SET-1 Questions and Answers

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

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

In [2]:
```
nums
```

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

In [3]:
```
for num in nums:
    if num % 2 == 0:
        print("Even number :: {}".format(num))
        continue
    print("Odd number :: {}".format(num))
```

```
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
```

2. Print the odd and even numbers from 1 to 20 via LIST COMPREHENSION.

In [4]:
```
even_nums = [num for num in range(1,21) if num%2 == 0]

odd_nums = [num for num in range(1,21) if num%2 != 0]
```

In [5]:
```
even_nums, odd_nums
```

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

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

In [6]:
```
[("4's muls: {} |||| 2's muls: {}".format(num,int(num/2))) for num in range(4,41) if
```

Out[6]:
```
["4's muls: 4 |||| 2's muls: 2",
 "4's muls: 8 |||| 2's muls: 4",
 "4's muls: 12 |||| 2's muls: 6",
 "4's muls: 16 |||| 2's muls: 8",
 "4's muls: 20 |||| 2's muls: 10",
 "4's muls: 24 |||| 2's muls: 12",
 "4's muls: 28 |||| 2's muls: 14",
 "4's muls: 32 |||| 2's muls: 16",
```

```
"4's muls: 36 |||| 2's muls: 18",
"4's muls: 40 |||| 2's muls: 20"]
```

4. Print the numbers which are multiples of 4 from range 4 to 40 and which gives quotient greater than 10 after dividing 50 by the number.

In [7]:
```
[num for num in range(4,41) if num%4 == 0 if 50//num > 2]
```

Out[7]:
```
[4, 8, 12, 16]
```

In [8]:
```
50//4         # --> Floor division
```

Out[8]:
```
12
```

5. Edit the tuple via LIST and STARRED Expression.

In [9]:
```
tup1 = (2,3,4,5,6)
```

In [10]:
```
tup1
```

Out[10]:
```
(2, 3, 4, 5, 6)
```

In [11]:
```
tup1[1]
```

Out[11]:
```
3
```

In [12]:
```
tup1[3:]
```

Out[12]:
```
(5, 6)
```

In [13]:
```
tup1[3] = 7
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-e5db0fcff5df> in <module>
----> 1 tup1[3] = 7

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

Via LIST

In [14]:
```
tup1 = list(tup1)
```

In [15]:
```
tup1[3] = 999
```

In [16]:
```
tup1 = tuple(tup1)
```

In [17]:
```
tup1
```

Out[17]:
```
(2, 3, 4, 999, 6)
```

Via STARRED Expression

In [18]:
```
(*tup1,)
```

Out[18]:
```
(2, 3, 4, 999, 6)
```

In [19]:
```
name = "raman"
```

In [20]:
```
(name, *tup1)
```

Out[20]:
```
('raman', 2, 3, 4, 999, 6)
```

In [21]:
```
(*tup1, name*3)
```

Out[21]:
```
(2, 3, 4, 999, 6, 'ramanramanraman')
```

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

CASE-I

In [22]:
```
nums = range(0,11,1)
```

In [23]:
```
nums
```

Out[23]:
```
range(0, 11)
```

In [24]:
```
num0 = 0
num1 = 1
for num in nums:
    if num == 0:
        print(num)
        p_val0 = num
    elif num == 1:
        print(num1)
        p_val1 = num
    else:
        res = (p_val0 + p_val1)
        print(res)
        p_val0 = p_val1
        p_val1 = res
```

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

CASE-II

In [25]:
```
def fibonacci(num):
    return 0 if num == 0 else 1 if num == 1 else fibonacci(num-1) + fibonacci(num-2)
```

In [26]:
```
for num in nums:
    print(fibonacci(num))
```

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

CASE-III

In [27]:
```
list(map(fibonacci,nums))
```

Out[27]:
```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

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

CASE-I : LAMBDA and REDUCE

In [28]:
```
from functools import reduce
```

In [29]:
```
num = 6
```

In [30]:
```
reduce(lambda x , y : x*y, [i+1 for i in range(num)])
```

Out[30]:
```
720
```

CASE-II : RECURSIVE

In [31]:
```
def factorial(num):
    return 1 if num <=1 else num * factorial(num-1)
```

In [32]:
```
print(factorial(num))
```

```
720
```

CASE-III : MAP Function

In [33]:
```
nums_lst = [5,6,7,8]
```

In [34]:
```
facts = list(map(factorial,nums_lst))
```

In [35]:
```
["Number is {} and Factorial is {}".format(nums_lst[i], facts[i]) for i in range(len
```

Out[35]:
```
['Number is 5 and Factorial is 120',
 'Number is 6 and Factorial is 720',
 'Number is 7 and Factorial is 5040',
 'Number is 8 and Factorial is 40320']
```

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

ARBITARY Function

In [36]:
```
def display_full_name(*last_names):
    for l_name in last_names:
        print("First name is Rajesh and Last Name is {}".format(l_name))
```

In [37]:
```
display_full_name('sehwag','dhoni','dravid')
```

```
First name is Rajesh and Last Name is sehwag
First name is Rajesh and Last Name is dhoni
First name is Rajesh and Last Name is dravid
```

FOR LOOP

In [38]:
```
for name in ('sharma','dhoni','gangully'):
    print("{} {}".format('rajesh',name))
```

```
rajesh sharma
rajesh dhoni
```

```
rajesh gangully
```

MAP and LAMBDA

In [39]:
```
list(map(lambda x: 'rajesh ' + str(x),('sharma','dhoni','gangully')))
```

Out[39]:
```
['rajesh sharma', 'rajesh dhoni', 'rajesh gangully']
```

STARRED Expression

In [40]:
```
'rajesh', *('sharma','verma','churma')
```

Out[40]:
```
('rajesh', 'sharma', 'verma', 'churma')
```

9. Print different names using KEYWORD Arguments.

In [41]:
```
def display_names(**kwargs):
    for i in range(len(kwargs['f_name'])):
        print(kwargs['f_name'][i], kwargs['m_name'][i], kwargs['l_name'][i])
```

In [42]:
```
display_names(f_name=['raj'],m_name=['kumar'],l_name=['sharma'])
```

```
raj kumar sharma
```

In [43]:
```
display_names(f_name=['raj','aro'],m_name=['kumar','chumed'],l_name=['sharma','chumd
```

```
raj kumar sharma
aro chumed chumda
```

10. Print the HCF using FOR Loop and LIST COMPREHENSION.

CASE-I

In [44]:
```
num1, num2 = 10, 20
```

In [45]:
```
if num1 < num2:
    smaller = num1
else:
    smaller = num2

hcf = []
for i in range(1,smaller+1):
    if num1%i==0 & num2%i==0:
        hcf.append(i)

print("HCF of {} and {} is {}".format(num1,num2,hcf[-1]))
```

```
HCF of 10 and 20 is 10
```

In [46]:
```
hcf
```

Out[46]:
```
[1, 2, 5, 10]
```

LIST COMPREHENSION

In [47]:
```
hcf_lc = []
[hcf_lc.append(i) if num1%i==0 & num2%i==0 else 0 for i in range(1,min(num1,num2)+1)
print("HCF of {} and {} is {}".format(num1,num2,hcf_lc[-1]))
print("LCM of {} and {} is {}".format(num1,num2,hcf_lc[1]))
```

```
HCF of 10 and 20 is 10
LCM of 10 and 20 is 2
```

In [48]:
```
hcf_lc
```

Out[48]: [1, 2, 5, 10]

10.How to transpose a MATRIX using LISTS

In [49]:
```
matrix = [
    [1,2,3,4],
    [5,6,7,8],
    [9,10,11,12]
]
```

CASE-I: Using WHILE Loop

In [50]:
```
temp_matrix = []
matrix_transposed = []

j=0
while j in range(0,len(matrix[0])):
    i=0
    for row in matrix:
        print(row[j],i)
        temp_matrix.append(row[j])
        i+=1
    j+=1

j=0
for i in range(0,len(matrix[0])):
    matrix_transposed.append(temp_matrix[j:j+3])
    j += len(matrix)
```

```
1 0
5 1
9 2
2 0
6 1
10 2
3 0
7 1
11 2
4 0
8 1
12 2
```

In [51]:
```
temp_matrix
```

Out[51]: [1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 12]

In [52]:
```
matrix_transposed
```

Out[52]: [[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]

In [53]:
```
matrix
```

Out[53]: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]

CASE-II: Only FOR Loop

In [54]:
```
tmat = []
for i in range(len(matrix[0])):
    lst = []
    for row in matrix:
        print(row[i])
        lst.append(row[i])
    print('----')
    tmat.append(lst)
```

```
1
5
9
----
2
6
10
----
3
7
11
----
4
8
12
----
```

In [55]:
```
tmat
```

Out[55]:
```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

CASE-III: Via LIST COMPREHENSION

In [56]:
```
trans_mat = []
tmp_mat = [row[i] for i in range(len(matrix[0])) for row in matrix]

j=0
for i in range(0,len(matrix[0])):
    trans_mat.append(tmp_mat[j:j+3])
    j += len(matrix)
```

In [57]:
```
trans_mat
```

Out[57]:
```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

11.Perform BINARY SEARCH

In [58]:
```
import numpy as np
import pandas as pd
```

In [59]:
```
player_score = np.random.randint(29,159,11)
```

In [60]:
```
player_score
```

Out[60]:
```
array([ 66,  91,  48,  65,  59, 107,  40, 131,  65, 123,  53])
```

In [61]:
```
player_score.sort()
```

In [62]:
```
player_score
```

Out[62]:
```
array([ 40,  48,  53,  59,  65,  65,  66,  91, 107, 123, 131])
```

In [63]:
```
def binary_search(array,val):
    l = 1
    r = len(array)
    print("Entered array ", array,'\n')
    print("Value to search ", val,'\n')
    if l < r:
        arr_mid_idx = int(np.ceil(r/2))
        print("Smaller Array Mid val IDX ", arr_mid_idx,'\n')
        arr_mid_value = array[arr_mid_idx]
        print("Smaller Array Mid value ", arr_mid_value,'\n')
```

```
        if val < arr_mid_value:
            new_array = array[0:arr_mid_idx]
            binary_search(new_array,val)
        elif val > arr_mid_value:
            new_array = array[arr_mid_idx:]
            binary_search(new_array,val)
        elif val == arr_mid_value:
            print("{} found".format(val))
        else:
            print("{} not found".format(val))
    elif (l == r) & (val == array[0]):
        print("{} found".format(val))
    else:
        print("{} not found".format(val))
```

In [64]:
```
binary_search(player_score,190)
```

```
Entered array  [ 40  48  53  59  65  65  66  91 107 123 131]

Value to search  190

Smaller Array Mid val IDX  6

Smaller Array Mid value  66

Entered array  [ 66  91 107 123 131]

Value to search  190

Smaller Array Mid val IDX  3

Smaller Array Mid value  123

Entered array  [123 131]

Value to search  190

Smaller Array Mid val IDX  1

Smaller Array Mid value  131

Entered array  [131]

Value to search  190

190 not found
```

12. Find common elements in two arrays

In [65]:
```
A = np.random.randint(30, 5000, 4000)
```

In [66]:
```
B = np.random.randint(30, 5000, 4000)
```

In [67]:
```
A[0:10]
```

Out[67]:
```
array([ 458, 2892, 3244,  264, 3421, 3217, 3478, 2607, 3955, 2376])
```

In [68]:
```
B[0:10]
```

Out[68]:
```
array([4122, 2447,  353, 2477,  537, 4761, 4323, 3079, 1445,  111])
```

Approach-1

In [69]:
```
def cross_prod_tup(arr1,arr2):
    cross_prod = []
```

```
    common_elements = []
    for i in range(len(arr1)):
        for j in range(len(arr2)):
            cross_prod.append((arr1[i], arr2[j]))

    for val in cross_prod:
        if val[0] == val[1]:
            common_elements.append(val[0])
    return set(common_elements)
```

In [70]:
```
%%timeit
cross_prod_tup(A,B)
```

```
32.3 s ± 5.99 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Approach-2

In [71]:
```
set_A = set(A)
set_B = set(B)
```

In [72]:
```
%%timeit
list(filter(lambda x : x in set_A,set_B))
```

```
1.69 ms ± 223 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

Approach-3

In [73]:
```
%%timeit
set_A.intersection(B)
```

```
2.25 ms ± 134 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Approach-4

In [74]:
```
%%timeit
set_A & set_B
```

```
448 µs ± 98.3 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

13. HASH in Python

HASH in detail: https://www.journaldev.com/17357/python-hash-function

In [75]:
```
hash(3487364)
```

Out[75]:
```
3487364
```

In [76]:
```
hash(189090213902949847832748234)
```

Out[76]:
```
654238644410231826
```

In [5]:
```
hash('My name is Rajesh SHarma')
```

Out[5]:
```
6948892566641039714
```

In [2]:
```
hash('My name is Rajesh Sharma')
```

Out[2]:
```
-2921062826119943623
```

In [3]:
```
hash('My name is Rajesh Sharma')
```

Out[3]:
```
-2921062826119943623
```

```
%reset
```

In [80]:

```
Nothing done.
```