Assignment 1 : Practice Questions on Python

Problem 1 : Write a function that inputs a number and prints the multiplication table of that number

In [1]:

```
def multiplicationTable():
    x = int(input('Enter a number ?'))
    for i in range(1,11) :
        ans = i * x
        print(ans)

multiplicationTable()
```

```
Enter a number ?5
5
10
15
20
25
30
35
40
45
50
```

Problem 2 : Write a program to print twin primes less than 1000. If two consecutive odd numbers are both prime then they are known as twin primes

In [2]:

```
import math
import math
def checkPrime(x):
    # we are checking upto square root of x  to reduce number of iterations
    for i in range(2,math.floor(math.sqrt(x))):
        if x % i == 0:
            return False
    return True

def printTwinPrimes():
    i = 3
    while i <= 1000:
        if checkPrime(i) == True and checkPrime(i+2) == True:
            print(i,i+2)
        i += 2

printTwinPrimes()
```

```
3 5
5 7
7 9
9 11
11 13
13 15
15 17
17 19
23 25
29 31
35 37
41 43
47 49
59 61
71 73
101 103
```

```
107 109
137 139
149 151
167 169
179 181
191 193
197 199
227 229
239 241
269 271
281 283
311 313
347 349
359 361
419 421
431 433
461 463
521 523
569 571
599 601
617 619
641 643
659 661
809 811
821 823
827 829
839 841
857 859
881 883
```

Problem 3 : Write a program to find out the prime factors of a number.

In [20]:

```
import math
def primeFactors(x):
    #part 1
    # Print the number of 2s that divide x
    while x % 2 == 0:
        print(2)
        x //= 2

    #part 2
    # As n is now odd we only check for odd numbers form now on.
    i = 3
    while i <= math.floor(math.sqrt(x)):
        while x % i == 0:
            print(i)
            x = x // i
        i += 2

    #part 3
    # If x is a prime number this code will handle it as part 2 only chekcs upto square root of x
    if x > 2:
        print(x)

primeFactors(13)
```

```
13
```

Problem 4: Write a program to implement these formulae of permutations and combinations

In [7]:

```
import math
def permutation(n,r):
    ans = math.factorial(n) // math.factorial(n-r)
    return ans

def combination(n,r):
    ans = permutation(n,r) // math.factorial(r)
```

```
    return ans

print(permutation(5,3))
print(combination(7,4))
```

```
60
35
```

Problem 5: Write a function that converts a decimal number to binary number

In [6]:

```
def decToBin(n):
    # we use shift operators here
    # If we right shift the given number we get the last bit then print 1 or 0 accordingly
    # assume int size is 64bits
    # returns the converted binary number as a string
    ans = ''
    for i in range(1,65):
        x = n >> 64 - i
        ans = ans + str(x & 1)
    return ans


print(decToBin(521))
```

```
0000000000000000000000000000000000000000000000000000001000001001
```

Problem 6 : Write a function cubesum() that accepts an integer and returns the sum of the cubes of individual digits of that number. Use this function to make functions PrintArmstrong() and isArmstrong() to print Armstrong numbers and to find whether is an Armstrong number.

In [5]:

```
def cubeSum(x):
    # we are doing x modulo 10 to get its last digit and then integer divide by 10 to make a right
shift in decimal form
    sum = 0
    while x > 0 :
        d = x % 10
        sum += d ** 3
        x = x // 10
    return sum

print(cubeSum(123))

def isArmstrong(x):
    return cubeSum(x) == x

def PrintArmstrong(n):
    for i in range(1,n+1):
        if isArmstrong(i):
            print(i)

PrintArmstrong(1000)
```

```
36
1
153
370
371
407
```

Problem 7 : Write a function prodDigits() that inputs a number and returns the product of digits of that number

In [13]:

```
def prodDigits(x):
    prod = 1
    while x > 0 :
        d = x % 10
        prod *= d
        x = x // 10
    return prod

x = int(input('Enter a number: '))
print(prodDigits(x))
```

```
Enter a number: 123
6
```

Problem 8 : Finding Multiplicative Digital Root and Persistance

In [19]:

```
# Has a dependency on problem 7 run problem 7 first

def MDRP(x):
    i = 0
    # to check if the product has reached to a one digit number we use modulo 10
    while x % 10 != x :
        x = prodDigits(x)
        i = i + 1
    return (x,i)


def MDR(x):
    ans = MDRP(x)
    return ans[0]

def MPersistence(x):
    ans = MDRP(x)
    return ans[1]

print(MDR(86))
print(MPersistence(86))
```

```
6
3
```

Problem 9 : Write a function sumPdivisors() that finds the sum of proper divisors of a number.

In [35]:

```
def sumPdivisors(x):
    sum = 0
    for i in range(1,x):
        if x % i == 0:
            sum += i
    return sum

print(sumPdivisors(6))
```

```
6
```

Problem 10 : Write a program to print all the perfect numbers in a given range

In [37]:

```
# has a dependency on problem 9
def printPerfectNumbers(x):
```

```
    for i in range(1,x+1):
        sumation = sumPdivisors(i)
        if i == sumation:
            print(i)

printPerfectNumbers(100)
```

```
6
28
```

Problem 11 : Write a program to find amicable numbers

In [45]:

```
def isAmicableNumbers(x,y):
    sumPx = sumPdivisors(x)
    sumPy = sumPdivisors(y)
    if sumPx == y and sumPy == x:
        return True
    else:
        return False

# Here all the pairs of numbers between 1 and x has been generated

def amicableNumbers(x):
    for i in range(1,x):
        for j in range(i+1,x):
            if isAmicableNumbers(i,j) == True:
                print(i,j)

print(amicableNumbers(1000))
```

```
220 284
None
```

Problem 12 : Write a program which can filter odd numbers in a list by using filter function

In [49]:

```
def isOdd(x):
    return x % 2 == 1

def filterOdd(x):
    return list(filter(isOdd,x))

def filterOddLambda(x):
    return list(filter(lambda i : i % 2 == 1,x))

print(filterOdd([1,2,3,10,17,20,34]))
print(filterOddLambda([1,2,3,10,17,20,34]))
```

```
[1, 3, 17]
[1, 3, 17]
```

Problem 13: Write a program which can map() to make a list whose elements are cube of elements in a given list

In [51]:

```
def MapToCube(x):
    return list(map(lambda i : i ** 3,x))

print(MapToCube([1,2,3,10,17,20,34]))
```

```
[1, 8, 27, 1000, 4913, 8000, 39304]
```

Problem 14 : Write a program which can map() and filter() to make a list whose elements are cube of even number in a given list

In [57]:

```
def MapToCubeAfterFilter(x):
    return list(
        map(lambda y : y **3 ,
        filter(lambda i : i % 2 == 0,x)))

print(MapToCubeAfterFilter([1,2,3,10,17,20,34]))
```

```
[8, 1000, 8000, 39304]
```

In [ ]:

```

```

In [ ]:

```

```