Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

```
NAME = "Shaurya Jaiswal"
COLLABORATORS = ""
```

1.(1 point):

Duplicate the elements of a list a given number of times. Example: dupli([a,b,c],3). X = [a,a,a,b,b,b,c,c,c]

```
def dupli(l,N):
    l1=[]
    ''' return a list with elements duplicated N times '''
    # YOUR CODE HERE
    for i in range(len(l)):
      for j in range(N):
        l1=l1+[l[i]]
    print(l1)
    raise NotImplementedError()
    raise NotImplementedError()
l=['a','b','c','e']
N=int(input("Enter times to be repeated:"))
dupli(l,N)
```

```
Enter times to be repeated:6
```

```
dupli(['a','b','c'],3)
```

```
NotImplementedError                       Traceback (most recent call last)
```

```
"""Test for dupli"""
assert(dupli(['a','b','c'],3)) == ['a', 'a', 'a', 'b', 'b', 'b', 'c', 'c', 'c']
```

2. (1 point):

Create a list containing all integers within a given range. Example: range_build(4,9). L = [4,5,6,7,8,9]

```
def range_build(n,m):
    l1=[]
    '''Create a list containing all integers within a given range(n,m)'''
    # YOUR CODE HERE
    for i in range(n,m+1):
      l1=l1+[i]
    print(l1)
    raise NotImplementedError()

n=int(input("Enter value of n:"))
m=int(input("Enter value of m:"))
range_build(n,m)
```

```
Enter value of n:4
Enter value of m:13
[4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-5-4849251e7dfa> in <module>()
     10 n=int(input("Enter value of n:"))
     11 m=int(input("Enter value of m:"))
---> 12 range_build(n,m)

<ipython-input-5-4849251e7dfa> in range_build(n, m)
      6       l1=l1+[i]
      7     print(l1)
----> 8     raise NotImplementedError()
      9
     10 n=int(input("Enter value of n:"))

NotImplementedError:
```

SEARCH STACK OVERFLOW

```
"""Test for range_build"""
assert(range_build(50,60)) == [50,51,52,53,54,55,56,57,58,59,60]
```

3.(point 1):

Determine whether two positive integer numbers are coprime.

Two numbers are coprime if their greatest common divisor equals 1.

Example:

coprime(35, 64) --> True.

coprime(48,68) --> False.

coprime(48,69) --> True.

```
def coprime(a,b):
    '''Two numbers a,b are coprime if their greatest common divisor equals 1'''
    # YOUR CODE HERE
    while(b%a!=0):
      k=a
      a=b%a
      b=k
    if(a==1):
      print("True")
    else:
      print("False")
    raise NotImplementedError()
a=int(input("Enter value of a:"))
b=int(input("Enter value of b:"))
if(a<b):
  coprime(a,b)
else:
  coprime(b,a)
```

```
Enter value of a:48
Enter value of b:64
False
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-12-3e7815145d71> in <module>()
     14 b=int(input("Enter value of b:"))
     15 if(a<b):
---> 16   coprime(a,b)
     17 else:
     18   coprime(b,a)

<ipython-input-12-3e7815145d71> in coprime(a, b)
     10     else:
     11       print("False")
---> 12     raise NotImplementedError()
     13 a=int(input("Enter value of a:"))
     14 b=int(input("Enter value of b:"))

NotImplementedError:
```

SEARCH STACK OVERFLOW

```
print(coprime(19,4))
```

```
'''Testing for numbers are coprime or not '''
assert(coprime(99,3))==False
assert(coprime(19,4))==True
```

4.(point 1):

Determine the prime factors of a given positive integer (2).
Construct a list containing the prime factors and their multiplicity.
Example:
factrlist(315) --> L = [[3,2],[5,1],[7,1]].

```
def factrlist(N):
    l2=[]
    for i in range(2,N):
      if(N%i==0):
        k=0
        for j in range(2,i):
          if(i%j==0):
            k=1
            break
        if (k==0):
          m=0
          l1=[]
          k=1
          l1=l1+[i]
          while(N%(i*i)==0):
              k+=1
              i=i*i
          l1=l1+[k]
        if(m==0):
          l2=l2+[l1]
        m=1
    print(l2)
    '''create list of prime factors of given positive integer (N)'''
    # YOUR CODE HERE

N=int(input("Enter Number:"))
factrlist(N)
raise NotImplementedError()
```

```
Enter Number:65
[[5, 1], [13, 1]]
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-28-fb1303f11d14> in <module>()
     26 N=int(input("Enter Number:"))
     27 factrlist(N)
---> 28 raise NotImplementedError()

NotImplementedError:
```

SEARCH STACK OVERFLOW

```
"""Test for factrlist"""
assert([3, 2]) in factrlist(9)
assert([3, 1]) in factrlist(15)
```

5.(point 1):

Write a Python program to remove a key from a dictionary

```
def rmkey(d,key):
    '''remove kth key from dictionary '''
   # YOUR CODE HERE
   raise NotImplementedError()



"""Tests for rmkey"""
d = {1:2, 2:3, 3:4}
assert (1 not in rmkey(d, 1).keys())



import numpy as np


def sortAscending(vector):
    # Sort vector in descending order
    # YOUR CODE HERE
    raise NotImplementedError()



"""Test for sortAscending"""


def classwiseAverage(X, Y):
  '''
  X: NxF matrix of features
  Y: Nx1 matrix of class labels; we can have two classes [0, 1]
  Return:
    Xmean: 2xF average feature for each class
  '''
  # YOUR CODE HERE
  raise NotImplementedError()


"""Test for classwiseaVerage"""
```