PYTHON

CONCEPTOS PRELIMINARES

Cuando se habla de programar, se hace referencia a la codificación de un algoritmo utilizando un lenguaje de programación. Ahora bien, existen los llamados ***lenguajes de alto nivel*** y ***lenguajes de bajo nivel*** (también denominados lenguajes de máquina o ensambladores). Las computadoras sólo ejecutan programas escritos en lenguajes de bajo nivel, por lo tanto los programas escritos en lenguajes de alto nivel deben "traducirse" antes de ejecutarse.

Existen dos tipos de programas que "traducen" lenguajes de alto nivel a lenguajes de bajo nivel: los **intérpretes** y los **compiladores**.

Un **intérprete** va "leyendo" un programa o ***código fuente*** y lo va ejecutando, es decir traduce el programa poco a poco, lee cada instrucción o sentencia y la ejecuta.

Por el contrario, un **compilador** lee el programa completo y lo traduce todo de una vez, generando de ese modo lo que se conoce como ***código ejecutable***. Una vez que un programa fue compilado, puede ejecutarse repetidamente sin volver a traducirlo, utilizando directamente el ***ejecutable***.

A pesar de que estas "traducciones" a veces pueden llevar tiempo, la programación en lenguajes de alto nivel es mucho más sencilla; escribir programas en un lenguaje de alto nivel toma menos tiempo, los programas son más cortos, más fáciles de leer y corregir.

Python es un lenguaje de alto nivel interpretado. Existen dos maneras de usar su intérprete: *modo de comando* y *modo de guión*. En el primero se escriben las sentencias de Python y el intérprete muestra inmediatamente el resultado. Alternativamente (*modo de guión*), se puede escribir el programa en un archivo y usar el intérprete para ejecutar el contenido de dicho archivo. En este caso, el archivo con las intrucciones se denomina *guión* o *script*.

En nuestro caso y para trabajar más cómodamente con este nuevo lenguaje utilizaremos lo que se conoce como **IDE** o **Entorno de Desarrollo Integrado** (**I**ntegrated **D**evelopment **E**nvironment). Esta es una aplicación que incluye un editor de texto para escribir nuestros programas, el intérprete del lenguaje que permitirá ejecutarlo y una serie de herramientas que nos facilitarán tanto la escritura como la corrección o depuración del código.

PARA TENER EN CUENTA

- Python es sensible a mayúsculas y minúsculas, por lo tanto si nombramos variables como **Num** y **num**, las considerará como dos variables diferentes.
- Las ***palabras reservadas*** (palabras propias del lenguaje, como *print*, *if*, *input*, etc.) deben escribirse en minúsculas.
- Los comentarios pueden escribirse en una línea completa o a continuación de cualquier sentencia comenzando siempre con el símbolo **#** (numeral).
- No es necesario escribir una instrucción específica para el comienzo y fin de un programa (como lo hacíamos en PSeInt). Se pueden comenzar escribiendo sentencias desde la primera línea.
- Python exige que el código esté correctamente indentado según las estructuras que se utilicen, caso contrario el código no se ejecutará de manera deseada o saltarán errores de ejecución.

ESTRUCTURA de un PROGRAMA: Sentencias básicas

Asignación

Las asignaciones se realizan utilizando el signo **=** (igual). En relación con esta sentencia, algo para destacar es que en Python no es necesario definir las variables y su tipo, pero sí es una buena práctica inicializarlas utilizando asignaciones. Al asignar un determinado valor a una variable, queda definido el tipo de la misma.

```
Numero = 53  # Esta variable es de tipo Entero (Int)

Real = 5.9   # Esta variable es de tipo Real (Float)

Cadena = "Hola mundo" # Esta variable es de tipo Cadena (String / str)

Logica_1 = False # Esta variable es de tipo Lógica (Boolean)

Logica_2 = True # Esta variable es de tipo Lógica (Boolean)
```

Salida de datos o información

Para mostrar información en pantalla usaremos la función[1] **print()**, que permite desplegar cadenas de caracteres o bien el contenido de variables, tal como se observa en los siguientes ejemplos:

```
Texto = "Este es un texto de ejemplo"

Numero = 25

# Se puede imprimir texto escribiendo entre comillas
print ("Este texto se mostrará en pantalla")

# Se puede imprimir en pantalla el valor de una variable
print (Texto)

# Se pueden imprimir números
print (Numero)
```

Al ejecutar el código precedente se podrá ver:

```
Este texto se mostrará en pantalla
Este es un texto de ejemplo
25
>>>
```

1 Uno de los tipos de sentencias de Python que más utilizaremos; su sintaxis se caracteriza porque a continuación del nombre se escriben entre paréntesis los datos con los que trabajará o argumentos.

Entrada de datos

En Python el ingreso de datos desde teclado se hace a través de una asignación y utilizando la función **input()**. Se escribe el nombre de la variable que contendrá el valor ingresado por el usuario y se le asigna **input** seguida de paréntesis. Dentro de estos paréntesis podemos escribir el texto que le mostraremos al usuario. Por ejemplo:

```
nom=input("Ingrese su nombre: ")
```

o

```
print ("Ingrese su nombre:")
nom=input()
```

Estos códigos son equivalentes. En ambos casos la variable ***nom*** recibirá y almacenará el nombre ingresado.
Algo muy importante a tener en cuenta con esta sentencia es que por defecto lo ingresado será de tipo *cadena* o *string*, por lo tanto si se desea ingresar datos numéricos, deberemos utilizar adicionalmente las funciones **int()** o **float()** según sea el caso. Por ejemplo:

```
nom=input("Ingrese nombre: ")
horas=int(input("Ingrese horas trabajadas: "))
valor=float(input("Ingrese valor hora: "))
salario=horas*valor
print (nom,"por sus ", horas, "horas trabajadas, cobrará $", salario)
```

Al ejecutar el código anterior, en el IDE podremos visualizar lo siguiente:

```
Ingrese nombre: Lucía
Ingrese horas trabajadas: 43
Ingrese valor hora: 341.56
Lucía por sus  43 horas trabajadas, cobrará $ 14687.08
```

Condicionales

En Python el condicional se expresa por medio de la sentencia **if**, seguido de la condición a evaluar y finalizando la línea con **:** (dos puntos). En la línea siguiente y respetando la indentación correspondiente, se deberán escribir las instrucciones que deben llevarse a cabo si la condición evaluada resulta verdadera). Si es necesario, en la línea siguiente se puede escribir la sentencia **else** seguida de **:** (dos puntos) y abajo, siempre respetando la indentación, se escribirán las instrucciones a ejecutarse si la condición evaluada resulta ser falsa. Veamos el siguiente ejemplo:

```
# Se le pide al usuario una clave y se guarda en
# la variable Clave

Clave = input ("Ingrese su clave: ")

# Se corrobora que la clave ingresada sea la correcta

if Clave == "Mi_Clave":
    print ("Clave correcta, puede ingresar")
else:
    print ("Ha ingresado una clave errónea")
```

La ejecución del código anterior mostrará:

```
Ingrese su clave: Mi_Clave
Clave correcta, puede ingresar
>>>
```