assign_1_original

March 29, 2020

1 This is Assignment_01

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

[98]:
```
def mul_table(num):

    """
    This fuction prints multipliaction table of input number.
    """

    for i in range(1,11):
        print(str(num)+' * '+str(i) +' = '+ str(num*i))

n=int(input("Enter the input number. "))
mul_table(n)
```

```
Enter the input number. 174
174 * 1 = 174
174 * 2 = 348
174 * 3 = 522
174 * 4 = 696
174 * 5 = 870
174 * 6 = 1044
174 * 7 = 1218
174 * 8 = 1392
174 * 9 = 1566
174 * 10 = 1740
```

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.

[99]:
```
from math import sqrt

def check_prime(num):

    """
    This function checks whether input number is prime or not.
    """
```

```
    is_prime = False
    for i in range(2,round(sqrt(num))+1):
        if(num%i == 0):
            is_prime = True
            break;
    return not(is_prime)

lst=[]
for i in range(3,1000,2):
    if(check_prime(i)):
        if(check_prime(i+2)):
            print(" ( " + str(i) + ','+ str(i+2) + " )")
            lst.append((i,i+2))
        else:
            i=i+2
```

```
( 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 )
```

[100]:
```
print(len(lst))
```

```
35
```

- As shown in the above, number of twin primes between 1 and 1000 are 35.

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

[101]:
```
from math import sqrt

def check_prime(num):

    """
    This function checks whether input number is prime or not.
    """
    is_prime = False
    for i in range(2,round(sqrt(num))+1):
        if(num%i == 0):
            is_prime = True
            break;
    return not(is_prime)

def primefactor(num):

    """
    This function does prime factorisation of a number.
    """
    for i in range(2,round(sqrt(num))+1):

        if(num % i ==0 and check_prime(i)):

            print(str(i) + ',',end='')

            if(check_prime(num//i)):

                print( str(num//i), end=' ' )
                break;
            else:
                primefactor(num//i)
                break;
```

```
n=int(input("Enter the number for prime factorisation. \n"))
print("prime factors of", n , " are ")
primefactor(n)
```

```
Enter the number for prime factorisation.
56
prime factors of 56  are
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!

[102]:
```
def fact(num):

    """This function returns the factorial of input number."""

    if(num<=1):
        return 1

    prod=1
    for i in range(2,num+1):
        prod*=i
    return prod

def permNcomb(n,r):
    perm=(fact(n)/fact(n-r))
    comb=perm/fact(r)
    return  perm,comb

n=int(input("Enter n  "))
r=int(input("Enter r  "))

print("permutations and combinations are " + str(permNcomb(n,r)))
```

```
Enter n  10
Enter r  3
permutations and combinations are (720.0, 120.0)
```

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

[104]:
```
def dec2bin(num):
    """This function returns binary representation of a decimal number."""
    out=""
    while(num):
        out+=str(num%2)
        num=num//2
    else:
```

```
        return out[::-1]

n=int(input("Enter the number to convert into binary. "))
print(dec2bin(n))
```

```
Enter the number to convert into binary. 158
10011110
```

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.

[105]:
```
def order(num):
    """This function returns number of digits in input number"""
    n=0
    while(num):
        n+=1
        num=num//10
    else:
        return n


def cubesum(num,i):

    """This function returns the sum of cubes of digits of input number."""
    tot=0
    while(num):
        tot+=(num%10)**i
        num=num//10
    else:
        return tot

def isarmstrong(num):
    """This function tells whether number is armstrong or not."""
    if(cubesum(num,order(num))==num):
        return True
    else:
        return False


def PrintArmstrong(a,b):
    """This function prints all armstrong numbers."""

    for i in range(a,b+1):
        if(isarmstrong(i)):
            print(i)
```

```
c=int(input("Enter the number for which cubesum is to be found:: "))
print(cubesum(c,3))

a=int(input("Enter the range of armstrong numbers to be printed:" + " \n " +␣
 ↪"min:: "))
b=int(input(" max:: "))

PrintArmstrong(a,b)
```

```
Enter the number for which cubesum is to be found:: 134
92
Enter the range of armstrong numbers to be printed:
 min:: 1
 max:: 1000
1
2
3
4
5
6
7
8
9
153
370
371
407
```

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

[106]:
```
def prodDigits(num):
    """This function returns the product of digits of input number."""
    prod=1
    while(num):
        prod*=(num%10)
        num=num//10
    else:
        return prod

n=int(input("Enter input number:: "))
print(prodDigits(n))
```

```
Enter input number:: 123456
720
```

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.

[107]:
```
def MDR(num):
    """This function returns the MDR of input number."""
    count=0
    while(num>10):
        num=prodDigits(num)
        count+=1
    else:
        return (num,count)

n=int(input("Enter input number:: "))
out=MDR(n)
print("(MDR {}, Mpersistence {})".format(out[0],out[1]))
```

```
Enter input number:: 86
(MDR 6, Mpersistence 3)
```

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, 12, 18

[108]:
```
def sumPdivisors(num):

    """This function returns the sum of proper divisors of a number."""
    tot=0
    for i in range(1,num):
        if(num%i == 0):
            tot+=i
    return tot

n=int(input("Enter input number for finding sum of proper divisors:: "))
sumPdivisors(n)
```

```
Enter input number for finding sum of proper divisors:: 36
```

[108]: 55

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.

[110]:
```
def perfectnum(a,b):
    """This function prints all perfect numbers in a given range."""
    for i in range(a,b+1):
```

```
        if(sumPdivisors(i) == i):
            print(str(i))


a=int(input("Enter the range of perfect numbers to be printed:" + " \n " + "min:
 ↪: "))
b=int(input(" max:: "))
print("Perfect numbers in the given range are::")
perfectnum(a,b)
```

```
Enter the range of perfect numbers to be printed:
 min:: 1
 max:: 1000
Perfect numbers in the given range are::
6
28
496
```

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.

[111]:
```
def amicable(a,b):
    """This function prints amicable numbers in the given range."""
    for i in range(a,b+1):
        for j in range(a,b+1):
            if(sumPdivisors(i)==j and sumPdivisors(j)==i and i!=j):
                print(str(i)+"and"+str(j)+"are amicable numbers")


a=int(input("Enter the range of amicable numbers to be printed:" + " \n " +␣
 ↪"min:: "))
b=int(input(" max:: "))
print("Perfect numbers in the given range are::")
amicable(a,b)
```

```
Enter the range of amicable numbers to be printed:
 min:: 1
 max:: 300
Perfect numbers in the given range are::
220and284are amicable numbers
284and220are amicable numbers
```

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

[112]:
```
lst=range(100)
filt_odd= lambda x: x%2!=0

odd=list(filter(filt_odd,lst))
print(odd)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41,
43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81,
83, 85, 87, 89, 91, 93, 95, 97, 99]
```

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

[113]:
```
lst=range(100)
odd=list(map(lambda x: x**3,lst))
print(odd)
```

```
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 2744, 3375,
4096, 4913, 5832, 6859, 8000, 9261, 10648, 12167, 13824, 15625, 17576, 19683,
21952, 24389, 27000, 29791, 32768, 35937, 39304, 42875, 46656, 50653, 54872,
59319, 64000, 68921, 74088, 79507, 85184, 91125, 97336, 103823, 110592, 117649,
125000, 132651, 140608, 148877, 157464, 166375, 175616, 185193, 195112, 205379,
216000, 226981, 238328, 250047, 262144, 274625, 287496, 300763, 314432, 328509,
343000, 357911, 373248, 389017, 405224, 421875, 438976, 456533, 474552, 493039,
512000, 531441, 551368, 571787, 592704, 614125, 636056, 658503, 681472, 704969,
729000, 753571, 778688, 804357, 830584, 857375, 884736, 912673, 941192, 970299]
```

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

[114]:
```
lst=range(100)
odd=list(map(lambda x: x**3,filter(lambda x:x%2==0,lst)))
print(odd)
```

```
[0, 8, 64, 216, 512, 1000, 1728, 2744, 4096, 5832, 8000, 10648, 13824, 17576,
21952, 27000, 32768, 39304, 46656, 54872, 64000, 74088, 85184, 97336, 110592,
125000, 140608, 157464, 175616, 195112, 216000, 238328, 262144, 287496, 314432,
343000, 373248, 405224, 438976, 474552, 512000, 551368, 592704, 636056, 681472,
729000, 778688, 830584, 884736, 941192]
```