optional_1

November 27, 2020

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

[1]:
```
#function for printing multiplication table of input number
def multiplicationTable():
    # for taking input from user in integer format
    number = int(input())
    #for printing multiplication table for input number
    print("--Multiplication table of",number, "is--")
    for i in range(1, 11):
       print(number, 'x', i, '=', number * i )
multiplicationTable()
```

```
2
--Multiplication table of 2 is--
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

2 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

[10]:
```
def isPrime(n):
    for i in range(2,n):
        if(n%i==0):
            return False
    return True
```

```
print("Twin Primes between 1 and 1000 are : -")
for i in range(2,1000):
    if(isPrime(i) and isPrime(i+2)):
        print (i,',',i+2)
```

```
Twin Primes between 1 and 1000 are : -
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
```

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

[11]:
```
#function for printing prime factors of a number
def prime_factors(n):
    while n % 2 == 0:
        print (2),
        n = n / 2
    for i in range(3,int(n ** 0.5)+1,2):
        while n % i== 0:
            print (i),
            n = n / i
    if n > 2:
        print (n)

m = int(input()) #For taking input from user
print("Prime Factors of",m,"are - ")
prime_factors(m)
```

```
56
Prime Factors of 56 are -
2
2
2
7.0
```

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!

[12]:
```
#function for returning factorial of a number
def factorial(n):
    fact = 1;
    for i in range(2,n+1):
        fact = fact * i;
    return fact;
#function for returning Number of permutations of n objects taken r at a time
def nPr(n, r):
    pnr = factorial(n) / factorial(n - r);
    return pnr;
#function for returning Number of combinations of n objects taken r at a time
def nCr(n, r):
    return (factorial(n) / (factorial(r) * factorial(n - r)))
```

```
print("Number of permutations of 8 objects, taken 5 at a time is ",nPr(8, 5))
print("Number of combinations of 8 objects, taken 5 at a time is ",nCr(8, 5))
```

```
Number of permutations of 8 objects, taken 5 at a time is  6720.0
Number of combinations of 8 objects, taken 5 at a time is  56.0
```

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

[13]:
```
def decimalToBinary(n):
    binaryNum = [0] * n
    i = 0
    while (n > 0):
        binaryNum[i] = n % 2
        n = int(n / 2)
        i += 1
    for j in range(i - 1, -1, -1):
        print(binaryNum[j], end = "")
print("Binary number of 45 is - ",end="")
decimalToBinary(45)
```

```
Binary number of 45 is - 101101
```

6 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

[14]:
```
#function to add cube of digits
def cubesum(n):
    temp = n
    sum = 0
    while temp > 0:
        digit = temp % 10
        sum += digit ** 3
        temp //= 10
    return sum

def isArmstrong(x):
    if (x == cubesum(x)):
        return True
    else:
        return False

def PrintArmstrong(min,max):
```

```
    for i in range(min,max):
        if(isArmstrong(i)):
            print(i)
print("Armstrong number in range 1-1000 are :-")
PrintArmstrong(1,1000)
```

```
Armstrong number in range 1-1000 are :-
1
153
370
371
407
```

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

[15]:
```
# Function to get product of digits
def prodDigits(n):
    product = 1
    while (n != 0):
        product = product * (n % 10)
        n = n // 10
    return product
print("Product of digits of number 4671 is :-",prodDigits(4671))
```

```
Product of digits of number 4671 is :- 168
```

8 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

[16]:
```
def MDR(n):
    temp = str(n)
    while len(temp) > 1:
        temp = str(prodDigits(int(temp)))
    return int(temp)
```

```
print("MDR of 86 : ",MDR(86))
print("MDR of 341 : ",MDR(341))
def MPersistence(n):
    temp = str(n)
    pers = 0
    while len(temp) > 1:
        temp = str(prodDigits(int(temp)))
        pers += 1
    return pers
print("MPersistence of 86 : ",MPersistence(86))
print("MPersistence of 341 : ",MPersistence(341))
```

```
MDR of 86 :  6
MDR of 341 :  2
MPersistence of 86 :  3
MPersistence of 341 :  2
```

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

[17]:
```
def sumPdivisors(n) :
    sum = 0
    for i in range(2, int(n**0.5)+ 1) :
        if (n % i == 0) :
            if (i == int(n / i)) :
                sum = sum + i
            else :
                sum = sum + (i + int(n / i))
    return (sum + 1)
print("sum of proper divisors of 36 is :",sumPdivisors(36))
```

```
sum of proper divisors of 36 is : 55
```

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

[18]:
```
def perfectNumbers(min,max):
    for i in range(min,max):
        if(i == sumPdivisors(i)):
```

```
            print(i)
print("Perfect numbers in range 1-1000 are :")
perfectNumbers(1,1000)
```

```
Perfect numbers in range 1-1000 are :
1
6
28
496
```

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.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

[19]:
```
def amicablePairs(x, y) :

    if (sumPdivisors(x) != y) :
        return False

    return (sumPdivisors(y) == x)

def amicableNumbers(min,max):
    for i in range(min,max):
        for j in range(i+1,max):
            if(amicablePairs(i,j)):
                print(i,j)

print("Amicable number in range 1-2000 :" )
amicableNumbers(1,2000)
```

```
Amicable number in range 1-2000 :
220 284
1184 1210
```

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

[20]:
```
list1 = [101,10,1,24,2,3,4,5,9,13,22,17,99]
def Odd(x):
    if x % 2 != 0:
        return True
```

```
    else:
        return False

odd_list = filter(Odd , list1)
for x in odd_list:
    print(x)
```

```
101
1
3
5
9
13
17
99
```

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

[21]:
```
list2 = [1,2,3,4,5,6]
def cube(x):
    return x*x*x
cube_list = map(cube, list2)
for x in cube_list:
    print(x)
```

```
1
8
27
64
125
216
```

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

[22]:
```
list3 = [101,10,1,24,2,3,4,5,9,13,22,17,99]

def Even(x):
    if x % 2 == 0:
        return True
    else:
        return False

even_list = filter(Even, list3)
```

```
even_cube_list = map(cube, even_list) #using cube function made in problem 13

for x in even_cube_list:
    print(x)
```

```
1000
13824
8
64
10648
```