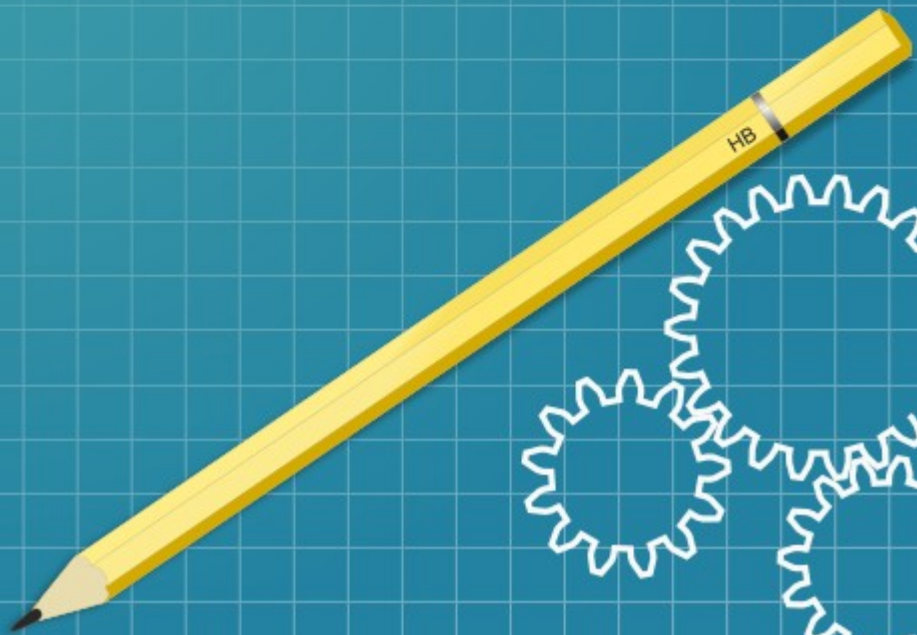
INTRODUCCIÓN AL LENGUAJE PYTHON



INDICE

- Características de Python
- Tipos de datos básicos
- Conversiones de tipo (explícita e implícita)
- Identificadores y comentarios
- Variables y expresiones
- Formato programa en Python
- Instrucciones de E/S
- Estructuras de control

Características de Python

- Es un lenguaje **interpretado**, no compilado, usa **tipado dinámico**, **fuertemente tipado**.
- Es **multiplataforma**, lo cual es ventajoso para hacer ejecutable su código fuente entre varios sistema operativos.
- Es un lenguaje de programación **multiparadigma**, el cual soporta varios paradigma de programación como orientación a objetos, **estructurada**, programación imperativa y, en menor medida, programación funcional.
- En Python, el formato del código (p. ej., la indentación) es estructural.

Características de Python

- El **fuertemente tipado** significa que el tipo de valor no cambia repentinamente. Un string que contiene solo dígitos no se convierte mágicamente en un número. Cada cambio de tipo requiere una conversión explícita.

```
# varible "valor1" es integer, varible "valor2" es string
valor1, valor2 = 2, "5"
# el metodo int() es para convertir a integer
total = valor1 + int(valor2)
# el metodo str() es para convertir a string
print "El total es: " + str(total)
```

Características de Python

- El **tipado dinámico** significa que los objetos en tiempo de ejecución (valores) tienen un tipo, a diferencia del tipado estático donde las variables tienen un tipo

```
# "variable" guarda un valor integer
variable = 11
print variable, type(variable)
# "variable" guarda un valor string
variable = "activo"
print (variable), type(variable)
```

Características de Python

Ventajas de programar en Python

- Simplificado y rápido: Este lenguaje simplifica mucho la programación, es un gran lenguaje para scripting.
- Elegante y flexible: El lenguaje ofrece muchas facilidades al programador al ser fácilmente legible e interpretable.
- Programación sana y productiva: Es sencillo de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y fomenta la productividad.
- Ordenado y limpio: es muy legible y sus módulos están bien organizados.
- Portable: Es un lenguaje muy portable. Podemos usarlo en prácticamente cualquier sistema de la actualidad.
- Comunidad: Cuenta con un gran número de usuarios. Su comunidad participa activamente en el desarrollo del lenguaje.

Tipos de datos básicos

- En cualquier lenguaje de programación de alto nivel se manejan tipos de datos. Los tipos de datos definen un conjunto de valores que tienen una serie de características y propiedades determinadas.
- En Python, todo valor que pueda ser asignado a una variable tiene asociado un tipo de dato
- Un tipo de dato establece qué valores puede tomar una variable y qué operaciones se pueden realizar sobre la misma.

Tipos de datos básicos

Números enteros

- El tipo de los números enteros es int. Este tipo de dato comprende el conjunto de todos los números enteros, pero como dicho conjunto es infinito, en Python el conjunto está limitado realmente por la capacidad de la memoria disponible. No hay un límite de representación impuesto por el lenguaje.

Tipos de datos básicos

Números de punto flotante

- Para representar el mayor número posible de los números reales con las limitaciones de memoria (tamaños de palabra de 32 y 64 bits), se adaptó la notación científica de representación de números reales al sistema binario (que es el sistema que se utiliza en programación para representar los datos e instrucciones). En esta notación científica, los números se representan así:

Número	Notación científica
101,1	$1{,}011 * 10^{2}$
0,032	$3{,}2 * 10^{-2}$

Tipos de datos básicos

- El tipo float sirve para representar cualquier número real (siempre teniendo en cuenta que es una aproximación lo más precisa posible). Por tanto para longitudes, pesos, frecuencias, …, en los que prácticamente es lo mismo 3,3 que 3,30000000000000003 el tipo float es el más apropiado.

Tipos de datos básicos

- A un número float puedes darle formato al número de la siguiente manera:

```
>>> real = 1.1 + 2.2  # real es un float
>>> print(real)
3.3000000000000003  # Representación aproximada de 3.3
>>> print(f'{real:.2f}')
3.30  # real mostrando únicamente 2 cifras decimales
```

- En Python la clase que representa los valores booleanos es **bool**. Esta clase solo se puede instanciar con dos valores/objetos: True para representar verdadero y False para representar falso.

Tipos de datos básicos

- Otro tipo básico de Python son las secuencias o cadenas de caracteres. Este tipo es conocido como string aunque su clase verdadera es str.
- Formalmente, un string es una secuencia inmutable de caracteres en formato Unicode.
- Para crear un string, simplemente tienes que encerrar entre comillas simples '' o dobles "" una secuencia de caracteres.
- Se puede usar indistintamente comillas simples o dobles, con una particularidad. Si en la cadena de caracteres se necesita usar una comilla simple, tienes dos opciones: usar comillas dobles para encerrar el string, o bien, usar comillas simples pero anteponer el carácter \ a la comilla simple del interior de la cadena.

Tipos de datos básicos

Conocer el tipo de una variable

- Existen dos funciones para conocer el tipo de una variable: Son type() e isinstance():
  - type() recibe como parámetro un objeto y te devuelve el tipo del mismo.
  - isinstance() recibe dos parámetros: un objeto y un tipo. Devuelve True si el objeto es del tipo que se pasa como parámetro y False en caso contrario.

Conversiones de tipo (explícita e implícita)

- En Python no se puede realizar la conversión explícita, por lo que tenemos que realizar una conversión implícita, para ello tenemos las funciones:
  - str(): Devuelve la representación en cadena de caracteres del objeto que se pasa como parámetro.
  - int(): Devuelve un int a partir de un número o secuencia de caracteres.
  - float(): Devuelve un float a partir de un número o secuencia de caracteres.

Identificadores y comentarios

- Un identificador es el nombre empleado para identificar una variable, una función, una clase, un módulo u otro objeto.
- Un identificador comienza con una letra (de la A a la Z o de la a a la z) o con un guión bajo (_) seguido de cero o más letras, guiones bajos y números.
- Python distingue mayúsculas de minúsculas.
- Python NO permite signos de puntuación como @, $ y %, excepto el guión bajo (_).

Identificadores y comentarios

- Un identificar no pueden coincidir con palabras clave del lenguaje

and	del	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	with
def	finally	in	print	yield

Identificadores y comentarios

- Los comentarios en Python, al igual que sucede en otros lenguajes de programación, sirven para explicar a las personas que puedan leer el programa en el futuro, qué es lo que hace el programa, así como explicar algunas partes del código. Estos comentarios son ignorados por las computadoras cuando ejecutan el código.
- El uso de comentarios aclaratorios es aconsejable y es “buen estilo” de programación

Identificadores y comentarios

- En python los comentarios se pueden poner de dos formas:
  - Escribiendo el símbolo almohadilla delante de la línea de texto donde está el comentario.
  - Escribiendo triple comilla doble (“””) al principio y al final del comentario (que puede ocupar más de una línea).

Variables, constantes y expresiones

- Cuando uses variables y constantes, usa siempre nombres significativos con lo respecto a lo que almacenan.
- El uso de constantes es necesario para hacer el programa más claro y mantenible.
- Usar un flag o bandera si necesitamos recordar en un punto del programa algo que ocurrió anteriormente. También se suelen usar para forzar la salida de un bucle en un determinado caso.

Variables, constantes y expresiones

- Las constantes son escritas en letras MAYÚSCULAS y separadas las palabras con el carácter underscore _.
- Las variables deben escribirse en minúsculas, si la variable está formada por varias palabras, la segunda y consecutivas palabras empezarán en mayúsculas.

Variables, constantes y expresiones

Signo negativo	-
Paréntesis	()
Aritméticos	+ - * / // suma, resta, multiplicación, división y división entera
	% resto de la división o el módulo
	** Exponente
Relacionales	== != < > <= >= igual, distinto, menor, mayor, menor o igual, mayor o igual
Lógicos	and or not
Asignación	= += -= *= /* //= **= %=
Asignación aumentada	Var +=1

• Formato programa en Python

- Antes de iniciar es bueno definir que significa "indentar". En principio esta palabra no es aceptada en castellano pues es un término adaptado del inglés "indentation" y que debería traducirse como "sangrado" aunque en el contexto de ciencias de la computación suele usarse mucho más el anglicismo "indentar" que el término castizo "sangrar" e incluso en algunas ocasiones como el vulgarismo "identar" resultado de remover la n .
- En general un código adecuadamente indentado/sangrado es más fácil de leer para un humano pues visualmente le indica que nivel de jerarquía tiene cada fragmento del código, aunque usualmente cuando un computador interpreta el código de la mayoría de los lenguajes de programación el resultado es indiferente si se cuenta o no con una buena indentación.

• Formato programa en Python

- La indentación se utiliza para delimitar bloques, por lo que todas las sentencias del mismo bloque deben tener la misma indentación.

Instrucciones de E/S

- Para pedir información al usuario, debe utilizar las funciones integradas en el interprete del lenguaje, así como los argumentos de línea de comandos.
  - La función raw_input() siempre devuelve un valor de cadenas de caracteres.
  - La función input() siempre devuelve un valor numérico.

Instrucciones de E/S

- La forma general de mostrar información por pantalla es mediante una consola de comando, generalmente podemos mostrar texto y variables separándolos con comas, para este se usa la sentencia print.

Estructuras de control: condicional

- Condicional simple

```
if cond:
    instrucciones
```

- Condicional

```
if cond:
    instrucciones
else:
    instrucciones
```

Estructuras de control:condicional

- Condicional compuestole

```
if cond:
    instrucciones
elif cond:
    instrucciones
else:
    instrucciones
```

Estructura de control:iterativas

```
while cond:
    instrucciones


for var in (valor1,valor2):
    instrucciones
```

Estructura de control. Consejos

- Intentar siempre no repetir código . Por ejemplo, cuando tenemos que escribir el mismo texto pero con diferente resultado no poner varias instrucciones ESCRIBIR. Usar una variable para el resultado y así tenemos una sola instrucción ESCRIBIR. De esta forma si cambiara el texto solo tendría que cambiar una instrucción

Estructura de control. Consejos

- Si se trata de problemas de casuística
  - Determinar si se trata de casos excluyentes o solapados (o mezcla de ambos).
  - Usar la rama del SI para poner los casos más fáciles, que no se bifurcan más
  - Estudiar la lógica del problema concreto e identificar el orden en que deben evaluarse las condiciones para que no se repitan
  - Si en la rama del SI y en la del NO se repite código, corregirlo

Estructura de control. Consejos

- Acerca de los bucles
  - Es muy común al principio programar un bucle infinito. Dentro del bucle debe existir alguna instrucción que haga que la condición del bucle pase a ser Falsa, para poder salir del bucle.
  - Hay que pensar que la condición del bucle debe ser verdadera para que el bucle de vueltas. Cuando se sale del bucle, en la línea siguiente seguro que se cumple la condición contraria.
  - Repasar los casos límite, es muy común que el bucle de una vuelta más o menos de lo que se espera.
  - Empezar programando el caso general. Comprobar si los casos particulares entran en el general, y si no es así, incluirlo en la casuística.