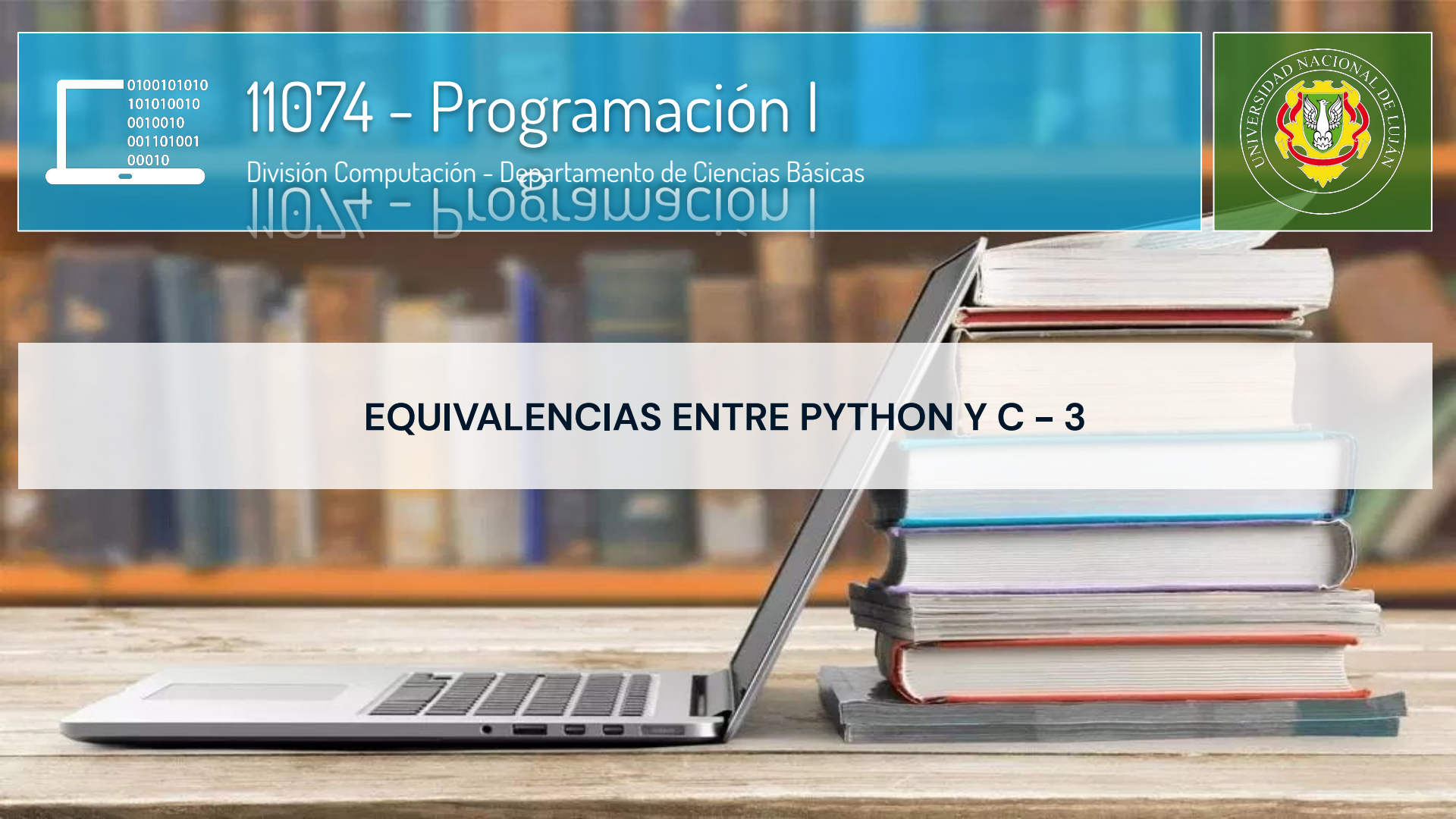

EQUIVALENCIAS ENTRE PYTHON Y C – 3

Equivalencias entre Python y C- Funciones

Repaso de Conceptos Básicos

Piezas de un Programa

❖ Funciones

- En programación, una función podríamos definirla como una parte o porción de un programa que calcula un valor de manera independiente al resto del programa.

- Una función tiene tres componentes importantes:

  Los **parámetros**(valores que recibe la función como entrada)

  El **código** (instrucciones y operaciones que tiene la función)

  El **resultado** (o valor de retorno o sea el valor final que entrega la función).

Piezas de un Programa

❖ Funciones

Una función puede ser llamada desde cualquier parte del programa que la contiene, posterior a su declaración/definición. **Se la invoca** con su **nombre, seguido de una lista opcional de argumentos (o parámetros)**. Los argumentos van entre paréntesis y, si hubiera más de uno, separados por comas.

Debe haber el mismo número de parámetros en la declaración (**Parámetros Formales**), que en la llamada (**Parámetros Actuales**). Además deben ser del mismo tipo y existe una relación de orden, son posicionales (**correspondencia en orden, tipo y número**).

Piezas de un Programa

- **Definición de función:**

  En **Python:**

```
def <nombre_función>(lista de parámetros):
    <cuerpo_función>
    return   #puede retornar valores o no
```

  En **C:**

```
<tipo_de_retorno><nombre_función> (<lista de parámetros>){

<implementación de la función>
}
```

Piezas de un Programa

❖ Consideraciones:

En **C,** una función **siempr**e retorna **un solo valo**r (en casos eventuales podría no devolver nada si se usa void como tipo de retorno). Si fuese necesario devolver más de un valor, debería realizarse por medio de los parámetros, que dejarían de ser sólo de entrada, para convertirse en parámetros de entrada y salida. Esto se implementa pasando referencias a los argumentos (no los argumentos en sí).

Parámetros: hay 3 tipos:

Entrada:valores proporcionados por el llamador, **Salida**:valores calculados en la función y devueltos al llamador para su proceso y **Entrada/Salida**: son proporcionados por el llamador, modificados por el procedimiento y devueltos al llamador.

Piezas de un Programa

❖ Consideraciones:

Los parámetros se clasifican en **Parámetros por Valor** y **Parámetros Variable o por Referencia.**

Por Valor: se enumeran por su nombre en la declaración, son unidireccionales, proporcionan información al procedimiento pero no devuelven valores, son de **Entrada**.

Por Referencia: se usan tanto para transmitir como para recibir valores entre llamador y llamados, pueden actuar como de **Salida** o **Entrada/Salida**.

Es usual que las funciones se definan a continuación de la función main(), en cuyo caso es necesario, para cada una de ellas, escribir el prototipo antes del main(), como se muestra en el ejemplo siguiente:

Piezas de un Programa

```
nt main(){
    int operando1, operando2, resultado;
    operando1 = 5;
    operando2 = 10;
    resultado = suma(operando1, operando2);
    printf("%d + %d = %d \n", operando1, operando2, resultado);
    return 0;
}#include <stdio.h>
/* Prototipo de la función 'suma()' */
int suma(int a, int b);
i
int suma(int a, int b{
    int resultado;
    resultado = a + b;
    return resultado;
}
```

Piezas de un Programa

❖ Resumiendo:

La lista de **parámetros formales** está siempre entre **()**. Puede tener 1 o más identificadores. Los identificadores van separados por **, .**

La lista de **parámetros actuales** puede ser una lista de variables, constantes o expresiones.

La **correspondencia entre parámetros actuales y formales** se da por la **posición**, o sea que **no tienen por qué** tener el mismo nombre.

Piezas de un Programa

❖ Resumiendo:

¿Cuándo utilizar parámetros Valor o por Referencia ?

Cuando la información a pasar no tiene que ser devuelta, el parámetro formal puede ser por **Valor**(Entrada). Si se tiene que devolver información en el mismo, debe ser por **Referencia**(Salida). Si se pasa información que será modificada en el procedimiento, el parámetro formal será por **Referencia**(Entrada/Salida).

Piezas de un Programa

❖ Ámbito de las Variables

Un programa puede tener distintas funciones. Las funciones en lo que una variable puede ser utilizada se conoce como **Ámbito** de la variable, o sea que **ámbito** es la sección del programa donde esa variable es válida (su zona de actuación).

En el caso particular de constantes y variables definidas dentro del cuerpo de una función (incluyendo las de la función main()), pueden emplearse en cualquier punto dentro de este bloque, **exclusivamente**. A estas declaraciones se las denomina **locales**, ya que su existencia y alcance están limitados por la función que las contiene. En cambio, cuando las **declaraciones se realizan fuera de cualquier función**, se las considera **globales,** y su alcance es el de todas las funciones del programa.

Piezas de un Programa

❖ Variables Locales y Globales

Variable Local: declarada dentro de una función y solo puede ser accedida en ella. Fuera de la misma no es conocida.

Variable Global: declarada fuera de cualquiera de las funciones. Puede ser usada por todas las funciones.

En caso de que existan 2 variables con el mismo nombre, y una sea global y la otra local, son **distintas**, pero **dentro de la función** la **global no tiene** actuación. Se desaconseja el uso de variables globales, salvo que su presencia sea estrictamente justificada,ya que impide la reutilización del código, entre otras cosas. En cambio, el uso de variables locales contribuye a una mejor legibilidad del programa y minimiza la probabilidad de errores por referencias incorrectas.

"Enseñar no es transferir conocimientos, sino crear las posibilidades de su construcción; quien enseña aprende al enseñar y quien aprende enseña al aprender"

Paulo Freire