In [333…
```
import keyword
```

In [334…
```
dir(keyword)
```

Out[334…
```
['__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'iskeyword',
 'kwlist',
 'main']
```

In [3]:
```
keyword.iskeyword('if')
```

Out[3]:
```
True
```

In [4]:
```
keyword.iskeyword('tutu')
```

Out[4]:
```
False
```

In [5]:
```
(keyword.kwlist)
```

Out[5]:
```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

In [6]:
```
len(keyword.kwlist)
```

Out[6]:
```
33
```

Identifiers

In [7]:
```
global_x = 1
```

In [8]:
```
global_x
```

Out[8]:
```
1
```

In [9]:
```
global_x == 2
```

Out[9]:
```
False
```

Comments

In [10]:
```
"""
This is the next chapter of
the
module
1
"""
```

Out[10]:
```
'\nThis is the next chapter of \nthe \nmodule\n1\n'
```

Multi-line Statements

In [11]:
```
1 + 2 + 3 +\
4 + 5
```

Out[11]:
```
15
```

Python Memory Storage Location

In [12]:
```
rajesh = 30
```

In [13]:
```
id(rajesh)
```

Out[13]:
```
1600029712
```

In [14]:
```
help(id)
```

```
Help on built-in function id in module builtins:

id(obj, /)
    Return the identity of an object.

    This is guaranteed to be unique among simultaneously existing objects.
    (CPython uses the object's memory address.)
```

In [15]:
```
ram = 30
```

In [16]:
```
id(ram)
```

Out[16]:
```
1600029712
```

In [17]:
```
jai = 32
```

In [18]:
```
id(jai)
```

Out[18]: 1600029776

In [19]:
```
1705346128 - 1705346064
```

Out[19]: 64

In [20]:
```
jeetu = 33
```

In [21]:
```
id(jeetu)
```

Out[21]: 1600029808

In [22]:
```
1705346160 - 1705346128
```

Out[22]: 32

In [23]:
```
jitendra = 44
```

In [24]:
```
id(jitendra)
```

Out[24]: 1600030160

In [25]:
```
1705346512 - 1705346160
```

Out[25]: 352

In [26]:
```
name = 'rajesh'
```

In [27]:
```
id(name)
```

Out[27]: 2391318510312

In [28]:
```
last_name = 'sharma'
```

In [29]:
```
id(last_name)
```

Out[29]: 2391318560528

In [30]:
```
first_name = 'rajesh'
```

In [31]:
```
id(first_name)
```

Out[31]: 2391318510312

In [32]:
```
id(name) == id(first_name)
```

Out[32]: True

In [33]:
```
x = 257
```

In [34]:
```
id(x)
```

Out[34]: 2391318317392

In [35]:
```
x = 30
```

In [36]:
```
id(x)
```

Out[36]:
```
1600029712
```

In [37]:
```
rajesh = 257
```

In [38]:
```
id(rajesh)
```

Out[38]:
```
2391318317744
```

In [39]:
```
rajesh = x
```

In [40]:
```
id(rajesh)
```

Out[40]:
```
1600029712
```

In [41]:
```
y=268
```

In [42]:
```
id(y)
```

Out[42]:
```
2391318317936
```

In [43]:
```
rajesh = 268
```

In [44]:
```
id(y), id(rajesh)
```

Out[44]:
```
(2391318317936, 2391318318064)
```

Scenario:I

In [45]:
```
z = 306
```

In [46]:
```
id(z)
```

Out[46]:
```
2391318318160
```

In [47]:
```
p = 887
```

In [48]:
```
id(z), id(p)
```

Out[48]:
```
(2391318318160, 2391318318352)
```

In [49]:
```
p = 306
```

In [50]:
```
id(z), id(p)
```

Out[50]:
```
(2391318318160, 2391318318384)
```

In [51]:
```
z = 887
```

In [52]:
```
id(z), id(p)
```

Out[52]:
```
(2391318318352, 2391318318384)
```

Scenario-II

In [53]:
```
ww = 'raman' ## 'raman' is created here and assigned a memory location ending with 2
```

In [54]:
```
id(ww)
```

Out[54]: 2391318635440

In [55]:
```
yy = 'taman' ## 'taman' is created here and assigned a memory location ending with 3
```

In [56]:
```
id(ww), id(yy)
```

Out[56]: (2391318635440, 2391318636896)

In [57]:
```
yy = 'raman' ## Before running this line both 'raman' and 'taman' already exist in t
             ## memory location with 'raman' thats ending with 248
```

In [58]:
```
id(ww), id(yy) ## Now, 'taman' no more exists
```

Out[58]: (2391318635440, 2391318635440)

In [59]:
```
ww = 'taman' ## Here, we are initializing 'taman' to ww, now in this line 'taman' wi
             ## thus gets a new memory location
```

In [60]:
```
id(ww), id(yy)
```

Out[60]: (2391317343176, 2391318635440)

In [61]:
```
ww = yy
```

In [62]:
```
id(ww), id(yy)
```

Out[62]: (2391318635440, 2391318635440)

Scenario-II : Solved

In [63]:
```
ww = 'bat'
```

In [64]:
```
id(ww)
```

Out[64]: 2391289260832

In [65]:
```
yy = 'taman'
```

In [66]:
```
id(ww), id(yy)
```

Out[66]: (2391289260832, 2391318648680)

In [67]:
```
xx = 'bat'
```

In [68]:
```
id(ww), id(yy), id(xx)
```

Out[68]: (2391289260832, 2391318648680, 2391289260832)

In [69]:
```
ww = 'taman'
```

In [70]:
```
id(ww), id(yy), id(xx)
```

Out[70]:
```
(2391318648680, 2391318648680, 2391289260832)
```

In [71]:
```
ww = yy
```

In [72]:
```
id(ww), id(yy)
```

Out[72]:
```
(2391318648680, 2391318648680)
```

Scenario:III

In [73]:
```
ppp = 259
```

In [74]:
```
qqq = 209
```

In [75]:
```
id(ppp), id(qqq)
```

Out[75]:
```
(2391317365264, 1600035440)
```

In [76]:
```
qqq = 259
```

In [77]:
```
id(ppp), id(qqq)
```

Out[77]:
```
(2391317365264, 2391317364880)
```

Python allocates the same memory location in case of integers when values are between the range of [-5, 256]. If values are outside this range then new memory location is used.

It is the pure python implementation as they believed that mostly integers of this range are used frequently.

Tuples

In [78]:
```
tup = ('ram','kumar','sharma','bharadwaj','saanvlodiya')
```

In [79]:
```
tup
```

Out[79]:
```
('ram', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')
```

In [80]:
```
empty_tuple, single_element_tuple = tuple(), ('manohar')
```

In [81]:
```
empty_tuple, len(empty_tuple), type(empty_tuple), single_element_tuple, len(single_e
```

Out[81]:
```
((), 0, tuple, 'manohar', 7, str)
```

In [82]:
```
empty_tuple, single_element_tuple = tuple(), ('manohar',)
```

In [83]:
```
empty_tuple, len(empty_tuple), type(empty_tuple), single_element_tuple, len(single_e
```

Out[83]:
```
((), 0, tuple, ('manohar',), 1, tuple)
```

The difference between the ('manohar') and ('manohar',) is that w/o comma(,) python will take it as a string else a tuple.

In [84]:
```
tup[0], tup[-1]
```

Out[84]: ('ram', 'saanvlodiya')

In [85]:
```
tup[::-1]
```

Out[85]: ('saanvlodiya', 'bharadwaj', 'sharma', 'kumar', 'ram')

In [86]:
```
tup[::-2]
```

Out[86]: ('saanvlodiya', 'sharma', 'ram')

In [87]:
```
tup
```

Out[87]: ('ram', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')

In [88]:
```
'hari', *tup
```

Out[88]: ('hari', 'ram', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')

In [89]:
```
'hari', *tup[0:2]
```

Out[89]: ('hari', 'ram', 'kumar')

In [90]:
```
'hari', *tup[::-1]
```

Out[90]: ('hari', 'saanvlodiya', 'bharadwaj', 'sharma', 'kumar', 'ram')

In [91]:
```
'hari', *tup[::-2]
```

Out[91]: ('hari', 'saanvlodiya', 'sharma', 'ram')

Editing a tuple

CASE-I :: Using List

In [92]:
```
tup = list(tup)
```

In [93]:
```
tup
```

Out[93]: ['ram', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya']

In [94]:
```
tup[0] = 'jai'
```

In [95]:
```
tup = tuple(tup)
```

In [96]:
```
tup
```

Out[96]: ('jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')

CASE-II :: Using Starred expression

By using * we can unpack the elements of the iterables and we can unpack the elements or values of dictionaries**

In [97]:
```
tup
```

Out[97]: ('jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')

In [98]:
```
'hari', *tup[0:]
```

Out[98]:
```
('hari', 'jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')
```

In [99]:
```
'hari', *tup[1:]
```

Out[99]:
```
('hari', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')
```

In [100…
```
'hari', 300, *tup[0:]
```

Out[100…
```
('hari', 300, 'jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')
```

In [101…
```
"{}".format(tup)
```

Out[101…
```
"('jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')"
```

SETS are mutable but frozensets are not

In [102…
```
set_a = {1,2,3,4,4}
```

In [103…
```
set_a
```

Out[103…
```
{1, 2, 3, 4}
```

In [104…
```
set_a.add(("Rama","Shyama"))
```

In [105…
```
set_a
```

Out[105…
```
{('Rama', 'Shyama'), 1, 2, 3, 4}
```

In [106…
```
set_a.add("Cat")
```

In [107…
```
set_a
```

Out[107…
```
{('Rama', 'Shyama'), 1, 2, 3, 4, 'Cat'}
```

In [108…
```
frozen_set_b = frozenset((1,2,3,4,5,5))
frozen_set_b
```

Out[108…
```
frozenset({1, 2, 3, 4, 5})
```

In [109…
```
frozen_set_b.add(a)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-109-bc258a6c0b76> in <module>
----> 1 frozen_set_b.add(a)

AttributeError: 'frozenset' object has no attribute 'add'
```

In [110…
```
frozen_set_b.intersection(set_a)
```

Out[110…
```
frozenset({1, 2, 3, 4})
```

In [111…
```
new_list = [100,200,300,400,500,'raje']
```

In [112…
```
1, *(1,2,3,4,(5,6,7)), type((1, *(1,2,3,4,(5,6,7))))
```

Out[112… (1, 1, 2, 3, 4, (5, 6, 7), tuple)

In [113…
```
a =()
a
```

Out[113… ()

In [114…
```
new_list
```

Out[114… [100, 200, 300, 400, 500, 'raje']

In [115…
```
'a', *new_list, type(('a', *new_list))
```

Out[115… ('a', 100, 200, 300, 400, 500, 'raje', tuple)

In [116…
```
players_dict = {'name':['rajesh','ramesh','suresh','mohitest','jyotesh'],
                'best_score':[100,200,300,400,500]}
```

In [117…
```
players_dict.get('nam')
```

In [118…
```
players_dict.get('name'), players_dict.get('best_score')
```

Out[118… (['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'],
[100, 200, 300, 400, 500])

In [119…
```
players_dict
```

Out[119… {'name': ['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'],
'best_score': [100, 200, 300, 400, 500]}

In [120…
```
'{}'.format(tup), 'a',*tup
```

Out[120… ("('jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya')",
'a',
'jai',
'kumar',
'sharma',
'bharadwaj',
'saanvlodiya')

In [121…
```
type('{}'.format(tup))
```

Out[121… str

In [122…
```
unpacked_tuple = '{}'.format(tup)
```

In [123…
```
unpacked_tuple.split("'")[1::2]
```

Out[123… ['jai', 'kumar', 'sharma', 'bharadwaj', 'saanvlodiya']

In [124…
```
'{name}{best_score}'.format(**players_dict)
```

Out[124… "['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'][100, 200, 300, 400, 500]"

In [125…
```
"{name}".format(**players_dict)
```

Out[125… "['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh']"

In [126…
```
"{best_score}".format(**players_dict)
```

Out[126… '[100, 200, 300, 400, 500]'

In [127…
```
def dict_print(d):
    print("{name}{best_score}".format(**d))
```

In [128…
```
dict_print(players_dict)
```

['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'][100, 200, 300, 400, 500]

In [129…
```
dict_unpacked = "{name}{best_score}".format(**players_dict)
```

In [130…
```
type(dict_unpacked)
```

Out[130… str

In [131…
```
dict_unpacked
```

Out[131… "['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'][100, 200, 300, 400, 500]"

In [132…
```
dict_unpacked.split("][")
```

Out[132…
```
["['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'",
 '100, 200, 300, 400, 500]']
```

In [133…
```
for val in dict_unpacked.split("]["):
    print(val)
```

```
['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'
100, 200, 300, 400, 500]
```

In [134…
```
player_name = "{name}".format(**players_dict)
player_score = "{best_score}".format(**players_dict)
```

In [135…
```
player_name
```

Out[135… "['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh']"

In [136…
```
type(player_name)
```

Out[136… str

In [137…
```
for val in player_name.split('"'):
    print(val)
```

['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh']

In [138…
```
player_score
```

Out[138… '[100, 200, 300, 400, 500]'

In [139…
```
type(player_score)
```

Out[139… str

In [140…
```
for val in player_score.split('"'):
    print(val)
```

[100, 200, 300, 400, 500]

In [141…
```
players_dict['name']
```

Out[141… ['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh']

In [142…
```
players_dict.keys()
```

Out[142… dict_keys(['name', 'best_score'])

In [143…
```
players_dict.values()
```

Out[143… dict_values([['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh'], [100, 200, 300, 400, 500]])

In [144…
```
help(__builtins__.__doc__)
```

```
No Python documentation found for "Built-in functions, exceptions, and other object
s.\n\nNoteworthy: None is the `nil' object; Ellipsis represents `...' in slices.".
Use help() to get the interactive help utility.
Use help(str) for help on the str class.
```

In [145…
```
dir(__builtins__)
```

Out[145…
```
['ArithmeticError',
 'AssertionError',
 'AttributeError',
 'BaseException',
 'BlockingIOError',
 'BrokenPipeError',
 'BufferError',
 'BytesWarning',
 'ChildProcessError',
 'ConnectionAbortedError',
 'ConnectionError',
 'ConnectionRefusedError',
 'ConnectionResetError',
 'DeprecationWarning',
 'EOFError',
 'Ellipsis',
 'EnvironmentError',
 'Exception',
 'False',
 'FileExistsError',
 'FileNotFoundError',
 'FloatingPointError',
 'FutureWarning',
 'GeneratorExit',
 'IOError',
 'ImportError',
 'ImportWarning',
 'IndentationError',
 'IndexError',
 'InterruptedError',
 'IsADirectoryError',
 'KeyError',
 'KeyboardInterrupt',
 'LookupError',
 'MemoryError',
 'ModuleNotFoundError',
 'NameError',
 'None',
 'NotADirectoryError',
 'NotImplemented',
 'NotImplementedError',
 'OSError',
 'OverflowError',
 'PendingDeprecationWarning',
 'PermissionError',
 'ProcessLookupError',
```

```
 'RecursionError',
 'ReferenceError',
 'ResourceWarning',
 'RuntimeError',
 'RuntimeWarning',
 'StopAsyncIteration',
 'StopIteration',
 'SyntaxError',
 'SyntaxWarning',
 'SystemError',
 'SystemExit',
 'TabError',
 'TimeoutError',
 'True',
 'TypeError',
 'UnboundLocalError',
 'UnicodeDecodeError',
 'UnicodeEncodeError',
 'UnicodeError',
 'UnicodeTranslateError',
 'UnicodeWarning',
 'UserWarning',
 'ValueError',
 'Warning',
 'WindowsError',
 'ZeroDivisionError',
 '__IPYTHON__',
 '__build_class__',
 '__debug__',
 '__doc__',
 '__import__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'abs',
 'all',
 'any',
 'ascii',
 'bin',
 'bool',
 'bytearray',
 'bytes',
 'callable',
 'chr',
 'classmethod',
 'compile',
 'complex',
 'copyright',
 'credits',
 'delattr',
 'dict',
 'dir',
 'display',
 'divmod',
 'enumerate',
 'eval',
 'exec',
 'filter',
 'float',
 'format',
 'frozenset',
 'get_ipython',
 'getattr',
 'globals',
 'hasattr',
 'hash',
 'help',
 'hex',
```

```
'id',
'input',
'int',
'isinstance',
'issubclass',
'iter',
'len',
'license',
'list',
'locals',
'map',
'max',
'memoryview',
'min',
'next',
'object',
'oct',
'open',
'ord',
'pow',
'print',
'property',
'range',
'repr',
'reversed',
'round',
'set',
'setattr',
'slice',
'sorted',
'staticmethod',
'str',
'sum',
'super',
'tuple',
'type',
'vars',
'zip']
```

OPERATORS

In [146…
```
## + , - , / , * , ** , % , //
x, y = 15, 2

print(x+y)
print(x-y)
print(x/y)
print(x*y)
print(x**y)
print(x//y)
print(x%y)
```

```
17
13
7.5
30
225
7
1
```

In [147…
```
15**2
```

Out[147…
```
225
```

In [148…
```
15%2
```

```
1
```

Out[148…

In [149…
```
(-15%2)
```

Out[149… 1

In [150…
```
-15/2
```

Out[150… -7.5

In [151…
```
(-15//2)
```

Out[151… -8

In [152…
```
15.9/2
```

Out[152… 7.95

In [153…
```
(15//2)
```

Out[153… 7

In [154…
```
pp = ()
```

In [155…
```
type(pp)
```

Out[155… tuple

In [156…
```
1+2+3+\
4+5
```

Out[156… 15

In [157…
```
10&4, 10|5, 1|2, 10&8
```

Out[157… (0, 15, 3, 8)

In [158…
```
a = b =4
i=0
while i in range(0,4):
    if a == b:
        print('hello')
        i+=1
    continue
    print('ramu')
```

```
hello
hello
hello
hello
```

In [159…
```
test_tup = (12,3,4,5,66,)
print(isinstance(a,tuple))
print(isinstance(test_tup,tuple))
print(isinstance(test_tup,(tuple,list)))
```

```
False
True
True
```

```
bool(10)
```

In [160…

Out[160… True

In [161…
```
x = [10,20,12,30,40,50,60,70,80,90]
```

In [162…
```
list(reversed(x)), sorted(x)
```

Out[162…
```
([90, 80, 70, 60, 50, 40, 30, 12, 20, 10],
 [10, 12, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [163…
```
a,b,c = 10,20,30
```

In [164…
```
print(a is b)
print(b is c)
print(a is c)
print(a is not b)
print(b is not c)
print(a is not c)
```

```
False
False
False
True
True
True
```

In [165…
```
a in x, b in x, c in x, a not in x, b not in x, c not in x
```

Out[165… (True, True, True, False, False, False)

In [166…
```
1&2, 1|2
```

Out[166… (0, 3)

In [167…
```
(3==3) & (1==2)
```

Out[167… False

In [168…
```
2&4
```

Out[168… 0

In [169…
```
if (3==3) & (1&2):
    print('hello')
else:
    print('not matched')
```

```
not matched
```

In [170…
```
10<<4
```

Out[170… 160

In [171…
```
30 is not 30
```

Out[171… False

In [172…
```
30 is not 35
```

Out[172… True

WHILE with ELSE

User-defined exception

In [173…
```
class Error(Exception):
    pass

class LengthGreater(Error):
    val_new = 10
    val_old = 14
    pass

class LengthSmaller(Error):
    val_old = 18
    pass
```

In [174…
```
hasattr(LengthGreater,'val_new')
```

Out[174… True

In [175…
```
hasattr(LengthSmaller,'val_new'), hasattr(LengthSmaller,'val_old')
```

Out[175… (False, True)

In [176…
```
getattr(LengthSmaller,'val_old')
```

Out[176… 18

In [177…
```
try:
    player_runs = []
    idx = 0
    total_runs = 0
    scores = input("Enter batsment scores of last 5 five matches(comma separated)\n>
    player_runs = [int(val) for val in scores.split(",")]
except ValueError as exception:
    print('\n',exception.__class__.__name__ +' :: '+ 'Wrong input or no input. ' + s

try:
    if len(player_runs) > 5:
        length = len(player_runs)
        print("Provided scores are {}".format(player_runs))
        player_runs = player_runs[::-1][0:5]
        print("Last 5 matches scores are {}".format(player_runs))
        while idx < len(player_runs):
            total_runs += player_runs[idx]
            idx +=1
            print(total_runs,'\n')
        raise LengthGreater
    elif len(player_runs) < 5 and len(player_runs) > 0:
        length = len(player_runs)
        print(player_runs)
        raise LengthSmaller
    elif len(player_runs) == 0:
        print("Kindly check the entered data.")
    else:
        print("Last 5 matches scores are {}".format(player_runs))
        while idx < len(player_runs):
            total_runs += player_runs[idx]
            idx +=1
            print(total_runs,'\n')
        else:
```

```
            print('Reached higher index','\n')
except IndexError as exception:
    print(exception.__class__.__name__ +' :: '+ str(exception),'\n')
except LengthSmaller:
    print("You have entered scores of last {} matches. Kindly enter scores of last 5
except LengthGreater:
    print("You have entered scores of last {} matches. We have considered the scores
finally:
    if total_runs > 400:
        print("\nBatsmen in really good form with an average score of {} in last 5 m
    elif total_runs == 0:
        print("")
    else:
        print("\nBatsmen average score is {} in last 5 matches.".format(round(total_
```

```
Provided scores are [6, 7, 8, 9, 2, 4, 5]
Last 5 matches scores are [5, 4, 2, 9, 8]
5

9

11

20

28

You have entered scores of last 7 matches. We have considered the scores of only las
t 5 matches.

Batsmen average score is 6 in last 5 matches.
```

In [178…
```
abs(-10), abs(0), abs(False), abs(True)
```

Out[178…
```
(10, 0, 0, 1)
```

In [179…
```
bool(9),bool(0),bool(1),bool(True),bool('True'),bool('False'),bool(False)
```

Out[179…
```
(True, False, True, True, True, True, False)
```

In [180…
```
all([3,4,5,6,7,8,'m','True','False',True,False])
```

Out[180…
```
False
```

In [181…
```
all([3,4,5,6,7,8,'m','True','False',True])
```

Out[181…
```
True
```

In [182…
```
any([3,4,5,6,7,8,'m','True','False',True,False])
```

Out[182…
```
True
```

In [183…
```
any(['False',False])
```

Out[183…
```
True
```

In [184…
```
divmod(15.5,2)
```

Out[184…
```
(7.0, 1.5)
```

In [185…
```
from functools import reduce
```

In [186…
```
(lambda x, y : x + y)(2,3)
```
Out[186… 5

In [187…
```
add_lambda = lambda x,y:x+y
```

In [188…
```
reduce(add_lambda, [1, 2, 3, 4, 5])
```
Out[188… 15

In [189…
```
filter_lambda = lambda x: x%2==0
```

In [190…
```
list(filter(filter_lambda,[1,2,3,4,5,6,7,8,9]))
```
Out[190… [2, 4, 6, 8]

In [191…
```
map_lambda = lambda x: x**2
```

In [192…
```
list(map(map_lambda,[1,2,3,4,5,6,7,8,9]))
```
Out[192… [1, 4, 9, 16, 25, 36, 49, 64, 81]

In [193…
```
players_dict = {'name':['rajesh','ramesh','suresh','mohitest','jyotesh'],
                'best_score':[100,200,300,400,500]}
```

In [194…
```
for i,j in enumerate(players_dict):
    print(i,"--> Index")
    print(j,"--> key")
```
```
0 --> Index
name --> key
1 --> Index
best_score --> key
```

In [195…
```
for i, (k,v) in enumerate(players_dict.items()):
    print("key :: {}".format(k))
    print("Values :: {}".format(v))
```
```
key :: name
Values :: ['rajesh', 'ramesh', 'suresh', 'mohitest', 'jyotesh']
key :: best_score
Values :: [100, 200, 300, 400, 500]
```

In [196…
```
hasattr(players_dict,'keys'),hasattr(players_dict,'values'), hasattr(players_dict,'s
```
Out[196… (True, True, False, False)

In [197…
```
getattr(players_dict,'keys'), getattr(players_dict,'values')
```
Out[197… (<function dict.keys>, <function dict.values>)

In [198…
```
class players:
    name = 'Rama'
    last_name = 'Shyama'
```

In [199…
```
hasattr(players,'name')
```
Out[199… True

In [200…
```
getattr(players,'name')
```

Out[200…
```
'Rama'
```

FOR LOOPS

STEP-WISE printing the numbers

In [201…
```
for i in range(4,41,4):
    print(i, i//3)
```

```
4 1
8 2
12 4
16 5
20 6
24 8
28 9
32 10
36 12
40 13
```

BREAK and CONTINUE

In [202…
```
nums = [num for num in range(1,21)]
```

In [203…
```
nums
```

Out[203…
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

BREAK the WHILE Loop after incrementing the value of i

In [204…
```
i = 1
while i < max(nums):
    if type(i) == int:
        print(i)
        i += 1
        print(i)
    else:
        print('Inside HI')
    i += 2
    print(i)
    break;
    print('Outise HI')
```

```
1
2
4
```

In [205…
```
i = 1.0
while i < max(nums):
    if type(i) == int:
        print(i)
        i += 1
        print(i)
    else:
        print('Inside HI')
    i += 2
    print(i)
    continue;
    print('Outise HI')
```

```
Inside HI
```

```
3.0
Inside HI
5.0
Inside HI
7.0
Inside HI
9.0
Inside HI
11.0
Inside HI
13.0
Inside HI
15.0
Inside HI
17.0
Inside HI
19.0
Inside HI
21.0
```

BREAK doesn't execute any code written after break command and CONTINUE restart the LOOP as soon it reads the continue statement.

ODD and EVEN numbers via single FOR Loop

In [206…
```
for val in nums:
    if val % 2 == 0:
        print('Even number',val)
        continue
    print('Odd number', val)
```

```
Odd number 1
Even number 2
Odd number 3
Even number 4
Odd number 5
Even number 6
Odd number 7
Even number 8
Odd number 9
Even number 10
Odd number 11
Even number 12
Odd number 13
Even number 14
Odd number 15
Even number 16
Odd number 17
Even number 18
Odd number 19
Even number 20
```

LISTS

In [207…
```
names = ['dhoni','rahul','sachin']
```

In [208…
```
names
```

Out[208…
```
['dhoni', 'rahul', 'sachin']
```

In [209…
```
names.append('raja')
```

In [210…
```
names
```

Out[210…
```
['dhoni', 'rahul', 'sachin', 'raja']
```

In [211…
```
names.insert(1,'kumble')
```

In [212…
```
names
```

Out[212…
```
['dhoni', 'kumble', 'rahul', 'sachin', 'raja']
```

In [213…
```
names.remove('raja')
```

In [214…
```
names
```

Out[214…
```
['dhoni', 'kumble', 'rahul', 'sachin']
```

In [215…
```
names.extend(['kohli','ishant'])
```

In [216…
```
names
```

Out[216…
```
['dhoni', 'kumble', 'rahul', 'sachin', 'kohli', 'ishant']
```

In [217…
```
names.append(['mayank',20])
```

In [218…
```
names
```

Out[218…
```
['dhoni', 'kumble', 'rahul', 'sachin', 'kohli', 'ishant', ['mayank', 20]]
```

In [219…
```
names.pop(2)
```

Out[219…
```
'rahul'
```

In [220…
```
names
```

Out[220…
```
['dhoni', 'kumble', 'sachin', 'kohli', 'ishant', ['mayank', 20]]
```

In [221…
```
del names[0]
```

In [222…
```
names
```

Out[222…
```
['kumble', 'sachin', 'kohli', 'ishant', ['mayank', 20]]
```

In [223…
```
names.reverse()
```

In [224…
```
score = [10,20,30,40]
```

In [225…
```
score.sort(reverse=True)
```

In [226…
```
score
```

Out[226…
```
[40, 30, 20, 10]
```

In [227…
```
names.extend(['a','bb','ccc','ddddddddd'])
```

In [228…
```
names
```

Out[228…
```
[['mayank', 20],
 'ishant',
 'kohli',
 'sachin',
```

```
'kumble',
'a',
'bb',
'ccc',
'ddddddddd']
```

In [229…
```
def return_len(val):
    return len(val)
```

In [230…
```
return_len(names[2])
```

Out[230… 5

In [231…
```
score
```

Out[231… [40, 30, 20, 10]

In [232…
```
score.extend([5,90,12,19,15,29,56])
```

In [233…
```
score
```

Out[233… [40, 30, 20, 10, 5, 90, 12, 19, 15, 29, 56]

In [234…
```
score.sort()
```

In [235…
```
score
```

Out[235… [5, 10, 12, 15, 19, 20, 29, 30, 40, 56, 90]

In [236…
```
names[1:]
```

Out[236… ['ishant', 'kohli', 'sachin', 'kumble', 'a', 'bb', 'ccc', 'ddddddddd']

In [237…
```
sorted(names[1:],key=return_len)
```

Out[237… ['a', 'bb', 'ccc', 'kohli', 'ishant', 'sachin', 'kumble', 'ddddddddd']

In [238…
```
score
```

Out[238… [5, 10, 12, 15, 19, 20, 29, 30, 40, 56, 90]

In [239…
```
score[::2]
```

Out[239… [5, 12, 19, 29, 40, 90]

In [240…
```
score[2::2]
```

Out[240… [12, 19, 29, 40, 90]

In [241…
```
score[::2] + score[2::2]
```

Out[241… [5, 12, 19, 29, 40, 90, 12, 19, 29, 40, 90]

In [242…
```
(score[::2] + score[2::2]).count(12)
```

Out[242… 2

In [243…
```
(score[::2] + score[2::2]).count(90)
```

Out[243… 2

In [244…
```
for val in (score[::2] + score[2::2]):
    print(val)
```

```
5
12
19
29
40
90
12
19
29
40
90
```

In [332…
```
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [245…
```
for idx in range(len((score[::2] + score[2::2]))):
    print(idx)
```

```
0
1
2
3
4
5
6
7
8
9
10
```

In [246…
```
[((num/2),num) for num in range(4,41,4)]
```

Out[246…
```
[(2.0, 4),
 (4.0, 8),
 (6.0, 12),
 (8.0, 16),
 (10.0, 20),
 (12.0, 24),
 (14.0, 28),
 (16.0, 32),
 (18.0, 36),
 (20.0, 40)]
```

Transpose a MATRIX using LISTS

In [247…
```
matrix = [
    [1,2,3,4],
    [5,6,7,8],
    [9,10,11,12]
]
```

In [248…
```
matrix
```

Out[248…
```
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

In [249…
```
lst = []

for i in range(0,4):
    tmp_lst = []
    for row in matrix:
        print(row)
        print(row[i])
        print('----')
        tmp_lst.append(row[i])
    lst.append(tmp_lst)

print(lst)
print('****')
print(tmp_lst)
```

```
[1, 2, 3, 4]
1
----
[5, 6, 7, 8]
5
----
[9, 10, 11, 12]
9
----
[1, 2, 3, 4]
2
----
[5, 6, 7, 8]
6
----
[9, 10, 11, 12]
10
----
[1, 2, 3, 4]
3
----
[5, 6, 7, 8]
7
----
[9, 10, 11, 12]
11
----
[1, 2, 3, 4]
4
----
[5, 6, 7, 8]
8
----
[9, 10, 11, 12]
12
----
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
****
[4, 8, 12]
```

In [250…
```
matrix
```

Out[250…
```
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

In [251…
```
[[row[i] for row in matrix] for i in range(0,4)]
```

Out[251…
```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

In [252…
```
mixed_lst = [(1,2,3,4,5),4,5,6,7,[6,7,8,9]]
```

In [253…
```
mixed_lst
```

Out[253… `[(1, 2, 3, 4, 5), 4, 5, 6, 7, [6, 7, 8, 9]]`

In [254…
```
type(mixed_lst)
```

Out[254… `list`

In [255…
```
mixed_tuple = (12,3,4,5,[4,5,6,7],[5,6,7,8])
```

In [256…
```
mixed_tuple
```

Out[256… `(12, 3, 4, 5, [4, 5, 6, 7], [5, 6, 7, 8])`

In [257…
```
type(mixed_tuple)
```

Out[257… `tuple`

TUPLES of Single ELEMENT

In [258…
```
single_el_tup = ('rajesh',)
```

In [259…
```
single_el_tup
```

Out[259… `('rajesh',)`

In [260…
```
type(single_el_tup)
```

Out[260… `tuple`

In [261…
```
singl_el = ('rajesh',1,2,3)
```

In [262…
```
singl_el
```

Out[262… `('rajesh', 1, 2, 3)`

In [263…
```
type(singl_el)
```

Out[263… `tuple`

In [264…
```
score
```

Out[264… `[5, 10, 12, 15, 19, 20, 29, 30, 40, 56, 90]`

In [265…
```
score[0:4] * 2
```

Out[265… `[5, 10, 12, 15, 5, 10, 12, 15]`

In [266…
```
(single_el_tup * 4)
```

Out[266… `('rajesh', 'rajesh', 'rajesh', 'rajesh')`

In [267…
```
del single_el_tup[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-267-172a98499aec> in <module>
----> 1 del single_el_tup[0]

TypeError: 'tuple' object doesn't support item deletion
```

In [268…
```
score
```

Out[268…
```
[5, 10, 12, 15, 19, 20, 29, 30, 40, 56, 90]
```

In [269…
```
new_score = score[-5:-1]
```

In [270…
```
new_score
```

Out[270…
```
[29, 30, 40, 56]
```

In [271…
```
tuple(score[-9:-2])
```

Out[271…
```
(12, 15, 19, 20, 29, 30, 40)
```

In [272…
```
tuple(sorted(score[-9:-2],reverse=True))
```

Out[272…
```
(40, 30, 29, 20, 19, 15, 12)
```

SET

One important point with sets is that it used flower braces for its definition. However, by using the same flower braces dictionaries are also defined. Having said that the beauty of python is if we define key and values inside flower braces then it will treat it as a dictionary else it will treat it as a SET.

In [273…
```
set1 = {1,2,3,4,5}
```

In [274…
```
type(set1)
```

Out[274…
```
set
```

In [275…
```
set1
```

Out[275…
```
{1, 2, 3, 4, 5}
```

In [276…
```
set1.update([3,4,5,6,7,8,'rajesh'])
```

In [277…
```
set1
```

Out[277…
```
{1, 2, 3, 4, 5, 6, 7, 8, 'rajesh'}
```

In [278…
```
set1.add(10)
```

In [279…
```
set1
```

Out[279…
```
{1, 10, 2, 3, 4, 5, 6, 7, 8, 'rajesh'}
```

In [280…
```
set1.update((1,11,22), [33,44,55,66,0])
```

In [281…
```
set1
```
Out[281… {0, 1, 10, 11, 2, 22, 3, 33, 4, 44, 5, 55, 6, 66, 7, 8, 'rajesh'}

In [282…
```
set1.discard(66)
```
In [283…
```
set1
```
Out[283… {0, 1, 10, 11, 2, 22, 3, 33, 4, 44, 5, 55, 6, 7, 8, 'rajesh'}

In [284…
```
set1.remove(4)
```
In [285…
```
set1
```
Out[285… {0, 1, 10, 11, 2, 22, 3, 33, 44, 5, 55, 6, 7, 8, 'rajesh'}

In [286…
```
set1.pop()
```
Out[286… 0

In [287…
```
set1
```
Out[287… {1, 10, 11, 2, 22, 3, 33, 44, 5, 55, 6, 7, 8, 'rajesh'}

In [288…
```
set1.clear()
```
In [289…
```
set1
```
Out[289… set()

In [290…
```
A = {1,2,3,4,5}

B = {3,4,5,6,7}
```
In [291…
```
A.symmetric_difference(B)        # Unique of both the sets means only A and only B e
```
Out[291… {1, 2, 6, 7}

1. Starred Expression

In [292…
```
{*(A - B),*(B - A)}
```
Out[292… {1, 2, 6, 7}

2. Normal SET operations

In [293…
```
(A - B).union(B - A)
```
Out[293… {1, 2, 6, 7}

In [294…
```
(A.difference(B).union(B.difference(A))).intersection(A.union(B))
```
Out[294… {1, 2, 6, 7}

In [295…
```
A.union(B)
```
Out[295… {1, 2, 3, 4, 5, 6, 7}

In [296…
```
A.intersection(B)
```
Out[296… {3, 4, 5}

In [297…
```
A & B
```
Out[297… {3, 4, 5}

In [298…
```
A | B
```
Out[298… {1, 2, 3, 4, 5, 6, 7}

In [299…
```
A - B                # elements which are in A but not in B
```
Out[299… {1, 2}

In [300…
```
B - A                # elements which are in B but not in A
```
Out[300… {6, 7}

In [301…
```
A.symmetric_difference(B)
```
Out[301… {1, 2, 6, 7}

In [302…
```
A ^ B
```
Out[302… {1, 2, 6, 7}

In [303…
```
C = {3,4,5}
```

In [304…
```
C.issubset(A)
```
Out[304… True

In [305…
```
C.issuperset(B)
```
Out[305… False

In [306…
```
B.issuperset(C)
```
Out[306… True

In [307…
```
A[1:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-307-92e8a113811e> in <module>
----> 1 A[1:]

TypeError: 'set' object is not subscriptable
```

Frozen SETS

These are the fixed or unchangable SETS.

In [308…
```
AA = frozenset({1,2,3,4})
```

```
BB = frozenset({3,4,5,6})
```

In [309…

In [310…
```
AA - BB
```

Out[310…
```
frozenset({1, 2})
```

In [311…
```
BB- AA
```

Out[311…
```
frozenset({5, 6})
```

In [312…
```
AA.union(BB)
```

Out[312…
```
frozenset({1, 2, 3, 4, 5, 6})
```

In [313…
```
BB.union(AA)
```

Out[313…
```
frozenset({1, 2, 3, 4, 5, 6})
```

In [314…
```
BB.intersection(AA)
```

Out[314…
```
frozenset({3, 4})
```

In [315…
```
AA.intersection(BB)
```

Out[315…
```
frozenset({3, 4})
```

Deep v/s Shallow Copy

Deep copy -- Altogether creates a new object with the content of copied object. Both objects are independent and with no references to each other. Use .copy()

Shallow copy -- A new object is created with a reference to the memory location of the parent or copied object.

In [316…
```
csk = ['Dhoni','Ambati','Raina']
```

In [317…
```
id(csk)
```

Out[317…
```
2391335506504
```

In [318…
```
mps = csk
```

In [319…
```
id(mps), id(csk)
```

Out[319…
```
(2391335506504, 2391335506504)
```

In [320…
```
mps
```

Out[320…
```
['Dhoni', 'Ambati', 'Raina']
```

In [321…
```
new_team = csk.copy()
```

In [322…
```
id(new_team)
```

Out[322…
```
2391336591560
```

In [323…
```
new_team
```

Out[323… ['Dhoni', 'Ambati', 'Raina']

In [324…
```
csk.append('Bravo')
```

In [325…
```
csk
```

Out[325… ['Dhoni', 'Ambati', 'Raina', 'Bravo']

In [326…
```
id(csk)
```

Out[326… 2391335506504

In [327…
```
mps
```

Out[327… ['Dhoni', 'Ambati', 'Raina', 'Bravo']

In [328…
```
id(mps)
```

Out[328… 2391335506504

In [329…
```
new_team
```

Out[329… ['Dhoni', 'Ambati', 'Raina']

In [330…
```
id(new_team)
```

Out[330… 2391336591560