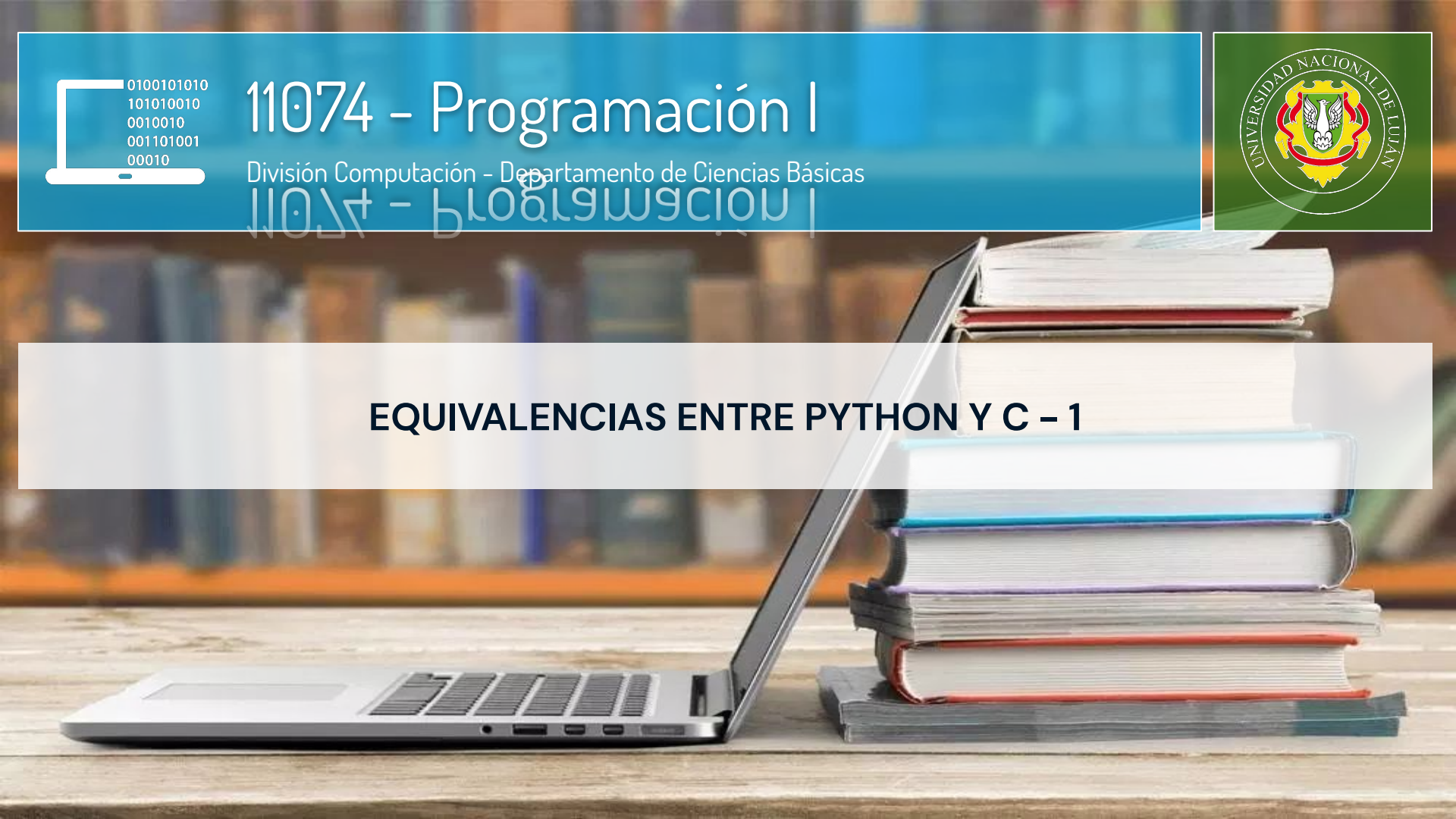

EQUIVALENCIAS ENTRE PYTHON Y C – 1

Equivalencias entre Python y C

Repaso de Conceptos Básicos

EJEMPLOS DE LENGUAJES

- Ada
- Basic
- Pascal
- C, C++, C#
- Cobol
- Java
- JavaScript
- PHP
- Smalltalk
- Python

Estructura de un programa en C

Un programa sencillo escrito en C puede presentar, entre otras, las siguientes partes:

- Directivas al preprocesador.
- Prototipos de funciones.
- La función main().
- Las definiciones de las demás funciones elaboradas por el programador.

Ya hemos detallado estas características y vamos a profundizar en la función principal **main()** para poder enfocarnos en lo necesario para usar este lenguaje y las sentencias equivalentes a las utilizadas en Python.

Estructura de un programa en C

❖ La función main()

Esta función es un tanto “especial”, en cuanto que es el “punto de partida” por el cual comienza la ejecución del programa. A partir de aquí podrán invocarse otras funciones. Además, como parte del cuerpo de la función main(), se pueden encontrar:

- Declaración de constantes (por lo general estas declaraciones se hacen fuera del cuerpo del main() para que sean accesibles a otras funciones).
- Declaración de tipos (estas declaraciones suelen hacerse fuera del cuerpo del main() para que sean accesibles a otras funciones).
- Declaración de variables (son las variables locales del main()).
- Sentencias (instrucciones) que conforman la lógica de la función main().

Piezas de un Programa

❖ Declaración de constantes

Las constantes son inalterables (de ahí su nombre), en general se las nombra usando mayúsculas y se declaran como se muestra a continuación:

```
const <tipo de dato> <nombre de la constante> = <valor>;
```

ejemplo:

```
const float PI = 3.14;

const char *MENSAJE = "Buenos días!";
```

También podríamos declarar una constante utilizando un tipo de directivas al procesador: `#define PI 3.14`

En este caso, cada vez que el preprocesador reconozca la palabra PI en el código, la reemplazará por el número 3,14.

La declaración de constantes no tiene su equivalencia en Python.

Piezas de un Programa

❖ Declaración de tipos y variables

Los diferentes objetos de información con los que trabaja un programa C se conocen en conjunto, como datos. Todos los datos tienen un tipo asociado. Un dato puede ser un simple carácter, un valor entero o un número decimal, entre otros.

El lenguaje C es conocido como **fuertemente tipado** (strongly-typed), en cuanto a que es **obligatorio** para el programador **asignar un tipo determinado** a cada dato procesado antes de utilizarlos. Ejemplos:

```
int numero;   char mensaje;  float promedio;
```

Es una **diferencia fundamental con Python** ya que su tipado es dinámico.

Piezas de un Programa

Los tipos de datos manejados por **Python**

Tipo	Ejemplo
int (Enteros)	1 127 -122 0
float(Reales)	1,27 32,615 -0,1368 1,0
char(Caracter)	“1” ”s” “?”
String	”Hola Mundo” “Anna no duerme” “127”
bool(Booleano)	True(Verdadero) False(False)

En **C**

Tipo	Ejemplo
int (Enteros)	1 127 -122 0
float(Reales)	1,27 32,615 -0,1368 1,0
char(Caracter)	“1” ”s” “?”
double(Reales)	

Booleano no existe, 0 indica falso.

Piezas de un Programa

En C también tenemos tipificados:

signed int , unsigned int, short int, signed short int ,unsigned short int, long int , signed long int,unsigned long int

Para ser estrictos, los tipos de datos básicos del lenguaje C son: **char, int, float y double**. Las palabras reservadas **signed, unsigned, short y long** son modificadores que permiten ajustar el tipo de dato al que se aplican. Por ejemplo, si se antepone **unsigned** al tipo **int**, se forma un tipo de **dato entero sin signo**. Si se aplica el modificador **long,** el tipo **entero es largo** (mayor rango de valores).

Piezas de un Programa

Palabra reservada void

En el lenguaje C, la palabra reservada **void no es un tipo en sí**, aunque en ciertas circunstancias puede utilizarse como tal. Por ejemplo, en la implementación de una **función** podría indicarse que ésta **“no retorna ningún valor”** mediante el uso de void:

```
void mostrar_menu()
```

Existen otros modificadores de tipos en C que iremos viendo más adelante.

Piezas de un Programa

Tipos de datos definidos por el usuario

Todos los tipos de datos estudiados hasta ahora son los elementales provistos por el lenguaje, y pueden utilizarse directamente. Sin embargo, un aspecto muy interesante de lenguajes como C es su capacidad para que el programador cree estructuras de datos a partir de estos datos simples, que favorecen la legibilidad de los programas y simplifican su mantenimiento. Los tipos de datos definidos por el usuario, que se desarrollarán a lo largo de la asignatura, se clasifican en:

Piezas de un Programa

Tipos de datos definidos por el usuario

Escalares definidos por el usuario.

Estructuras

Arreglos de caracteres.

Arreglos en general

Archivos.

Punteros

Piezas de un Programa

❖ Construcción de sentencias básicas

En un programa de computadora, las sentencias describen las acciones algorítmicas que deben ejecutarse. Expresado de otra manera, un programa es una secuencia de sentencias, que pueden clasificarse en:

- **Ejecutables**: Especifican operaciones de cálculos aritméticos y entrada/salida de datos.

- **No ejecutables**: No realizan acciones concretas, pero ayudan a la legibilidad del programa y no afectan su ejecución.

Dentro de las sentencias ejecutables, existen aquellas que permiten llamar a una función y las que se utilizan para asignar un valor a una variable.

Piezas de un Programa

Estrictamente, la **asignación** es una operación que **sitúa un valor determinado en una posición de la memoria.**

variable = expresión

variable es un identificador válido declarado con anterioridad **(en C)** y expresión es una variable, constante, literal o fórmula para evaluar.

En **Python**:

```
numero = 5
```

En **C**:

```
int numero;
numero = 5;
```

Piezas de un Programa

En cuanto a las **etiquetas** de las variables existen en **C** las mismas restricciones que en **Python**:

Válidos	**No Válidos**
Num1	1num (no puede comenzar con nro)
num_1	num 1 (no puede contener espacios)
	num*1(no puede contener símbolos matemáticos)
	num-1(no puede contener símbolos matemáticos)

Diferencia MAYÚSCULAS de minúsculas

num1 ≠ NUM1 ≠ Num1

Piezas de un Programa

Hay algunas asignaciones un tanto diferentes de las convencionales; éstas corresponden al **contador** y al **acumulador**. Un **contador** es una variable entera que se incrementa, cuando se ejecuta, en una unidad o en una cantidad constante:

```
int contador; (no vale para Python)
contador = 0;
contador = contador + 1;
contador++; /* equivalente a contador = contador + 1 */
Contador- -; /* equivalente a contador = contador - 1 */
contador= contador + 1 (en Python)
contador +=1#equivalente a contador=contador + 1 en Python
```

Piezas de un Programa

Por su parte, un **acumulador** es una **variable que se incrementa en una cantidad variable**. La sintaxis es la misma que en Python:

```
acum = acum + valor; /* acumulación */
acum += valor; /* acumulación */
```

Piezas de un Programa

Expresiones

- Operaciones que podemos hacer sobre los datos, su resultado sería una expresión
- Según el tipo de dato varían las operaciones y su expresión o resultado
- Una **expresión es la combinación de datos y operadores.**

Operadores

En los lenguajes de programación, un operador se aplica sobre una (operador unario) o dos (operador binario) variables para modificar o utilizar de alguna manera su valor.

Piezas de un Programa

Un ejemplo de **operador unario** es el ya visto operador de incrementación para lenguaje **C**: **++,** existe también **--** para decrementar en 1.

El operador de asignación es un buen ejemplo de **operador binario** en ambos lenguajes: letras **=** 'A';

En los casos típicos los distintos operadores pueden clasificarse en:

- Aritméticos.
- Relacionales y lógicos.
- De manejo de bits.

Piezas de un Programa

- **Operadores aritméticos**

Python

- ✔ **+**
- ✔ **-**
- ✔ *****
- ✔ **/**
- ✔ **%**
- ✔ **//** división entera
- ✔ ******

C

+	suma
-	resta
*****	multiplicación
/	división
%	resto de división entera

pow(x,y)(incluir librería **Math.h.**) potencia

Piezas de un Programa

A continuación se listan los operadores citados, en orden de precedencia **decreciente**:

++ , – –
*** , / , %**
+ , –

Además, para los operadores de igual grado de precedencia, la evaluación se hace de izquierda a derecha.

Los paréntesis cambian la precedencia de las operaciones y, por lo tanto, su orden de evaluación.

Piezas de un Programa

- **Operadores para cadenas de caracteres**

En **Python**

+	Suma o concatenación (unir)
***dígito**	Multiplicar (repetir dígito veces el string o char)

En **C** hay que usar funciones para manejo de strings y para ello incluir la biblioteca **string.h**. Algunas de estas funciones son:

Función	**Descripción**
char strcpy(char s1, char s2)	Copia s2 en s1 y retorna s1
char strncpy(char s1, char s2, int n)	Copia hasta n caracteres de s2 en s1 y retorna s1
char strcat(char s1, char s2)	Concatena s2 a s1 y regresa s1
char strncat(char s1, char s2, int n)	Concatena hasta n caracteres de s2 a s1 y retorna s1
int strlen (char s1)	Devuelve la longitud de la cadena s1
strcmp (char s1, char s2)	Devuelve 0 si las cadenas representadas por s1 y s2 son iguales, o un valor menor que cero si s1 precede alfabéticamente a s2.

Piezas de un Programa

- **Operadores relacionales y lógicos**

La lista de **operadores relacionales**, tanto en **Python** como en **C** es la siguiente:

Operador	Acción
>	Mayor que
>=	Mayor o igual que
<	Menor
<=	Menor o igual que
==	Igual
!=	Distinto

Piezas de un Programa

- **Operadores relacionales y lógicos**

La lista de **operadores lógicos** es la siguiente:

Operador		Acción
Python	**C**	
and	**&&**	y (conjunción)
or	**||**	o (disyunción)
not	**!**	no(negación)

En lenguaje **C todo valor distinto de 0** significa **“verdadero”**, mientras que el **valor 0** significa **“falso”**. Así, una expresión en la que intervienen operadores relacionales o lógicos genera un valor booleano: **1** para indicar **“verdadero”** y **0** para indicar **“falso”.**

Piezas de un Programa

Operadores relacionales y lógicos

Estos operadores tienen menor grado de precedencia respecto de los aritméticos y, entre ellos, se verifica el siguiente orden de precedencia (de mayor a menor):

!

> , >= , < , <=

== , !=

&&

||

Piezas de un Programa

- **Palabras Reservadas**
  Son palabras que el lenguaje utiliza como órdenes o instrucciones por lo tanto no pueden ser utilizadas como nombre de variables y son estas:

En **Python**:

and as assert break class continue def del elif else except False finally for from global if import in is lambda None nonlocal not pass raise return True try while with or yield

En **C**:

auto break case char const continue default do double else enum extern float for goto if int long register restrict return short signed sizeof static struct switch typedef union unsigned void volatile while

"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