prabhudayala@gmail.com_op1

July 18, 2019

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

[1]:
```
def mul_table(num):
    '''
    This function computes and prints the multiplication table of a number
    '''

    for i in range(1, 11):
        print(f"{num} * {i} = {num*i}")
    return


num = int(input("Enter a Number: "))
mul_table(num)
```

```
Enter a Number: 10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

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

[2]:
```
def twin_num(lst):
    """
    This function generates twin prime number pairs
    """
    for i in lst:
        j = i+2
        if i and j in lst:
            print(f"({i}, {j})")
```

```
def prime_num(num):
    """
    This function finds out a number is prime or not
    """
    prime = True
    if num == 1:
        prime = False
    else:
        for i in range(2, num // 2):
            if num % i == 0:
                prime = False
    if prime == True:
        return num


# Lists of numbers from 1 to 1000
lst = list(range(1, 1000))
# Finds out all the odd numbers from the above list
odd_lst = list(filter(lambda x: (x % 2 != 0), lst))
# Finds out all the primes from the odd numbers
prime_lst = list(filter(prime_num, odd_lst))

print("The twin prime pairs are:")
twin_num(prime_lst)
```

```
The twin prime pairs 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- Write a program to find out the prime factors of a number. Example: prime factors of 56 - 2, 2, 2, 7

[3]:
```python
def is_prime(num):
    """
    This function returns true if a number is prime
    """
    for i in range(2, num):
        if num % i == 0:
            return False
    return True


def factor_of_no(num):
    """
    This function prints the prime factorisation of a number
    """
    # If Number is divisible by 2 then keep on deviding it by 2
    while num % 2 == 0:
        print(2, end=" ")
        num /= 2
        num = int(num)
    # If Number is not divisoble by 2 then check with prime nos. greater than 2
    for i in range(3, num+1):
        while num % i == 0:
            if is_prime(i):
                print(i, end=" ")
                num /= i
                num = int(num)
```

```
# Asks for Input
num = int(input("Enter a Number: "))
factor_of_no(num)
```

```
Enter a Number: 88
```

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!

[4]:
```
def fact(num):
    """
    This is a recursive function to find the factorial of a given number
    """
    return 1 if num <= 1 else (num * fact(num-1))


def pnr(n, r):
    """
    Computes P(n,r) given n and r values
    """
    pnr = fact(n) / fact(n-r)
    return pnr


def cnr(n, r):
    """
    Computes C(n, r) given n and r values
    """
    cnr = pnr(n, r) / fact(r)
    return cnr


# Asks for user Input
n = int(input("Enter Value for n: "))
r = int(input("Enter Value for r: "))
print(f"P({n},{r}) = {pnr(n, r)}")
print(f"C({n},{r}) = {cnr(n, r)}")
```

```
Enter Value for n: 5
Enter Value for r: 2
P(5,2) = 20.0
C(5,2) = 10.0
```

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

[5]:
```
def dec_to_bin(num):
    """
    This function converts Decimal to Binary
```

```
    """
    bin_no = []
    while num:
        while num % 2 == 0:
            num /= 2
            bin_no.append(0)
        while num % 2 != 0:
            num = num // 2
            bin_no.append(1)
    for ele in reversed(bin_no):
        print(ele, end=" ")


# Asks for user Input
num = int(input("Enter a Decimal Number: "))
dec_to_bin(num)
```

```
Enter a Decimal Number: 8
```

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.

[6]:
```
from functools import reduce


def cubesum(num):
    """
    This function returns the sum of the cubes of individual digits of a number
    """
    # Converting the number to an Integer List
    num_list = list(map(int, str(num)))
    # Creating a list cubenum by taking cube of each element in the Integer␣
 ↪List
    cubenum = list(map(lambda x: x**3, num_list))
    # Finding the cube sum of the number
    sum = reduce(lambda x, y: x + y, cubenum)
    return sum


def isArmstrong(num):
    """
    This function checks if a number is Armstrong Number or not
    """
    if num == cubesum(num):
        return True
    return False
```

```
def PrintArmstrong(num1, num2):
    """
    This function prints Armstrong Numbers on a range
    """
    for num in range(num1, num2+1):
        if isArmstrong(num):
            print(num)


# Asks for user Input
num1 = int(input("Enter a Starting Range: "))
num2 = int(input("Enter a Ending Range: "))
print(f"Armstrong Numbers from {num1} to {num2}: ")
PrintArmstrong(num1, num2)
```

```
Enter a Starting Range: 45
Enter a Ending Range: 1000
Armstrong Numbers from 45 to 1000:
153
370
371
407
```

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

[7]:
```
from functools import reduce


def prodDigits(num):
    """
    This function returns the product of digits of a Number
    """
    # Converting Number to a list of digits
    list_num = list(map(int, str(num)))
    # Calculating product of digits
    prod = reduce(lambda x, y: x * y, list_num)
    return prod


# Asks for user Input
num = int(input("Enter a Number: "))
print(f"Product of digits of the Number {num} = {prodDigits(num)}")
```

```
Enter a Number: 456
Product of digits of the Number 456 = 120
```

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

[8]:
```
from functools import reduce


def prodDigits(num):
    """
    This function returns the product of digits of a Number
    """
    list_num = list(map(int, str(num)))
    product = reduce(lambda x, y: x * y, list_num)
    return product


def MDR(num):
    """
    This function returns the Multiplicative Digital Root (MDR) of a Number
    """
    while num // 10 != 0:
        print(num, end=" > ")
        num = prodDigits(num)
    print(num, end="\n")
    return num


def MPersistence(num):
    """
    This function returns the multiplicative persistance of a Number
    """
    count = 0
    while num // 10 != 0:
        num = prodDigits(num)
        count += 1
    return count


# Asks for user Input
num = int(input("Enter a Number: "))
print(f"MDR of {num} = {MDR(num)}\nMpersistence of {num} = {MPersistence(num)}")
```

```
Enter a Number: 45
45 > 20 > 0
MDR of 45 = 0
```

```
Mpersistence of 45 = 2
```

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

[10]:
```
def sumPdivisors(num):
    """
    This function returns the sum of proper divisors of a number
    """
    sum_of_div = 0
    for div in range(1, num):
        if num % div == 0:
            print(div, end=" ")
            sum_of_div += div
    return sum_of_div


# Asks for user Input
num = int(input("Enter a Number: "))
print(f"\nSum of All Proper Divisors of {num} = {sumPdivisors(num)}")
```

```
Enter a Number: 45
1 3 5 9 15
Sum of All Proper Divisors of 45 = 33
```

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]:
```
def sumPdivisors(num):
    """
    This function returns the sum of proper divisors of a number
    """
    sum_of_div = 0
    for div in range(1, num):
        if num % div == 0:
            sum_of_div += div
    return sum_of_div


def Pdivisors_in_range(num1, num2):
    """
    This function prints all the perfect numbers in a given range
    """
    for num in range(num1, num2+1):
        if num == sumPdivisors(num):
            print(num, end=" ")
    return ""
```

```
# Asks for user Input
num1 = int(input("Enter Starting Range: "))
num2 = int(input("Enter Ending Range: "))

print(f"Perfect Number in the Range {num1} to {num2}")
print(Pdivisors_in_range(num1, num2))
```

```
Enter Starting Range: 1
Enter Ending Range: 5555
Perfect Number in the Range 1 to 5555
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

[12]:
```
def sumPdivisors(num):
    """
    This function returns the sum of proper divisors of a number
    """
    sum_of_div = 0
    for div in range(1, num):
        if num % div == 0:
            sum_of_div += div
    return sum_of_div


def amicable_num_in_range(num1, num2):
    """
    This function prints the Amicable Number pairs in a given range
    """
    repetative = []
    for i in range(num1, num2+1):
        j = sumPdivisors(i)
        if (sumPdivisors(j) == i) and (i != j):
            repetative.append(j)
            if i in repetative:
                continue
            print(f"({i}, {j})", end="  ")
    return ""


# Asks for user Input
num1 = int(input("Enter Starting Range: "))
num2 = int(input("Enter Ending Range: "))
```

```
print(f"The Amicable Number pairs are in range {num1} to {num2}")
print(amicable_num_in_range(num1, num2))
```

```
Enter Starting Range: 1
Enter Ending Range: 500
The Amicable Number pairs are in range 1 to 500
(220, 284)
```

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

[13]:
```
# list of numbers from 1 to 50
numbers = list(range(1, 50))
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
# Printing the odd number list
print(odd_numbers)
```

```
[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]
```

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

[14]:
```
# list of numbers from 1 to 10
numbers = list(range(1, 10))
cube_of_numbers = list(map(lambda x: x**3, numbers))
# Printing the cube of given list
print(cube_of_numbers)
```

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

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

[15]:
```
# list of numbers from 1 to 50
numbers = list(range(1, 50))
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
cube_of_even_numbers = list(map(lambda x: x**3, even_numbers))
# Printing the cube of given list
print(cube_of_even_numbers)
```

```
[8, 64, 216, 512, 1000, 1728, 2744, 4096, 5832, 8000, 10648, 13824, 17576,
21952, 27000, 32768, 39304, 46656, 54872, 64000, 74088, 85184, 97336, 110592]
```

==========End Of File==========