**ESCUELA POLITÉCNICA NACIONAL**

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**ALGORITMOS Y ESTRUCTURAS DE DATOS (TSDS)**

|  |  |
| --- | --- |
|  | |
| ASIGNATURA: | ALGORITMOS Y ESTRUCTURAS DE DATOS |
| PROFESOR:  FECHA: | Ing. Lorena Chulde  15-11-2024 |
| PERÍODO ACADÉMICO: | 2024-B |
|  | |

**Taller**

|  |
| --- |
|  |
| **TÍTULO:** |
|  |
|  |
|  |
|  |

**Estudiante**

Felipe Zapata

**Parte 1: Funciones. Paso de parámetro por valor y por referencia**

# Paso por valor y referencia

Dependiendo del tipo de dato que enviemos a la función, podemos diferenciar dos comportamientos:

* **Paso por valor**: Se crea una copia local de la variable dentro de la función.
* **Paso por referencia**: Se maneja directamente la variable, los cambios realizados dentro de la función le afectarán también fuera.

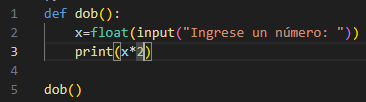
Tradicionalmente:

* **Los tipos simples se pasan por valor**: Enteros, flotantes, cadenas, lógicos...
* **Los tipos compuestos se pasan por referencia**: Listas, diccionarios, conjuntos...

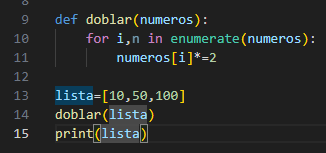
## Paso de parámetro por valor

Los números se pasan por valor y crean una copia dentro de la función, por eso no les afecta externamente lo que hagamos con ellos:

**Ejemplo:** crear una función doblar valor que multiplique a un número por dos





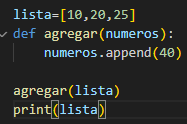




## Paso de parámetro por referencia

Sin embargo las listas u otras colecciones, al ser tipos compuestos se pasan por referencia, y si las modificamos dentro de la función estaremos modificándolas también fuera:

**Ejemplo:** crear una función doblar valor que multiplique los elementos de una lista





**Uso de la función id().**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

El ejemplo anterior nos podría llevar a pensar que si en vez de añadir un elemento a x, hacemos x=[], estaríamos destruyendo la lista original. Sin embargo esto no es cierto.

Texto

Descripción generada automáticamente

Texto

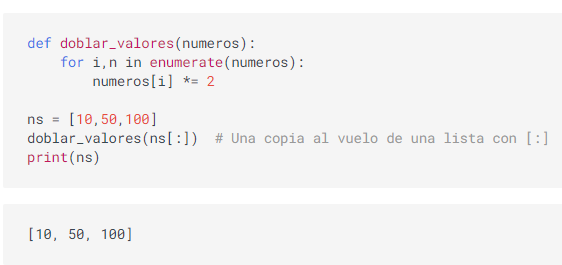
Descripción generada automáticamente

Para modificar los tipos simples podemos devolverlos modificados y reasignarlos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

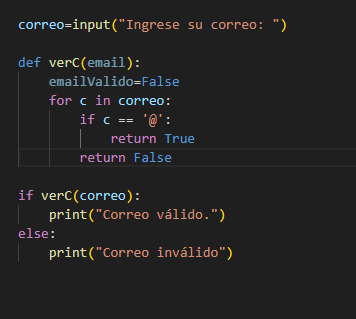
Y en el caso de los tipos compuestos, podemos evitar la modificación enviando una copia:



**Parte 2: Ejercicios con Funciones**

1. Solicitar al usuario que ingrese su dirección email. Imprimir un mensaje indicando si la dirección es válida o no, valiéndose de una función para decidirlo.

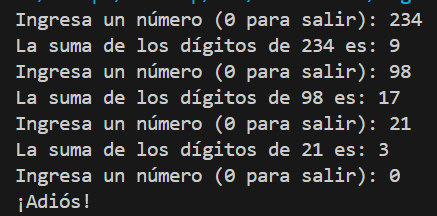
Nota: El correo se considerará válido si tiene el símbolo "@".





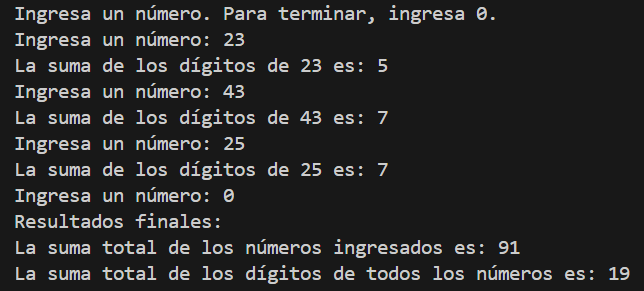
1. Pedir que el usuario ingrese valores hasta que ingrese el 0. Por cada valor, mostrar la suma de sus dígitos. Use una función que sume.

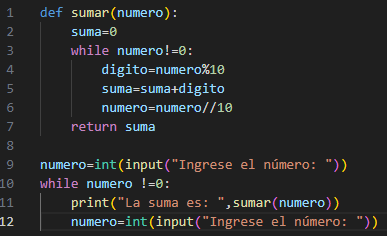
<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E3.py>



1. Solicitar al usuario que ingrese valores hasta que ingrese el cero. Por cada valor ingresado sumar sus dígitos y mostrarlos en pantalla. Al finalizar, calcular la suma total de todos los valores ingresados y la suma de sus dígitos. (TAREA)

<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E4.py>







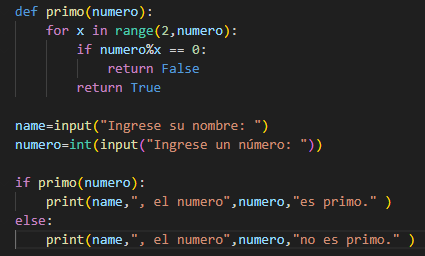
1. Solicitar al usuario que ingrese un número entero e imprimir en pantalla si es primo o no. Utilice una función booleana que lo decida.



119 2 119 7  **NO**

19 59 49 17

1 0

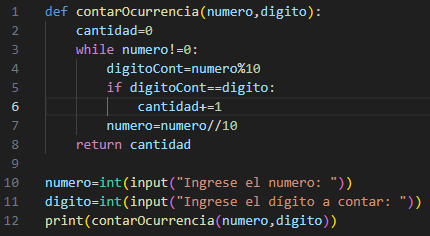




1. Imprimir en pantalla la cantidad de ocurrencias de un dígito que se encuentra en un número entero ingresado por el usuario.

Nota: El usuario digitará tanto el número entero como el dígito.

Utilice una función que calcule la frecuencia.

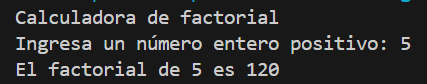




1. Calcular el factorial de un numero ingresado por el usuario.

Nota: Utilice una o más funciones, según sea necesario.

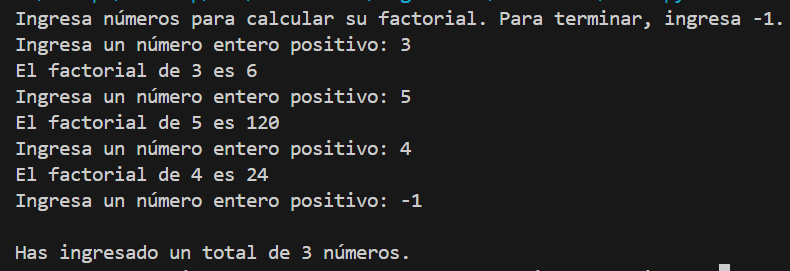
<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E5.py>



1. Escribir un programa que pida números al usuario, mostrar el factorial de cada uno y, al finalizar, la cantidad total de números leídos en total.

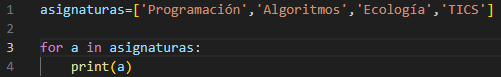
Nota: Utilice una o más funciones, si es necesario. Tarea

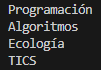
<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E6.py>



**Parte 3: Ejercicios con Listas**

1. Programa que imprime las asignaturas definidas en una lista.



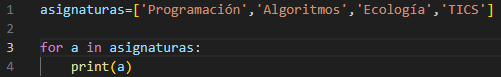


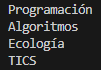
1. Imprimir el tamaño de una lista dada



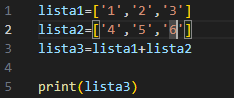


1. Programa que imprime las asignaturas con for (elemento por elemento)





1. Unir dos listas y mostrar en una tercera lista



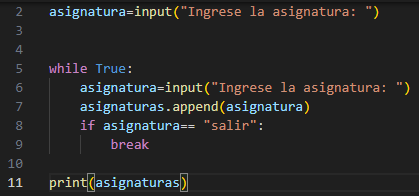


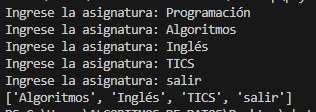
1. Agregar un elemento quemado a una lista usando append()

<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E7.py>

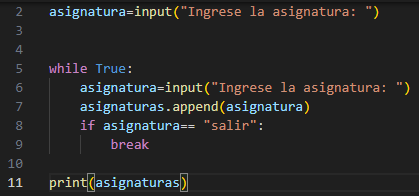


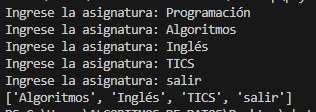
1. Agregar un elemento pidiendo al usuario a una lista usando append()





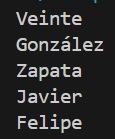
1. Insertar las asignaturas en la lista e imprimirlas.



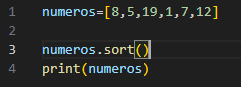


1. Crear una lista e inicialízala con 5 cadenas de caracteres. Copia los elementos de la lista en otra lista, pero en orden inverso, y muestra sus elementos por la pantalla.

<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E8.py>



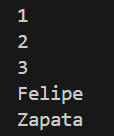
1. **Con función sort:** ordenar un arreglo de elementos de menor a mayor





1. Crear una fn para recorrer una lista.

<https://github.com/FelipeZapata137/Algoritmos-/blob/main/Parcial%202/D7E9.py>



**Método burbuja:**

Crear una función **“ordenar ()”** para ordenar elementos de un array de menor a mayor.

def ordenar(array):

    tamanio = len(array)

#for para recorrer las posiciones de 0 a 5

    for i in range(0,tamanio-1):

        #con este bucle comparamos

        for j in range(0,tamanio-1):

            if array[j] > array[j+1]:

                aux = array[j]

                array[j] = array[j+1]

                array[j+1] = aux

    return array

numeros = [6,3,8,2,7]

print("la lista original es: ", numeros)

#ordenar(numeros)

print("la lista ordenada es: ", ordenar(numeros))

Explicar el algoritmo paso a paso

1. Definición de la función ordenar
2. Entrada: Se recibe una lista llamada 'array'.
3. Variable 'tamanio': Se calcula la longitud de la lista con len(array).
4. 2. Primer bucle for (índice i)

• Función: Recorre la lista un número de veces igual a tamano - 1.

• Rango: range(0, tamanio - 1) (es decir, desde el índice 0 hasta el penúltimo índice).

Este bucle externo asegura que todas las posiciones se comparen las veces necesarias.

1. Segundo bucle for (índice j)

• Función: En cada iteración del primer bucle, este segundo bucle recorre la lista desde el índice 0 hasta el penúltimo elemento no ordenado (por eso también usa tamanio - 1 como límite).

• Propósito: Compara pares consecutivos de elementos.

1. Comparación en el if

• Condición: if array[j] > array[j + 1]

- Si el elemento actual es mayor que el siguiente, entonces hay que intercambiarlos.

• Intercambio:

1. Se guarda temporalmente el valor de array[j] en aux.
2. array[j] toma el valor de array[j + 1].
3. array[j + 1] recibe el valor almacenado en aux.
4. Retorno

Al finalizar todos los bucles, la lista estará ordenada y se devuelve con return array.

Ejecución del código

1. Lista original: numeros = [6, 3, 8, 2, 7].
2. Salida intermedia: El mensaje 'la lista original es: ' imprime la lista original.
3. Llamada a ordenar(numeros): Ordena la lista usando el algoritmo explicado.
4. Resultado final: El mensaje 'la lista ordenada es: ' imprime la lista ordenada, que será [2, 3, 6, 7, 8].