El lenguaje Python

El lenguaje Python fue creado a fines de la década de 1980, y publicado por primera vez en 1991. Su creador es Guido van Rossum, un holandés nacido en 1956, que actualmente trabaja para la empresa Dropbox, además de coordinar el desarrollo del lenguaje.

Guido van Rossum

El lenguaje debe su nombre a un programa de televisión británico.

Las características del lenguaje tienden a que el programador escriba un código fácilmente legible, lo que lo hace ideal para el aprendizaje.

Logo de Python

Muchas aplicaciones y sitios Web actualmente en uso están escritas en Python (como Youtube, por ejemplo).

La versión actual es Python3, que presentó notables diferencias respecto a su antecesor Python2, que, por cuestiones de compatibilidad sigue siendo utilizado.

El “shebang” (#!)

En el sistema operativo GNU/Linux, todos nuestros programas en Python comenzarán con: `#! /usr/bin/env python3`

Este renglón sirve para decirle al Sistema Operativo que lo que está a continuación es código Python.

Mostrando información por pantalla

Para mostrar algo por pantalla, utilizaremos la función `print()`. Si queremos hacer un programa que muestre un cartel que diga: “Hola, mundo”, lo haríamos así:

```
#! /usr/bin/env python3
print("Hola, mundo")
```

Variables

Una variable es un lugar de memoria al que se le asigna un nombre. En ese lugar de la memoria, podemos guardar un dato (un número, un texto, etc). Podemos imaginar las variables como si fueran cajas que tienen una etiqueta. Dentro de esas cajas hay guardado un dato.

Por ejemplo, podemos guardar el texto “Daniel” en la variable “nombre”, y luego mostrarlo.

```
#! /usr/bin/env python3
nombre = "Daniel"
print("Hola ", nombre)
```

Este programa mostrará: “Hola, Daniel”

Ingreso de datos por el usuario

Si queremos solicitarle al usuario que ingrese un dato, utilizaremos la función `input()`. El dato ingresado lo guardaremos en una variable, para utilizarlo luego. Por ejemplo, podemos hacer un programa que le pida al usuario que escriba su nombre; el programa lo guardará en la variable “nombre”. Luego, mostrará un saludo:

```
#! /usr/bin/env python3
nombre = input("Escriba su nombre")
print("Hola ", nombre)
```

Ingreso de datos por el usuario: la funcion eval()

Si queremos que el usuario ingrese un número, podemos utilizar la función `eval()`, para asegurarnos de que el usuario no ingresará ninguna otra cosa. Por ejemplo, si queremos pedirle al usuario que ingrese dos números, para mostrar su suma:

```
#! /usr/bin/env python3
numero1 = eval(input("Escriba un número"))
numero2 = eval(input("Escriba otro número"))
suma = numero1 + numero2
print("La suma es ", suma)
```

Operadores matemáticos:

Suma: `+`	Resta: `-`	Multiplicación: `*`	División: `/`
Potencia: `**`	División entera: `//`	Resto de la división: `%`	

Estructura if

Frecuentemente, queremos que una operación se lleve a cabo solamente si se cumple una determinada condición. Para estos casos, utilizamos la función if, que se escribe así:

```
if condición:
    acción si la condición es verdadera
    otra acción si la condición es verdadera
```

Para tener en cuenta:

- Después de la condición se escriben dos puntos (:)
- Las acciones que se ejecutarán si la condición es verdadera, deben tener una sangría.

Por ejemplo: Pedirle al usuario que ingrese dos números y mostrar su suma. Si el primer número ingresado es mayor que el segundo, mostrar también su resta.

```
#! /usr/bin/env python3
numero1 = eval(input("Escriba un número"))
numero2 = eval(input("Escriba otro número"))
suma = numero1 + numero2
print("La suma es ", suma)
if numero1 > numero2:
    resta = numero1 - numero2
    print("La resta es ", resta)
```

A veces, podemos necesitar que se ejecuten ciertas acciones si la condición es verdadera, y otras distintas si la condición es falsa. Para esto se utiliza **if … else**, que se escribe así:

```
if condición:
    acción si la condición es verdadera
    otra acción si la condición es verdadera
else:
    acción si la condición es falsa
    otra acción si la condición es falsa
```

Por ejemplo: Pedirle al usuario que ingrese dos números y mostrar la resta del mayor menos el menor:

```
#! /usr/bin/env python3
numero1 = eval(input("Escriba un número"))
numero2 = eval(input("Escriba otro número"))
if numero1 > numero2:
    resta = numero1 - numero2
else:
    resta = numero2 - numero1
print("La resta es ", resta)
```

Vemos que la última línea muestra el valor de la variable **resta**, sea que se haya calculado en el **if** o en el **else**. (Para pensar: ¿qué hace este programa si los números fueran iguales?)

Operadores lógicos:

Si queremos que la estructura if evalúe más de una condición, podemos usar los operadores lógicos **and** y **or**.

Por ejemplo: Pedirle al usuario que ingrese dos números, e indicar si todos ellos son positivos.

```
#! /usr/bin/env python3
numero1 = eval(input("Escriba un número"))
numero2 = eval(input("Escriba otro número"))
if numero1 > 0 and numero2 > 0:
      print("Ambos son positivos")
```

Otro ejemplo: Pedirle al usuario que ingrese dos números, e indicar si alguno de ellos es negativo:

```
#! /usr/bin/env python3
numero1 = eval(input("Escriba un número"))
numero2 = eval(input("Escriba otro número"))
if numero1 < 0 or numero2 < 0:
      print("Se ha ingresado al menos un número negativo")
```

Otros operadores lógicos:

mayor: **>**	menor: **<**	mayor o igual: **>=**	menor o igual: **<=**
igual: **==**	distinto: **!=**	no: **not**	

Estructura if … elif … elif … elif … else

En ocasiones, queremos elegir uno entre varios caminos diferentes (más de dos). En este caso, recurrimos a la siguiente estructura:

```
if unaCondición:
      una acción
      otra acción
elif otraCondición:
      una acción
elif otraCondiciónMás:
      una acción
      otra acción
else:
      una acción
```

Podemos incluir tantos “elif” como necesitemos, y dentro de cada bloque, tantas acciones como hagan falta. Recordemos que la sangría indica a qué bloque pertenece cada instrucción.

Ejemplo: Solicitarle al usuario que ingrese un número. Indicar si es positivo, negativo o cero.

```
#! /usr/bin/env python3
numero = eval(input("Escriba un número"))
if numero > 0:
    print "Es positivo"
elif numero == 0:
    print "Es cero"
else:
    print "Es negativo"
```

Estructuras de repetición en Python: while

Las computadoras son máquinas particularmente útiles para realizar tareas repetitivas. Por esto, todos los lenguajes de programación proveen estructuras de repetición. Una de las estructuras de repetición de Python es la estructura while. Su sintaxis es la siguiente:

```
while condición:
      una acción a repetir
      otra acción a repetir
      …
```

Las acciones a repetir se escriben con una sangría. Una vez que se alcanza la última acción con sangría, el programa vuelve a la línea de while y evalúa nuevamente la condición. Si esta es verdadera, reptie una vez más las “acciones a repetir”. Esto continuará hasta que la condición sea falsa.

Ejemplo: Mostrar todos los números naturales menores que 100.

```
#! /usr/bin/env python3
numero = 1
while numero < 100:
    print(numero)
    numero = numero + 1
print("Fin del programa")
```