Python

Kalpana S

12-07-2024

In [ ]:
```
import os
import keyword
import operator
from datetime import datetime
import sys
```

Keywords

keywords are the reserved words in python and cannot be used as an identifier.

In [ ]:
```
# List of all python Keywords
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'brea
k', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finall
y', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonloc
al', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yiel
d']
```

In [ ]:
```
len(keyword.kwlist) # python contains 35 key words
```

Out[4]: 35

```
# Identifiers

An identifier is a name given to entities like class,functions,Variables
etc. It helps to differentiate one from another.

-->Identifier can't start with a digit
--> can't use special symbols
--> keywords can't be used as identifiers

#Identifiers can be a combination of letters(a-z) in lowercase or
uppercase and also contain numbers ,underscore.
```

In [ ]:
```
var =  10

var_ = 12

var12 = 30

A = 10
```

In [ ]:
```
A
```

Out[7]:
```
10
```

Comments in Python

Comments can be used to explain the code for more readability. --> single line comment by using "#" symbol --> multiline comment by using triple quotes.

In [ ]:
```
## Single line comment

'''
Multiple
line
comment
'''
```

Statements

Instructions that a Python interpreter can execute.

In [ ]:
```
# single line statement
a = 10+20+30
a
```

Out[8]:
```
60
```

In [ ]:
```
# Single line statement
b = ['a','b','c','d']
b
```

Out[9]:
```
['a', 'b', 'c', 'd']
```

In [ ]:
```
# Multiline statement
s1 = 20+30\
+40+50+\
+20+30

s1
```

Out[10]:
```
190
```

Indentation

Indentation refers to the spaces at the begining of a code line.It is very important as python uses indentation to indicate a block of code.If the indentation is not correct we will endup with IndentationError.

In [ ]:
```
a = 20
if a == 20:
    print(' a is equal to 10')
```

```
 a is equal to 10
```

In [ ]:
```
a = 20
if a == 20:
print(' a is equal to 10')
```

```
  Cell In[12], line 3
    print(' a is equal to 10')
    ^
IndentationError: expected an indented block after 'if' statement on line
2
```

Docstrings

--> Docstrings provide a convenient way of associating documentation with functions, classes, methods or modules.

--> They appear right after the definition of a function,method,class or module.

In [ ]:
```
def square (num):
    '''Square function : This funstion will return the square of a number''
    return num**2
square(5)
```

Out[13]:
```
25
```

Varibales

A python variable is a reserved memory location to store values.A variable is created the moment you first assign a value to it.

In [ ]:
```
 a = 12

a
```

Out[15]:
```
12
```

In [ ]:
```
id(a)
```

Out[16]:
```
140729621433112
```

In [ ]:
```
hex(id(a)) # Memory address of the variable
```

Out[18]:
```
'0x7ffe2b184b18'
```

Data types in python

Fundamental Data types

1.Integer --- eg: 1,2,23,69 etc whole number

2.Float ----- eg: 1.2,1.0,2.0 etc contains decimal

3.Complex --- eg: 2i+j

4.Bollean --- True,False

Derived or series data types

1.List : sequence of data seperated by comma and enclosed in square brakets and mutable.

eg : [1,2,2.0, "text"]

2.Tuple : sequence of data seperated by comma ,enclosed in parenthesis and immutable.
eg: (1,2,3,4,5)

3. string : enclosed in quotes

eg: "string", "abs" etc

4.Dictionary : key value pair enclosed in curlybraces.

eg : dict = {name : ram, age : 15}

In [ ]:
```
## Numeric data type

val = 10 # integer
print(val)
print(type(val)) # type of data
```

```
10
<class 'int'>
```

In [ ]:
```
## Float data type

var1 = 10.0
print(var1)
print(type(var1))
```

```
10.0
<class 'float'>
```

In [ ]:
```
# complex data type

var2 = 10+12j
print(var2)
print(type(var2))
```

```
(10+12j)
<class 'complex'>
```

In [ ]:
```
## Boolean data type

bool1 = True
bool2 = False

print(type(bool1))
print(type(bool2))
```

```
<class 'bool'>
<class 'bool'>
```

In [ ]:
```
print(bool(0))
print(bool(1))
```

```
False
True
```

Strings

In [ ]:
```
# string creation
str1 = "Kalpana"
print(str1)


str2 = "Hello world"
print(str2)
```

```
Kalpana
Hello world
```

In [ ]:
```
## Length of string
len(str2) # no of characters in string
```

Out[10]:
```
11
```

In [ ]:
```
len(str1)
```

Out[11]:
```
7
```

In [ ]:
```
## String indexing

str = "PYTHON"

print(len(str))

print(str[0]) # first character in string
print(str[1])
print(str[2])
print(str[3])
print(str[4])
print(str[5])# last character in string
print(str[-1])# last character in string
```

```
6
P
Y
T
H
O
N
N
```

In [ ]:
```
str[len(str)-1]
```

Out[14]:
```
'N'
```

In [ ]:
```
# string slicing
str3 = 'HELLO WORLD'

print(len(str3))
str3[0:5]
```

```
11
```

Out[16]:
```
'HELLO'
```

In [ ]:
```
str3[0:2] # retrieve first 2 characters from string
```

Out[17]:
```
'HE'
```

In [ ]:
```
## Upadate & Delete string

str3
```

Out[18]:
```
'HELLO WORLD'
```

In [ ]:
```
## strings are immutable

str3[0:2] = "HI"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[19], line 3
      1 ## strings are immutable
----> 3 str3[0:2] = "HI"

TypeError: 'str' object does not support item assignment
```

In [ ]:
```
del str3  # delete a string
```

In [ ]:
```
# string concatenation
str1 = "HELLO"
str2 ="RAM"

str3 = str1+str2
print(str3)
```

```
HELLORAM
```

In [ ]:
```
str1 = "HELLO"
str2 ="RAM"

str3 = str1+" " +str2
print(str3)
```

```
HELLO RAM
```

In [ ]:
```
# Iterating through a sring

str = "PYTHON"

for i in str:
    print(i)
```

```
P
Y
T
H
O
N
```

In [ ]:
```
for i in enumerate(str): # enumerate give index
    print(i)
```

```
(0, 'P')
(1, 'Y')
(2, 'T')
(3, 'H')
(4, 'O')
(5, 'N')
```

In [ ]:
```
# string membership

str = "welcome to python tutorial"

print("welcome" in str) # chesk substring welcome present in the string or
print("python" in str)
print("hello" in str)
```

```
True
True
False
```

In [ ]:
```
# string partioning

str1 = str.partition("to")

print(str1)
```

```
('welcome ', 'to', ' python tutorial')
```

In [ ]:
```
str1 = str.rpartition("t")

print(str1)
```

```
('welcome to python tu', 't', 'orial')
```

In [ ]:
```
# string functions

string = "  PYTHON  "
string
```

Out[32]:
```
'  PYTHON  '
```

In [ ]:
```
# strip remove white space from begining and end
print(string.strip())#remove white space from begining and end
print(string.lstrip())#remove white space from begining of string
print(string.rstrip())#remove white space from end of string
```

In [ ]:
```
str5 = "*****HELLO WORLD******"
print(str5.strip('*'))
print(str5.lstrip('*'))
print(str5.rstrip('*'))
```

```
HELLO WORLD
HELLO WORLD******
*****HELLO WORLD
```

In [ ]:
```
str6 = "WEL come"

str6.upper()
```

Out[40]:
```
'WEL COME'
```

In [ ]:
```
str6.lower()
```

Out[41]:
```
'wel come'
```

In [ ]:
```
str6.replace("WEL" ,"ram") #replace "WEL" with "ram"
```

Out[42]:
```
'ram come'
```

In [ ]:
```
str6.replace(" ","")
```

Out[43]:
```
'WELcome'
```

In [ ]:
```
str6.count("e") # count the number of times "e" is present in string
```

Out[44]:
```
1
```

In [ ]:
```
str6.startswith('W')
```

Out[45]:
```
True
```

In [ ]:
```
str6.endswith('e')
```

Out[47]:
```
True
```

In [ ]:
```
my_str = "HELLO WELCOME TO PYTHON TUTORIAL "
my_str1 = my_str.split()
my_str1
```

Out[49]:
```
['HELLO', 'WELCOME', 'TO', 'PYTHON', 'TUTORIAL']
```

In [ ]:
```
## CENTER -- center align the string

str = "WELCOME"

str.center(100)
```

Out[1]:
```
'                                                WELCOME
'
```

In [ ]:
```
str1 = str.center(100,'*')
str1
```

Out[3]:
```
'************************************************WELCOME*************************
*************************'
```

In [ ]:
```
# rjust -- right align the string

str2 = str.rjust(50)
print(str2)
str2 = str.rjust(50,'*')
print(str2)
```

```
                                                 WELCOME
**************************************************WELCOME
```

In [ ]:
```
# find -- Find the location of word
str7 = 'one two three four five six  two seven'

loc = str7.find("two")
loc
```

Out[14]: 4

In [ ]:
```
loc = str7.rfind("two") # last occurence of word "two" in string
loc
```

Out[15]: 29

In [ ]:
```
loc = str7.index("two")
loc
```

Out[10]: 4

In [ ]:
```
loc = str7.rindex("two") # last occurence word "two" in string
loc
```

Out[16]: 29

In [ ]:
```
str7 = "123pythonHELLO"

print(str7.isalpha())
print(str7.isalnum())
print(str7.isdecimal())
print(str7.isnumeric())
```

```
False
True
False
False
```

In [ ]:
```
str8 = "PYTHON"

print(str8.isupper())
print(str8.islower())
```

```
True
False
```

LIST

In [ ]:
```
list = [] # empty list
```

In [ ]:
```
 print(type(list))
```

```
<class 'list'>
```

In [ ]:
```
list = [ 1,2,3,4,5]

list1 = [1,1,2,3,[5,"hello"],["a","b"]] # nested list
len(list1)
```

Out[19]: 6

In [ ]:
```
# List indexing
list2 = ["amala",[1,2,3],"A",45,23]
list2[0] # retrieve the first element of the list
```

Out[21]: 'amala'

In [ ]:
```
list2[-1] # last element of list
```

Out[23]: 23

In [ ]:
```
list2[1][1]
```

Out[22]: 2

In [ ]:
```
### LIST SLICING

# list[start:stop:step]
# start - inclusive
# stop - exclusive

my_list = ["ram","krishna","seetha","kalpana","ananya","aniruth"]

my_list[0:3] # return the elements from 0 to 3,
```

Out[24]: ['ram', 'krishna', 'seetha']

In [ ]:
```
my_list[1:5:2]
```

Out[25]: ['krishna', 'kalpana']

In [ ]:
```
my_list[:2] #return first two elements
```

Out[26]: ['ram', 'krishna']

In [ ]:
```
## APPEND -- Add an item to the end of the list

my_list.append('Jaisvi')
my_list
```

Out[27]: ['ram', 'krishna', 'seetha', 'kalpana', 'ananya', 'aniruth', 'Jaisvi']

In [ ]:
```
# INSERT -- Add an item at specific index location

my_list.insert(1,"banu")
my_list
```

Out[29]:
```
['ram', 'banu', 'krishna', 'seetha', 'kalpana', 'ananya', 'aniruth', 'Jais
vi']
```

In [ ]:
```
## Remove -- remove the element

my_list.remove("kalpana")
my_list
```

Out[30]:
```
['ram', 'banu', 'krishna', 'seetha', 'ananya', 'aniruth', 'Jaisvi']
```

In [ ]:
```
## Remove -- remove last element of list

my_list.pop()
my_list
```

Out[31]:
```
['ram', 'banu', 'krishna', 'seetha', 'ananya', 'aniruth']
```

In [ ]:
```
# remove an item at specified index location
my_list.pop(2)
my_list
```

Out[32]:
```
['ram', 'banu', 'seetha', 'ananya', 'aniruth']
```

In [ ]:
```
# change value of string
my_list = [1,20,30,50,60,80,90,100]

my_list[0] = 1000
my_list
```

Out[35]:
```
[1000, 20, 30, 50, 60, 80, 90, 100]
```

In [ ]:
```
my_list[3] = 25
my_list
```

Out[37]:
```
[1000, 20, 30, 25, 60, 80, 90, 100]
```

In [ ]:
```
my_list.clear()
my_list
```

Out[39]:
```
[]
```

In [ ]:
```
# Join the list

list1 = [1,2,3,4]
list2 = [5,6,7,8,9]

list3 = list1+list2
list3
```

Out[40]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```
list4 = [10,20,30]
list5 = [50,60,70]

list4.extend(list5)
list4
```

Out[54]:
```
[10, 20, 30, 50, 60, 70]
```

In [ ]:
```
10 in list4
```

Out[55]:
```
True
```

In [ ]:
```
100 in list4
```

Out[56]:
```
False
```

In [ ]:
```
# REVERSE
list3 = ["amith",45,12,100,"text"]
list3.reverse()

list3
```

Out[61]:
```
['text', 100, 12, 45, 'amith']
```

In [ ]:
```
list3[::-1]
```

Out[62]:
```
['amith', 45, 12, 100, 'text']
```

In [ ]:
```
my_list = [10,20,56,78,12,2,3]

my_list.sort()# Acending order
my_list
```

Out[64]:
```
[2, 3, 10, 12, 20, 56, 78]
```

In [ ]:
```
my_list.sort(reverse = True)
my_list
```

Out[65]:
```
[78, 56, 20, 12, 10, 3, 2]
```

In [ ]:
```
my_list.count(2)
```

Out[66]:
```
1
```

ALL

-- returns True if all elements in a list are true & returns False if any element in the list is False.

ANY

-- Returns True if any element in the list is True.

In [ ]:
```
list = [ 1,2,3,"hi",5.0]

list1 = all(list)
list1
```

Out[2]:
```
True
```

In [ ]:
```
list = [0,1,2,3]

print(all(list)) # 0 considered as False
```

```
False
```

In [ ]:
```
print(any(list))
```

```
True
```

List Comprehension

[expression for item in list]

In [ ]:
```
str = "PYTHON"
list = [i for i in str]
list
```

In [ ]:
```
list1 = [i for i in range(20) if i%2==0]# display even numbers between 0-20
list1
```

Out[7]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

In [ ]:
```
list2 = [i**2 for i in range(10)]
list2
```

Out[8]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [ ]:
```
# multiply whole list by 10
list1 = [1,2,3,4,5]
list1 = [i*10 for i in list1]
print(list1)
```

```
[10, 20, 30, 40, 50]
```

In [ ]:
```
## Extract numbers from a string

mystr = "one 1 two 2 three 3 four 4 five 5 six 7 7"
num = [i for i in mystr if i.isdigit()]
num
```

Out[10]:
```
['1', '2', '3', '4', '5', '7', '7']
```

In [ ]:
```
# extract text from a string
mystr = "one 1 two 2 three 3 four 4 five 5 six 7 7"
char = [i for i in mystr if i.isalpha()]
char
```

Out[11]:
```
['o',
 'n',
 'e',
 't',
 'w',
 'o',
 't',
 'h',
 'r',
 'e',
 'e',
 'f',
 'o',
 'u',
 'r',
 'f',
 'i',
 'v',
 'e',
 's',
 'i',
 'x']
```

Tuples

Iterating over the elements of a tuple is faster compared to iterating over a list.

In [ ]:
```
# Tuple creation
tuple = ()
t1 = (1,2,3,4)
t2 = ("to","hello","hello")
t3 = ("ram",1,2,69,(1,2))
t4 = (1.0,89,5.2,"hi")
t5 = ("goa",45,[1,2],{20,30},(1,2,3))
```

In [ ]:
```
print(type(t1))
```

```
<class 'tuple'>
```

In [ ]:
```
# tuple indexing

t1 = ("abc",20,30,40,50)
print([t1[0]]) # retrieve first element of tuple
```

```
['abc']
```

In [ ]:
```
t1[-1] #retrieve last element
```

Out[15]:
```
50
```

In [ ]:
```
t1[0][1] # nested indexing,access second character of first element
```

Out[16]:
```
'b'
```

In [ ]:
```
# Tuple slicing
tuple = ('ONE','two','Three','Four','Five','SIX')
tuple[0:3]
```

Out[17]:
```
('ONE', 'two', 'Three')
```

In [ ]:
```
tuple[2:4]
```

Out[18]:
```
('Three', 'Four')
```

In [ ]:
```
tuple[-1:]
```

Out[19]:
```
('SIX',)
```

In [ ]:
```
tuple[: :-1]
```

Out[20]:
```
('SIX', 'Five', 'Four', 'Three', 'two', 'ONE')
```

In [ ]:
```
tuple[:] # return whole tuple
```

Out[21]:
```
('ONE', 'two', 'Three', 'Four', 'Five', 'SIX')
```

In [ ]:
```
# Count

tuple = (1,2,3,3,4,5,6,6,6)
tuple.count(6) # occuence of 6 in tuple
```

Out[22]:
```
3
```

In [ ]:
```
tuple.count(2)# occuence of 2 in tuple
```

Out[23]:
```
1
```

In [ ]:
```
tuple.count(3) ## occuence of 3 in tuple
```

Out[24]:
```
2
```

In [ ]:
```
# Tuple Membership
tuple
```

Out[25]:
```
(1, 2, 3, 3, 4, 5, 6, 6, 6)
```

In [ ]:
```
1 in tuple
```

Out[26]:
```
True
```

In [ ]:
```
10 in tuple
```

Out[27]:
```
False
```

In [ ]:
```
# Index Position
tuple = ("ram","Krishna",50,100,400,"hello")

tuple
```

Out[29]:
```
('ram', 'Krishna', 50, 100, 400, 'hello')
```

In [ ]:
```
tuple.index('Krishna')
```

Out[31]:
```
1
```

In [ ]:
```
tuple.index(50)
```

Out[32]:
```
2
```

In [ ]:
```
# Sorting

sorted(tuple)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last
t)
Cell In[33], line 3
      1 # Sorting
----> 3 sorted(tuple)

TypeError: '<' not supported between instances of 'int' and 'str'
```

In [ ]:
```
tuple = (10,50,89,12,42,41,0,23,2)

sorted(tuple) # asending order
```

Out[34]:
```
[0, 2, 10, 12, 23, 41, 42, 50, 89]
```

In [ ]:
```
sorted(tuple,reverse = True)# Descending order
```

Out[35]:
```
[89, 50, 42, 41, 23, 12, 10, 2, 0]
```

Sets

1.Unordered & Unindexed collection of items. 2.Set elements are unique.duplicate elements not allowed. 3.immutable 4.set itself is mutable.we can add or remove items from it.

In [ ]:
```
set = {1,2,3,4,5,6}

set
```

Out[36]:
```
{1, 2, 3, 4, 5, 6}
```

In [ ]:
```
print(type(set))
```

```
<class 'set'>
```

In [ ]:
```
len(set)
```

Out[38]:
```
6
```

In [ ]:
```
set1 ={1,2,3,3,3,56,5,5}
set1 # duplicates not allowed
```

Out[39]:
```
{1, 2, 3, 5, 56}
```

In [ ]:
```
set2 ={"one","two","three","four","five"}
set2
```

Out[41]:
```
{'five', 'four', 'one', 'three', 'two'}
```

In [ ]:
```
# adding elemnts

set2.add("six")
set2
```

Out[42]:
```
{'five', 'four', 'one', 'six', 'three', 'two'}
```

In [ ]:
```
# add multiple elements
set2.update(["seven","BaseExceptionGroupeight","eight","ten"])
set2
```

Out[51]:
```
{'BaseExceptionGroupeight',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'ten',
 'three',
 'two'}
```

In [ ]:
```
# Remove
set2.remove('eight')
set2
```

Out[53]:
```
{'BaseExceptionGroupeight',
 'five',
 'four',
 'one',
 'seven',
 'ten',
 'three',
 'two'}
```

In [ ]:
```
set2.discard('seven')
set2
```

Out[54]:
```
{'BaseExceptionGroupeight', 'five', 'four', 'one', 'ten', 'three', 'two'}
```

In [ ]:
```
set2.clear()
set2
```

Out[56]:
```
set()
```

In [ ]:
```
del set2
set2
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[58], line 2
      1 del set2
----> 2 set2

NameError: name 'set2' is not defined
```

SET OPERATION

In [ ]:
```
a ={10,20,30,40,50}
b= {40,50,60,70}
c= {60,70,80,90,100}
```

In [ ]:
```
a|b # union of a and b (alla elements from both sets,no duplicates)
```

Out[61]:
```
{10, 20, 30, 40, 50, 60, 70}
```

In [ ]:
```
a.union(b)
```

Out[62]:
```
{10, 20, 30, 40, 50, 60, 70}
```

In [ ]:
```
a.union(b,c)
```

Out[63]:
```
{10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
```

In [ ]:
```
a.update(b,c)
a
```

Out[65]:
```
{10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
```

In [ ]:
```
A = {1,2,3,4,5,6,7}
B = {6,7,8,9,10,11,12}
```

In [ ]:
```
A & B # Intersecton of A and B (common items from both the sets)
```

Out[69]:
```
{6, 7}
```

In [ ]:
```
A.intersection(B)
```

Out[70]:
```
{6, 7}
```

In [ ]:
```
# Difference

A = {1,2,3,4,5,6,7}
B = {6,7,8,9,10,11,12}

A-B # set of elements that only in A not in B
```

Out[71]:
```
{1, 2, 3, 4, 5}
```

In [ ]:
```
A.difference(B)
```

Out[73]:
```
{1, 2, 3, 4, 5}
```

In [ ]:
```
B - A # set of elements that only in B not in A
```

Out[72]:
```
{8, 9, 10, 11, 12}
```

In [ ]:
```
B.difference(A)
```

Out[75]:
```
{8, 9, 10, 11, 12}
```

In [ ]:
```
# Symmetric difference
A^B # set of elements in both A and B ,not in both
```

Out[76]:
```
{1, 2, 3, 4, 5, 8, 9, 10, 11, 12}
```

In [ ]:
```
A.symmetric_difference(B)
```

Out[77]:
```
{1, 2, 3, 4, 5, 8, 9, 10, 11, 12}
```

In [ ]:
```
A.symmetric_difference_update(B)
A
```

Out[79]:
```
{1, 2, 3, 4, 5, 6, 7}
```

Subset,Superset, and Disjoint

In [ ]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40,50}
```

In [ ]:
```
B.issubset(A) # if all elements of B are present in A
```

Out[82]:
```
True
```

In [ ]:
```
A.issuperset(B)
```

Out[83]:
```
True
```

In [ ]:
```
C.isdisjoint(A) # two sets are said to be disjoints sets having no common e
```

Out[84]:
```
True
```

Other buitin functions

In [ ]:
```
A
```

Out[85]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [ ]:
```
sum(A)
```

Out[86]:
```
45
```

In [ ]:
```
max(A)
```

Out[87]:
```
9
```

In [ ]:
```
min(A)
```

Out[88]:
```
1
```

In [ ]:
```
len(A)
```

Out[89]:
```
9
```

Dictionary data type

1.Dictionary is a mutable data type in Python. 2.A python dictionary is a collection of key and value pairs separated by a colon (:) & enclosed in curly braces {}. 3.Keys must be unique in a dictionary, duplicate values are allowed.

In [ ]:
```
dictionary = dict() #empty dictionary
dictionary
```

Out[1]:
```
{}
```

In [ ]:
```
mydict = {"name" : 'Kalpana', "city" : 'Hyd',"state": "TS"}
mydict
```

Out[7]:
```
{'name': 'Kalpana', 'city': 'Hyd', 'state': 'TS'}
```

In [ ]:
```
mydict = {1:'one' , 2:'two' , 3:'three'} # dictionary with integer keys
mydict
```

Out[8]:
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [ ]:
```
mydict = {1:'one' , 'A':'two' , 3:'three'} # dictionary with mixed keys
mydict
```

Out[9]:
```
{1: 'one', 'A': 'two', 3: 'three'}
```

In [ ]:
```
mydict.keys() # Return Dictionary Keys using keys() method
```

Out[10]:
```
dict_keys([1, 'A', 3])
```

In [ ]:
```
mydict.values() # Return Dictionary Values using values() method
```

Out[11]:
```
dict_values(['one', 'two', 'three'])
```

In [ ]:
```
mydict.items() # Access each key-value pair within a dictionary
```

Out[12]:
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

In [ ]:
```
mydict1 = {'Name':'Amar' , 'ID': 10001 , 'DOB': 1991 , 'job' :'Analyst'}
mydict1
```

Out[16]:
```
{'Name': 'Amar', 'ID': 10001, 'DOB': 1991, 'job': 'Analyst'}
```

In [ ]:
```
mydict1['Name']
```

Out[17]:
```
'Amar'
```

In [ ]:
```
mydict1.get('job')
```

Out[18]:
```
'Analyst'
```

Add,Remove and Change items

In [ ]:
```
mydict2 = {'Name':'Hanshu' , 'ID': 1001 , 'DOB':2002 , 'job' :'Data Scienti
mydict2
```

Out[21]:
```
{'Name': 'Hanshu', 'ID': 1001, 'DOB': 2002, 'job': 'Data Scientist'}
```

In [ ]:
```
mydict2['Name'] = 'Bhanu'
mydict2
```

Out[22]:
```
{'Name': 'Bhanu', 'ID': 1001, 'DOB': 2002, 'job': 'Data Scientist'}
```

In [ ]:
```
mydict2['city'] ='Hyd' # adding an item in dictionary
mydict2
```

Out[24]:
```
{'Name': 'Bhanu',
 'ID': 1001,
 'DOB': 2002,
 'job': 'Data Scientist',
 'city': 'Hyd'}
```

In [ ]:
```
mydict2.pop('job') # removing an item
mydict2
```

Out[25]:
```
{'Name': 'Bhanu', 'ID': 1001, 'DOB': 2002, 'city': 'Hyd'}
```

In [ ]:
```
mydict2.popitem()# random item is removed
```

Out[27]:
```
('DOB', 2002)
```

In [ ]:
```
mydict2
```

Out[28]:
```
{'Name': 'Bhanu', 'ID': 1001}
```

In [ ]:
```
del [mydict2['ID']] # removing an item
```

In [ ]:
```
mydict2.clear()
mydict2
```

Out[32]:
```
{}
```

In [ ]:
```
# Dictionary membership
mydict3 = {'Name':'RK' , 'ID': 1001 , 'DOB':2002 , 'job' :'Data Scientist'}
mydict3
'Name' in mydict3 # Test if a key is in a dictionary or not.
```

Out[34]:
```
True
```

In [ ]:
```
'DOB' in mydict3 # Membership test can be only done for keys.
```

Out[39]:
```
True
```

In [ ]:
```
'Address' in mydict3
```

Out[37]:
```
False
```

ALL/ANY

The all() method returns: True - If all all keys of the dictionary are true False - If any key of the dictionary is false The any() function returns True if any key of the dictionary is True. If not, any() returns False

In [ ]:
```
all(mydict3)
```

Out[42]:
```
True
```

In [ ]:
```
any(mydict3)
```

Out[41]:
```
True
```

Dictionary Comprehension

{key : value for var in iterable} {i : i**2 for i in range(10)}

In [ ]:
```
dict = {i:i*2 for i in range(10)}
dict
```

Out[46]:
```
{0: 0, 1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}
```

In [ ]:
```
square_dict = { i:i**2 for i in range (10)}
square_dict
```

Out[45]:
```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```