Assignment -1

Practice_Questions_On_Functions

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

```
# Considering table is printed till 10 nos.

def multiplication_table(number):
  for num in range(1,11):
    print("{} * {} = {}".format(number,num,(number*num)))

#Taking an example
multiplication_table(2)
```

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

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.

Solution 2 Approach

1) Start from 3 as we want to filter twin primes that are 2 consecutive odd nos, so no use of checking 2.

2)Add all prime nos from 3-1000 in a list

3)Filter the twin primes from the prime list.

```
from math import sqrt
from math import ceil

primeNumbers = []
twinPrimeList1 = []

#Function to check if number is prime or not
def isPrime(number):
  for num in range(2,int(ceil(sqrt(number)))):
```

```
    if((number%num)==0):
      return False
  return True

#Function to generate all prime nos.
def prime_nos():
  for number in range(3,1001,2):
    if(isPrime(number)):
      primeNumbers.append(number)
  return primeNumbers

#Function to print the twins
def twinPrimeNos(primeNumberList):
  for index in range(1,len(primeNumberList)):
    if(primeNumberList[index]-primeNumberList[index-1]==2):
      twins=()
      twins = (primeNumberList[index-1],primeNumberList[index])
      twinPrimeList1.append(twins)

  return twinPrimeList1

def twinsTuple(n1,n2):
  twins=()
  twins = (n1,n2)
  return twins

primeNumberList = prime_nos() #Getting all primes from 3 to 1000

#Sln1
twinPrimeList1 = twinPrimeNos(primeNumberList)# Filtering all twin primes
print(twinPrimeList1)

#Sln2
twinPrimeList2 = [twinsTuple(primeNumberList[index-1],primeNumberList[index]) for i
print(twinPrimeList2)
```

```
[(3, 5), (5, 7), (7, 9), (9, 11), (11, 13), (17, 19), (23, 25), (29, 31), (41,
[(3, 5), (5, 7), (7, 9), (9, 11), (11, 13), (17, 19), (23, 25), (29, 31), (41,
```

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

Solution 3 Approach

1) Starting with first prime no, divide the given number until it is is dividable

2)Go to next prime number

3)Repeat this until reminder is 0

```
from math import ceil
```

```
from math import ceil

prime_factors = []
#Function to check if number is prime or not
def isPrime(number):
  for num in range(2,int(ceil(sqrt(number)))):
    if((number%num)==0):
      return False
  return True

# Function to find prime factors
def primeFactors(number):
  current_prime =2
  while(number>1):
    print("{} is current number and {} is current prime factor".format(number,curre
    if(isPrime(current_prime)):
      if(number%current_prime == 0):
        prime_factors.append(current_prime)
        number = number/current_prime

      else:
        current_prime +=1
    else:
      current_prime +=1

  return prime_factors

prime_factors = primeFactors(56)
print(prime_factors)
```

```
56 is current number and 2 is current prime factor
28.0 is current number and 2 is current prime factor
14.0 is current number and 2 is current prime factor
7.0 is current number and 2 is current prime factor
7.0 is current number and 3 is current prime factor
7.0 is current number and 4 is current prime factor
7.0 is current number and 5 is current prime factor
7.0 is current number and 6 is current prime factor
7.0 is current number and 7 is current prime factor
[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!

Solution 4 Approach

1) Create a function that finds factorial

2) create 2 more function for perm. and comb. formula and pass 2 arguments like given in questions n &r

```
def factorial(number):
  if(number==1):
    return 1
  else:
    return number*factorial(number-1)

def perm(n,r):
  if(n>=r):
    return (factorial(n)/factorial(n-r))
  else:
    return "n should be greater than equal to r"

def comb(n,r):
  if(n>=r):
    return perm(n,r)/factorial(r)
  else:
    return "n should be greater than equal to r"

print(perm(5,3))
print(comb(5,3))
```

```
60.0
10.0
```

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

Solution 5 Approach

1)create a function binary with the logic and convert the final string to str

2) join the list

```
binaryList=[]
binaryNumber = ""
def binary(number):
  while(number>1):
    binaryList.append(number%2)
    number = int(number/2)

  binaryList.append(1)
  return [str(x) for x in reversed(binaryList)]

numberList = binary(3)
binaryNumber = ''.join(numberList)
print(binaryNumber)
```

```
11
```

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.

Solution 6 Approach

1)Create cubesum() which will have cubes of all digits

2)Create 2 more functions to find and print Armstrong number(Considering numbers till 1000)

```
armstrongList=[]
def cubesum(number):
  sum = 0
  while(number>0):
    reminder = number%10
    number = int(number/10)
    sum = reminder**3 + sum

  return sum

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

def PrintArmstrong():
  armstrongList = [num for num in range(1,1001) if(isArmstrong(num))]
  print(armstrongList)

PrintArmstrong()
```

```
[1, 153, 370, 371, 407]
```

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

Solution 7 Approach

1)Create prodDigits() which will have product of all digits

```
def prodDigits(number):
  prod_of_digits = 1
  while(number>0):
    reminder = number%10
    number = int(number/10)
    prod_of_digits = reminder*prod_of_digits

  print(prod_of_digits)
```

```
prodDigits(333)
```

```
27
```

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

Solution 8 Approach

1)Create prodDigits() which will have product of all digits

2)Create MDR() and MPersistence in which we'll have a counter to count no. of times it got divided or simple we can say until the number gets<10

```
def MDR(number):
  prod_of_digits = 1
  while(number>0):
    reminder = number%10
    number = int(number/10)
    prod_of_digits = reminder*prod_of_digits

  return prod_of_digits

def MPersistence(number):
  count =0
  mdr_number=number
  while(mdr_number>10):
    mdr_number = MDR(mdr_number)
    count+=1

  print("MDR is {} and MPersistence is {}".format(mdr_number,count))


MPersistence(86)
```

```
MDR is 6 and MPersistence is 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, 18

Solution 9 Approach

1)Create sumPdivisors() and check till number/2 times as we know, no number can be divisble perfectly if it is more than it's half.

```
from math import ceil

divisorsList=[]
def sumPdivisors(number):
  for num in range(1,int(ceil(number/2))+1):
    if(number%num==0):
      divisorsList.append(num)

  return divisorsList


divisorsList = sumPdivisors(36)
print(divisorsList)
```

```
    [1, 2, 4, 7, 14]
```

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

Solution 10 Approach

1)Create sumPdivisors() and check till number/2 times as we know, no number can be divisble perfectly if it is more than it's half. 2)Finally check if the sum of diivisors is equak to number. If yes then add in result list. Note- We'll take nos from 1-1000.

```
from math import ceil


resultList=[]
def sumPdivisors(number):
  divisorsList=[]
  for num in range(1,int(ceil(number/2))+1):
    if(number%num==0):
      divisorsList.append(num)

  if(sum(divisorsList)==number):
    return True

  else:
    return False


for num in range(1,1001):
```

```
for num in range(1,1001):
  if(sumPdivisors(num)):
    resultList.append(num)

print(resultList)
```

```
[1, 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

Solution 11 Approach

1)Create sumPdivisors() and check till number/2 times as we know, no number can be divisble perfectly if it is more than it's half. 2)Create a function isAmicable() to check if 2 nos are amicable or not 3)print the pairs

Note- We'll take nos from 1-1000.

```
def sumOfDivisors(number):
  divisorsList=[]
  for num in range(1,int(ceil(number/2))+1):
    if(number%num==0):
      divisorsList.append(num)
  return divisorsList

def isAmicable(num1, num2):

  sumNum1 = sum(sumOfDivisors(num1))
  sumNum2 = sum(sumOfDivisors(num2))
  if((sumNum1==num2) and (num1==sumNum2)):
    return True
  return False

amicableList=[]
for num1 in range(1,1000):
  for num2 in range(num1+1,1001):
    if(isAmicable(num1,num2)):
      pair=()
      pair = (num1,num2)
      amicableList.append(pair)

print(amicableList)
```

```
[(220, 284)]
```

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

```
a = [1,2,3,4,5,6,7,8,9,10]
# Hint from : https://www.geeksforgeeks.org/python-filter-even-values-from-a-list/
oddFilter= lambda x: x%2!=0

b = list(filter(oddFilter,a))
print(b)
```

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

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

```
cubeNos = lambda x: x**3

a = [1,2,3,4,5,6]
b= list(map(cubeNos,a))
print(b)
```

```
[1, 8, 27, 64, 125, 216]
```

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

```
evenFilter= lambda x: x%2==0
cubeNos = lambda x: x**3

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

b = list(map(cubeNos,list(filter(evenFilter,a))))
print(b)
```

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

✓ 0s completed at 10:58 AM