1. Sección- Programas básicos en Python

Unai Pérez-Goya
Área de Lenguajes y Sistemas Informáticos

Curso 2024/2025

1. Sección-

Programas básicos en Python

Sesión VI: Repaso y Recursividad

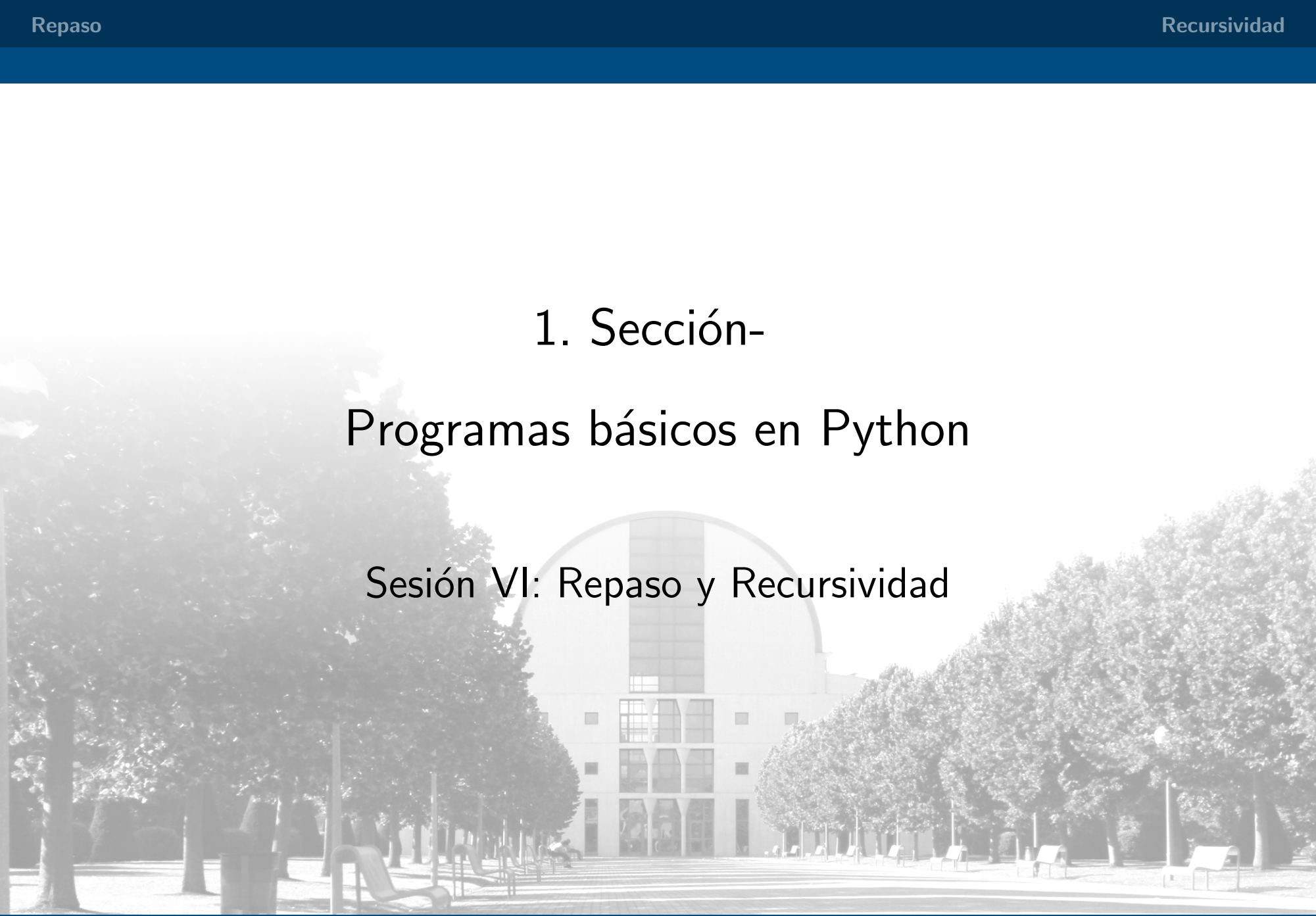


Tabla de contenidos

Repaso

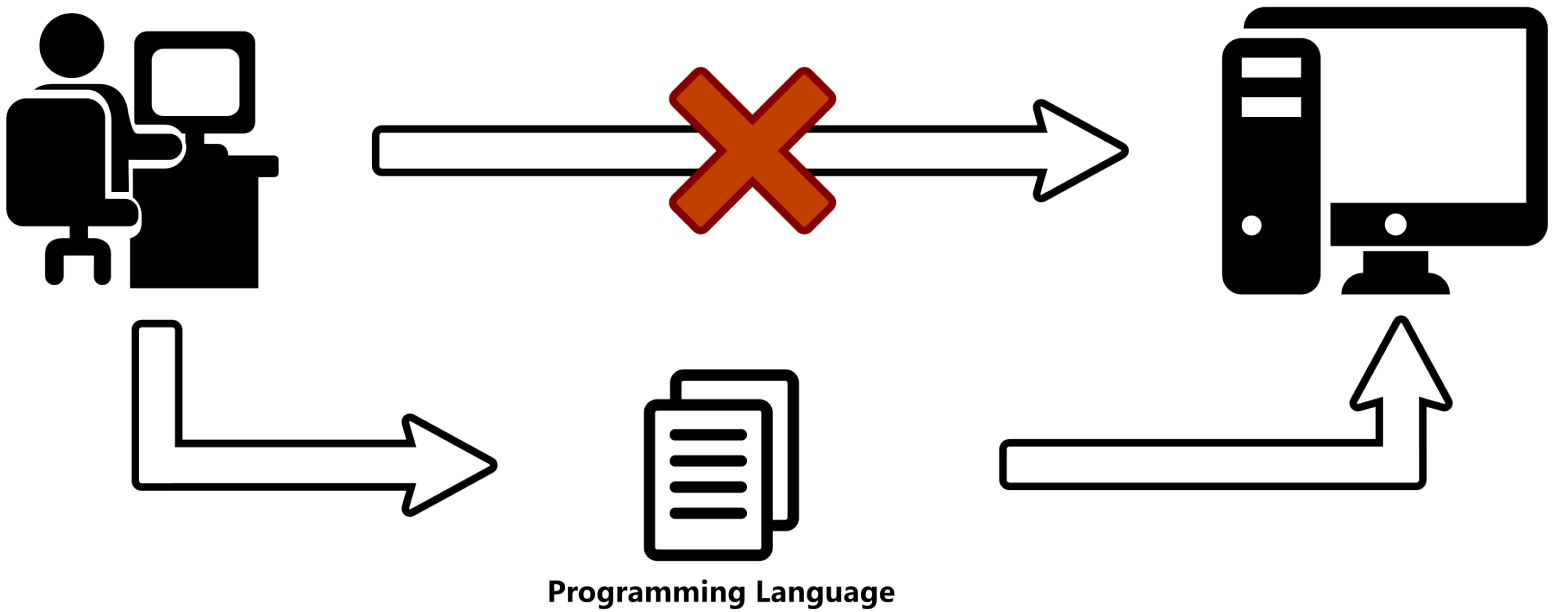
Sesión I

Lista de Contenidos

- Lenguajes de programación;
- Lenguajes compilados vs. interpretados;
- Introducción a Python;
- Algunos ejemplos en Python.

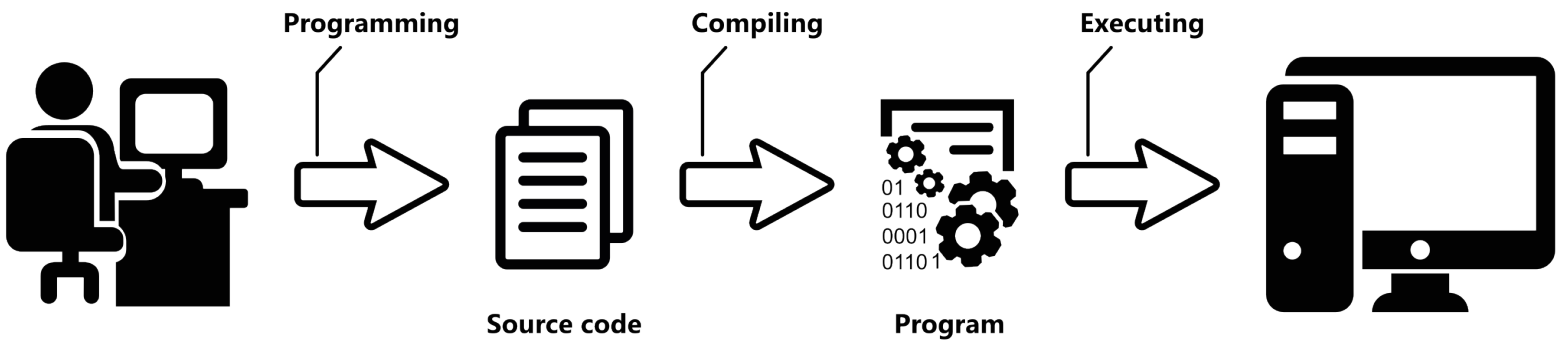
Sesión I

Lenguajes de Programación



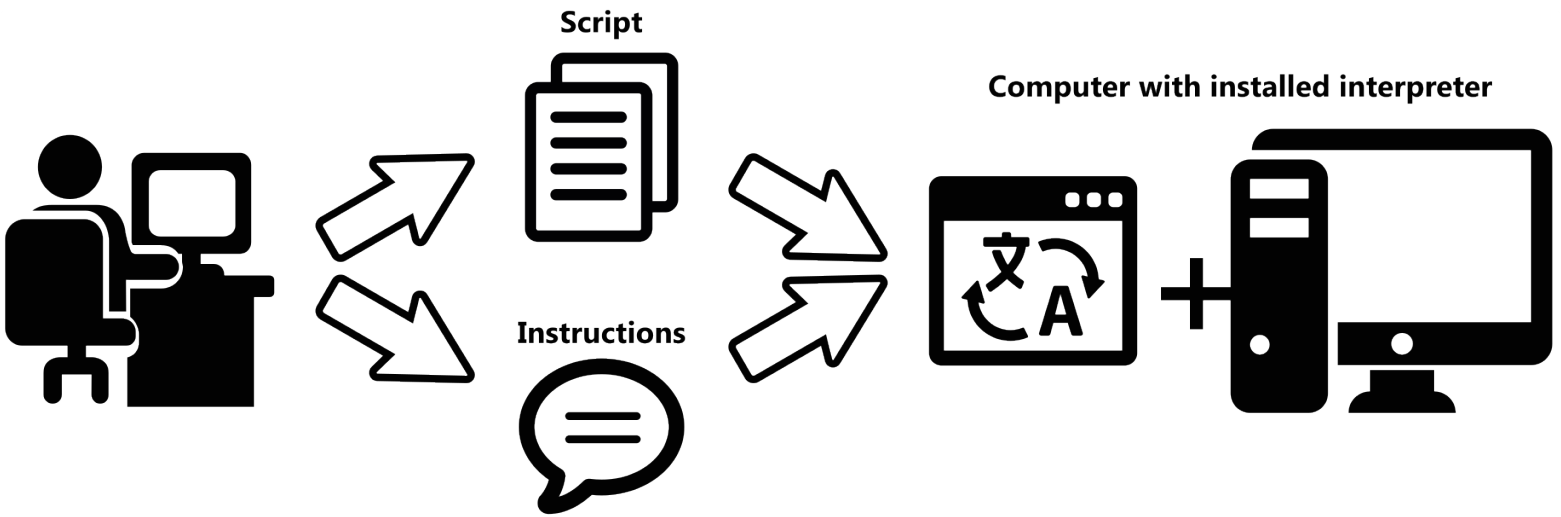
Sesión I

Proceso de compilación



Sesión I

Compilados vs. interpretados



Sesión II

Resumen de contenidos

- Variables en Python y su uso
- Asignación y declaración
- Tipos de variables.
  - Variables numéricas
  - Variables de texto
  - Booleanos
- Constantes

Ejercicio 1: Manipulación de Diferentes Tipos de Datos

- **Ejercicio 1** Crea variables de diferentes tipos de datos (números enteros, flotantes, cadenas de texto y booleanos). Aquí están las tareas a realizar:
  - Asigna un número entero a una variable y un número `float` a otra.
  - Convierte el número entero a una cadena de texto y concaténalo con el número flotante convertido a cadena.
  - Muestra el resultado de la concatenación y el tipo de datos del resultado.

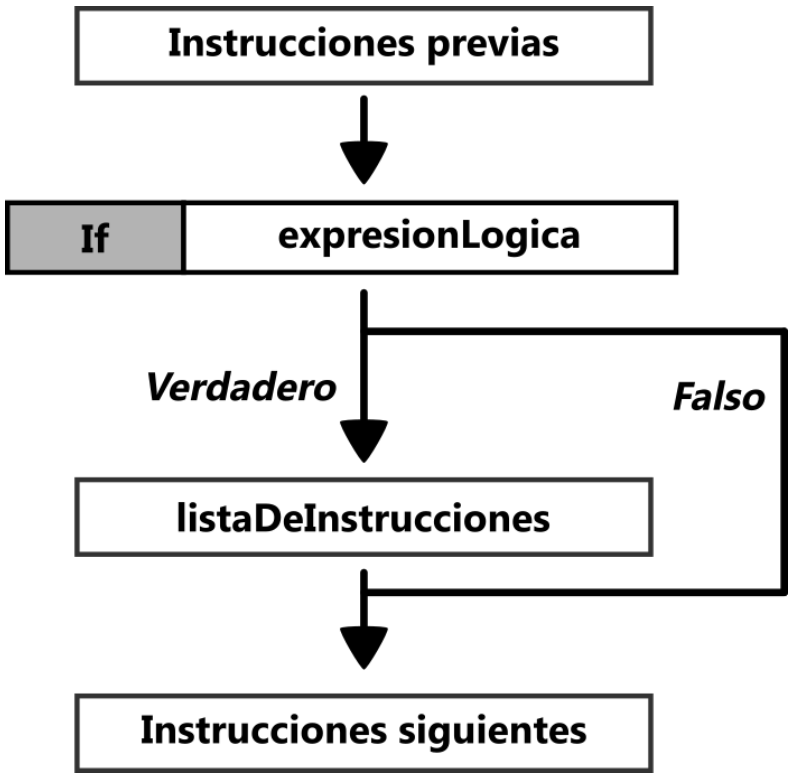
Ejercicio 2: Conversión de Tipos de Datos

- **Ejercicio 2** Realiza conversiones entre tipos de datos y usa las conversiones en cálculos. Aquí están las tareas a realizar:
  - Asigna una cadena de texto que pueda representar un número entero.
  - Convierte la cadena a un número entero y calcula la suma con otro número entero.
  - Convierte el resultado de la suma a un número `float` y muestra el resultado final junto con su tipo de datos.

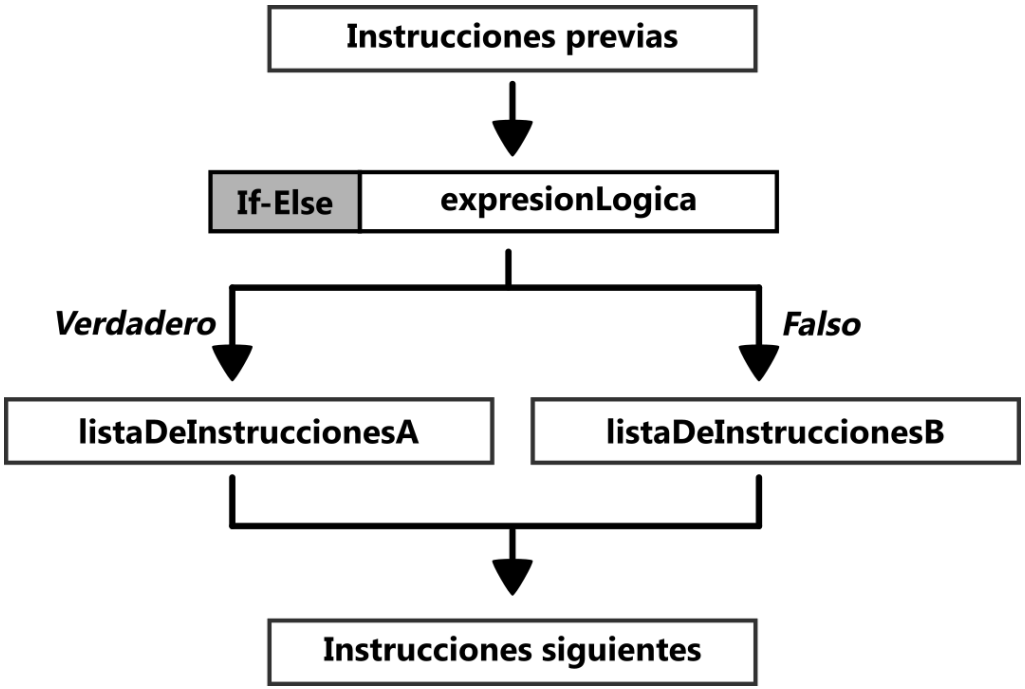
Sesión III

- Programas secuenciales y sus límites
- Estructuras de control:
  1. Estructura `if`;
  2. Estructura `if-else`;
- Funciones, encapsulación y ámbito de las variables

Sesión III


Instrucciones previas
If
expresionLogica
Verdadero
Falso
listaDeInstrucciones
Instrucciones siguientes

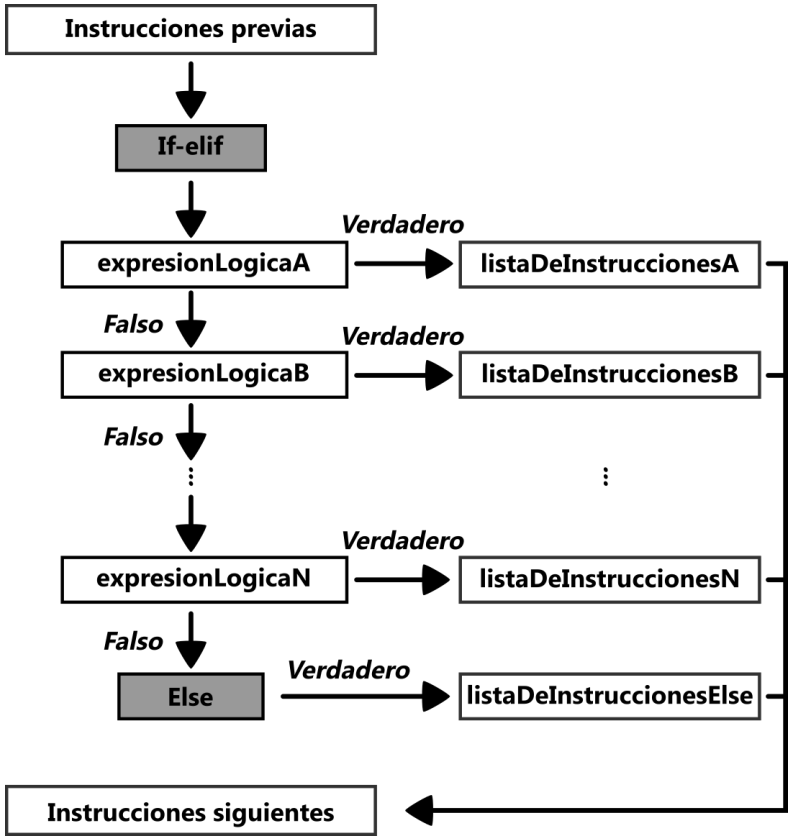
Sesión III


Instrucciones previas
If-Else
expresionLogica
Verdadero
Falso
listaDeInstruccionesA
listaDeInstruccionesB
Instrucciones siguientes

Sesión IV

- Interacción con el usuario;
- Estructura `if-elif`.
- Estructura `if-elif-else`.
- Funciones sobre variables de texto `str`.

Sesión IV


Instrucciones previas
If-elif
Verdadero
expresionLogicaA
listaDeInstruccionesA
Falso
Verdadero
expresionLogicaB
listaDeInstruccionesB
Falso
Verdadero
expresionLogicaN
listaDeInstruccionesN
Falso
Verdadero
Else
listaDeInstruccionesElse
Instrucciones siguientes

Ejercicio 3: Calculadora Básica

- **Ejercicio 3** Crea un programa que funcione como una calculadora básica. El programa debe solicitar al usuario que ingrese dos números y una operación matemática (suma, resta, multiplicación o división). Luego, debe realizar la operación correspondiente y mostrar el resultado.

Sesión V y VI

- Listas `list`;
- Estructura For [each] `for`
- For in range() `for`
- Estructura `while`

Ejercicio 4: Eliminar Elementos de una Lista

- **Ejercicio 4** Escribe una función llamada `eliminar_elemento` que reciba dos parámetros, una lista y un elemento. La función debe eliminar la primera aparición de ese elemento en la lista, sin utilizar el método `.remove()`. El programa debe modificar y devolver la lista.

Ejercicio 5: Factorial

- **Ejercicio 5** Crea una función que calcule el factorial de un número.

Factorial

- Para abordar el ejercicio, primero debemos entender el cálculo del factorial
- El factorial de un número entero no negativo n (denotado como $n!$)
- Se define como el producto de todos los números positivos desde 1 hasta n.
- Definición formal:
  - $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$ cuando $n > 0$
  - $0! = 1$ (uso común)

Recursividad

Una función que se llama a sí misma

- Si un problema grande puede dividirse en instancias más pequeñas del mismo problema... eso significa que el problema puede resolverse de manera recursiva. Un ejemplo es el factorial.
- La mayoría de los lenguajes de programación permiten la recursividad, lo que permite que una función se llame a sí misma.
- Un algoritmo recursivo es aquel que expresa la solución a un problema en forma de una llamada a sí mismo.
- Tiene las siguientes características:
  - **Caso base:** La función debe tener al menos un caso base, que detendrá la recursión, evitando llamadas infinitas.
  - **Llamada recursiva:** La función se llama a sí misma, normalmente con un argumento modificado que se aproxima al caso base.

Ejercicio 6: Factorial utilizando recursividad

- **Ejercicio 6** Crea una función que calcule el factorial de un número utilizando recursividad. Es obligatorio resolver el ejercicio usando recursividad.

Factorial

- El factorial puede definirse de manera recursiva.
- Esto significa que la función se llama a sí misma para resolver el problema.
- La idea es descomponer el cálculo del factorial en términos del factorial del número y los factoriales de números menores.
- Definición básica:
  - El factorial de un número entero no negativo n se denota como $n!$.
  - Se define de la siguiente manera:
    - $n! = n \times (n-1)!$ cuando $n > 0$
    - $0! = 1$ (por convención)

Ejemplo 1: Solicitar valores hasta obtener el correcto

- **Ejemplo 1** Escribe un programa que solicite un valor entero al usuario y continúe solicitándolo hasta que se proporcione un valor adecuado.

Ejemplo 1: Solución

```
1 def solicitar_numero_entero():
2   numero = input('Introduce un numero:')
3   if str.isdigit(numero):
4     numero = int(numero)
5     return numero
6   else:
7     print('No has introducido un numero entero.')
8   return solicitar_numero_entero()
```

Código: Solicitar un número entero utilizando recursividad.

Ejercicio 7: Solicitar operador

- **Ejercicio 7** Crea una función que solicite un operador matemático válido hasta que se obtenga uno adecuado.

Ejemplo 2: Cuenta regresiva

- **Ejemplo 2** Define una función que realice una cuenta regresiva. Este ejercicio debe completarse utilizando recursividad.

Ejemplo 2: Solución

- Utilizando recursividad, se puede repetir una lista de instrucciones `n` veces.

```
1 def cuenta_regresiva(n):
2     if n == 0:
3         return None
4     else:
5         print(n)
6     cuenta_regresiva(n-1)
7
8 cuenta_regresiva(5)
```

Código: Solución de cuenta regresiva.

Ejercicio 8: División con decimales

- **Ejercicio 8** Realiza divisiones con decimales utilizando operadores enteros. El programa calculará hasta un máximo de 5 decimales.