Module-1 :: Revision

Contents

Q1: Where Python store the memory locations of objects internally?

In [6]:
```
import ctypes
```

In [1]:
```
player1, player2 = "MSD", "Azhar"
```

In [2]:
```
id(player1), id(player2)
```

Out[2]:
```
(2873712587640, 2873712590272)
```

In [3]:
```
player3 = player1
```

In [4]:
```
id(player3)
```

Out[4]:
```
2873712587640
```

In [7]:
```
ctypes.cast(id(player1), ctypes.py_object).value, ctypes.cast(id(player3), ctypes.py
```

Out[7]:
```
('MSD', 'MSD')
```

In [8]:
```
player1 = 'A'+player1[1:]
```

In [11]:
```
id(player1), id(player2), id(player3)
```

Out[11]:
```
(2873712666752, 2873712590272, 2873712587640)
```

In [9]:
```
ctypes.cast(id(player1), ctypes.py_object).value, ctypes.cast(id(player3), ctypes.py
```

Out[9]:
```
('ASD', 'MSD')
```

So, in pure python we cannot really see the memory location data structure as `id(x)` is a Cpython implementation and such details are not exposed to the user.

Q2: Print Pyramid?

In [4]:
```
from IPython.display import Image
Image("Pattern_Scenario1.jpg",width=300,height=300)
```

Out[4]:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

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

In [76]:

```
## Problem - I:
num = 15
xi = 1
for i in range(1,num+1):
    print()
    xi += i
    for j in range(xi-i,xi):
        if j <= num:
            print(j, end=' ')

# ----------XX----------
# ----------XX----------

## Problem - II:
num = 10
next_i = 0
for i in range(1,num+1):
    print()
    for j in range(i,num+1):
        print(j, end = ' ')
for xj in range(j-1,0,-1):
    print()
    for xi in range(xj,i+1):
        print(xi,end = ' ')
```

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

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

Q3. Print the odd and even numbers from 1 to 20 in a single for loop with a tag of odd and even.

In [84]:
```
nums = [num for num in range(1,21)]
```

In [85]:
```
nums
```

Out[85]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

In [90]:
```
## Via For loop:
for num in nums:
    if num%2 == 0:
        print('{} is Even Number'.format(num))
        continue;
    else:
        print('{} is Odd Number'.format(num))
```

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

In [106…
```
odd_nums, even_nums = [num for num in nums if num%2 != 0], [num for num in nums if n
for i in odd_nums:
    print("{} is odd number".format(i))
print("\n")
for j in even_nums:
    print("{} is even number".format(j))
```

```
1 is odd number
3 is odd number
5 is odd number
7 is odd number
9 is odd number
11 is odd number
13 is odd number
15 is odd number
17 is odd number
19 is odd number


2 is even number
4 is even number
6 is even number
8 is even number
10 is even number
12 is even number
14 is even number
16 is even number
18 is even number
20 is even number
```

Q4. Print the multiple of 4 till 40 along with multiple of 2 via LIST COMPREHENSION.

In [108…
```
["4 muls {}, 2 muls {}".format(i*4,i*2) for i in range(1,11)]
```

Out[108…
```
['4 muls 4, 2 muls 2',
 '4 muls 8, 2 muls 4',
 '4 muls 12, 2 muls 6',
 '4 muls 16, 2 muls 8',
 '4 muls 20, 2 muls 10',
 '4 muls 24, 2 muls 12',
 '4 muls 28, 2 muls 14',
 '4 muls 32, 2 muls 16',
 '4 muls 36, 2 muls 18',
 '4 muls 40, 2 muls 20']
```

Q5. Print the numbers which are multiples of 4 from range 4 to 40 and which gives quotient greater than 4 after dividing 50 by the number.

In [122…
```
[num for num in range(1*4,10*4) if num%4 ==0 if 50//num >= 4]
```

Out[122…
```
[4, 8, 12]
```

Q6. Edit the tuple via LIST and STARRED Expression.

In [127…
```
tup = tuple(val for val in range(1,10))
tup
```

Out[127…
```
(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

In [141…
```
## Via Starred Expressions
new_tup = ("99",99,*tup)
print(new_tup,'\n')

## Via
print("99,99,{}".format(tup))
```

```
('99', 99, 1, 2, 3, 4, 5, 6, 7, 8, 9)

99,99,(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

Q7. Print Fibonacci Series via FOR Loop, Recursive Function and MAP Function.

In [147…
```
nums=range(0,11,1)
```

In [148…
```
nums
```

Out[148…
```
range(0, 11)
```

In [150…
```
def fibonacci(num):
    if num==0:
        return 0
    elif num==1:
        return 1
    else:
        return fibonacci(num-1) + fibonacci(num-2)
```

In [154…
```
for i in nums:
    print(fibonacci(i))
```

```
0
1
1
2
3
5
8
13
21
34
55
```

Q8. Print the Factorial of a number via LAMBDA, Recursive Function and MAP Function.

In [155…
```
from functools import reduce
```

In [160…
```
fact = lambda x,y:x*y
```

In [162…
```
reduce(fact,[1,2,3,4,5])
```

Out[162…
```
120
```

In [163…
```
reduce((lambda x, y : x*y), [i+1 for i in range(5)])
```

Out[163…
```
120
```

Recursive Function

In [166…
```
def factorial(num):
    if num <=1:
        return 1
    else:
        return num * factorial(num-1)
```

In [167…
```
factorial(5)
```

Out[167…
```
120
```

MAP Function

In [168…
```
find_facts = [5,6,7,2]
for val in find_facts:
    print(factorial(val))
```

```
120
720
5040
2
```

In [169…
```
list(map(factorial,find_facts))
```

Out[169… `[120, 720, 5040, 2]`

Q9. Print one First name with multiple last names using ARBITARY Function or STARRED Expression.

In [175…
```
def print_names(*names):
    f_name = 'Ram'
    for l_name in names:
        print(f_name,l_name)
```

In [177…
```
print_names('Sharma','Kapoor','Arora')
```

```
Ram Sharma
Ram Kapoor
Ram Arora
```

In [187…
```
list(map(print_names,('Sharma','Jolly','Lala','Pandey')))
```

```
Ram Sharma
Ram Jolly
Ram Lala
Ram Pandey
```

Out[187… `[None, None, None, None]`

Q10. Print different names using KEYWORD Arguments.

In [197…
```
def print_names(**kwargs):
    print('Ram',kwargs['l_name1'])
    print('Ram',kwargs['l_name2'])
    print('Ram',kwargs['l_name3'])
```

In [198…
```
print_names(l_name1='Sharma',l_name2='Jollu',l_name3='Titu')
```

```
Ram Sharma
Ram Jollu
Ram Titu
```

Q11. Print the LCM and HCF using FOR Loop and LIST COMPREHENSION.

In [227…
```
num1, num2 = 10,15

lcf = []
for num in range(2,max(num1,num2)+1):
    if len(lcf) == 0:
        if num1%num == 0 and num2%num == 0:
            lcf.append(num)
        else:
            print("{} is not a lowest common factor of {} and {}".format(num, num1,
    elif len(lcf) == 1:
```

```
        print('\n ### {} is a Lowest Common Factor ###'.format(lcf[-1]))
        break;
```

```
2 is not a lowest common factor of 10 and 15
3 is not a lowest common factor of 10 and 15
4 is not a lowest common factor of 10 and 15

 ### 5 is a Lowest Common Factor ###
```

In [245…
```
num1, num2 = 10,20

hcf = []
for num in range(1,min(num1,num2)+1):
    if num1%num == 0 and num2%num == 0:
        hcf.append(num)
    else:
        print("{} is not a highest common factor of {} and {}".format(num, num1, num

if len(hcf) > 1:
    print('\n ### {} is the Highest Common Factor ###'.format(hcf[-1]))
else:
    print('\n ### {} is the Highest Common Factor ###'.format(hcf[-1]))
```

```
3 is not a highest common factor of 10 and 20
4 is not a highest common factor of 10 and 20
6 is not a highest common factor of 10 and 20
7 is not a highest common factor of 10 and 20
8 is not a highest common factor of 10 and 20
9 is not a highest common factor of 10 and 20

 ### 10 is the Highest Common Factor ###
```

In [256…
```
players = {'f_name':['Virat','Sachin','Sourav','Mahendra','Parthiv'],
           'l_name':['Kohli','Tendulkar','Gangully','Dhoni','Patel']}
players
```

Out[256…
```
{'f_name': ['Virat', 'Sachin', 'Sourav', 'Mahendra', 'Parthiv'],
 'l_name': ['Kohli', 'Tendulkar', 'Gangully', 'Dhoni', 'Patel']}
```

In [257…
```
for index, name in enumerate(players):
    print(index,name)
```

```
0 f_name
1 l_name
```

In [258…
```
for index, name in enumerate(players.keys()):
    print(index,name)
```

```
0 f_name
1 l_name
```

In [259…
```
for index, name in enumerate(players.values()):
    for idx, player in enumerate(name):
        print("{} --> {}".format(idx,player))
```

```
0 --> Virat
1 --> Sachin
2 --> Sourav
3 --> Mahendra
4 --> Parthiv
0 --> Kohli
1 --> Tendulkar
2 --> Gangully
3 --> Dhoni
4 --> Patel
```

In [267…
```
"{f_name}{l_name}".format(**players)
```

Out[267…
```
"['Virat', 'Sachin', 'Sourav', 'Mahendra', 'Parthiv']['Kohli', 'Tendulkar', 'Gangull
y', 'Dhoni', 'Patel']"
```

In [268…
```
first_name = players['f_name']
last_name = players['l_name']
```

In [270…
```
list(zip(first_name,last_name))
```

Out[270…
```
[('Virat', 'Kohli'),
 ('Sachin', 'Tendulkar'),
 ('Sourav', 'Gangully'),
 ('Mahendra', 'Dhoni'),
 ('Parthiv', 'Patel')]
```

In [271…
```
for a, b in list(zip(first_name,last_name)):
    print("{} --> {}".format(a,b))
```

```
Virat --> Kohli
Sachin --> Tendulkar
Sourav --> Gangully
Mahendra --> Dhoni
Parthiv --> Patel
```

In [291…
```
with open('test_file.txt','w+') as f:
    f.write("My name is Rajesh. \n This file I have created for revision purpose.")
```

In [297…
```
with open('test_file.txt','r+') as f1:
    print(f1.tell())
    data = f1.read(4)
    print(f1.tell())
    print(f1.seek(1))
    print(f1.tell())
```

```
0
4
1
1
```

In [298…
```
data
```

Out[298…
```
'My n'
```

In [301…
```
new_dict = dict([('Raj','esh'),('Aa','rav'),('Su','resh')])
new_dict
```

Out[301…
```
{'Raj': 'esh', 'Aa': 'rav', 'Su': 'resh'}
```

In [305…
```
new_dict.items()
```

Out[305…
```
dict_items([('Raj', 'esh'), ('Aa', 'rav'), ('Su', 'resh')])
```

In [307…
```
for idx, pair in enumerate(new_dict.items()):
    print(pair)
```

```
('Raj', 'esh')
('Aa', 'rav')
('Su', 'resh')
```

In [304…
```
for i, (key, val) in enumerate(new_dict.items()):
    print(key)
    print(val)
```

```
Raj
esh
Aa
```

```
rav
Su
resh
```

In [309…
```
f2 = open('test_file.txt','r')
```

In [310…
```
f2.tell()
```

Out[310… 0

In [311…
```
f2.read(5)
```

Out[311… 'My na'

In [312…
```
f2.tell()
```

Out[312… 5

In [313…
```
f2.seek(2)
```

Out[313… 2

In [314…
```
f2.read()
```

Out[314… ' name is Rajesh. \n This file I have created for revision purpose.'

In [315…
```
f2.seek(5)
```

Out[315… 5

In [316…
```
f2.read()
```

Out[316… 'me is Rajesh. \n This file I have created for revision purpose.'

In [318…
```
f2.seek(7)
```

Out[318… 7

In [319…
```
f2.tell()
```

Out[319… 7

In [320…
```
f2.read(8)
```

Out[320… ' is Raje'

In [321…
```
import numpy as np
```

In [335…
```
np.eye(3,2),np.eye(3,3),np.diag(v=[3,2,1,4,0,0,0,9,8]),np.ones(shape=(3,3)),np.zeros
```

Out[335…
```
(array([[1., 0.],
        [0., 1.],
        [0., 0.]]),
 array([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.]]),
 array([[3, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 2, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 0, 0, 0, 0, 0, 0],
```

```
       [0, 0, 0, 4, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 9, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 8]]),
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]]),
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]]),
array([ 1.        ,  1.47368421,  1.94736842,  2.42105263,  2.89473684,
        3.36842105,  3.84210526,  4.31578947,  4.78947368,  5.26315789,
        5.73684211,  6.21052632,  6.68421053,  7.15789474,  7.63157895,
        8.10526316,  8.57894737,  9.05263158,  9.52631579, 10.        ]))
```

In [336…
```
a,b = np.eye(3,3),np.eye(3,3)
```

In [338…
```
a,b
```

Out[338…
```
(array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]]),
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]]))
```

In [339…
```
np.logical_and(a,b), np.logical_or(a,b)
```

Out[339…
```
(array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]]),
array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]]))
```

In [340…
```
a,b = np.ones(shape=(3,3)),np.eye(3,3)
```

In [341…
```
a,b
```

Out[341…
```
(array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]]),
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]]))
```

In [342…
```
np.logical_and(a,b), np.logical_or(a,b)
```

Out[342…
```
(array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]]),
array([[ True,  True,  True],
       [ True,  True,  True],
       [ True,  True,  True]]))
```

In [350…
```
A=np.array([[1,2,3,9],
            [4,5,6,0],
            [7,8,9,2]])
```

In [351…
```
A
```

Out[351…
```
array([[1, 2, 3, 9],
       [4, 5, 6, 0],
       [7, 8, 9, 2]])
```

In [352…
```
np.argmax(A),np.argmin(A) ## Return indices of max and min value in the matrix
```
Out[352… (3, 7)

In [353…
```
np.argmax(A,axis=0), np.argmin(A,axis=0)
```
Out[353… (array([2, 2, 2, 0], dtype=int64), array([0, 0, 0, 1], dtype=int64))

In [354…
```
np.argmax(A,axis=1), np.argmin(A,axis=1)
```
Out[354… (array([3, 2, 2], dtype=int64), array([0, 3, 3], dtype=int64))

In [358…
```
list(zip([1,2,3],[4,5]))
```
Out[358… [(1, 4), (2, 5)]

In [359…
```
np.linspace(1,10,2)
```
Out[359… array([1., 10.])

Implement Reduce

In [23]:
```
a = [1,2,3,4,5]
```

In [10]:
```
l = len(a)
```

In [11]:
```
l
```
Out[11]: 5

In [25]:
```
i = 0
result = []
result.append(a[i])
while i < l-1:
    tmp = result[-1] * a[i+1]
    result.append(tmp)
    i += 1
```

In [26]:
```
result
```
Out[26]: [1, 2, 6, 24, 120]

In [15]:
```
red = result[-1]
```

In [16]:
```
red
```
Out[16]: 15

In [17]:
```
from functools import reduce
```

In [24]:
```
reduce(lambda x,y:x*y,a)
```
Out[24]: 120

Implement Concatenate

In [151…
```
a = [[1,2,3,8],[4,5,6,4],[7,8,9,1],[9,0,0,9]]
```

In [152…
```
a
```

Out[152…
```
[[1, 2, 3, 8], [4, 5, 6, 4], [7, 8, 9, 1], [9, 0, 0, 9]]
```

In [153…
```
len(a)
```

Out[153…
```
4
```

In [154…
```
i=0
concatenate = []
for i in range(len(a)):
    if i == len(a)-(len(a)-1):
        break;
    for j in range(len(a[i])):
        vals = []
        for k in range(len(a)):
            vals.append(a[k][j])
        concatenate.append(vals)
    i += 1
```

In [155…
```
concatenate
```

Out[155…
```
[[1, 4, 7, 9], [2, 5, 8, 0], [3, 6, 9, 0], [8, 4, 1, 9]]
```