Python-Basics-Assignment I

June 24, 2020

In [1]: 1. Write a function that inputs a number and prints the multiplication table of that nu

In [1]:
```
def mulTable():
    n=int(input("Enter the number to print the multiplication table"))
    for i in range (1,11):
        print('{} * {} ={}'.format(n,i,n*i))

mulTable()
```

```
Enter the number to print the multiplication table5
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
```

In [ ]: 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]:
```
def check_prime(n):   #function to check whether the no is prime or not
    for i in range(2, n):
        if n % i == 0:
            return 0 #False
    return 1   #True


#function to generate twin prime nos
def generate_twins(low, high):
    print("Twin prime nos are: \n")
    for i in range(low, high):
        j = i + 2     #prime nos should have difference of 2 between them
```

```
                #print("Twin prime nos are: \n")
                if(check_prime(i) and check_prime(j)):   #Only True and True is True else Fals
                    print("{:d} and {:d}".format(i, j))

        start,end=map(int,input().split())  # taking two input values by map function
        generate_twins(start, end)
```

```
1 15
Twin prime nos are:

1 and 3
3 and 5
5 and 7
11 and 13
```

In [ ]: 3.Write a program to find out the prime factors of a number. Example: prime factors of

In [3]:
```
n=int(input("Enter the number to find the prime factors\n"))

        def primeFact(n):

            #divide he number fully by 2 until condition fails
            while n%2==0:
                print(2,end=' ')
                n=n//2
           #after dividing by 2 odd number will be left starting from 3
            for i in range(3,n+1):
                while n%i==0:
                    print(i,end=' ')
                    n=n//i

            #checks whether the prime is greater than 2 and the number was left from above for
            if n>2:
                print(n,sep=' ')

        primeFact(n)
```

```
Enter the number to find the prime factors
56
2 2 2 7
```

In [ ]: 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)!)

In [4]:
```
# for permutation
```

```
        def factorial(n):
            fact=1
            #if fact==1:
             #   return 1
            #else:
            for i in range(1,n+1):
                fact=fact*i
            return fact

            #formula of permutation
        def npr(n,r):
            npr=factorial(n)/factorial(n-r)
            print("Permutation of {} and {} is {}".format(n,r,npr))



        def ncr(n,r):
            ncr=factorial(n)/(factorial(r)*factorial(n-r))
            #return ncr
            print("Combination of {} and {} is {}".format(n,r,ncr))

        n,r=map(int,input("Enter value of n and r by giving space : ").split())
        npr(n,r)
        ncr(n,r)
```

```
Enter value of n and r by giving space : 4 2
Permutation of 4 and 2 is 12.0
Combination of 4 and 2 is 6.0
```

In [ ]: 5.Write a function that converts a decimal number to binary number

In [5]:
```
def dtob(num):
    binary=0
    rem=0
    i=1
    while num>0:
        rem = int(num)%2
        binary = (rem*i)+ binary       #adding the remainder on left of binary
        num = num/2
        i= i*10
    print(binary)


n=int(input("Enter number to convert it into binary: "))
                        #update i, so that on next iteration,   # rem will be added to
dtob(n)
```

```
Enter number to convert it into binary: 8
```

1000

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

In [6]:
```
def cubeSum(n):
    sum=0
    while n>0:
        rem=n%10
        sum=sum+rem*rem*rem
        n=n//10
    print(sum)

def printArmstrong(lower,upper):
    #lower = int(input("Enter lower range: "))
    #upper = int(input("Enter upper range: "))

    for num in range(lower,upper + 1):
   # initialize sum
        sum = 0

   # find the sum of the cube of each digit
        temp = num
        while temp > 0:
            digit = temp % 10
            sum += digit ** 3
            temp //= 10

        if num == sum:
            print(num)

def isArmstrong(n1):
    add=0
    temp=n1
    while temp>0:
        rem=temp%10
        add=add+rem**3
        temp=temp//10

    if add==n1:
        print("{} is an armstrong number".format(n1))
    else:
        print("{} is not an armstrong number".format(n1))

n=int(input("Enter the number to find the cube sum of its individual digits: "))
cubeSum(n)
```

```
        print("-----------------------------------------------------")
        print("For printing Armstrong numbers within range ")
        lower = int(input("Enter lower range: "))
        upper = int(input("Enter upper range: "))
        printArmstrong(lower,upper)

        print("-----------------------------------------------------")
        print("For checking whether the entered no is Armstrong or not")
        n1=int(input("Enter the number to check whether it is armstrong or not: "))
        isArmstrong(n1)
        print("-----------------------------------------------------")
```

```
Enter the number to find the cube sum of its individual digits: 123
36
-----------------------------------------------------
For printing Armstrong numbers within range
Enter lower range: 1
Enter upper range: 1000
1
153
370
371
407
-----------------------------------------------------
For checking whether the entered no is Armstrong or not
Enter the number to check whether it is armstrong or not: 143
143 is not an armstrong number
-----------------------------------------------------
```

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

In [7]:
```
def prodDigits(n):
    product=1
    while n>0:
        rem=n%10
        product=product*rem
        n=n//10
    return product

n=int(input("Enter the number to find the product of digits:  "))
ans=prodDigits(n)
print(ans)
```

```
Enter the number to find the product of digits:  142
8
```

In [ ]:
```
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 multi

Using the function prodDigits() of previous exercise write functions MDR() and
MPersistence() that input a number and return its multiplicative digital root and multi
```

In [8]:
```
def prodDigits(n):
    product=1
    while n>0:
        rem=n%10
        product=product*rem
        n=n//10
    return product

def MDR(n):
        c=prodDigits(n)
        while c>10:
            c=prodDigits(c)
        print("Multiplicative digital root: ",c)

def MPersistence(n):
        c=prodDigits(n)
        count=1
        while c>10:
            c=prodDigits(c)
            count=count+1
        print("Multiplicative persistence: ",count)
n=int(input("Enter any Number to find multiplicative digital root and multiplicative pe
MDR(n)
MPersistence(n)
```

```
Enter any Number to find multiplicative digital root and multiplicative persistence: 341
Multiplicative digital root:  2
Multiplicative persistence:  2
```

In [ ]:
```
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
For example proper divisors of 36 are 1, 2, 3, 4, 6, 9,12, 18
```

In [9]:
```
def sumPDivisors(n):
    for i in range(1,n):
        if n%i==0:
            print(i,end=' ')

num=int(input("Enter the number to find the proper divisors"))
sumPDivisors(num)
```

```
Enter the number to find the proper divisors36
1 2 3 4 6 9 12 18
```

```
In [ ]: 10. A number is called perfect if the sum of proper divisors of that number is equal to
        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 [10]: def sumPDivisors(n):
             sum=0
             for i in range(1,n):
                 if n%i==0:
                     print(i,end=' ')
                     sum=sum+i
             print()
             print("Sum of Proper Divisors: ",sum)

             if sum==n:
                 print("{} is perfect number".format(n))
             else:
                 print("{} is not a perfect number".format(n))
         num=int(input("Enter the number to find the proper divisors"))
         sumPDivisors(num)
```

```
Enter the number to find the proper divisors28
1 2 4 7 14
Sum of Proper Divisors:  28
28 is perfect number
```

```
In [ ]: 11.Two different numbers are called amicable numbers if the sum of the proper divisors
            For example 220 and 284 are amicable numbers. Sum of proper divisors of 220 = 1+2+4
            Write a function to print pairs of amicable numbers in a range
```

```
In [11]: n1=int(input('Enter number 1: '))
         n2=int(input('Enter number 2: '))
         sum1=0
         sum2=0
         for i in range(1,n1):
             if n1%i==0:
                 sum1+=i
         for j in range(1,n2):
             if n2%j==0:
                 sum2+=j
         if(sum1==n2 and sum2==n1):
             print('{} and {} are Amicable Numbers'.format(n1,n2))
         else:
             print('{} and {} are not Amicable Numbers'.format(n1,n2))
```

```
Enter number 1: 220
Enter number 2: 284
```

220 and 284 are Amicable Numbers

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

In [12]:
```
list1 = [40, 21, 6, 54, 77, 93, 9]

# iterating each number in list
for num in list1:

    # checking condition
    if num % 2 != 0:
        print(num, end = " ")
```

21 77 93 9

In [16]:
```
list1 = [40, 21, 6, 54, 77, 93, 9]

odd_nos = list(filter(lambda x: (x % 2 != 0), list1))

print("Odd numbers in the list: ", odd_nos)
```

Odd numbers in the list: [21, 77, 93, 9]

In [ ]: 13. Write a program which can map() to make a list whose elements are cube of elements

In [17]:
```
li=[]
for i in range(1,11):
    li.append(i)
print(li)
print()

cube_nos=list(map(lambda x:x**3,li))

print(cube_nos)
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]

In [ ]: 14. Write a program which can map() and filter() to make a list whose elements are cube

In [18]:
```
li = [1,2,3,4,5,6,7,8,9,10]

#first filter the even nos from list using filter and then apply map to find the cube
eve_num = list(map(lambda x: x**3, filter(lambda   x: x%2==0, li)))

print(eve_num)
#for i in eve_num:
 #   print(i,end=' ')
```

```
[8, 64, 216, 512, 1000]
```

```
In [15]: lis=[]
         l1=[]

         for i in range(1,11):
             lis.append(i)
         print(lis)


         #print even nos from list using filer function
         print("To print the even nos from list \n")
         filter_even=filter(lambda  x:x%2==0,lis)

         for i in filter_even:
             #print(i,end=' ')
             l1.append(i)
         print(l1)

         print()


         #print cube of even nos using map function
         print("To print the cube of even nos\n")
         cube=map(lambda x:x**3, l1)

         for j in cube:
             print(j,end=' ')
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
To print the even nos from list

[2, 4, 6, 8, 10]

To print the cube of even nos

8 64 216 512 1000
```

```
In [ ]:
```