Assignment1

June 13, 2020

1. function return input - Multiplication Table

[1]:
```
def mul_table(table_no, max_records=10):
  table = {}
  for i in range(1, max_records + 1, 1):
    table[str(i) + '*' + str(table_no)] = i*table_no
  return table

print(mul_table(int(input("Enter table number : "))))
```

```
Enter table number : 5
{'1*5': 5, '2*5': 10, '3*5': 15, '4*5': 20, '5*5': 25, '6*5': 30, '7*5': 35,
'8*5': 40, '9*5': 45, '10*5': 50}
```

2. display Twin prime_numbers below 1000

[2]:
```
def find_prime_pairs(n):

    sieve = [True] * n

    if n > 0:
        sieve[0] = False
    if n > 1:
        sieve[1] = False

    for number in range(2, int(n ** 0.5) + 1): #iterate sqrt of n to
        if sieve[number]:
            for index in range(number * number, n, number): # set squared terms␣
 ↪are not prime
                sieve[index] = False

    return [(a, b) for b, a in enumerate(range(0, n - 2), start=2) if sieve[a]␣
 ↪and sieve[b]] #enumerate numbers n-2 (for pairs) if both the pairs are true␣
 ↪with step=2

print(*find_prime_pairs(int(input("Enter number : "))), sep='\n')
```

```
Enter number : 153
(3, 5)
```

```
(5, 7)
(11, 13)
(17, 19)
(29, 31)
(41, 43)
(59, 61)
(71, 73)
(101, 103)
(107, 109)
(137, 139)
(149, 151)
```

3. Prime Factors

[3]:
```
import math
def primeFactors(n):

    # Print the number of two's that divide n
    while n % 2 == 0:
        print(2)
        n = n / 2

    # n must be odd at this point
    # so a skip of 2 ( i = i + 2) can be used
    for i in range(3,int(math.sqrt(n))+1,2):

        # while i divides n , print i ad divide n
        while n % i== 0:
            print (i)
            n = n / i

    # Condition if n is a prime
    # number greater than 2
    if n > 2:
        print (n)

primeFactors(int(input("enter number to find primefactor : ")))
```

```
enter number to find primefactor : 153
3
3
17.0
```

4. formulae of permutations and combinations

p(n, r) = n! / (n-r)!

(n, r) = n! / (r!*(n-r)!) = p(n,r) / r!

[4]:
```
def perm_comb(objects, time):

  x,y = objects, time
```

```
  def fact(num):
    factorial = 1
    for i in range(1, num+1):
      factorial = factorial+i
    return factorial

  def permutation(x, y):
    return fact(x) / fact(x-y)

  def combination(x, y):
    return fact(x)/(fact(y) * fact(x-y))

  return print("Permutation is {}".format(permutation(x, y)),"\n", "No. of␣
↪combination is {}".format(combination(x,y)))

perm_comb(int(input("Enter number of objects: ")),int(input("Enter time : ")))
```

```
Enter number of objects: 5
Enter time : 2
Permutation is 2.2857142857142856
 No. of combination is 0.5714285714285714
```

5. Decimal to Binary

[5]:
```
def dec_to_bin(x):
    return int(bin(x)[2:])
dec_to_bin(int(input("enter a decimal number to convert binary : ")))
```

```
enter a decimal number to convert binary : 10
```

[5]:
```
1010
```

6. Cubesum, isArmstrong, Armstrong

[6]:
```
def cubesum(x):
  for i in str(x):
    i = int(i)
    yield i**3

def PrintArmStrong(x):
  print("Armstrong number is {}".format(x))

def isArmStrong(x, CubeSum):
  CubeSum = sum(CubeSum)
  if x == int(CubeSum):
    print("Given number is armstrong number")
    PrintArmStrong(x)
  else:
```

```
    print("Given number is not Armstrong Number")


num = int(input("Enter number : "))
CubeSum = list(cubesum(num))
isArmStrong(num, CubeSum)
```

```
Enter number : 153
Given number is armstrong number
Armstrong number is 153
```

7. prodDigits()

[7]:
```
def prodDigits(num):
  prodig = 1
  for i in num:
    prodig*=int(i)
  return prodig

prodDigits(input("Enter a number to find pro digits : "))
```

```
Enter a number to find pro digits : 153
```

[7]:
```
15
```

8. MDR & MPersistence

[8]:
```
temp=list()

def MPersistence(x):
  return len(x)

def MDR(num):
  temp.append(num)
  num = prodDigits(num)

  if num >= 10:
    temp.append(str(num))
    MDR(str(num))
  else:
    temp.append(str(num))
    mdr = list(set(temp))
    mdr = [int(i) for i in mdr]
    mdr.sort(reverse=True)
    print("MDR is {}".format(mdr))
    print("MPR is {}".format(MPersistence(mdr)))

MDR(input("Enter number : "))
```

```
Enter number : 86
MDR is [86, 48, 32, 6]
MPR is 4
```

9. sumPdivisors()

[13]:
```
def sumPdivisors(num):
  devisors=[]
  for i in range(1, num, 1):
    if num%i == 0:
      devisors.append(i)
    else:
      pass
  return sum(devisors)

print(sumPdivisors(int(input("Enter number : "))))
```

```
Enter number : 153
81
```

10. ProperSumDevisors()

[14]:
```
def ProperSumDevisors(num):

  perfectSumDevisors=[]
  for i in range(1, num+1, 1):
    if i==1:
      perfectSumDevisors.append(i)
    else:
      devisors = sumPdivisors(i)
      if devisors == i:
        perfectSumDevisors.append(devisors)
      else:
        pass
  return perfectSumDevisors

print(ProperSumDevisors(int(input("Enter Number : "))))
```

```
Enter Number : 86
[1, 6, 28]
```

11. Amicable numbers proper devisor pair

[15]:
```
def Amicable(num):
  divisors = {}
  pairs = []
  for i in range(1, num+1, 1):
    divisors[str(i)] = str(sumPdivisors(i))

  for key in divisors.keys():
```

```
    val = divisors.get(key)

    if divisors.get(val) == key:
      pairs.append((key, val))

  return [pair for pair in pairs if pair[0] != pair[1]]

print(Amicable(int(input("Enter number : "))))
```

```
Enter number : 1000
[('220', '284'), ('284', '220')]
```

12. filter() odd no

[16]:
```
import numpy as np
num = np.arange(1, 100, 1) # input list
print([odd for odd in filter(lambda x: 0 if x % 2 == 0 else x, num)])
```

```
[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. cubeOfelmnts() in a list

[17]:
```
num = np.arange(1, 1000, 1) # input list

cubeOfelmnts = map(lambda x: dict({str(x) : x**3}) if x**3 in num else False,␣
 ↪num)

print([i for i in cubeOfelmnts if i!=False])
```

```
[{'1': 1}, {'2': 8}, {'3': 27}, {'4': 64}, {'5': 125}, {'6': 216}, {'7': 343},
{'8': 512}, {'9': 729}]
```

14. map(), filter() whose elements are cube of even number in a given list

[13]:
```
import numpy as np
num = np.arange(1, 11, 1) # input list
cubeOfelmnts = map(lambda x: x**3, filter(lambda x: x if x % 2 == 0 else False,␣
 ↪num))
print([i for i in cubeOfelmnts if i!=False])
```

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

[ ]:
```

```