Dictionary

* Dictionary, Range
  ↓
  * another Data Structure in Python
  * It is used to represent structured data.
    ↓
    * excel
    *

* d = {'name': 'ABC', 'age': 23, 'sal': 10000.00}

  print (d['name'])

  output: ABC

* d = {'emp1': {'name': 'ABC', 'age': 23, 'add': 16-1-33/c},
  'emp2': {'name': 'ab', 'age': 22,
  'add': '16-1'}}

  print (d['emp1']['add'])

  output: 16-1-33/c

key — Value

dictonary = (key, value)

dictionary properties :-

* Is a collection of key value pair
* A key, value par is called item
* item is separated by comma
* key & value are separated by :
* key can not be duplicated whereas value can be duplicated
* key can not be mutable, it will be only immutable whereas values can be any type
* All the item's should be kept inside { } separated by comma.
* Dictionary is not a seq, so indexing & slicing is not allowed, concatenation & repetation is also not allowed
* Membership operator then it checks for the key and return "True" if the key is present in the dictionary
* identity operation is possible
* Dictionary is Mutable

Ex:- tp1 = ('name', 'age', 'sal')

d = {tp1 : ('ABC', 23, 1000.00)}

d[tp1] ↵

output : ('ABC', 23, 1000.0)

* Creation of dictionary
  * If you know the values

d = {'emp' : {'name' : 'Abc', 'age' : 30,}}

print(d) ↵

* Using eval function

d1 = eval(input ('Enter a dictionary'))

print (d1) ↵

output : Enter a dictionary : {10:20, 'a':'abc}

* Using dict function:

dict function takes a seq of inner seq where inner seq will have pair of values out of that pair the first value will be key & second value is value.

[( ), ( ), ( )]

↘ sequence of sequence

Ex:-

* d = dict([(1,100), (2.9,2900), ('a', 'Python')])

print(d)

output: 1:100, 2.9:2900, 'a': 'Python'

Zip:

* ~~list~~ * zip([10, 20, 30.9], (20, 30.9, 10))

zip at 0.0819

* To see the list add the list before

list(zip([10, 20, 30.9], (20, 30.9, 10))

output: [(10, 20), (20, 30.9), (30.9, 10)]

* Accessing a value inside a dictionary :

* We use square bracket notation

```
print(d['emp1']['age'])
```

output :- 23

* Update a dictionary

* If key is not available

```
d['emp4'] = {'name': 'PQR', 'age': 29,
             'sal': 2800.00}
```

key

```
print(d)
```

output :

```
{'emp1': ..... 'emp2': ..... 'emp3': ...
..... 'emp4': 'PQR', '29', '2800.00'}
```

* If key is available

```
d['emp1'] = {'name': 'PQR', 'age': 29,
             'sal': 28000.00}
```

Key

Methods on dictionary

```
print(dir(dict))
```

* i) clear ✓
  ii) copy
  iii) fromkeys ✓
  iv) get ✓
  v) items
  vi) keys ✓
  vii) pop ✓
  viii) popitems ✓
  ix) setdefault ✓
  x) update ✓
  xi) values ✓

* get, setdefault, update :

```
print(d.get('emp1'))
```

If key is present

get :- Return the value for key if key is in the dictionary, else default

Ex :- print(d.get('emp1')) → key

output : {'name': 'abc', 'age': 23, 'address': 16-1-33/C}

* Get - if key is not present

```
print(d.get('emps'))
```

output: None

* parse a message when a key is not present

```
print(d.get('emps', 'This is not present'))
```

output: This is not present

Set default:

* Insert key with a value of default if key is not in the dictionary. Else returns value of the key

* print(d.setdefault('emp1', 'present in it'))

output: 'name': 'ABC', 'age': 23, 'sal': '2800'

If the ~~value~~ [key] is not present in "d" then it prints the key & value which is passed.

* print(d.setdefault('emps', 'present in it'))
print(d)

{'emp1': ------, 'emp2': ------,
'emp3': ---, 'emps': 'present in it'}

Update:

* It is similar as extend in list

print(d)
d1 = {11: 11000, 22: 22000, 33: 33000}

d.update(d1)
print(d1)
print(d)

output: {11: 11000, 22: 22000, 33: 33000}

* pop, popitem

pop :- which removes the value

d.pop('emps')
print(d)

output: 'emp1': --- 'emp2' -----

* If the key is not present & we are trying to delete it. Thus, it gonna give a "key Error".

Ex:- print (d.pop('emp5'))

Key Error : 'emp5'

* popitem

It returns & removes some key values.

print(d)

d.popitem()

* If we won't mention any key values thus it randomly removes the value.

* key

* To check the keys inside a dictionary

print (d.keys())

output : emp1, emp2, emp3

* Values

* To check the values inside a dictionary

print(d.values())

output : name, abc, age, 23 ...

* Fromkey : Creates a new dictionary with keys from iterable & values set to value.

```
d1 = d2.fromkeys ( [1,2,3,4] ,10 )
print (d1)
```

* clear :

To clear the dictionary & leaves empty dictionary behind.

```
d.clear ()
print (d)
```