(https://www.darshan.ac.in/)

Python Programming - 2101CS405

Lab - 7

SACHIN PATADIYA

22010101142

Functions

In [ ]:
```
# will throw an error
## UnboundLocalError: local variable 'a' referenced before assignment
a = 15
def change():
    a = a + 5
    print(a)
change()
```

In [4]:
```
# when to use global
a = 15
def change():
    global a
    a = a + 5
    print(a)
change()
```

```
20
```

01) WAP to count simple interest using function.

SI =(principle(amount) * rate of interest * time)/ 100

In [3]:
```
def findInterest(p,r,n):
    ans=(p*r*n)/100
    print(f"simple interest is {ans}")

p=float(input("Enter Principle : "))
r=float(input("Enter rate : "))
n=float(input("Enter Time : "))
findInterest(p,r,n)
```

```
Enter Principle : 100
Enter rate : 2
Enter Time : 1
simple interest is 2.0
```

02) WAP that defines a function to add first n numbers.

In [7]:
```
def SumOfN(n):
    return ((n*(n+1))/2)

n=int(input("Enter n :"))
print(f"Sum of first {n} number is {SumOfN(n)}")
```

```
Enter n :6
Sum of first 6 number is 21.0
```

03) WAP to find maximum number from given two numbers using function.

In [10]:
```
def maxOfTwo(a,b):
    if a>b:
        return a
    else:
        return b

a=int(input("Enter First number :"))
b=int(input("Enter Second number :"))
print(f"maximum number is {maxOfTwo(a,b)}")
```

```
Enter First number :1
Enter Second number :2
maximum number is 2
```

04) WAP that defines a function which returns 1 if the number is prime otherwise return 0.

Only iteration in loops are counted

In [56]:
```
countOfItr=0
def findPrime(n):
    global countOfItr
    for i in range(2,((n//2)+1)):
        countOfItr=countOfItr+1
        if n%i==0:
            return 0
    return 1
n=int(input("Enter number :"))
print(f"{findPrime(n)}")
print(f"number of iteration is = {countOfItr}")
```

```
Enter number :7
1
number of iteration is = 2
```

In [20]:
```

```

```
Enter Number :  9973
No. of iteration :  16
1
```

05) Write a function called primes that takes an integer value as an argument and returns a list of all prime numbers up to that number.

In [68]:
```
def findPrime(n):
    for i in range(2,((n//2)+1)):
        if n%i==0:
            return 0
    return 1

primeList=[]
noOfPrime=0
n=int(input("Enter n : "))
for i in range(2,n):
    if findPrime(i):
        primeList.append(i)
        noOfPrime=noOfPrime+1
print(primeList)
print(f"no of prime is {noOfPrime}")
```

```
Enter n : 1000
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149,
151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 22
9, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311,
313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 40
1, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487,
491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 59
3, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673,
677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 77
3, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877,
881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 98
3, 991, 997]
no of prime is 168
```

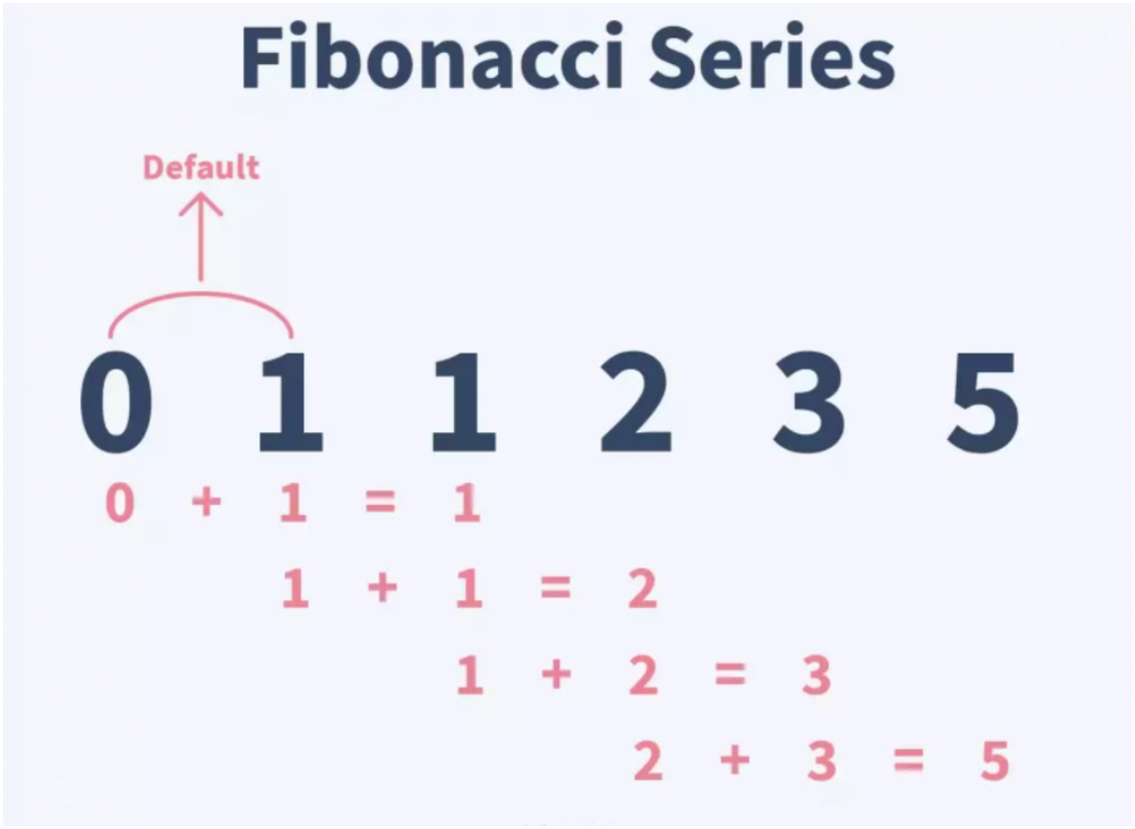
In [56]:

```
Enter Number: 1000
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149,
151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 22
9, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311,
313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 40
1, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487,
491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 59
3, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673,
677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 77
3, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877,
881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 98
3, 991, 997]
No of primes :  168
No. of iteration :  1764
```

In [55]:

```
Enter Number: 1000
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149,
151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 22
9, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311,
313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 40
1, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487,
491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 59
3, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673,
677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 77
3, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877,
881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 98
3, 991, 997]
No of primes: 168
No. of iteration: 929
```

06) WAP to generate Fibonacci series of N given number using function name fibbo. (e.g. 0 1 1 2 3 5 8...)



In [7]:
```
def fibonacci(n):
    prev=0
    save=1
    print(prev,save,end=" ")
    for i in range(2,n):
        next=prev+save
        prev=save
        save=next
        print(next,end=" ")

n=int(input("Enter Number : "))
fibonacci(n)
```

```
0 1 1 2 3 5
```

07) WAP to find the factorial of a given number using recursion.

In [5]:
```
def factorial(n):
    if(n<=1):
        return 1
    return (n* factorial(n-1))
n = int(input("Enter a number : "))
print(factorial(n))
```

```
Enter a number : 5
120
```

08) WAP to implement simple calculator using lamda function.

In [6]:
```
def myCal(n1,n2,ch):
    match ch :
        case "+":
            return (lambda n1,n2: n1+n2)(n1,n2)
        case "-":
            return (lambda n1,n2: n1-n2)(n1,n2)
        case "*":
            return (lambda n1,n2: n1*n2)(n1,n2)
        case "/":
            return (lambda n1,n2: n1/n2)(n1,n2)
        case "%":
            return (lambda n1,n2: n1%n2)(n1,n2)
    return "invalid operater"
n1 = int(input("Enter a first number : "))
n2 = int(input("Enter a second number : "))

ch = input("Enter operation you want to perform (+,-,*,/,% for modulo)")
print(myCal(n1,n2,ch))
```

```
Enter a first number : 3
Enter a second number : 5
Enter operation you want to perform (+,-,*,/,% for modulo)+
8
```

09)Write a Python program that accepts a hyphen-separated sequence of words as input and prints the words in a hyphen-separated sequence after sorting them alphabetically

Sample Items : green-red-yellow-black-white

Expected Result : black-green-red-white-yellow

In [7]:
```
s = "green-red-yellow-black-white"
mystr = s.split("-")
mystr.sort()
s = ("-").join(mystr)
print(s)
```

```
black-green-red-white-yellow
```

10) Write a python program to implement all function arguments type

Positional arguments

Default argument

Keyword arguments (named arguments)

Arbitrary arguments (variable-length arguments args and kwargs)

In [ ]:
```
def myFun(n1,n2=1):
    print(n1*n2)
def myFun1(*n):
    print(n)
```

In [58]:
```
# Default argument
myFun(4)
```

In [ ]:
```
# Keyword argument
myFun(n2=5,n1=4)
```

In [ ]:
```
# Arbitrary arguments awrgs
myFun1(4,5,6,7,8,9,0)
```

01) WAP to calculate power of a number using recursion.

In [8]:
```
def myRecPorwer(base,power):
    if(power <=0):
        return 1
    else:
        return base * myRecPorwer(base,power-1)

b = int(input("Enter a base number : "))
p = int(input("Enter a power number : "))
print(myRecPorwer(b,p))
```

```
Enter a base number : 2
Enter a power number : 4
16
```

02) WAP to count digits of a number using recursion.

In [16]:
```
def coundDigit(digit):
    if digit<=0:
        return 0
    else:
        return 1+coundDigit(digit//10)

print(f"total digit is {coundDigit(12345)}")
```

```
total digit is 5
```

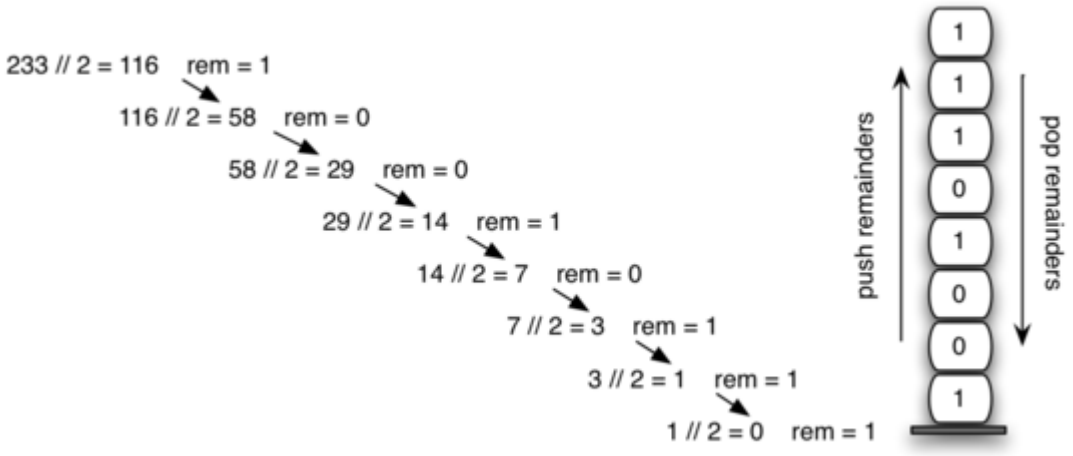
03) WAP to reverse an integer number using recursion.

In [21]:
```
revese=0
def reverseNum(num):
    global revese
    if num==0:
        return revese
    else:
        revese=(revese*10)+(num%10)
        return reverseNum(num//10)

print(reverseNum(123456))
```

```
654321
```

04) WAP to convert decimal number into binary using recursion.



Decimal To Binary Converstion:

Let the decimal number be : 14

2	14	0
2	7	1
2	3	1
2	1	1
	0	

$(14)_{10} = (1110)_2$

Let the decimal number be : 22

2	22	0
2	11	1
2	5	1
2	2	0
2	1	1
	0	

$(21)_{10} = (10110)_2$

In [41]:
```
binary=0
list=[]
def DecimalToBinery(num):
    global binary
    global list
    if num==0:
        return list
    else:
        list.append(str(num%2))
        return DecimalToBinery(num//2)
"".join(DecimalToBinery(21))
```

Out[41]: '10101'

Map , Filter , Reduce

In [ ]:
```
The filter() method filters the given sequence with the help of a function
```

In [ ]:
```
numbers = [1, 2, 3, 4, 5]

squared_numbers = list(map(lambda x: x * x, numbers))

print(squared_numbers)
```

[1, 4, 9, 16, 25]

The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

In [ ]:
```
# a list contains both even and odd numbers.
seq = [0, 1, 2, 3, 5, 8, 13]

# result contains odd numbers of the list
result = filter(lambda x: x % 2 != 0, seq)
print(list(result))

# result contains even numbers of the list
result = filter(lambda x: x % 2 == 0, seq)
print(list(result))
```

The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.This function is defined in “functools” module.

In [ ]:
```
 # importing functools for reduce()
import functools

# initializing list
lis = [1, 3, 5, 6, 2]

# using reduce to compute sum of list
print("The sum of the list elements is : ", end="")
print(functools.reduce(lambda a, b: a+b, lis))

# using reduce to compute maximum element from list
print("The maximum element of the list is : ", end="")
print(functools.reduce(lambda a, b: a if a > b else b, lis))
```