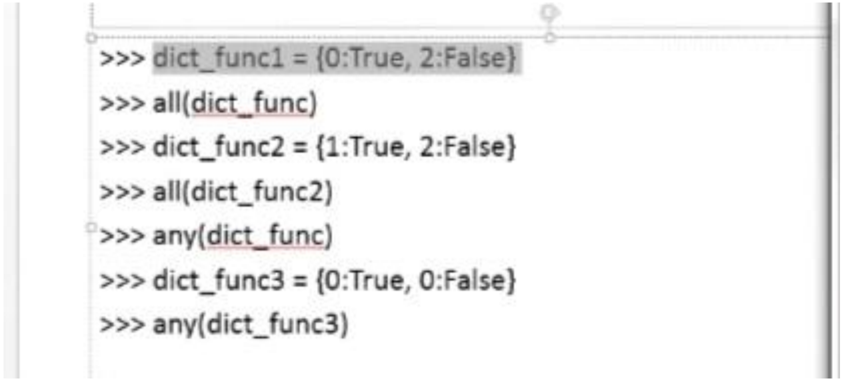


Built-In Functions Used on Dictionaries

Built-in Functions	Description
len()	The *len()* function returns the number of items (*key:value* pairs) in a dictionary.
all()	The *all()* function returns Boolean True value if all the keys in the dictionary are True else returns False.
any()	The *any()* function returns Boolean True value if any of the key in the dictionary is True else returns False.
sorted()	The *sorted()* function by default returns a list of items, which are sorted based on dictionary keys.

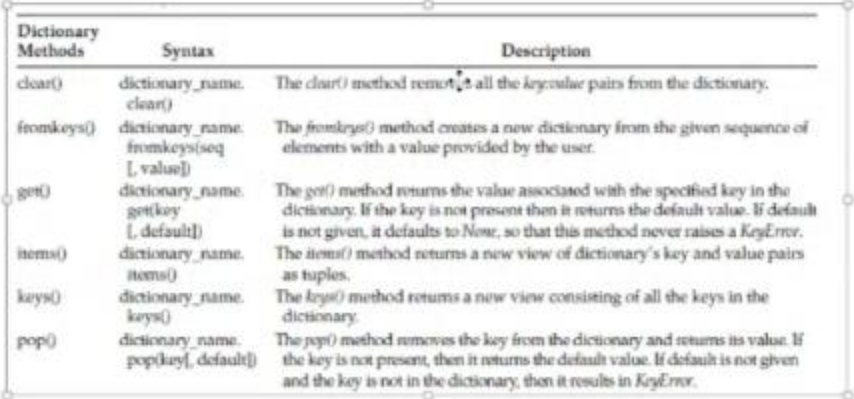
- In Python, any non-zero integer value is True, and zero is interpreted as False.
- The sorted() function returns the sorted list of keys by default in ascending order without modifying the original ***key:value*** pairs

```
{"r
v_pi
= {
ttic
["gi
, 'c
0, '
["le
rec
1#64
["le
ardc
```

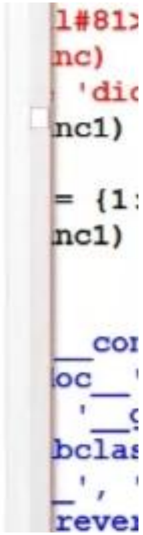


```
>>> dict_func1 = {0:True, 2:False}
>>> all(dict_func)
>>> dict_func2 = {1:True, 2:False}
>>> all(dict_func2)
>>> any(dict_func)
>>> dict_func3 = {0:True, 0:False}
>>> any(dict_func3)
```

Dictionary Methods- >>> dir(dict)



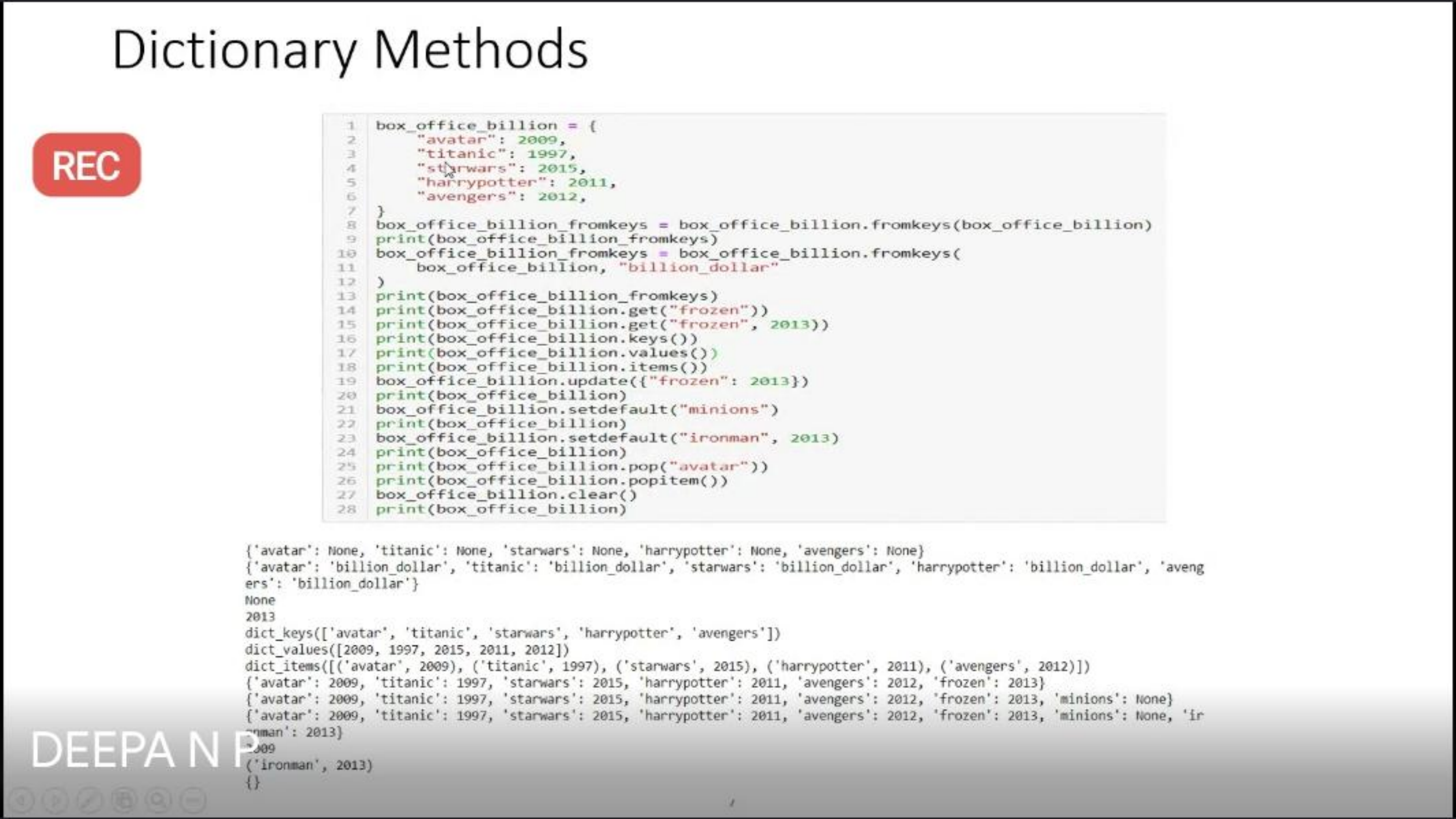
Dictionary Methods	Syntax	Description
clear()	dictionary_name. clear()	The *clear()* method removes all the *key:value* pairs from the dictionary.
fromkeys()	dictionary_name. fromkeys(seq [, value])	The *fromkeys()* method creates a new dictionary from the given sequence of elements with a value provided by the user.
get()	dictionary_name. get(key [, default])	The *get()* method returns the value associated with the specified key in the dictionary. If the key is not present then it returns the default value. If default is not given, it defaults to *None*, so that this method never raises a *KeyError*.
items()	dictionary_name. items()	The *items()* method returns a new view of dictionary's key and value pairs as tuples.
keys()	dictionary_name. keys()	The *keys()* method returns a new view consisting of all the keys in the dictionary.
pop()	dictionary_name. pop(key[, default])	The *pop()* method removes the key from the dictionary and returns its value. If the key is not present, then it returns the default value. If default is not given and the key is not in the dictionary, then it results in *KeyError*.



```
1#81
nc)
 'di
nc1)

= {1
nc1)


 __co
oc__
 '__
bclas
_',
rever
```



12:52

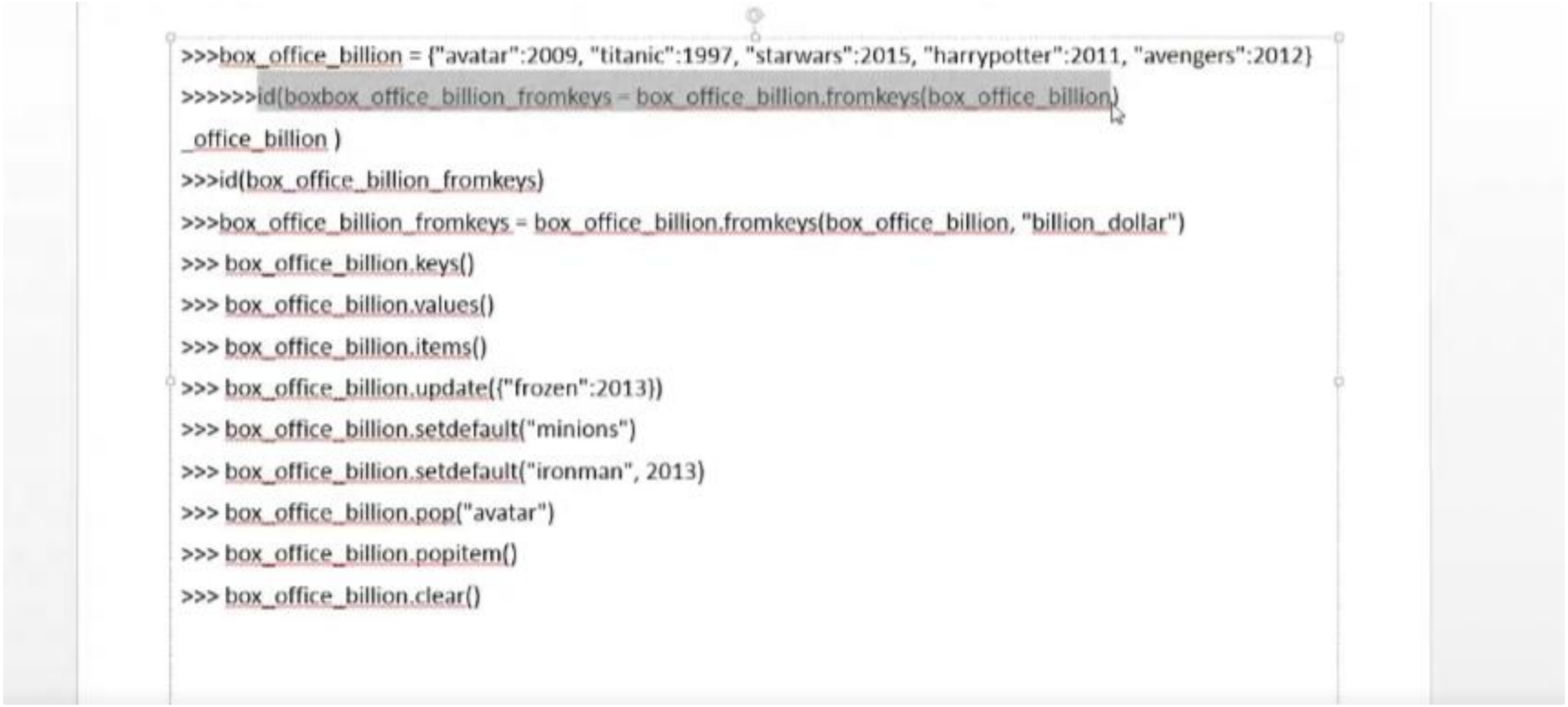
Dictionary Methods

REC

```python
box_office_billion = {
    "avatar": 2009,
    "titanic": 1997,
    "starwars": 2015,
    "harrypotter": 2011,
    "avengers": 2012,
}
box_office_billion_fromkeys = box_office_billion.fromkeys(box_office_billion)
print(box_office_billion_fromkeys)
box_office_billion_fromkeys = box_office_billion.fromkeys(
    box_office_billion, "billion_dollar"
)
print(box_office_billion_fromkeys)
print(box_office_billion.get("frozen"))
print(box_office_billion.get("frozen", 2013))
print(box_office_billion.keys())
print(box_office_billion.values())
print(box_office_billion.items())
box_office_billion.update({"frozen": 2013})
print(box_office_billion)
box_office_billion.setdefault("minions")
print(box_office_billion)
box_office_billion.setdefault("ironman", 2013)
print(box_office_billion)
print(box_office_billion.pop("avatar"))
print(box_office_billion.popitem())
box_office_billion.clear()
print(box_office_billion)
```

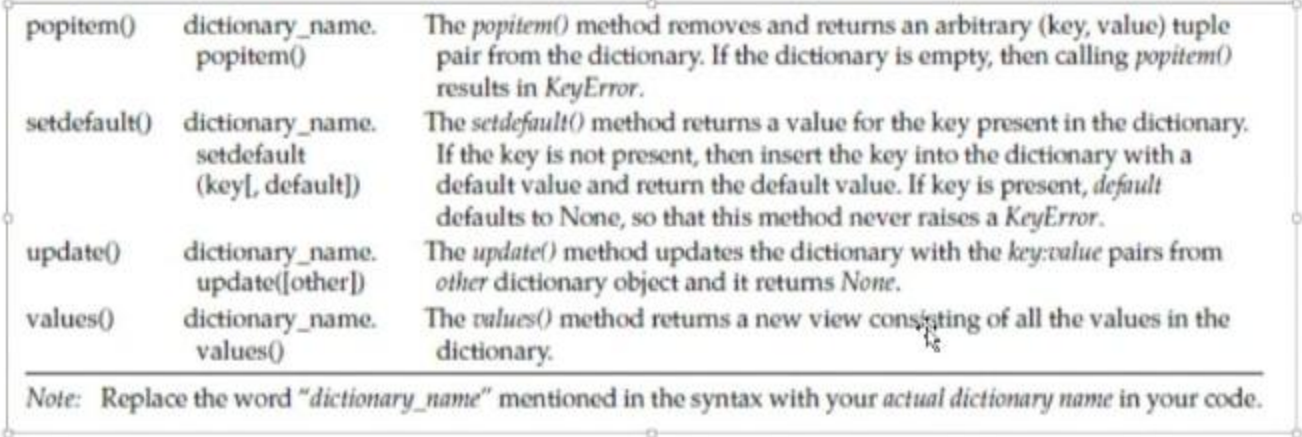
```
{'avatar': None, 'titanic': None, 'starwars': None, 'harrypotter': None, 'avengers': None}
{'avatar': 'billion_dollar', 'titanic': 'billion_dollar', 'starwars': 'billion_dollar', 'harrypotter': 'billion_dollar', 'aveng
ers': 'billion_dollar'}
None
2013
dict_keys(['avatar', 'titanic', 'starwars', 'harrypotter', 'avengers'])
dict_values([2009, 1997, 2015, 2011, 2012])
dict_items([('avatar', 2009), ('titanic', 1997), ('starwars', 2015), ('harrypotter', 2011), ('avengers', 2012)])
{'avatar': 2009, 'titanic': 1997, 'starwars': 2015, 'harrypotter': 2011, 'avengers': 2012, 'frozen': 2013}
{'avatar': 2009, 'titanic': 1997, 'starwars': 2015, 'harrypotter': 2011, 'avengers': 2012, 'frozen': 2013, 'minions': None}
{'avatar': 2009, 'titanic': 1997, 'starwars': 2015, 'harrypotter': 2011, 'avengers': 2012, 'frozen': 2013, 'minions': None, 'ir
onman': 2013}
2009
('ironman', 2013)
{}
```

DEEPA N P



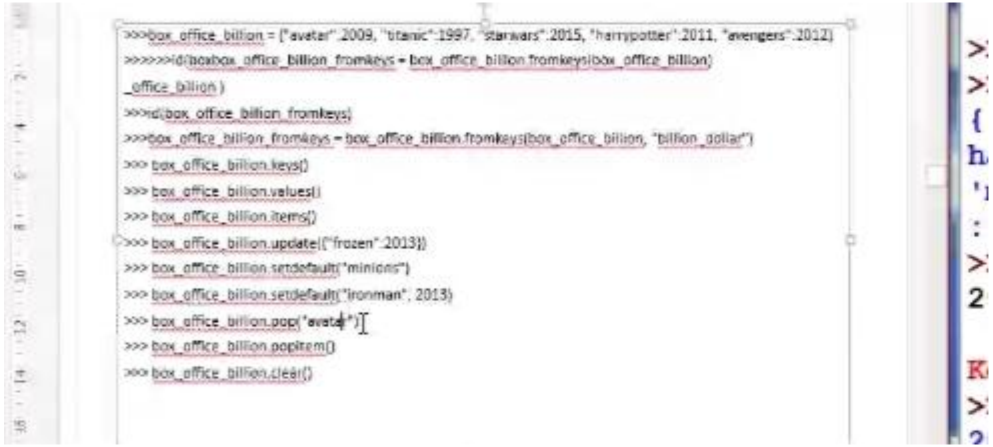
```
>>>box_office_billion = {"avatar":2009, "titanic":1997, "starwars":2015, "harrypotter":2011, "avengers":2012}
>>>>>>id(boxbox_office_billion_fromkeys = box_office_billion.fromkeys(box_office_billion)
_office_billion )
>>>id(box_office_billion_fromkeys)
>>>box_office_billion_fromkeys = box_office_billion.fromkeys(box_office_billion, "billion_dollar")
>>> box_office_billion.keys()
>>> box_office_billion.values()
>>> box_office_billion.items()
>>> box_office_billion.update({"frozen":2013})
>>> box_office_billion.setdefault("minions")
>>> box_office_billion.setdefault("ironman", 2013)
>>> box_office_billion.pop("avatar")
>>> box_office_billion.popitem()
>>> box_office_billion.clear()
```

Dictionary Methods- >>> dir(dict)



popitem()	dictionary_name. popitem()	The *popitem()* method removes and returns an arbitrary (key, value) tuple pair from the dictionary. If the dictionary is empty, then calling *popitem()* results in *KeyError*.
setdefault()	dictionary_name. setdefault (key[, default])	The *setdefault()* method returns a value for the key present in the dictionary. If the key is not present, then insert the key into the dictionary with a default value and return the default value. If key is present, *default* defaults to None, so that this method never raises a *KeyError*.
update()	dictionary_name. update([other])	The *update()* method updates the dictionary with the *key:value* pairs from *other* dictionary object and it returns *None*.
values()	dictionary_name. values()	The *values()* method returns a new view consisting of all the values in the dictionary.

Note: Replace the word "*dictionary_name*" mentioned in the syntax with your *actual dictionary name* in your code.



```
>>>box_office_billion = {"avatar":2009, "titanic":1997, "starwars":2015, "harrypotter":2011, "avengers":2012}
>>>>>>id(boxbox_office_billion_fromkeys = box_office_billion.fromkeys(box_office_billion)
_office_billion )
>>>id(box_office_billion_fromkeys)
>>>box_office_billion_fromkeys = box_office_billion.fromkeys(box_office_billion, "billion_dollar")
>>> box_office_billion.keys()
>>> box_office_billion.values()
>>> box_office_billion.items()
>>> box_office_billion.update({"frozen":2013})
>>> box_office_billion.setdefault("minions")
>>> box_office_billion.setdefault("ironman", 2013)
>>> box_office_billion.pop("avatar")
>>> box_office_billion.popitem()
>>> box_office_billion.clear()
```