Python_Assignment_1

June 13, 2020

1 1.multiplication table of number

[1]:
```
def multipication_table(num):
    print('Multiplication table of', num,':')
    for i in range(1,11):
        print(num,'x',i,'=',num*i)

multipication_table(19)
```

```
Multiplication table of 19 :
19 x 1 = 19
19 x 2 = 38
19 x 3 = 57
19 x 4 = 76
19 x 5 = 95
19 x 6 = 114
19 x 7 = 133
19 x 8 = 152
19 x 9 = 171
19 x 10 = 190
```

2 2.twin primes less than 1000

[2]:
```
#function for checking prime
def is_prime(n):
    if n==1:
        return 0
    for i in range(2,n):
        if n%i==0:
            return 0
    return 1

def twin_primes(num):
    for i in range(1,num-2,2):
        if is_prime(i) and is_prime(i+2):
            print('(',i,',',i+2,')')
```

```
twin_primes(1000)
```

```
( 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.prime factors of a number

[3]:
```
import math
def prime_factors(n):
    print('prime factors of ',n,'are:')
    while n%2==0: #number of 2's that divide n
        print(2,end=',')
        n=n//2
    for i in range(3,int(math.sqrt(n))+1,2):  #odd numbers
        while n%i==0:
            print(i,end=',')
            n=n//i

    if n>2: #if n is a prime number greater than 2
        print(n,end='\n')

prime_factors(56)
prime_factors(315)
```

```
prime factors of  56 are:
2,2,2,7
prime factors of  315 are:
3,3,5,7,
```

4 4.implement formulae of permutations and combinations

[4]:
```
#function to find factorial
def factorial(num):
    if num==1:
        return 1
    else:
        return num*factorial(num-1)

#function to find permutation
def npr(n,r):
    return factorial(n)/factorial(n-r)

#function to find combination
def ncr(n,r):
    return npr(n,r)/factorial(r)

print(npr(7,3))
print(ncr(7,3))
```

```
210.0
35.0
```

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

[5]:
```
def decimal_to_binary(num):
    if num>1:
        decimal_to_binary(num//2) #recursive function
    print(num%2,end='')
```

```
decimal_to_binary(99)
```

```
1100011
```

6 6.print Armstrong numbers and find whether number is an Armstrong or not

[6]:
```
def cubesum(num): #only for 3 digit numbers
    sum=0
    while num!=0:
        r=num%10
        sum+=pow(r,3)
        num//=10
    return sum

def isArmstrong(n): #whether num is an Armstrong number
    if cubesum(n)==n:
        print(n,' is an Armstrong number')
    else:
        print(n,' is not an Armstrong number')

isArmstrong(153)
isArmstrong(729)
```

```
153  is an Armstrong number
729  is not an Armstrong number
```

[7]:
```
def  PrintArmstrong(start,end): #printing all 3 digit armstrong numbers
    for i in range(start,end):
        if i==cubesum(i):
            print(i)

PrintArmstrong(100,1000)
```

```
153
370
371
407
```

7 7.function to find product of digits of a number

[8]:
```
def prodDigits(num): #function to find product of digits of a number
    product=1
    while num!=0:
        product*=num%10
        num//=10
    return product

prodDigits(1729)
```

```
[8]: 126
```

8 8.functions to find multiplicative digital root and multiplicative persistence

```
[9]: def MDR(n): #function to find multiplicative digital root
         while n>9:
             n=prodDigits(n)
         return n

     MDR(1729)
```

```
[9]: 2
```

```
[10]: def MPersistence(n): #function to find multiplicative persistence
          count=0
          while n>9:
              n=prodDigits(n)
              count+=1
          return count

      MPersistence(1729)
```

```
[10]: 3
```

9 9.function that finds the sum of proper divisors of a number

```
[11]: def sumPdivisors(n): #function to find sum of proper divisors of a number
          sum=0
          for i in range(1,n//2+1):
              if n%i==0:
                  sum+=i
          return sum

      sumPdivisors(36)
```

```
[11]: 55
```

10 10.program to print all the perfect numbers in a given range

[12]:
```python
def sumPdivisors(n): #function to find sum of properdivisors of a number
    sum=0
    for i in range(1,n//2+1):
        if n%i==0:
            sum+=i
    return sum

def printPerfect(start,end): #printing all perfect numbers
    for i in range(start,end+1):
        if i==sumPdivisors(i):
            print(i)

printPerfect(1,1000)
```

```
6
28
496
```

11 11.function to print pairs of amicable numbers in a range

[13]:
```python
def sumPdivisors(n): #function to find sum of properdivisors of a number
    sum=0
    for i in range(1,n//2+1):
        if n%i==0:
            sum+=i
    return sum

def amicablePairs(start,end): #function to find pairs of amicable numbers
    for i in range(start,end+1):
        sum1=sumPdivisors(i)
        sum2=sumPdivisors(sum1)
        if i!=sum1 and i==sum2:
            print('(',i,',',sum1,')')

amicablePairs(1,250)
```

```
( 220 , 284 )
```

12 12.filter odd numbers in a list by using filter function

[14]:
```python
def fun(num): #function that filters odd numbers
    if num%2==0:
        return 0
    else:
        return 1

numbers=[1,2,3,4,5,6,7,8,9] #numbers list

filtered=list(filter(fun,numbers)) #filtering odd numbers using filter function
print(filtered)
```

```
[1, 3, 5, 7, 9]
```

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

[15]:
```
def cube(num):
    return num**3

numbers=[1,2,3,4,5,6,7,8,9] #numbers list

result=map(cube,numbers) #mapping cubes of elements
print(list(result))
```

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

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

[16]:
```
def fun(num): #function that filters even numbers
    if num%2==0:
        return 1
    else:
        return 0

def cube(num):
    return num**3

numbers=[1,2,3,4,5,6,7,8,9] #numbers list

res=map(cube,filter(fun,numbers)) #map and filter function
print(list(res))
```

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

[ ]: