Practice Questions On Functions

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

In [3]:
```
def multiply(x):
    for i in range(1,11):
        print(f'{x} * {i} = {x*i}')


multiply(5)
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

1. 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 [5]:
```
def prime(x):
    i=[]
    for j in range(x):
        c=0
        # to check if the given number 'j' is prime number or not.
        for n in range(2,j):
            if j%n==0:
                c=1
                break
        #if 'j' satisfy the condition , then 'j' will be append to list 'i'
        if c==0:
            i.append(j)
    l=[]

    # to check two consecutive odd numbers are both prime
    for k in i:
        if len(l)<=1:
            l.append(k)
        elif len(l)==2:
            a,b=l
            # if two consecutive odd numbers are both prime, then it print the numbers in tuple
            if a+2==b:
                l=[]
                print(f'({a},{b})',end=' ')
                l.append(k)
            # if two consecutive odd numbers are not prime, then it empty the list append the 'b'
            # next 'k' values to list 'l'
            else:
                l=[]
                l.append(b)
                l.append(k)

        else:
            print('')

prime(1000)
```

```
(3,5) (11,13) (17,19) (29,31) (41,43) (59,61) (71,73) (101,103) (107,109) (137,139) (149,151) (1
```

```
79,181) (191,193) (197,199) (227,229) (239,241) (269,271) (281,283) (311,313) (347,349) (419,42
1) (431,433) (461,463) (521,523) (569,571) (599,601) (617,619) (641,643) (659,661) (809,811) (82
1,823) (827,829) (857,859) (881,883)
```

1. Write a program to find out the prime factors of a number. Example: prime factors of 56 - 2, 2, 2, 7

In [6]:
```
def prime_factor(x):
    a=x
    l=[]
    i=2
    while i<=x:
        if x%i==0:
            l.append(i)
            x=x/i
            i=2
        else:
            i=i+1
    print('the prime factor of {} is {}'.format(a,l))


prime_factor(56)
```

```
the prime factor of 56 is [2, 2, 2, 7]
```

1. Write a program to implement these formulae of permutations and combinations. Number of permutations of n objects taken r at a time: p(n, r) = n! / (n-r)!. Number of combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p(n,r) / r!

In [2]:
```
def permutation_combination(x,r):
    #this function return factorial of 'b'
    def factorial(b):
        a=1
        for i in range(1,b+1):
            a=a*i
        return a

    y=factorial(x)/factorial(x-r)
    z=y/factorial(r)
    print('the permutation of {} and {} is {}'.format(x,r,y))
    print('the combination of {} and {} is {}'.format(x,r,z))


permutation_combination(10,5)
```

```
the permutation of 10 and 5 is 30240.0
the combination of 10 and 5 is 252.0
```

1. Write a function that converts a decimal number to binary number

In [3]:
```
def decimal_to_binary(x):
    z=x
    l=[]
    i=2
    while x>=2:
        a=x%i
        l.append(a)
        x=x//i
        i=2
    if z == 0:
        l.append(0)
    else:
        l.append(1)

    l=list(reversed(l))
    print('the prime factor of {} is {}'.format(z,l))
```

```
decimal_to_binary(11)
```

```
the prime factor of 11 is [1, 0, 1, 1]
```

1. 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 [4]:
```
def PrintArmstrong(number):

    #this function returns the sum of the cubes of individual digits of that number
    def cubesum(num):
        li = [int(x) for x in str(num)]
        sum=0
        for i in li:
            sum=sum+i*i*i
        return sum

    #this function print Armstrong numbers and to find whether is an Armstrong number.
    def isArmstrong(numb):
        b=numb
        a=cubesum(number)
        if a==b:
            return 'the given number {} is Armstrong number'.format(b)
        else:
            return 'the given number {} is not Armstrong number'.format(b)

    print(isArmstrong(number))



PrintArmstrong(153)
```

```
the given number 153 is Armstrong number
```

1. Write a function prodDigits() that inputs a number and returns the product of digits of that number.

In [5]:
```
def prodDigits(num):
    li = [int(x) for x in str(num)]
    product=1
    for i in li:
        product=product*i
    print('the product of digits of number {} is {}'.format(num,product))

prodDigits(123)
```

```
the product of digits of number 123 is 6
```

1. If all digits of a number n are multiplied by each other repeating with the product, the one digit number obtained at last is called the multiplicative digital root of n. The number of times digits need to be multiplied to reach one digit is called the multiplicative persistance of n. Example: 86 -> 48 -> 32 -> 6 (MDR 6, MPersistence 3) 341 -> 12->2 (MDR 2, MPersistence 2) Using the function prodDigits() of previous exercise write functions MDR() and MPersistence() that input a number and return its multiplicative digital root and multiplicative persistence respectively

In [13]:
```
def MDR(num):

    Mper = []
    def MPersistence(l):
        persistant=0
        while len(l)!=1:
```

```
            s = [str(i) for i in l]
            r = int("".join(s))
            c=prodDigits(r)
            persistant=persistant+1
            l = [int(x) for x in str(c)]
        return c,persistant

    def prodDigits(n):
        li = [int(x) for x in str(n)]
        product=1
        for i in li:
            product=product*i
        return product

    l = [int(x) for x in str(num)]

    if len(l)==1:
        a=num
        print('MDR is {}, MPersistence is 0'.format(a))

    else:
        Mper = MPersistence(l)
        print(f'MDR is {Mper[0]}, MPersistence is {Mper[1]}')




MDR(341)
```

```
MDR is 2, MPersistence is 2
```

1. Write a function sumPdivisors() that finds the sum of proper divisors of a number. Proper divisors of a number are those numbers by which the number is divisible, except the number itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 12, 18

In [14]:
```
def sumPdivisors(num):
    l=[]
    for i in range(1,num):
        if num%i==0:
            l.append(i)
    print(l)

sumPdivisors(36)
```

```
[1, 2, 3, 4, 6, 9, 12, 18]
```

1. A number is called perfect if the sum of proper divisors of that number is equal to the number. For example 28 is perfect number, since 1+2+4+7+14=28. Write a program to print all the perfect numbers in a given range

In [15]:
```
def perfect(num):
    l=[]
    for i in range(1,num):
        if num%i==0:
            l.append(i)

    sum=0
    for i in l:
        sum=sum+i

    if sum==num:
        print('the given number {} is perfect number'.format(num))

    else:
        print('the given number {} is not perfect number'.format(num))
```

```
perfect(28)
```

```
the given number 28 is perfect number
```

11.Two different numbers are called amicable numbers if the sum of the proper divisors of each is equal to the other number. For example 220 and 284 are amicable numbers. Sum of proper divisors of 220 = 1+2+4+5+10+11+20+22+44+55+110 = 284 Sum of proper divisors of 284 = 1+2+4+71+142 = 220 Write a function to print pairs of amicable numbers in a range

In [20]:
```
def amicable(x):

    def sumof(num):
        l=[]
        for i in range(1,num):
            if num%i==0:
                l.append(i)
        sum=0
        for i in l:
            sum=sum+i
        return sum


    for i in range(2,x+1):
        a=sumof(i)
        b=sumof(a)
        if i==b and b<=x:
            print(f'({i},{a})',end=' ')


amicable(2000)
```

```
(6,6) (28,28) (220,284) (284,220) (496,496) (1184,1210) (1210,1184)
```

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

In [21]:
```
l=[10,53,105,450,64]
odd_number = list(filter(lambda y: (y%2 != 0) , l))
print(odd_number)
```

```
[53, 105]
```

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

In [22]:
```
l=[1,2,3,4,5]
cube = list(map(lambda x:x*x*x,l))
print(cube)
```

```
[1, 8, 27, 64, 125]
```

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 [23]:
```
l=[1,2,3,4,5]
even_number = list(filter(lambda y: (y%2== 0) , l))
cube = list(map(lambda x:x*x*x,even_number))
print(cube)
```

```
[8, 64]
```