Practice Questions On Functions

In [7]:
```
#1. Write a function that inputs a number and prints the multiplication table of

def multwo(nos):
    for i in range(1,nos):
        for j in range(1,nos):
            print('{:<2d} * {:<2} is {:<2d}'.format(j, i, j*i))
        print('********************')
multwo(nos=int(input()))
```

```
3  * 9  is 27
4  * 9  is 36
5  * 9  is 45
6  * 9  is 54
7  * 9  is 63
8  * 9  is 72
9  * 9  is 81
10 * 9  is 90
********************

1  * 10 is 10
2  * 10 is 20
3  * 10 is 30
4  * 10 is 40
5  * 10 is 50
6  * 10 is 60
7  * 10 is 70
8  * 10 is 80
9  * 10 is 90
10 * 10 is 100
```

In [2]:
```
#2. Write a program to print twin primes less than 1000. If two consecutive odd nu
#both prime then they are known as twin primes
def prim(n):
    for i in range(2, n):
        if n % i == 0:
            break
    else:
        return True
#prim()
def twinprims(n):
    l = []
    for i in range(2,n):
        if prim(i) == True:
            l.append(i)
    for k in l:
        for j in range(l[0], k):
            if k - j == 2:
                print('twin prime numbers are {:<4d} and {:<4d}'.format(j,k))
```

In [3]:
```
twinprims(1000)
```

```
twin prime numbers are 99   and 101
twin prime numbers are 101  and 103
twin prime numbers are 105  and 107
twin prime numbers are 107  and 109
twin prime numbers are 111  and 113
twin prime numbers are 125  and 127
twin prime numbers are 129  and 131
twin prime numbers are 135  and 137
twin prime numbers are 137  and 139
twin prime numbers are 147  and 149
twin prime numbers are 149  and 151
twin prime numbers are 155  and 157
twin prime numbers are 161  and 163
twin prime numbers are 165  and 167
twin prime numbers are 171  and 173
twin prime numbers are 177  and 179
twin prime numbers are 179  and 181
twin prime numbers are 189  and 191
twin prime numbers are 191  and 193
twin prime numbers are 195  and 197
```

In [1]:
```
#3. Write a program to find out the prime factors of a number. Example: prime fac
def primfact(n):
    i = 2
    l_p_fact = []
    while i * i <= n:
        if n % i:
            i += 1
        else:
            n //= i
            l_p_fact.append(i)
    if n > 1:
        l_p_fact.append(n)
    return l_p_fact
primfact(84)
```

Out[1]:
```
[2, 2, 3, 7]
```

In [1]:
```
#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)!. Nu
#combinations of n objects taken r at a time is: c(n, r) = n! / (r!*(n-r)!) = p(n,

n = int(input('enter a number:'))
r = int(input('enter a selection no: '))
count = 1
perm_count = 1
comb_count = 1
for i in range(1, n+1):
    count = count * i
for j in range(r, n+1-r):
    perm_count = perm_count * j
for k in range(1, r+1):
    comb_count = comb_count * k
permutation = count / perm_count
combination = permutation / comb_count
print('permutation of {} taken {} at a time is {}'.format(n, r, permutation))
print('combination of {} taken {} at a time is {}'.format(n,r,combination ))
```

```
enter a number:6
enter a selection no: 2
permutation of 6 taken 2 at a time is 30.0
combination of 6 taken 2 at a time is 15.0
```

In [1]:
```
#5. Write a function that converts a decimal number to binary number

def d2b(n):
    if n > 1:
        d2b(n // 2)
    print(n % 2, end='')
d2b(25)
```

```
11001
```

In [2]:
```
#6. Write a function cubesum() that accepts an integer and returns the sum of the
#individual digits of that number. Use this function to make functions PrintArmstr
#isArmstrong() to print Armstrong numbers and to find whether is an Armstrong numb
def cubes(n):
    a = 0
    b = str(n)
    for i in range(len(b)):
        a = a + int(b[i]) ** 3
    return a

def isArmstrong(a):
   if a == cubes(a):
    print('{} is armstrong no'.format(a))
   else:
    print('{} is not an armstrong number'.format(a))

isArmstrong(371)
isArmstrong(345)
```

```
371 is armstrong no
345 is not an armstrong number
```

In [16]:
```
#7. Write a function prodDigits() that inputs a number and returns the product of

def proddigits(n):
    b = 1
    a = str(n)
    for i in range(len(a)):
        b = b * int(a[i])
    return b

proddigits(249)
```

Out[16]: 72

In [19]:
```
#8.8. If all digits of a number n are multiplied by each other repeating with the
#digit number obtained at last is called the multiplicative digital root of n. Th
#times digits need to be multiplied to reach one digit is called the multiplicati
#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 an
#multiplicative persistence respectively


def MDR(n):
    mpr = proddigits(n)#249, 72, 14, 4
    if mpr <10:
        print(mpr)

    elif mpr >=10:
        n = mpr
    return proddigits(mpr)#14

MDR(249)
##Note: op dint come as expected.pls rectify
```

Out[19]: 14

In [1]:
```
#9. Write a function sumPdivisors() that finds the sum of proper divisors of a nu
#divisors of a number are those numbers by which the number is divisible, except
#number itself. For example proper divisors of 36 are 1, 2, 3, 4, 6, 9, 18

def proper(n):
    count = 0
    for i in range(1, n ):
        if n % i == 0 & i != n:
            print('{:<2d} is proper divisible of {:<2d}'.format(i, n))
proper(36)
```

```
1  is proper divisible of 36
2  is proper divisible of 36
3  is proper divisible of 36
4  is proper divisible of 36
6  is proper divisible of 36
9  is proper divisible of 36
12 is proper divisible of 36
18 is proper divisible of 36
```

In [2]:
```
#10. A number is called perfect if the sum of proper divisors of that number is e
#number. For example 28 is perfect number, since 1+2+4+7+14=28. Write a program t
#print all the perfect numbers in a given range

def properdiv(n):
    count = 0
    sums = 0
    for i in range(1,n):
        if n % i == 0 & i != n:
            count = count + i
    if count == n:
        print('{} is proper divisor'.format(n))
    else:
        print('{} is not proper divisor'.format(n))
properdiv(128)
properdiv(28)
```

```
128 is not proper divisor
28 is proper divisors
```

In [14]:
```
#12. Write a program which can filter odd numbers in a list by using filter funct
l = [1,4,6,2,3,7,9,17,31,23]
odd_n = list(filter(lambda x : x % 2 == 1, l))
odd_n
```

Out[14]: [1, 3, 7, 9, 17, 31, 23]

In [12]:
```
#13. Write a program which can map() to make a list whose elements are cube of el
l = [1,2,3,11,5,14,7]
cube_l = list(map(lambda x: x ** 3, l))
cube_l
```

Out[12]: [1, 8, 27, 1331, 125, 2744, 343]

In [11]:
```
#14. Write a program which can map() and filter() to make a list whose elements a
#even number in a given list

l = [1,5,6,8,2,44,67,14]
even_l = list(filter(lambda x: x % 2 == 0, l))
cube_l = list(map(lambda x: x ** 3, even_l))
cube_l
```

Out[11]: [216, 512, 8, 85184, 2744]