Practice Questions On Functions

Exercise :

1. Write a function that inputs a number and prints the multiplication table of that number
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.
3. Write a program to find out the prime factors of a number. Example: prime factors of 56 -2, 2, 2, 7
4. 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!
5. Write a function that converts a decimal number to binary number
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.
7. Write a function prodDigits() that inputs a number and returns the product of digits of that number.
8. 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
9. 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, 18
10. 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
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
12. Write a program which can filter odd numbers in a list by using filter function
13. Write a program which can map() to make a list whose elements are cube of elements in a given list
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 [1]:

```python
# 1) Write a function that inputs a number and prints the multiplication table of that number

import sys
import math

def multiplicationTable():
    '''
    This function prints multiplication table.
    '''
    try:
        print(multiplicationTable.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number: "))
        if validateNumber(number):
            for index in range(1,11):
                print("{0} X {1} = {2}".format(number,index,number*index))
    except:
        print("Unexpected error occured.",sys.exc_info()[0])
    finally:
        print("\n****End of result****")


def validateNumber(number):
    '''
    This function perform checks for positive whole number and returns 'True' if passes,
    else returns 'False'.
    '''
    try:
        if number <= 0:
            raise ValueError("Zero and negative numbers are not allowed.")
        return True
    except ValueError as e:
        print(e)
    return False

multiplicationTable()
```

```
    This function prints multiplication table.

****Start of result****

Enter a number: 5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50

****End of result****
```

In [2]:

```python
# 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.

def isPrimeNumber(number):
    '''
    This function returns true if number is prime, otherwise returs false.
    '''
    if(number % 2 == 0):
        return False
    squareRootOfNumber = int((math.sqrt(number)))
    divisorList = [x for x in range(2,(squareRootOfNumber+1)) if x % 2 != 0]
    for divisor in divisorList:
        if(number % divisor == 0):
            return False

    return True



def twinNumbers():
    '''
    This function prints twin prime numbers from 1 to 1000.
    '''
    try:
        print(twinNumbers.__doc__)
        print("****Start of result****\n")
        print("Twin Numbers : \n")
        twinList = []
        twinList.append(3)
        for number in range(5,998):
            if(isPrimeNumber(number)):
                twinList.append(number)
            else:
                continue

            if(len(twinList) == 1):
                continue
            elif(len(twinList) == 2 and abs(twinList[0] - twinList[1]) == 2):
                print("( {0} , {1} )".format(twinList[0],twinList[1]))

            del twinList[0]

    except:
        print("Unexpected error occured",sys.exc_info()[0])
    finally:
        print("\n****End of result****")


twinNumbers()
```

```
    This function prints twin prime numbers from 1 to 1000.

****Start of result****

Twin Numbers :

( 3 , 5 )
( 5 , 7 )
( 11 , 13 )
( 17 , 19 )
( 29 , 31 )
( 41 , 43 )
( 59 , 61 )
( 71 , 73 )
( 101 , 103 )
( 107 , 109 )
( 137 , 139 )
( 149 , 151 )
( 179 , 181 )
( 191 , 193 )
( 197 , 199 )
( 227 , 229 )
( 239 , 241 )
( 269 , 271 )
( 281 , 283 )
( 311 , 313 )
( 347 , 349 )
( 419 , 421 )
( 431 , 433 )
( 461 , 463 )
( 521 , 523 )
( 569 , 571 )
( 599 , 601 )
( 617 , 619 )
( 641 , 643 )
( 659 , 661 )
( 809 , 811 )
( 821 , 823 )
( 827 , 829 )
( 857 , 859 )
( 881 , 883 )

****End of result****
```

In [3]:
```python
# 3) Write a program to find out the prime factors of a number.
#    Example: prime factors of 56 - 2, 2, 2, 7


def listOfPrimeNumbers(number):
    '''
    This function returns the list of primes <= number/2
    '''
    listOfPrimes = []
    if isPrimeNumber(number) :
        listOfPrimes.append(number)
        return  listOfPrimes

    listOfPrimes.append(2)
    listOfPrimes.extend([x for x in range(2,(int(number/2)+1)) if x % 2 != 0 if isPrimeNumber(x)])

    return listOfPrimes



def primeFactors():
    '''
    This function prints prime factors of a number
    '''
    try :
        print(primeFactors.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number :"))
        holder = number
        result = []
        if(validateNumber(number)):
            if number == 1:
                print("Value should be greater than 1")
                return
            listOfPrimes = listOfPrimeNumbers(number)
            while number >= 1 :
                for i in listOfPrimes[::-1] :
                    if number % i == 0 :
                        number = int(number / i)
                        result.append(i)
                        break
                if number == 1:
                    print("Prime factorization of {0} is ".format(holder))
                    result.sort()
                    print(*result, sep=' X ')
                    break
                else :
                    listOfPrimes = [y for y in listOfPrimes if y <= number]
    except :
        print("Unexpected error occured",sys.exc_info()[0])
    finally :
         print("\n****End of result****")


primeFactors()
```

```
    This function prints prime factors of a number

****Start of result****

Enter a number :56
Prime factorization of 56 is
2 X 2 X 2 X 7

****End of result****
```

In [4]:
```
# 4) 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!

def factorial(number):
    '''
        This function returns factorial of the number
    '''
    if number == 1 or number == 0:
        return 1
    return number * factorial(number - 1)

def permutaionAndCombination():
    '''
        This function prints permutation and combination of n and r such that n >= r >= 0
    '''
    try:
        print(permutaionAndCombination.__doc__)
        print("****Start of result****\n")
        n = int(input("Enter a value for n : "))
        r = int(input("Enter a value for r : "))
        if r > n :
            print("please enter n >= r >= 0")
            return
        if validateNumber(n) and validateNumber(r) :
            nPr = factorial(n) / factorial(n-r)
            nCr = nPr / factorial(r)
            print("Number of permutations of {0} objects taken {1} at a time : p({0}, {1}) is {2}".for
mat(n,r,nPr))
            print("Number of combinations of {0} objects taken {1} at a time : c({0}, {1}) is {2}".for
mat(n,r,nCr))
    except:
        print("Unexpected error occured",sys.exc_info()[0])
    finally:
         print("\n****End of result****")


permutaionAndCombination()
```

```
        This function prints permutation and combination of n and r such that n >= r >= 0

****Start of result****

Enter a value for n : 4
Enter a value for r : 2
Number of permutations of 4 objects taken 2 at a time : p(4, 2) is 12.0
Number of combinations of 4 objects taken 2 at a time : c(4, 2) is 6.0

****End of result****
```

In [5]:
```python
# 5) Write a function that converts a decimal number to binary number

import sys

def decimalToBinary():
    '''
        This function performs Decimal to Binary conversion of a given number.
    '''
    try:
        print(decimalToBinary.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number: "))
        holder = number
        if number < 0 :
            raise ValueError("Negative numbers are not allowed.")
        result = []
        if number == 0 or number == 1 :
            result.append(number)
            print("Decimal to Binary conversion of {0} is {1}".format(number,result[0]))
            return
        while True :
            result.append(int(number % 2))
            number = int(number / 2)
            if number <= 1:
                result.append(number)
                break
        result.reverse()
        print("Decimal to Binary conversion of {0} is".format(holder))
        print(*result, sep='')


    except ValueError as valueError:
        print(valueError)
    except:
        print("Unexpected error occured",sys.exc_info()[0])
    finally:
        print("\n****End of result****")


decimalToBinary()
```

```
        This function performs Decimal to Binary conversion of a given number.

****Start of result****

Enter a number: 196
Decimal to Binary conversion of 196 is
11000100

****End of result****
```

In [6]:
```python
# 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.

import sys

def cubeSum(number):
    '''
        This function returns the sum of the cubes of individual digits of that number.
    '''
    totalSum = 0
    while True:
        remainder = int(number % 10)
        number = int(number / 10)
        totalSum = totalSum + remainder**3
        if number < 10 :
            totalSum = totalSum + number**3
            break
    return totalSum

def exponentiationSum(number):
    '''
        This function returns the sum of the exponentiation of individual digits of that number.
    '''
    totalSum = 0
    holder = number
    while True:
        remainder = int(number % 10)
        number = int(number / 10)
        totalSum = totalSum + remainder**(len(str(abs(holder))))
        if number < 10 :
            totalSum = totalSum + number**(len(str(abs(holder))))
            break
    return totalSum

def isArmstrong(exponentiationSumOfNumber , number):
    '''
        This function returns True if number is Armstrong otherwise returns False.
    '''
    return exponentiationSumOfNumber == number

def printArmstrong(number):
    '''
        This function prints Armstrong number.
    '''
    print("{0} is a Armstrong number.".format(number))


def armStrong():
    '''
        This function prints whether a given number is Armstrong or not.
    '''
    try:
        print(armStrong.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number : "))
        if number < 0:
            raise ValueError("Negative numbers are not allowed.")

        exponentiationSumOfNumber = 0
        if (len(str(abs(number)))) == 3 :
            exponentiationSumOfNumber = cubeSum(number)
        else :
            exponentiationSumOfNumber = exponentiationSum(number)
        if isArmstrong(exponentiationSumOfNumber,number) :
            printArmstrong(number)
        else:
            print("{0} is not a Armstrong number.".format(number))
    except ValueError as valueError:
        print(valueError)
    except:
        print("Unexpected error occured",sys.exc_info()[0])
    finally:
        print("\n****End of result****")



armStrong()
```

```
        This function prints whether a given number is Armstrong or not.

****Start of result****

Enter a number : 371
371 is a Armstrong number.

****End of result****
```

In [7]:
```
# 7) Write a function prodDigits() that inputs a number and returns the product of digits of that
#    number.

def prodDigits(number) :
    '''
        This function returns the product of digits of a given number.
    '''
    product = 1
    while True:
        remainder = int(number % 10)
        number = int(number / 10)
        product = product * remainder
        if number < 10 :
            product = product * number
            break

    return product


def calcProduct():
    '''
        This function returns the product of digits of a given number.
    '''
    try:
        print(prodDigits.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number : "))
        if number <= 0:
            raise ValueError("Zero and negative numbers are not allowed.")

        if number < 10 and number > 0 :
            return number

        return prodDigits(number)
    except ValueError as valueError:
        print(valueError)


print("the product of digits of a given number is {0}".format(calcProduct()))
print("\n****End of result****")
```

```
        This function returns the product of digits of a given number.

****Start of result****

Enter a number : 123
the product of digits of a given number is 6

****End of result****
```

In [8]:
```python
# 8) 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

def isMultiplicativeDigitalRoot(number) :
    '''
        This function returns the boolean value True if sum of remiander and quotient of a given numbe
r is <= 6, othersise returns False.
    '''
    remainder = int(number % 10)
    quotient = int(number / 10)
    return (remainder + quotient) <= 6


def MDR(number):
    '''
        This function returns MDR value of a given number.
    '''
    remainder = int(number % 10)
    quotient = int(number / 10)
    return remainder * quotient

def MPersistence(counter):
    '''
        This function increments given value by 1.
    '''
    return counter + 1


def calcMPersistence_MDR():
    '''
        This function prints multiplicative digital root and multiplicative persistence value of a giv
en number.
    '''
    try:
        print(calcMPersistence_MDR.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number : "))
        if number <= 0:
            raise ValueError("Zero and negative numbers are not allowed.")

        if number < 10 and number > 0 :
            return print("{0} (MDR {0}, MPersistence 0)".format(number))

        product = 0
        counter = MPersistence(0)
        if isMultiplicativeDigitalRoot(number):
            product = MDR(number)
        else:
            product = prodDigits(number)

        print("{0}".format(number),end=' -> ')
        print("{0}".format(product),end=' -> ')
        while True:
            if isMultiplicativeDigitalRoot(product) :
                print("{0} (MDR {0}, MPersistence {1})".format(MDR(product),counter),end='')
                break
            product = prodDigits(product)
            counter += MPersistence(0)
            print("{0}".format(product),end=' -> ')

    except ValueError as valueError:
        print(valueError)
    finally:
        print("\n\n****End of result****")


calcMPersistence_MDR()
```

```
        This function prints multiplicative digital root and multiplicative persistence value of a giv
en number.

****Start of result****

Enter a number : 341
341 -> 12 -> 2 (MDR 2, MPersistence 1)

****End of result****
```

In [9]:
```
# 9) 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, 18

def properDivisors(number,i):
    '''
        This function returns the remainder.
    '''
    return number % i


def sumPdivisors(properDivisorsList):
    '''
        This function returns summation of given list.
    '''
    return sum(properDivisorsList)



def calcSumProperDivisors():
    '''
        This function prints proper divisors and its sum, for a given number.
    '''
    try:
        print(calcSumProperDivisors.__doc__)
        print("****Start of result****\n")
        number = int(input("Enter a number : "))
        if number <= 0:
            raise ValueError("Zero and negative numbers are not allowed.")
        if number == 1:
            print("Proper divisor of 1 is 1")
            return
        properDivisorsList = [1]
        numberBy2 = int(number/2)
        for i in range(2,(numberBy2+1)):
            if properDivisors(number,i) == 0:
                properDivisorsList.append(i)

        print("Proper divisors of {0} are {1}".format(number,str(properDivisorsList)[1:-1]))
        print("Sum of a proper divisors of {0} is {1}".format(number,sumPdivisors(properDivisorsList
)))

    except ValueError as valueError:
        print(valueError)
    finally:
        print("\n****End of result****")


calcSumProperDivisors()
```

```
        This function prints proper divisors and its sum, for a given number.

****Start of result****

Enter a number : 36
Proper divisors of 36 are 1, 2, 3, 4, 6, 9, 12, 18
Sum of a proper divisors of 36 is 55

****End of result****
```

In [10]:
```
# 10) 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

def findPerfectNumbers():
    '''
        This function prints perfect number in given range.
    '''
    try :
        print(findPerfectNumbers.__doc__)
        print("****Start of result****\n")
        start = int(input("Enter starting number of a range : "))
        end = int(input("Enter ending number of a range : "))

        if not validateNumber(start):
            return
        if not validateNumber(end) :
            return

        if start >= end :
            raise ValueError("Please provide a range, starting number should be less than ending numbe
r.")

        properDivisorsList = []
        print("All perfect number between {0} - {1} : ".format(start,end),end = ' ')
        for number in range(start,(end+1)) :
            properDivisorsList = [x for x in range(1,(int(number/2)+1)) if properDivisors(number,x) ==
 0]
            if len(properDivisorsList) > 0 and number ==  sumPdivisors(properDivisorsList):
                print("{0}".format(number),end = ' ')
            elif len(properDivisorsList) == 0 and number == end:
                print("{0}".format(None),end = ' ')
            properDivisorsList.clear()


    except ValueError as valueError:
        print(valueError)
    except :
        print("Unexpected error occured",sys.exc_info()[0])
    finally:
        print("\n\n****End of result****")


findPerfectNumbers()
```

```
        This function prints perfect number in given range.

****Start of result****

Enter starting number of a range : 1
Enter ending number of a range : 100
All perfect number between 1 - 100 :  6 28

****End of result****
```

In [11]:
```
# 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.


def findAmicableNumbers():
    '''
        This function prints amicable number in given range.
    '''
    try :
        print(findAmicableNumbers.__doc__)
        print("****Start of result****\n")
        start = int(input("Enter starting number of a range : "))
        end = int(input("Enter ending number of a range : "))

        if not validateNumber(start):
            return
        if not validateNumber(end) :
            return

        if start >= end :
            raise ValueError("Please provide a range, starting number should be less than ending numbe
r.")

        properDivisorsList = []
        properDivisorsSumDict = {}
        print("All amicable number between {0} - {1} : ".format(start,end),end = ' ')
        for number in range(start,(end+1)) :
            if number not in properDivisorsSumDict:
                properDivisorsList = [x for x in range(1,(int(number/2)+1)) if properDivisors(number,x
) == 0]
                if len(properDivisorsList) > 0:
                    temp = sumPdivisors(properDivisorsList)
                    properDivisorsList2 = [y for y in range(1,(int(temp/2)+1)) if properDivisors(temp,
y) == 0]
                    if temp >= start and temp <= end and number == sumPdivisors(properDivisorsList2) a
nd number != temp:
                        properDivisorsSumDict[temp] = number
                properDivisorsList.clear()

        if len(properDivisorsSumDict) > 0 :
            print(*((v,k) for k,v in properDivisorsSumDict.items()), sep=' ')
        else:
            print("None")
    except ValueError as valueError:
        print(valueError)
    except :
        print("Unexpected error occured",sys.exc_info()[0])
    finally:
        print("\n****End of result****")


findAmicableNumbers()
```

```
        This function prints amicable number in given range.

****Start of result****

Enter starting number of a range : 200
Enter ending number of a range : 300
All amicable number between 200 - 300 :  (220, 284)

****End of result****
```

In [12]:
```
# 12) Write a program which can filter odd numbers in a list by using filter function

print(list(filter(lambda x : x % 2 != 0,[10,2,0,5,5,3,6,4,89,78,51])))
```

```
[5, 5, 3, 89, 51]
```

In [13]:
```
# 13)  Write a program which can map() to make a list whose elements are cube of elements in a given l
ist

print(list(map(lambda x : x*3,[10,2,0,5,5,3,6,4,89,78,51])))
```

```
[30, 6, 0, 15, 15, 9, 18, 12, 267, 234, 153]
```

In [14]:
```
# 14) Write a program which can map() and filter() to make a list whose elements are cube of even numb
er in a given list

print(list(map(lambda x: x*3,filter(lambda y:y % 2 != 0,[10,2,0,5,5,3,6,4,89,78,51]))))
```

```
[15, 15, 9, 267, 153]
```