NOCIONES BASICAS DE

Miguel Iván Bobadilla

Indice

Introducción

Python es un lenguaje de programación interpretado legible y limpio, fácil de aprender debido a la sencillez de su sintaxis. Es apto para aprender programación debido a que al tener una codificación simple facilita el aprendizaje y la depuración de código ya que no posee codificación compleja.

Python posee la ventaja de que podemos ahorrar mucho tiempo en desarrollo ya que las líneas de códigos son pocas a diferencia de otros lenguajes. Su simplicidad evita errores de códigos comunes y a una depuración rápida.

A pesar de que Python nos limita para proyectos complejos, para proyectos simples y rápidos es una buena opción, además de que si se busca aprender a programar este es una cátedra para todo principiante.

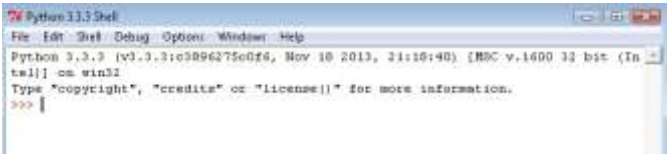
Aclaro que este libro está más orientado a las funciones básicas, no abundaremos funciones complejas ya que este libro está más orientado a principiantes o personas que quieran aprender a programar.

Para descargar Python lo puedes encontrar en el siguiente enlace:

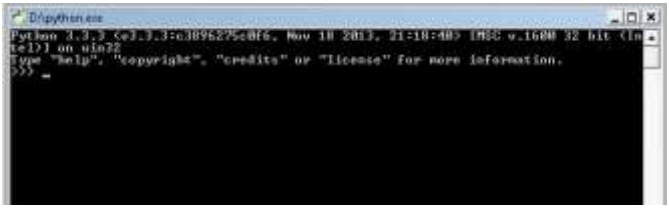
https://www.python.org/downloads/

Luego lo instalas. Al finalizar vas a inicio, todos los programas y seleccionas Python. Te saldrán varias opciones, la que usaremos para ejecutar codigo es IDLE (Python GUI) que llamaremos Python Sell. La otra opción Python (Command line) es para ejecutar líneas de comando en Python, no lo usaremos debido a que solo nos orientaremos a lo básico.

IDLE (Python GUI)



PYTHON (command line)



Sintaxis de Python

Datos

Python nos ofrece un modo sencillo de producir entrada y salida de datos gracias a su forma dinámica, esto quiere decir que no hay que especificar su tipo a la hora de declarar variables ya que lo detecta de forma automática.

Tenemos que tener en cuenta una regla básica, cuando el valor a introducir es una cadena de texto este debe ir entre comillas, para que python pueda distinguir los textos de los valores que no son textos.

En el siguiente ejemplo tenemos el mismo valor pero uno es texto y el otro no. Hay que tener esto muy presente para evitar errores posteriores en el código.

```
>>> 45
45
>>> "45"
'45'
```

Nota: También podemos utilizar comillas simples (´…´) para indicar cadena de texto.

Otra regla básica es que las palabras reservadas siempre irán en minúscula. Tener cuidado con las mayúscula y minúscula ya que Python es sensible a ello.

Tipos de datos básicos

Aunque en Python la escritura de código es dinámica, por lo tanto no es necesario especificar el tipo de variable en las declaraciones, en ciertas ocasiones especiales los tendremos que usar.

Tipo	Función
Int	Variable para números enteros.
Float	Variable para números decimales.
Chr	Variable para un caracter alfanumérico.
Str	Variable para cadenas alfanuméricas.
Bool	Variable condicional (falso o verdadero)

En caso de que quisiéramos conocer el tipo de dato de una variable utilizaremos la sentencia **Type ()**. Dentro del paréntesis colocamos el valor y hacemos enter. “x” representara el valor, variable o cualquier dato que queramos procesar.

Type (x)

Ejemplos:

```
>>> type(10)
<class 'int'>

>>> type(10.5)
<class 'float'>

>>> type("Hola")
<class 'str'>
```

Hagamos un ejemplo con los valores del primer ejemplo sobre la entrada de datos.

```
>>> type(45)
<class 'int'>
>>> type("45")
<class 'str'>
```

Si nos fijamos el primer valor sin las comillas lo detectó como entero y el segundo valor como cadena, por eso es importante tener en cuenta esta regla de entrada.

Asignando valores, variables y constantes

La ventaja de Python ser dinámico es que ahorramos mucho tiempo en el desarrollo del código, por eso no hay que estar declarando el tipo de datos cada vez que realizamos una declaración. Para declarar variables y constantes solo colocamos el nombre de la variable o constante, una igualdad y luego su valor.

Variable = x

```
>>> edad=20
>>> nombre="Juan"
>>> numero=8
>>> pago=200.50
```

Python asigna los valores a cada variable y lo almacena en memoria, con estas variables podemos realizar operaciones posteriores. Hagamos el siguiente ejemplo, a esas mismas variables identifiquemos su tipo.

```
>>> type(edad)
<class 'int'>
>>> type(nombre)
<class 'str'>
>>> type(numero)
<class 'int'>
>>> type(pago)
<class 'float'>
```

Como observamos detecta su tipo de forma automática.

Salida de datos

Para producir una salida de datos solo tenemos que utilizar la sentencia **Print ()**.

Print (x)

```
>>> print(edad)
20
>>> print(nombre)
Juan
>>> print(numero)
8
>>> print(pago)
200.5
```

También podemos usar **Print** para mostrar letreros.

```
>>> print("Hola mundo")
Hola mundo
```

Comentarios

Siempre en la programación es recomendable utilizar comentarios para que el código este mas organizado, esto nos ayuda a realizar una buena depuración de errores. También para recordar que hace una rutina o para que compañeros (si se trabaja en grupo) puedan entender la función del código.

x

```
>>> #esto es un comentario
```

Entrada de datos

Para recibir entrada de datos utilizamos la sentencia **Input ()**.

Variable = imput (texto)

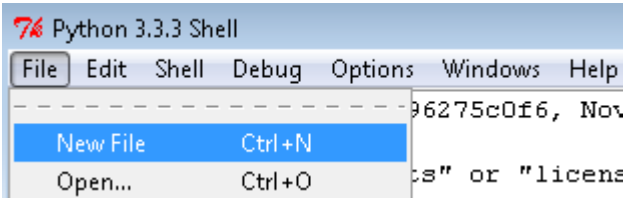
```
>>> edad=input("¿Cual es su edad?")
```

También podemos escribir el código de la siguiente manera si queremos que la entrada se reciba debajo del texto.

Print (texto)

Variable = input ()

Hagamos un simple ejemplo con entradas y salidas. Primero ejecutemos el Python Sell y seleccionamos File, luego New File.

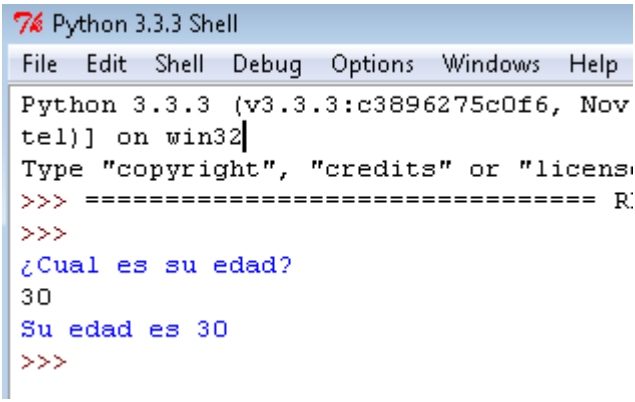


Se ejecutara otra ventana (esta ventana es el editor de Python), aquí escribimos el siguiente código.

```
print("¿Cual es su edad?")
edad=input()
print("Su edad es", edad)
```

Nota: Luego del letrero podemos hacer que aparezca el valor de la variable con solo colocar una coma después de la cadena de texto.

Seleccionamos File y luego en Save. Guardamos el archivo con el nombre que desee. Luego presionamos la tecla F5 y se ejecutara el Python Shell. Debería aparecer algo similar a esto:



Conversión de tipo de datos

En algunos casos habrá operaciones donde dos o más variables tendrán que interactuar y en ello se puede dar que una variable deba cambiar de tipo, como por ejemplo un problema donde tengamos que tomar un número de una cadena de texto o viceversa convertir un número en cadena de texto. También si queremos convertir en decimales a entero o viceversa.

Tipo (x)

```
>>> int(45.60)
45
>>> str(5)
'5'
>>> float(20)
20.0
```

Operadores aritméticos básicos

Operador	Operación
+	Suma
-	Resta
*	Multiplicación
/	División coma flotante
//	División sin coma flotante
%	Sobrante de una división
**	Potencia

```
>>> 2+2
4
>>> 2-2
0
>>> 2*2
4
>>> 2**2
4
>>> 2/2
1.0
>>> 2//2
1
>>> 2%2
0
```

Problema: Desarrolle un programa que reciba dos números de entrada y que realice las operaciones de la tabla de arriba.

Solución:

```
print("Ingrese primer numero")
a=input()
print("Ingrese segundo numero")
b=input()
c=int(a)+int(b)
print("La suma es: ",c)
c=int(a)-int(b)
print("La resta es: ",c)
c=int(a)*int(b)
print("La multiplicacion es: ",c)
c=int(a)**int(b)
print("La potencia es: ",c)
c=int(a)/int(b)
print("La divison con decimal es: ",c)
c=int(a)//int(b)
print("La divison sin decimal es: ",c)
c=int(a)%int(b)
print("El sobrebante de su division es: ",c)
```

Resultado:

```
Ingrese primer numero
2
Ingrese segundo numero
2
La suma es:  4
La resta es:  0
La multiplicacion es:  4
La potencia es:  4
La divison con decimal es:  1.0
La divison sin decimal es:  1
El sobrebante de su division es:  0
```

Posición numérica de caracteres en una cadena

Python cuenta la cantidad de caracteres en una cadena iniciando desde cero. Esto nos ayuda a identificar la posición de un carácter en dichas cadenas.

Cadena [x], siendo "x" un número entero.

```
>>> "Hola mundo"[5]
'm'
```

Nota: Python cuenta los espacios.

Como podemos observar en el anterior ejemplo si contamos desde 0 hasta llegar a 5 nos toparemos que la posición de la cadena en 5 es "m".

H	o	l	a		m	U	N	d	o
0	1	2	3	4	5	6	7	8	9

Colección de tipos de datos

Para almacenar cualquier tipo de elementos de datos se utilizan las tuplas y las listas. Las **tuplas** se identifican encerrándolas en "()" y son inmutables. Las **listas** se identifican encerrándolas en "[]" y pueden ser editadas.

Variable = (x,y,z)

Variable = [x,y,z]

```
>>> color=("rojo","amarillo","azul","verde")
>>> numeros=[1,2,3,6]
>>> color
('rojo', 'amarillo', 'azul', 'verde')
>>> numeros
[1, 2, 3, 6]
```

Si queremos saber si una variable es una lista o una tupla utilizaremos la ya conocida sentencia **type ()**.

```
>>> type(numeros)
<class 'list'>
>>> type(color)
<class 'tuple'>
```

También si deseamos podemos insertar tuplas dentro de listas y viceversa.

```
>>> enteros=(2,5,8)
>>> decimales=[3.5,7.8,2.4]
>>> enterosydecimales=[enteros,decimales]
>>> enterosydecimales
[(2, 5, 8), [3.5, 7.8, 2.4]]
```

Si queremos saber la posición de un elemento en una lista o tupla utilizamos la siguiente sintaxis. Recuerde que Python inicia desde 0 a contar.

Variable [posición]

```
>>> decimales
[3.5, 7.8, 2.4, 56.89]
>>> decimales[1]
7.8
```

Funcion len ()

La función **len ()** se utiliza para contar la cantidad de datos o elementos en una lista o tupla. Hagamos un ejemplo con el ejemplo de los enteros y decimales.

len (variable)