**Dictionary in Python**

**What is dictionary in Python?**

Python dictionary is an **unordered collection of items**. While other compound data types have only value as an element, **a dictionary has a key: value pair**.

Dictionaries are optimized to retrieve values when the key is known.

**How to create a dictionary?**

Creating a dictionary is as simple as placing items inside curly braces {} separated by **comma**.

An item has a key and the corresponding value expressed as a **pair, key: value.**

While **values** can be of any data type and can **repeat**,

**keys** must be of **immutable** **type** ***(string, number or tuple with immutable elements) and must be unique.***

# 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]}

# using dict()

my\_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair

my\_dict = dict([(1,'apple'), (2,'ball')])

directions = {}

directions[1]="East"

directions[2]="West"

directions[3]="North"

directions[4]="South"

del directions

print (directions)

**Add elements in Dictionary using Input()**

#------------Method-1-----------

n = 3

d = dict(input().split() for \_ in range(n))

print (d)

#------------Method-2-----------

n=int(input("Enter the no. of item u want in Dictionary"))

pair = dict()

for i in range(0,n):

word = input().split()

key = word[0]

value = word[1]

pair[key]=value

print(pair)

#### Nested Dictionary

# Empty nested dictionary

Dict = { 'Dict1': { },

         'Dict2': { }}

print("Nested dictionary 1-")

print(Dict)

# Nested dictionary having same keys

Dict = { 'Dict1': {'name': 'Ali', 'age': '19'},

         'Dict2': {'name': 'Bob', 'age': '25'}}

print("\nNested dictionary 2-")

print(Dict)

**Output:**

Nested dictionary 1-

{'Dict1': {}, 'Dict2': {}}

Nested dictionary 2-

{'Dict1': {'name': 'Ali', 'age': '19'}, 'Dict2': {'name': 'Bob', 'age': '25'}}

**How to access elements from a dictionary?**

my\_dict = {'name':'Jack', 'age': 26}

# Output: Jack

print(my\_dict['name'])

**How to change or add elements in a dictionary?**

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.

my\_dict = {'name':'Jack', 'age': 26}

# update value

my\_dict['age'] = 27

#Output: {'age': 27, 'name': 'Jack'}

print(my\_dict)

# add item

my\_dict['address'] = 'Downtown'

# Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}

print(my\_dict)

**Dictionary Functions**

**Access Value of an Item in Dictionary**

my\_dict = {'name':'Jack', 'age': 26}

print(my\_dict['name'])

print(my\_dict.get('age'))

**Access all Values of Dictionary**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

print(my\_dict.values())

**Access all Keys of Dictionary**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

print(my\_dict.keys())

**Iterate through all keys in a dictionary**

statesAndCapitals = {

'Gujarat' : 'Gandhinagar',

'Maharashtra' : 'Mumbai',

'Rajasthan' : 'Jaipur',

'Bihar' : 'Patna'

}

print('List Of given states:\n')

for state in statesAndCapitals:

print(state)

**Iterate through all keys in a dictionary in a specific order**

from collections import OrderedDict

statesAndCapitals = OrderedDict([

('Gujarat', 'Gandhinagar'),

('Maharashtra', 'Mumbai'),

('Rajasthan', 'Jaipur'),

('Bihar', 'Patna')

])

print('List Of given states:\n')

for state in statesAndCapitals:

print(state)

**Iterate through all values:**

statesAndCapitals = {

'Gujarat' : 'Gandhinagar',

'Maharashtra' : 'Mumbai',

'Rajasthan' : 'Jaipur',

'Bihar' : 'Patna'

}

print('List Of given capitals:\n')

for capital in statesAndCapitals.values():

print(capital)

**Iterating Through a Dictionary**

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

for i in squares:

print(squares[i])

Output: 1 9 25 49 81

**Iterate through all key, value pairs:**

statesAndCapitals = {

'Gujarat' : 'Gandhinagar',

'Maharashtra' : 'Mumbai',

'Rajasthan' : 'Jaipur',

'Bihar' : 'Patna'

}

print('List Of given states and their capitals:\n')

for state, capital in statesAndCapitals.items():

print(state, ":", capital)

**Using update() method**

my\_dict = {'name':'Jack', 'age': 26}

my\_dict['age'] = 27

my\_dict.update(age = 28)

**adding dict1 (key3, key4 and key5) to dict**

dict = {'key1':'geeks', 'key2':'for'}

print("Current Dict is: ", dict)

dict1 = {'key3':'geeks', 'key4':'is', 'key5':'fabulous'}

dict.update(dict1)

**Remove a particular item**

squares = {1:1, 2:4, 3:9, 4:16, 5:25}

print(squares.pop(4))

# Output: 16

print(squares)

# Output: {1: 1, 2: 4, 3: 9, 5: 25}

**Remove an arbitrary item**

print(squares.popitem())

# Output: (5, 25)

**Remove an item**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

del my\_dict[3]

print(my\_dict)

**Remove all item**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

my\_dict.clear()

print(my\_dict)

**Remove entire dictionary**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

del my\_dict

print(my\_dict)

## Python Dictionary Comprehension

Dictionary comprehension is an elegant and concise way to create new dictionary from an iterable in Python.

Dictionary comprehension consists of an expression pair (key: value) followed by for statement inside curly braces {}.

squares = {x: x\*x for x in range(6)}

print(squares)

# Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

**Use:**

# Lists to represent keys and values

keys = ['a','b','c','d','e']

values = [1,2,3,4,5]

# but this line shows dict comprehension here

myDict = { k:v for (k,v) in zip(keys, values)}

# We can use below too

# myDict = dict(zip(keys, values))

print (myDict)

**To copy the dictionary**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

my\_dict1 = my\_dict

print(my\_dict1)

my\_dict1.clear()

print(my\_dict)

1. **Shallow Copy**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

my\_dict1 = my\_dict.copy()

print(my\_dict1)

my\_dict1.clear()

print(my\_dict)

**fromkeys() Method**

x = ('key1', 'key2', 'key3')  
y = 0  
  
thisdict = dict.fromkeys(x, y)  
print(thisdict)

**Output: {'key1': 0, 'key2': 0, 'key3': 0}**

x = ('key1', 'key2', 'key3')  
  
thisdict = dict.fromkeys(x)  
print(thisdict)

**Output: {'key1': None, 'key2': None, 'key3': None }**

**setdefault() Method**

car = {

"brand": "Ford",

"model": "Mustang",

"year": 1964,

"color": "Green"

}

car.setdefault("color", "White")

print(car)

### Dictionary Membership Test

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

print(1 in squares)

# Output: True

print(2 not in squares)

# Output: True

print(49 in squares)

# Output: False

**Any and All are two built ins provided in python used for successive And/Or.**

## all() Function: if Empty, return True

s = {0: 'False', 1: 'False'}

print(all(s))

s = {1: 'True', 2: 'True'}

print(all(s))

s = {1: 'True', False: 0}

print(all(s))

s = {}

print(all(s))

# 0 is False

# '0' is True

s = {'0': 'True'}

print(all(s))

False

True

False

True

True

## any() Function: if Empty, return False

d = {0: 'False'}

print(any(d))

d = {0: 'False', 1: 'True'}

print(any(d))

d = {0: 'False', False: 0}

print(any(d))

d = {}

print(any(d))

# 0 is False

# '0' is True

d = {'0': 'False'}

print(any(d))

**False**

**True**

**False**

**False**

**True**

## Python sorted() function

pyDict = {'e': 1, 'a': 2, 'u': 3, 'o': 4, 'i': 5}

print(sorted(pyDict)) #Ascending Order

print(sorted(pyDict, reverse=True)) #Descending Order

**To find the number of entries.**

my\_dict = {1: 'Geeks', 2: 'For', 3:'Geeks'}

z = len(my\_dict)

print(z)

**Get key from value in Dictionary**

**Method-1**

# creating a new dictionary

my\_dict ={"java":100, "python":112, "c":11}

# list out keys and values separately

key\_list = list(my\_dict.keys())

val\_list = list(my\_dict.values())

print(key\_list[val\_list.index(100)])

print(key\_list[val\_list.index(112)])

**Method-2**

# function to return key for any value

def get\_key(val):

    for key, value in my\_dict.items():

         if val == value:

             return key

    return "key doesn't exist"

my\_dict ={"java":100, "python":112, "c":11}

print(get\_key(100))

print(get\_key(11))

**Second largest value in a Python Dictionary**

dictionary = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

val = list(dictionary.values())

val.sort()

res = val[-2]

print(res)