Darshan UNIVERSITY

योग: कर्मसु कौशलम्

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

Python Programming - 2101CS405

Lab - 7

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 [1]:

```
Enter Principle : 50000
Enter Rate : 2
Enter Time : 3
Simple interest is 3000.0
```

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

In [6]:

```
Enter Number : 6
Sum of Number is :  21
```

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

In [5]:

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

In [1]:

```
Enter Number :  9973
No. of iteration :  100
1
```

optional task : optimization

complete basic task first then try to optimize

In [1]:

```
# Example on how iterations are calculated
# Only iteration in loops are counted
iteration_count= 0
n = 5
for i in range(n):
    iteration_count += 1

    for j in range(n-i-1):
        iteration_count += 1
        print(" ", end = " ")
    for j in range(i+1):
        iteration_count += 1
        print("*", end = " ")
    print()
print("Total Iterations : ", iteration_count)
```

```
        * 
      * * 
    * * * 
  * * * * 
* * * * * 
Total Iterations :  30
```

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 [45]:

```
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, 17
3, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 2
69, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367,
373, 379, 383, 389, 397, 401, 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,
593, 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, 773, 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, 983, 991, 997]
No of primes :  168
No. of iteration :  6287
```

optional task : optimization

if basic task completed then try to optimize

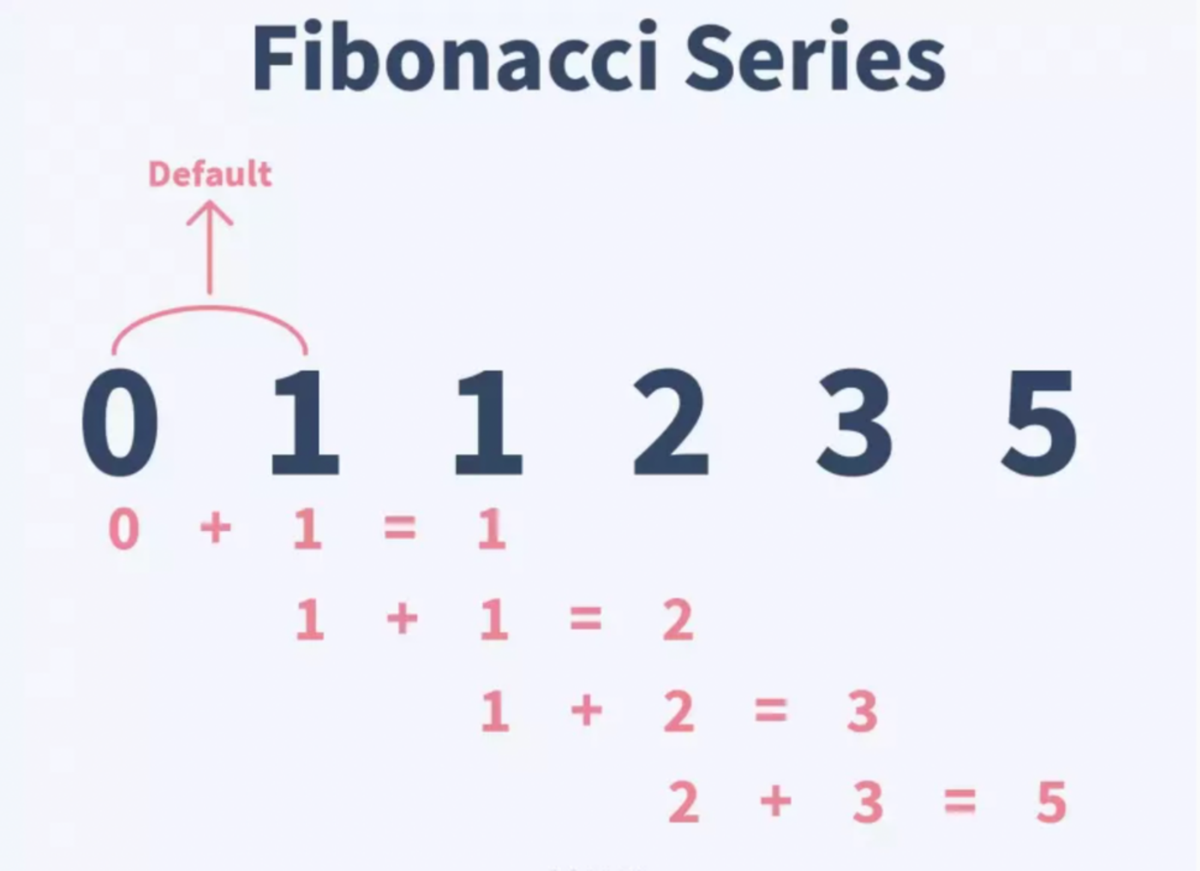
In [56]:

```
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, 17
3, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 2
69, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367,
373, 379, 383, 389, 397, 401, 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,
593, 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, 773, 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, 983, 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, 17
3, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 2
69, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367,
373, 379, 383, 389, 397, 401, 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,
593, 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, 773, 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, 983, 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 [21]:

```
Enter Number : 20
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181,
6765]
```

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

In [57]:

08) WAP to implement simple calculator using lamda function.

In [ ]:

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 [ ]:

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 [ ]:

```
# Positional argument
```

In [58]:

```
# Default argument
def defaultArgument (a,b=10) :
```

In [ ]:

```
# Keyword argument
```

In [ ]:

```
# Arbitrary arguments awrgs
def arbitraryArguments(*b):
```

In [ ]:

```
# Arbitrary arguments kwargs
def arbitraryArguments(**b):
```

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

In [ ]:

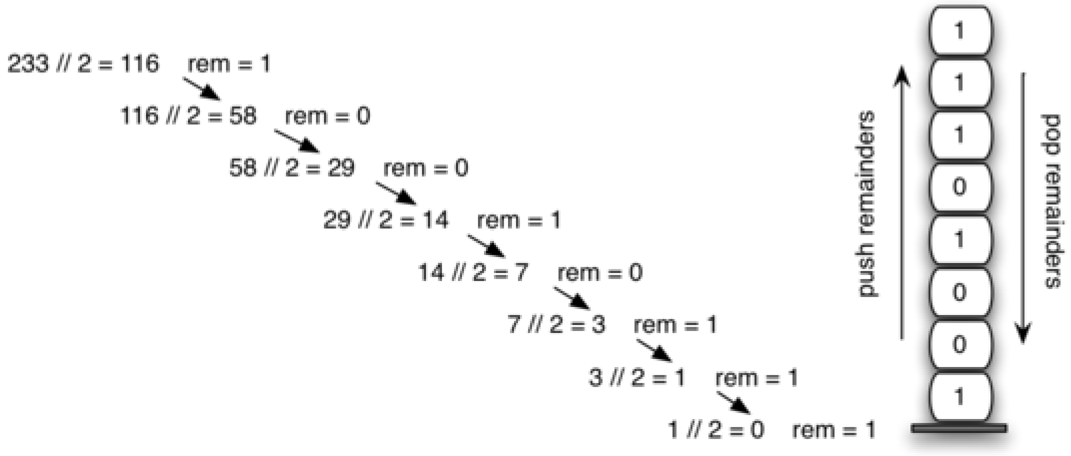
02) WAP to count digits of a number using recursion.

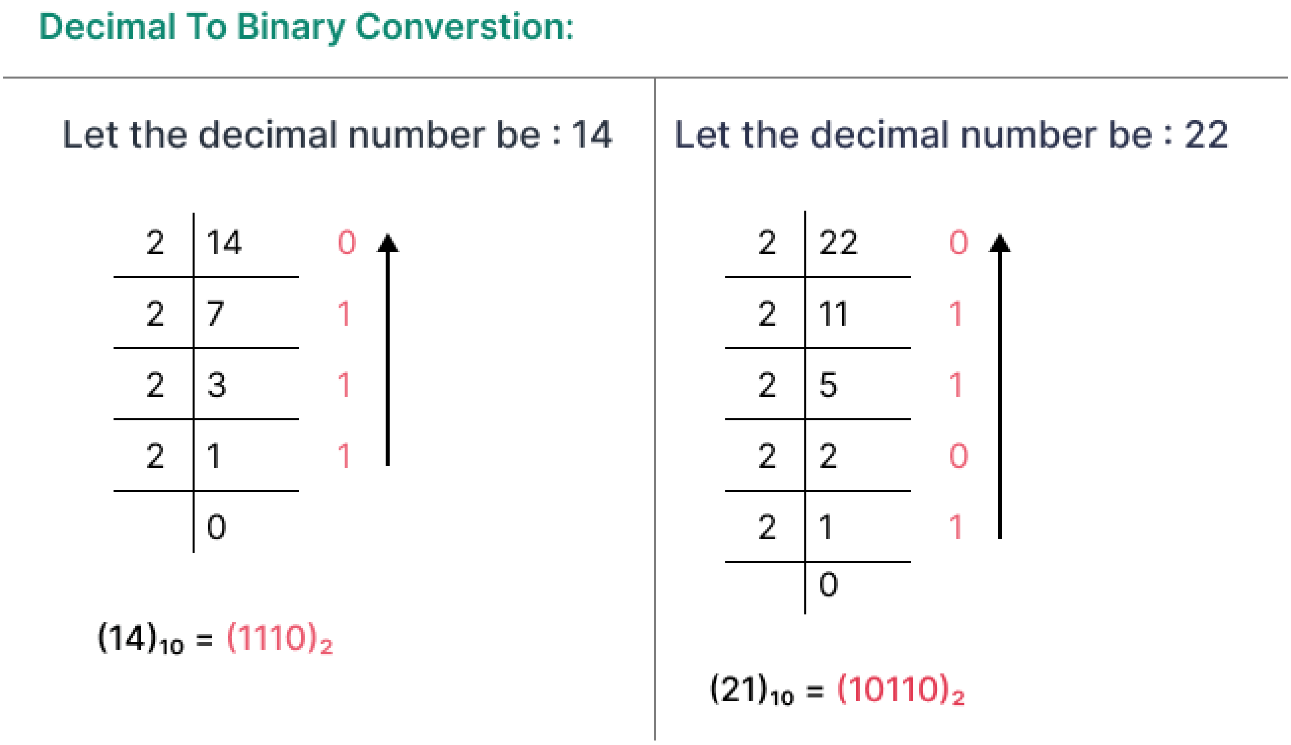
In [ ]:

03) WAP to reverse an integer number using recursion.

In [ ]:

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 [59]:

```
Enter Number : 42
Binary :  101010
```

Map , Filter , Reduce

map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

In [3]:

```python
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 [4]:

```python
# 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))
```

```
[1, 3, 5, 13]
[0, 2, 8]
```

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 [5]:

```
 # 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))
```

```
The sum of the list elements is : 17
The maximum element of the list is : 6
```