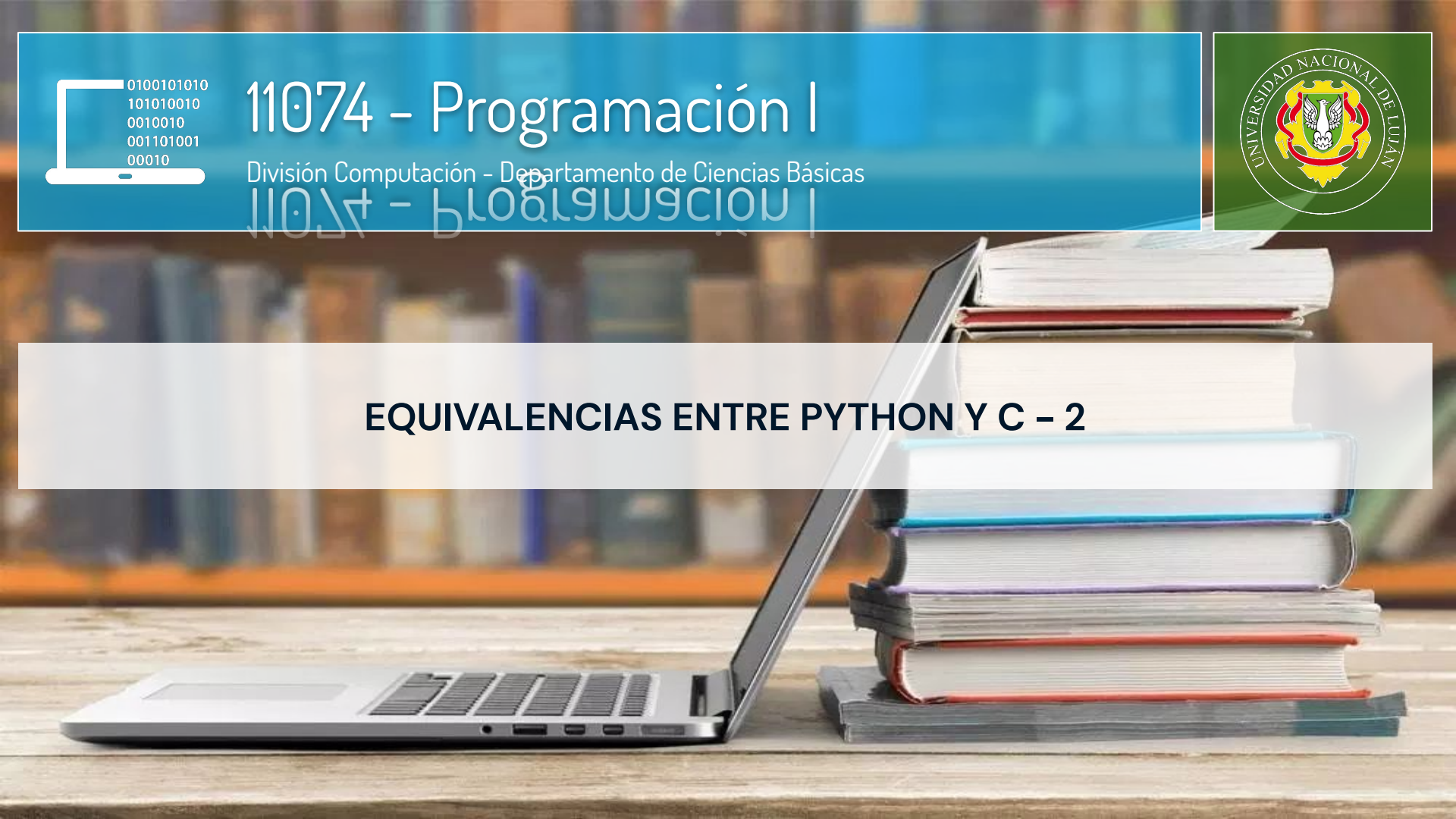

EQUIVALENCIAS ENTRE PYTHON Y C – 2

Equivalencias entre Python y C

Repaso de Conceptos Básicos

Ejemplos De Lenguajes

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

Piezas de un Programa

❖ Instrucciones

Son las órdenes con las que construimos nuestro programa:

Operaciones de entrada/salida

Un programa no es una entidad aislada. Necesita, en mayor o menor medida, interactuar con el “mundo exterior”, tanto para obtener los datos de entrada para procesar, como para enviar los resultados generados. A los mecanismos que permiten establecer esta “interfaz” entre el programa y su entorno se los denomina operaciones de entrada/salida. El ingreso de datos por teclado, la salida de resultados en pantalla o la escritura y la lectura de archivos, son algunos ejemplos de este tipo de operaciones.

Piezas de un Programa

En lenguaje C, las operaciones de entrada/salida están implementadas en la unidad de biblioteca **stdio.h** (standard input/output), que se ajusta al estándar ANSI. Por ahora podemos indicar que es un conjunto de funciones, definiciones de tipos y constantes, que puede incluirse dentro de un programa para su uso, reutilizando el código ya implementado.

Función		Acción
Python	**C**	
variable=input ( )	**scanf("%d", &variable);**	entrada por teclado
	fgets(variable, MAX_SIZE, stdin);	entrada por teclado

Piezas de un Programa

Las funciones **scanf() y printf()** (se usa para salida y la vemos más adelante) permiten **aplicar formato** a los datos de E/S, como por ejemplo, cantidad de decimales, justificación de texto, conversión a mayúsculas/minúsculas, numeración hexadecimal, etc.

Ambas funciones toman al menos **dos** argumentos (pueden ser más): **1**: La **cadena de formato**. **2**: Los **datos a enviar a la salida**, en el caso de **printf(),** o r**ecibir de la entrada**, en el caso de **scanf().** La cadena de formato es un literal encerrado entre comillas dobles (**“ “**) precedida de un símbolo **'%**'. La tabla siguiente muestra especificadores frecuentemente utilizados:

Piezas de un Programa

Especificador	Significado
c	Un carácter (char)
i,d	Número entero (int o double)
s	Cadena de caracteres (char *)
f	Número de punto flotante (float o double)
m.n	M cifras enteras y n decimales
-	Ajuste de texto a la izquierda
k	Número de cifras de un número entero
x,X	Número hexadecimal (0-9 A-F)

Piezas de un Programa

La cadena de formato puede incluir literales y caracteres especiales tales como

'\n'	Nueva línea
'\t'	Tabulación
'\0'	Carácter nulo (fin de cadena)

Ejemplo **scanf():**

```
char string[80];

scanf("%s",&string);
```

En esta función, el primer argumento es la cadena de formato y el segundo la dirección de las variables que recibirán los datos.

Piezas de un Programa

sintaxis: **scanf("%d", &variable);**

El primer parámetro de **scanf()** es la cadena de formato, que le indica a la función cómo debe interpretar el dato leído. El segundo parámetro corresponde a la variable en memoria donde quedará almacenado ese dato, y su tipo debe ser consistente con lo que se espera como entrada. Como puede observarse, la variable está precedida por el **operador unario &** . Esto permite que la función **scanf() reciba la dirección en memoria de la variable (y no su contenido)**, lo cual es lógico, ya que a scanf() le interesa conocer dónde debe poner el dato de entrada.

La función scanf() presenta un **problema crítico** que pone en riesgo la seguridad del programa que la ejecuta y, por esta razón, se desaconseja su uso.

Piezas de un Programa

La variable tiene lugar para almacenar una cantidad máxima de acuerdo a su declaración previa. No obstante, ¿qué sucedería si el usuario ingresara por teclado más de lo permitido? La información que no quepa en la variable estará "pisando" una porción de memoria que no le corresponde y que podría contener otros datos. Peor aún, el usuario podría sobreescribir a conciencia datos vitales para la ejecución del programa, y controlar su ejecución a su gusto. Este escenario ha sido uno de los ataques más populares en la historia de la computación, y se conoce como ataque por desborde de memoria (**buffer overflow attack**). Para evitar esta situación es recomendable el uso de una función de entrada más segura, como es el caso de **fgets()**.

```
fgets(variable, MAX_SIZE, stdin);
```

Piezas de un Programa

La característica más sobresaliente de **fgets()** es que recibe en el segundo parámetro la **cantidad máxima de caracteres que se leerán**. Además, el tercer parámetro establece el origen de los datos de entrada (en el ejemplo se utiliza la entrada estándar (**stdin**): el teclado).

Ejemplo:

```
#include <stdio.h>
#define MAX_SIZE 10
int main() {
    char cadena[MAX_SIZE];
    printf("Ingrese una cadena de caracteres \n");
    fgets(cadena, MAX_SIZE, stdin);
    printf("Cadena ingresada: %s \n", cadena);
    return 0;
}
```

Piezas de un Programa

Función		Acción
Python	**C**	
print(argumentos)	**printf(argumentos);**	salida por monitor

donde argumentos pueden ser texto, variables, constantes o combinación de todos

En C el formato puede controlarse mediante el uso de especificadores de formato, que son caracteres “especiales”.

Ejemplo :

```c
#include <stdio.h>
int main() {
    printf("El resultado de sumar  %d y %d es %d \n", 2, 3, 5);
    return 0;
}
```

Piezas de un Programa

Ejemplos printf():

printf(“%-1.3f\n”, 3.1415927);
3.141

printf(“IVA: 21,5%%\n”);
IVA: 21,5%

Piezas de un Programa

Lógica de la función principal

De acuerdo con el paradigma de programación estructurada, en toda función pueden encontrarse, entre otras cosas, sentencias que indiquen la realización de tres tipos de acciones: **secuencia, selección o iteración**.

Secuenciales: Son las asignaciones y las invocaciones a funciones. Ejemplo: **variable = expresión;**

```
num1 = 15;
num2 = 5;
resultado = suma(num1,num2); /* llamada a función */
resultado = resultado / 2;
...
```

Piezas de un Programa

Lógica de la función principal

Secuenciales: Son las asignaciones y las invocaciones a funciones.

variable = expresión;

Python

```
num1 = 15
num2 = 5
resultado = suma(num1,num2)    #suma:función
resultado = resultado / 2
```

...

C

```
num1 = 15;
num2 = 5;
resultado = suma(num1,num2); /*suma:función*/
resultado = resultado / 2
```

Piezas de un Programa

Lógica de la función principal

Selección: Las estructuras selectivas se utilizan para tomar decisiones lógicas.

En **Python** tenemos:

Estructura Condicional Simple:

```
if (condición):
    [código a ejecutar si la condición es verdadera]
```

Estructura Condicional Compuesta:

```
if (condición):
        [código a ejecutar si la condición es verdadera]
else:
        [código a ejecutar si la condición es falsa]...
```

Piezas de un Programa

Lógica de la función principal

Selección: Las estructuras selectivas se utilizan para tomar decisiones lógicas.

Condicionales anidados: Dentro del código perteneciente a cada opción pueden existir más estructuras condicionales. Ejemplo:

```
if (condición):
        if (condición):
            [código a ejecutar si la condición es verdadera]
        else:
            [código a ejecutar si la condición es falsa]
else:
        if (condición):
            [código a ejecutar si la condición es verdadera]
        else:
            [código a ejecutar si la condición es falsa]
```

Piezas de un Programa

Lógica de la función principal

Selección: Las estructuras selectivas se utilizan para tomar decisiones lógicas.

Condicionales encadenados: Cualquier estructura **if** se puede extender con cantidad ilimitada de **elif** e incluso también agregar **else** cuando ninguna de las condiciones se cumple. Ejemplo:

```
if (condición):
        [código a ejecutar si la condición es verdadera]
elif (condición):
        [código a ejecutar si la condición es verdadera]
elif (condición):
        [código a ejecutar si la condición es verdadera]
....
else:
    [código a ejecutar si la condición es falsa]
```

Piezas de un Programa

Lógica de la función principal

Selección: Las estructuras selectivas se utilizan para tomar decisiones lógicas.

En **C** tenemos:

Estructura de Alternativa Doble: La sentencia **if-else** se considera de alternativa doble (si se cumple cierta condición, entonces..., en caso contrario...), y tiene la siguiente estructura:

```
if (condición)
        acción 1;
Else
        acción 2;
```

Piezas de un Programa

Lógica de la función principal

Selección: Las estructuras selectivas se utilizan para tomar decisiones lógicas.

Cuando alguna de las alternativas tiene más de una instrucción, esa opción deberá llevar una llave de apertura '{' y otra de cierre '}'. Por ejemplo:

```
if (condición){
        acciones;
}
else{
        acciones;
}
```

Piezas de un Programa

Lógica de la función principal

Selección: Las estructuras selectivas se utilizan para tomar decisiones lógicas.

Estructura de Selección Múltiple: La sentencia **switch**, en cambio, se considera de **selección múltiple**, ya que el flujo de ejecución puede continuar por una cantidad ***N*** de alternativas posibles, según el valor de la expresión que se evalúa al principio:

```
switch (expresión){
        case Opcion1: < acción 1>;
                      break;
        case Opcion2: < acción 2>;
                      break;
              …
        case Opcion N: < acción N>;
                      break;
        default: < acción M>;
}
```

Piezas de un Programa

Lógica de la función principal

Iterativa o repetitiva:Las estructuras repetitivas (llamadas también bucles o ciclos) se utilizan para realizar varias veces el mismo conjunto de operaciones. Entre ellas se encuentran aquellas donde **la cantidad de repeticiones se conoce a priori** (**Ciclos Definidos)** y aquellas en las que **las repeticiones se realizan hasta que se cumple una condición lógica dada** (**Ciclos indefinidos o Condicionales**).

Ciclos Definidos: En **Python:**

```
for <variable> in  <iterable>:
    <cuerpo>
```

cuerpo se va a ejecutar tantas veces como valores tomados de **iterable** recibe **variable.**

Piezas de un Programa

Lógica de la función principal

Ciclos Definidos: En **Python:**Ejemplo:

```
suma = 0
cantidad = 10
for i in range(cantidad):
    nota = input("Ingrese la nota del alumno ")
    suma += int(nota)
```

Ciclos Definidos: En **C:**

```
for ( variable de control ; condición ; actualización variable de control ){

    < acción 1>
    < acción 2>
    ...
    }
```

Piezas de un Programa

Lógica de la función principal

Ciclos Definidos: En **C**:

El **encabezado de un bucle for tiene tres partes (separadas por ";").** En la **primera** se **inicializa la variable de control** y sólo se ejecuta una vez, antes de la primera iteración. La **segunda** es la **condición lógica** que debe cumplirse para que la próxima iteración se ejecute; esta condición se evalúa antes de cada iteración y, cuando deja de satisfacerse, el bucle **for** termina. La **tercera** parte del encabezado es la actualización de la variable de control y se ejecuta después de cada iteración. Ejemplo:

```
acum = 0;
for (i = 1; i <= 10; i = i+1 )
{
        acum = acum + i;
}
```

Piezas de un Programa

Lógica de la función principal

Ciclos Indefinidos o Condicionales: En un **ciclo condicional**, la cantidad de veces que se repetirá el bloque de código **no** se conoce de antemano; se requiere de la evaluación de una condición antes de realizar una nueva iteración.

En **Python**:

```
while condición:
    # bloque a repetir
```

La **condición** es un valor o expresión booleana que indica si se debe ejecutar o no la siguiente iteración. El **bloque a repetir** son todas las instrucciones que se ejecutarán en cada iteración del ciclo **while**. Es importante que la **condición** se vea afectada de alguna manera por el bloque a repetir, para evitar **bucles infinitos**.

Piezas de un Programa

Lógica de la función principal

Ciclos Indefinidos o Condicionales:

Ejemplo: En **Python**

```
palabra = ''
while palabra != 'TERMINAR':
    palabra = input('Ingrese una palabra (o "TERMINAR" para finalizar):')
    print('Usted ingresó', palabra)
```

En **C**: la estructura **while** trabaja exactamente igual que en Python

```
while (condición){
        < acción 1>
        < acción 2>
        …
}
```

Piezas de un Programa

Lógica de la función principal

Ciclos Indefinidos o Condicionales:

Ejemplo:

```
acum = 0;
i = 1;
while (i <= 10){
    acum = acum + i;
    i++;
}
```

En C tenemos otra estructura más dentro de esta clasificación y es **do-while** que se utiliza cuando se quiere asegurar que el ciclo se ejecuta al menos una vez, puesto que la evaluación de la condición lógica se hace al final de éste.

Piezas de un Programa

Lógica de la función principal

Ciclos Indefinidos o Condicionales: Sintaxis de **do-while**:

```
Do{
        < acción 1>
        …
}
while (condición);
```

Ejemplo:

```
acum = 0;
i = 1;
do{
    acum = acum + i;
    i++;
}
while (i <= 10)
```

Ejemplo de un programa en C

Este programa tiene como **única funcionalidad** mostrar en pantalla un menú de tres opciones. Si el usuario selecciona la opción “1”, se muestra la leyenda “Este es un mensaje!”; si selecciona la opción “2”, se muestra la leyenda “Este es otro mensaje!”; y si selecciona la opción “3”, el programa finaliza su ejecución.

```
#include <stdio.h>
/* Constantes */
#define OPCION_1 1
#define OPCION_2 2
#define OPCION_SALIR 3
/* Prototipos de funciones */
void mostrar_menu();
int leer_opcion();
void ejecutar(int);
```

Ejemplo de un programa en C

```c
/* Programa Principal */
int main(){
/* Declaración de la variable 'opcion' */
    int opcion;
    do{
        mostrar_menu();
        opcion = leer_opcion();
        ejecutar(opcion);
        }
    while (opcion != OPCION_SALIR);
    return 0;
}
```

Ejemplo de un programa en C

```c
/* Implementación de funciones */
void mostrar_menu(){
    printf("Elija una opción: \n");
    printf(" 1 - Mostrar un mensaje \n");
    printf(" 2 - Mostrar otro mensaje \n");
    printf(" 3 - Salir \n");
}
int leer_opcion(){
    int opcion;
    /* Se lee un número entero por teclado */
    scanf("%d", &opcion);
    return opcion;
}
```

Ejemplo de un programa en C

```
void ejecutar(int opcion){
    switch (opcion){
        case OPCION_1: printf("Este es un mensaje! \n");
                       break;
        case OPCION_2: printf("Este es otro mensaje! \n");
                       break;
        case OPCION_SALIR: printf("Saliendo... \n");
                       break;
        default      : printf("Opción incorrecta! \n");
    }
}
```

Ejemplo de un programa en C

En primer lugar, mediante la directiva al preprocesador **#include**, se incluye el archivo de cabecera **stdio.h** (provisto con el compilador de C) que permite, entre otras cosas, **tener acceso al teclado y a la pantalla** (periféricos de entrada y salida, respectivamente). Luego, usando la directiva **#define**, se definen tres constantes.

En la declaración de prototipos de funciones, es interesante observar la palabra reservada **void**. Ésta **no es un tipo de dato en sí** y sirve para indicar que, en este caso, las funciones **mostrar_menu() y ejecutar() no retornan valor alguno**.

"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