(https://colab.research.google.com/github/poojashah19/Applied-AI-Course/blob/main/Assignment%201/Assignment1.ipynb)

In [1]:
```
## 1
def print_multiplicationtable(num) :
  for index in range(1,11):
    print ('%d * %d = %d' % (num, index, num*index))
num = int(input("enter any number: "))
print_multiplicationtable(num)
```

```
enter any number: 17
17 * 1 = 17
17 * 2 = 34
17 * 3 = 51
17 * 4 = 68
17 * 5 = 85
17 * 6 = 102
17 * 7 = 119
17 * 8 = 136
17 * 9 = 153
17 * 10 = 170
```

In [ ]:
```
## 2
def print_twinprimes() :
  for index in range(2,1000) :
    next_num = index + 2;
    if(is_prime(index) and is_prime(next_num)):
      print ('%d, %d' % (index, next_num))

def is_prime(num) :
  for index in range(2, num) :
    if(num % index == 0) :
      return False
  return True

print_twinprimes()
```

```
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
```

In [ ]:
```
## 3
##referenced https://www.geeksforgeeks.org/print-all-prime-factors-of-a-given-number/ link for few steps.
import math
def print_primefactors(num) :
  while num % 2 == 0 :
    print('2')
    num = int(num/2)
  for index in range(3, int(math.sqrt(num))+1,2) :
    while num % index == 0:
      print(index)
      num = int(num/index)
  if(num > 2):
    print(num)

num = int(input("enter any number: "))
print_primefactors(num)
```

```
enter any number: 88
2
2
2
11
```

In [ ]:
```
## 4
def get_factorial(num) :
  result = 1
  for index in range(1, num+1) :
    result = result*index
  return result

def get_numOfPermutations(num, r):
  print("Number of Permutations of %d objects taken %d at a time:" % (num, r))
  print(int(get_factorial(num) / get_factorial(num - r)))

def get_numOfCombinations(num, r):
  print("Number of Combinations of %d objects taken %d at a time:" % (num, r))
  print(int( get_factorial(num) / ( get_factorial(r) * get_factorial(num-r)
)))

get_numOfPermutations(10,3)
get_numOfCombinations(10,3)
```

```
Number of Permutations of 10 objects taken 3 at a time:
720
Number of Combinations of 10 objects taken 3 at a time:
120
```

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

num = int(input("enter any number: "))
get_binary(num)
```

```
enter any number: 30
11110
```

In [ ]:
```
## 6
def cubesum(num):
  temp = num
  sum = 0
  while(temp > 0) :
    num1 = temp % 10
    sum += num1 ** 3
    temp //= 10
  return sum

def isArmstrong(num, sum) :
  if(num == sum) :
    print('%d is an armstrong number' % num)
  else :
    print('%d is not an armstrong number' % num)

def PrintArmstrong(num) :
  sum = cubesum(num)
  print('sum of cube of digits in %d is %d' % (num, sum))
  isArmstrong(num, sum)

num = int(input("enter any number: "))
PrintArmstrong(num)
```

```
enter any number: 98
sum of cube of digits in 98 is 1241
98 is not an armstrong number
```

In [ ]:
```
## 7
def prodDigits(num) :
  temp = num
  prod = 1
  while(temp > 0) :
    digit = temp % 10
    prod *= digit
    temp //= 10
  return prod

num = int(input("enter any number: "))
print('product of digits of %d is %d' % (num, prodDigits(num)))
```

```
enter any number: 438
product of digits of 438 is 96
```

In [ ]:
```
## 8
## I have implemented logic of MDR and MPersistence in a single function as bo
th has repetitive while loop.
def prodDigits(num) :
  temp = num
  prod = 1
  while(temp > 0) :
    digit = temp % 10
    prod *= digit
    temp //= 10
  return prod

def MDRandMPersistence(num) :
  prodOfDigits = prodDigits(num)
  count = 1
  while( prodOfDigits >= 10 ) :
    prodOfDigits = prodDigits(prodOfDigits)
    count += 1
  print('MDR is %d' % prodOfDigits)
  print('MPersistence is %d' % count)

MDRandMPersistence(486)
```

```
MDR is 8
MPersistence is 3
```

In [3]:
```
## 9
import math
def sumPdivisors(num) :
  result = 0
  for index in range(2, int(math.sqrt(num))+1):
    if(num % index == 0):
      if(index == num//index):
        result += index
      else :
        result += index + num//index
  return result+1  ## added 1 in final result as 1 is also a proper divisor.

num = int(input("enter any number: "))
sumPdivisors(num)
```

```
enter any number: 24
```

Out[3]:
```
36
```

In [ ]:
```
## 10
import math
def sumPdivisors(num) :
  result = 0
  for index in range(2, int(math.sqrt(num))+1):
    if(num % index == 0):
      if(index == num//index):
        result += index
      else :
        result += index + num//index
  return result+1  ## added 1 in final result as 1 is also a proper divisor.

num = int(input("enter any number: "))
if(sumPdivisors(num) == num) :
  print('%d is a perfect number' % num)
else :
  print('%d is not a perfect number' % num)
```

```
enter any number: 36
36 is not a perfect number
```

In [ ]:
```
## 11
## referred https://stackoverflow.com/questions/28267790/c-find-all-amicable-numbers-between-limits because i did not wanted to use nested for loop for larger range as nested loop was taking a lot of time
import math
def sumPdivisors(num) :
  result = 0
  for index in range(2, int(math.sqrt(num))+1):
    if(num % index == 0):
      if(index == num//index):
        result += index
      else :
        result += index + num//index
  return result+1  ## added 1 in final result as 1 is also a proper divisor.

def isAmicable(num, snum):
  if(sumPdivisors(num) == snum and sumPdivisors(snum) == num) :
    return True
  else :
    return False

for index in range(1, 10000):
  snum = sumPdivisors(index)
  if(index<snum and sumPdivisors(snum) == index):
    print(index, ',', snum)
```

```
220 , 284
1184 , 1210
2620 , 2924
5020 , 5564
6232 , 6368
```

In [10]:
```
## 12
num_list = [1,2,4,7,19,44,11,37,26,53,87,74,95,58]
def odd_numbers(num) :
  if(num % 2 != 0):
    return True
  else:
    return False

oddNumbers = list(filter(odd_numbers, num_list))
print(oddNumbers)
```

```
[1, 7, 19, 11, 37, 53, 87, 95]
```

In [5]:
```
## 13
list = [5,9,3,8,6]
cube = map(lambda x:x**3 , list)

for num in cube:
  print(num)
```

```
125
729
27
512
216
```

In [ ]:
```
## 14
num_list = [1,2,4,7,19,44,11,37,6,53,87,14,95,8]
def even_numbers(num) :
  if(num % 2 == 0):
    return True
  else:
    return False

evenNumbers = filter(even_numbers, num_list)
cube = map(lambda x:x**3 , evenNumbers)

result = set(cube)
print(result)
```

```
{64, 85184, 512, 8, 2744, 216}
```