

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, Decana de América)

Facultad de Ciencias Matemáticas



Computación Científica – Estructura de Datos

Grupo 11 – Integrantes:

- Pozo Velarde, Luis Felipe**
- Luis Enrique Huayta Huillcahuare**

1) Suma recursiva de elementos de un arreglo

PSEUDOCODIGO:

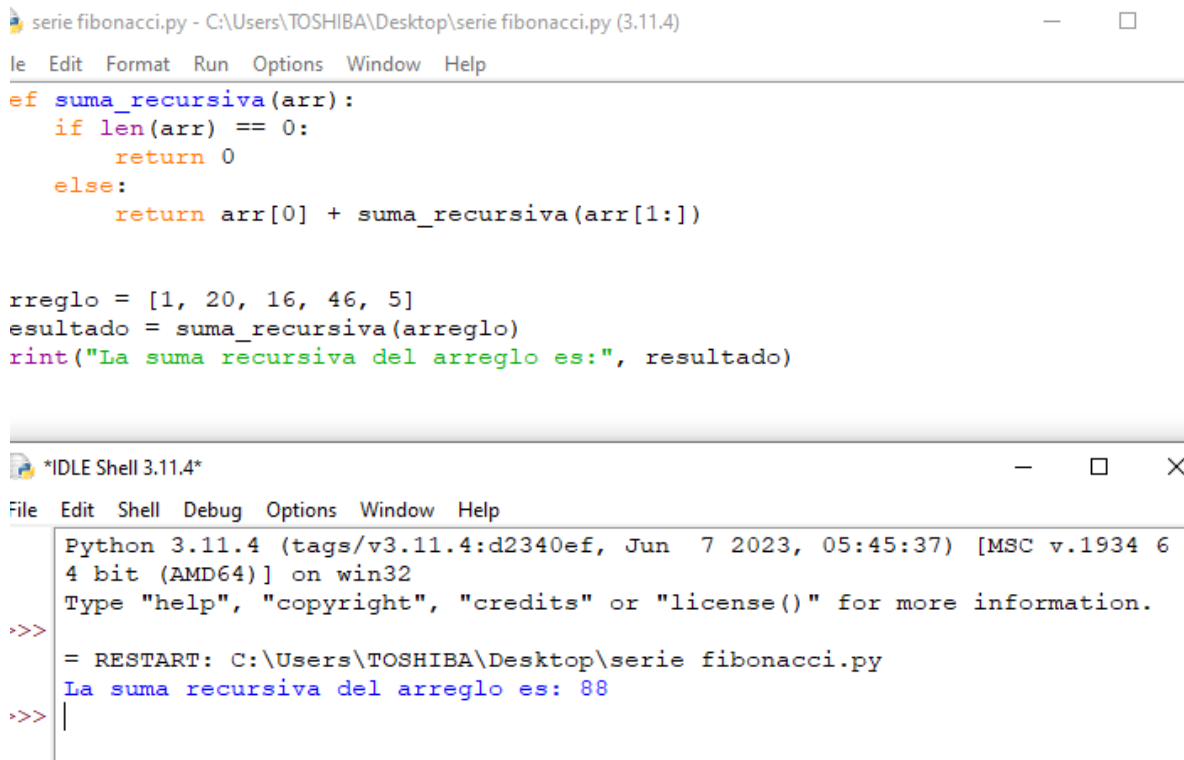
1. Se define una función llamada suma_recursiva que tome como argumento arreglo
2. Verifica si la longitud es igual a cero
3. Calcula la suma del primer elemento arreglo
4. Define un arreglo en números
5. Retorna el resultado de la suma

Prueba de escritorio:

arreglo = [1, 20, 16, 46, 5]

suma = 88

RESULTADO:



The image shows a screenshot of a Python IDE (likely IDLE) with two windows. The top window, titled 'serie fibonacci.py - C:\Users\TOSHIBA\Desktop\serie fibonacci.py (3.11.4)', contains the following Python code:

```
def suma_recursiva(arr):  
    if len(arr) == 0:  
        return 0  
    else:  
        return arr[0] + suma_recursiva(arr[1:])  
  
rreglo = [1, 20, 16, 46, 5]  
esultado = suma_recursiva(arreglo)  
rint("La suma recursiva del arreglo es:", resultado)
```

The bottom window, titled '*IDLE Shell 3.11.4*', shows the output of the script:

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 6  
4 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> = RESTART: C:\Users\TOSHIBA\Desktop\serie fibonacci.py  
La suma recursiva del arreglo es: 88  
>>> |
```

2)Serie Fibonacci

PSEUDOCODIGO:

1. La función Fibonacci es una función recursiva que calcula el enésimo termino
2. Si n es menor o igual a 1 la función devuelve n
3. De lo contrario la función se llama a si misma 2 veces
4. La variable números se establece en 5
5. Se imprime secuencia para indicar que demostrará la secuencia
6. Se ejecuta el bucle
7. Después de implementar todas las interacciones del bucle se habrán impreso los primeros 5 términos

Prueba de escritorio:

Iteración 0:

i = 0

fibonacci(0) = 0

Imprimir "0"

Iteración 1:

i = 1

fibonacci(1) = 1

Imprimir "1"

Iteración 2:

i = 2

fibonacci(2) = fibonacci(1) + fibonacci(0) = 1 + 0 = 1

Imprimir "1"

Iteración 3:

i = 3

fibonacci(3) = fibonacci(2) + fibonacci(1) = 1 + 1 = 2

Imprimir "2"

Iteración 4:

i = 4

fibonacci(4) = fibonacci(3) + fibonacci(2) = 2 + 1 = 3

Imprimir "3"

Secuencia de Fibonacci: 0 1 1 2 3

Código fuente:

```
def fibonacci(n):  
    if n <= 1:  
        return n  
  
    else:  
        return fibonacci(n - 1) + fibonacci(n - 2)
```

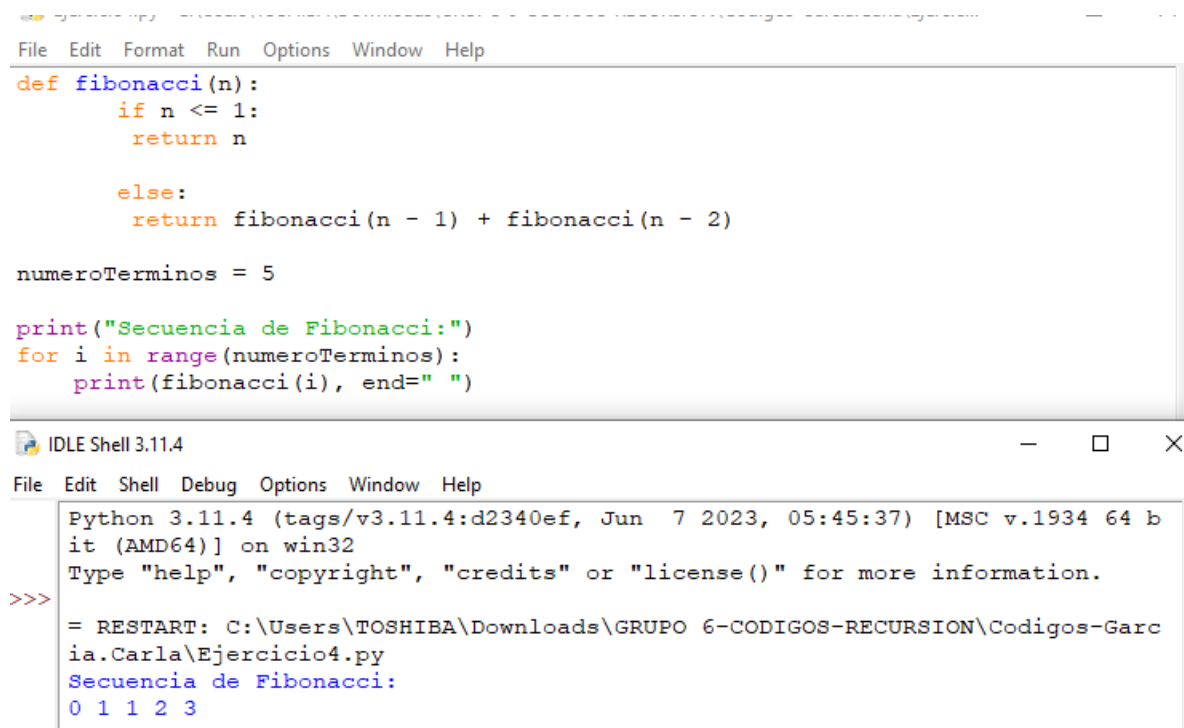
numeroTerminos = 5

print("Secuencia de Fibonacci:")

for i in range(numeroTerminos):

print(fibonacci(i), end=" ")

RESULTADO:



The screenshot shows the Python IDLE Shell 3.11.4 interface. The top window displays the source code for a Fibonacci sequence program. The bottom window shows the execution output, which includes the file path, the program name, and the resulting Fibonacci sequence: 0 1 1 2 3.

```
File Edit Format Run Options Window Help  
def fibonacci(n):  
    if n <= 1:  
        return n  
  
    else:  
        return fibonacci(n - 1) + fibonacci(n - 2)  
  
numeroTerminos = 5  
  
print("Secuencia de Fibonacci:")  
for i in range(numeroTerminos):  
    print(fibonacci(i), end=" ")  
  
IDLE Shell 3.11.4  
File Edit Shell Debug Options Window Help  
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 b  
it (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
= RESTART: C:\Users\TOSHIBA\Downloads\GRUPO 6-CODIGOS-RECURSION\Codigos-Garc  
ia.Carla\Ejercicio4.py  
Secuencia de Fibonacci:  
0 1 1 2 3  
...
```

3) Factorial

Factorial de un número es la multiplicación de todos los números enteros positivos, detrás de dicho número ejemplo:

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

PSEUDOCODIGO:

1. Pide un numero cualquiera
2. El código verifica si el numero ingresado es mayor o igual a cero utilizando la declaración if
3. Si el número no es negativo el código establece 2 variables
4. El código entra a bucle while esto se ejecuta mientras que x sea menor igual al número ingresado.
5. Dentro de dicho bucle se multiplica el valor actual de f con el de x, guarda el resultado en la variable f.
6. Después del que el bucle haya terminado de ejecutarse, el código muestra el resultado.

Prueba de escritorio:

1. Se coloca las variables
2. Se coloca un numero
n: 5
x: 1
f: 1
se evalúa que mientras que la variable x sea menor igual a uno
n = 5 y x = 1 como 1 es menor a 5 la condición se cumple
entonces multiplicamos lo que tiene f * x y esto se almacena en la variable f entonces se incrementa la variable x
n = 3
x = 2
f = 1
el ciclo retorna se vuelve y evaluamos si 2 es menor a 3 como se cumple entonces multiplicamos lo que tiene f * x y el resultado se almacena en f
n = 3
x = 2
f = 2

Código fuente:

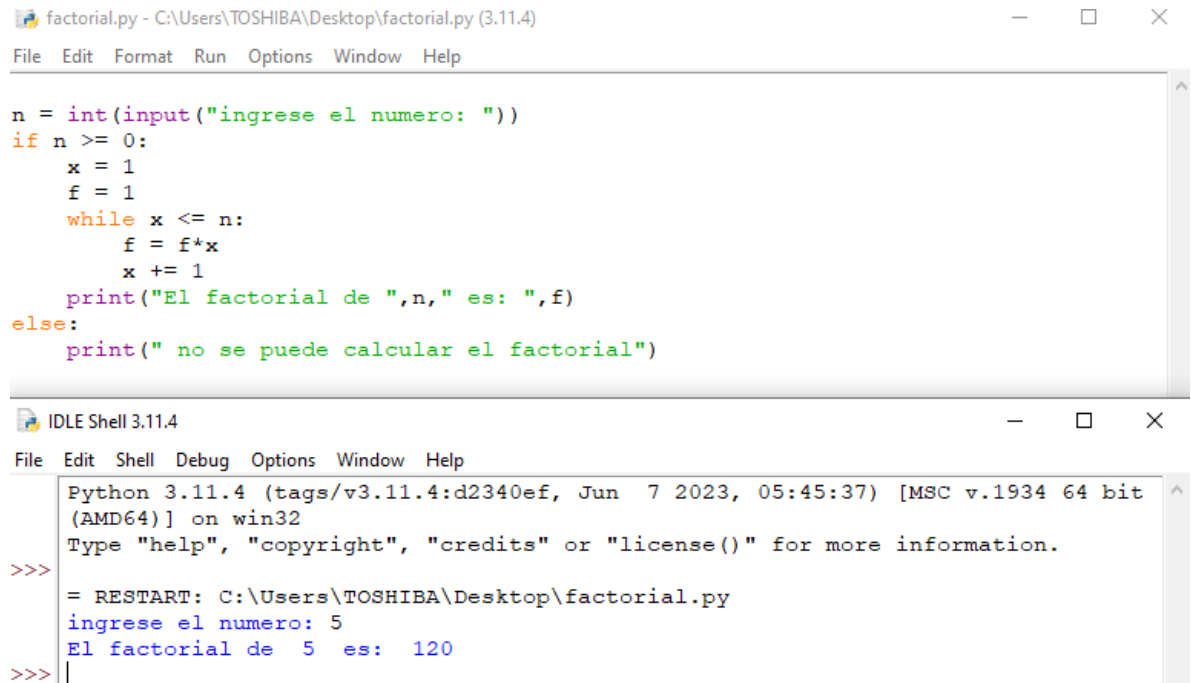
```
n = int(input("ingrese el numero: "))
if n >= 0:
    x = 1
    f = 1
```

```

while x <= n:
    f = f*x
    x += 1
print("El factorial de ",n," es: ",f)
else:
    print(" no se puede calcular el factorial")

```

RESULTADO:



The screenshot shows a Python IDE window titled 'factorial.py - C:\Users\TOSHIBA\Desktop\factorial.py (3.11.4)'. The code in the editor is as follows:

```

n = int(input("ingrese el numero: "))
if n >= 0:
    x = 1
    f = 1
    while x <= n:
        f = f*x
        x += 1
    print("El factorial de ",n," es: ",f)
else:
    print(" no se puede calcular el factorial")

```

Below the editor is the 'IDLE Shell 3.11.4' window. It shows the execution of the program:

```

Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\TOSHIBA\Desktop\factorial.py
ingrese el numero: 5
El factorial de 5 es: 120
>>>

```

4) Multiplicación de números naturales

PSEUDOCODIGO

1. Se solicita al usuario que ingrese la cantidad de factores a multiplicar
2. Se crea una lista vacía llamada números para almacenar los números que se ingresa
3. Se inicia un bucle for que se ejecutara la cantidad de veces especificadas
4. Dentro de este bucle, se solicita que se ingrese cada numero
5. Cada número ingresado se agrega a dicha lista
6. Se inicia una variable
7. Dentro de este bucle se inicia otro que recorre cada número de la lista
8. Una vez se han multiplicado todos los números de la lista
9. Finalmente se muestra el resultado

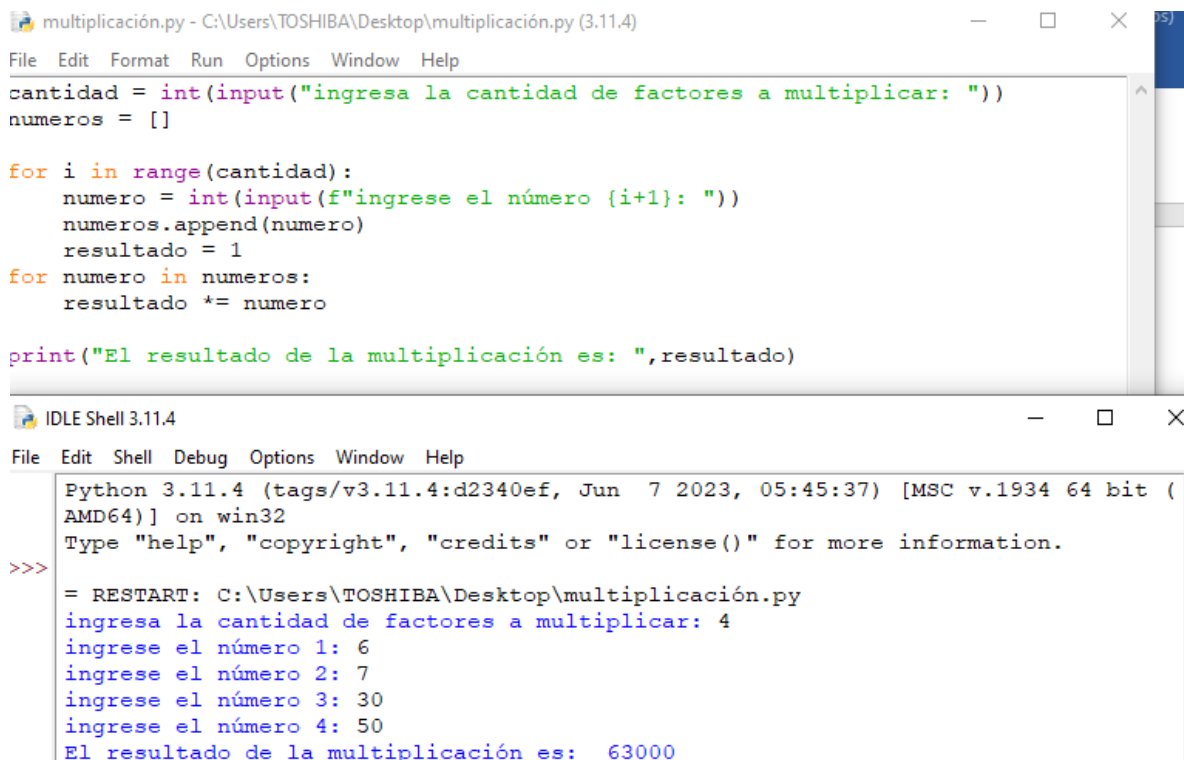
PRUEBA DE ESCRITORIO:

$$6 * 7 * 30 * 50 = 63\ 000$$

Código fuente:

```
cantidad = int(input("ingresa la cantidad de factores a multiplicar: "))
numeros = []
for i in range(cantidad):
    numero = int(input(f"ingrese el número {i+1}: "))
    numeros.append(numero)
    resultado = 1
for numero in numeros:
    resultado *= numero
print("El resultado de la multiplicación es: ",resultado)
```

Resultado:



The screenshot displays two windows from a Python IDE. The top window, titled 'multiplicación.py - C:\Users\TOSHIBA\Desktop\multiplicación.py (3.11.4)', shows the source code for a program that calculates the product of a user-defined number of factors. The code prompts the user for the number of factors, then for each factor, and finally prints the result. The bottom window, titled 'IDLE Shell 3.11.4', shows the program's execution. It starts with the Python version and system information, followed by a restart message. The user enters 4 for the number of factors, then provides the factors 6, 7, 30, and 50. The final output is 'El resultado de la multiplicación es: 63000'.

```
multiplicación.py - C:\Users\TOSHIBA\Desktop\multiplicación.py (3.11.4)
File Edit Format Run Options Window Help
cantidad = int(input("ingresa la cantidad de factores a multiplicar: "))
numeros = []

for i in range(cantidad):
    numero = int(input(f"ingrese el número {i+1}: "))
    numeros.append(numero)
    resultado = 1
for numero in numeros:
    resultado *= numero

print("El resultado de la multiplicación es: ",resultado)

IDLE Shell 3.11.4
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\TOSHIBA\Desktop\multiplicación.py
ingresa la cantidad de factores a multiplicar: 4
ingrese el número 1: 6
ingrese el número 2: 7
ingrese el número 3: 30
ingrese el número 4: 50
El resultado de la multiplicación es: 63000
```

RECUSIÓN

POTENCIA:

DEFINA UN MÉTODO RECURSIVO PARA ENCONTRAR **b^d**.

PSEUDOCÓDIGO:

Datos de entrada:

b = Numero base

d = Exponente de la base

Variables

Darle un valor a "base"

Darle un valor a "exponente"

Definimos una función "potencia" (base,exponente)

Si "exponente" == 0

Devolver 1

Si "exponente" == 1

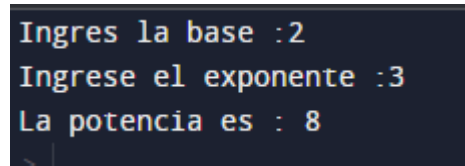
Devolver "base"

Si no se cumplen las anteriores

Un "numero" = b* potencia(b,d-1)

Devolver "numero"

Imprimir La potencia es función "potencia"

PRUEBA DE ESCRITORIO:

```
Ingres la base :2
Ingrese el exponente :3
La potencia es : 8
```

CÓDIGO FUENTE:

```
b = int(input("Ingresa la base :"))
```

```
d = int(input("Ingresa el exponente :"))
```

```
def potencia(b,d):
```

```
    if d == 0:
```

```
        return 1
```

```
    if d == 1:
```

```
        return b
```

```
    else:
```

```
        numero = b * potencia(b,d-1)
```

```
        return numero
```

```
print("La potencia es :",potencia(b,d))
```

MÁXIMO COMÚN DIVISOR (MCD):

DEFINIR UN MÉTODO RECURSIVO PARA ENCONTRAR EL MÁXIMO COMÚN DIVISOR ENTRE A Y B.

PSEUDOCÓDIGO:

Datos de entrada:

a = numero 1

b = numero 2

Variables:

Darle un valor a "numero 1"

Darle un valor a "numero 2"

Definimos una función MCD (numero 1, numero 2)

 Si "numero 2" == 0

 Retornar "numero 1"

 Sino cumple entonces

 Retornar función MCD (Numero 1, residuo del numero 1 y 2)

Imprimir función MCD (numero 1, numero 2)

PRUEBA DE ESCRITORIO:

a = 14

b = 245

```
Ingrese el primer numero :14
Ingrese el segundo numero :245
7
```

CÓDIGO FUENTE:

```
a=int(input("Ingrese el primer numero :"))
```

```
b=int(input("Ingrese el segundo numero :"))
```

```
def maximo_comun_divisor_recursivo(a, b):
```

```
    if b == 0:
```

```
        return a
```

```
    else:
```

```
        return maximo_comun_divisor_recursivo(b, a % b)
```

```
print (maximo_comun_divisor_recursivo(a, b))
```

ÁRBOL BINARIO:

BÚSQUEDA BINARIA RECURSIVA.

PSEUDOCÓDIGO:

Datos de entrada:

tamaño_lista = Define el tamaño que va a tener la lista

L = Lista vacia

Variables:

Damos el valor de tamaño de la lista

Definir una lista vacia

Para x en el rango del tamaño de la lista

 Creamos una variable "números" para ingresar números

 Almacenamos los "números" en la Lista vacia

Imprimimos la nueva lista llena

Damos el valor del numero que queremos buscar en la lista

Definimos una función de Búsqueda binaria (Numero a buscar, Lista, posInicial,posFinal)

 Si "posición inicial" es <= "posición final"

 Posición central es = ("posición inicial" + "posición final") // 2

 Si el valor de la posición del medio de la lista es menor al numero a buscar

 Retornar "Búsqueda binaria" (NUM,Lista, mid+1, posFinal)

 Si el valor de la posición del medio de la lista es mayor al numero a buscar

 Retornar "Búsqueda binaria" (NUM,Lista, posInicial, mid-1)

 Si no se cumplen las anteriores

 Retornar la posición media ya que no es ni mayor ni menor

 Sino se cumple ninguna

 Retornar -1

Creamos una variable "index" que tomara el valor de la función Búsqueda Binaria

Si "index" == -1

 Imprimir que no existe el valor, para buscar, en la lista

Si no se cumple entonces

 Imprimir que el valor solicitado se encuentra en la posición "index"

PRUEBA DE ESCRITORIO:

tamaño_lista = 4

L = [4 ,5 ,6 ,7]

a = 6

```
¿Cuantos numeros desea ingresar? :4
Ingrese numero para la lista :4
Ingrese numero para la lista :5
Ingrese numero para la lista :6
Ingrese numero para la lista :7
[4, 5, 6, 7]
Ingresa el numero a buscar :6
El valor solicitado se encuentra en la posición : 2
PS C:\Users\Felipe Pozo\Documents\Programacion>
```

CÓDIGO FUENTE:

```
tamaño_lista = int(input("¿Cuantos numeros desea ingresar? :"))
L = []
```

```
for x in range (tamaño_lista):
    numeros = int(input("Ingrese numero para la lista :"))
    L.append(numeros)
print(L)
```

```
a=int(input("Ingresa el numero a buscar :"))
```

```
def binary_search(a,L,low, high):
```

```
    if low <= high:
```

```
        mid = (high+low)//2
```

```
        if L[mid] < a:
```

```
            return binary_search(a,L,mid+1,high)
```

```
        elif L[mid] > a:
```

```
            return binary_search(a,L,low,mid-1)
```

```
        else:
```

```
            return mid
```

```
    return -1
```

```
index = binary_search(a,L,0,len(L))
```

```
if index == -1:
```

```
    print("El numero solicitado no existe.")
```

```
else:
```

```
print("El valor solicitado se encuentra en la posición :", index)
```