Python para ingenieros

Fundamentos y aplicaciones

Jorge De Los Santos

Versión 0.1

Contenido

Acerca de...

1. Introducción

¿Qué es Python?

Generalmente cuando se define un lenguaje de programación, se hace referencia a sus características más representativas. Siendo así, Python es un lenguaje de programación desarrollado a finales de los 80's y con sus primeras versiones liberadas en los albores de los 90's. Se caracteriza por ser interpretado, de tipado dinámico , multiparadigma y multiplataforma. Posee una licencia de código abierto (PSFL) desarrollada por la Python Software Foundation (PSF), de modo que el acceso al código fuente del proyecto está disponible para los desarrolladores. Lo anterior ha permitido, en gran medida, la proliferación de una comunidad sólida alrededor del mismo.

Es un lenguaje de propósito general, relativamente sencillo de aprender y con un potencial enorme. Sus aplicaciones pueden pasar desde un simple script para uso personal en las tareas comunes con el ordenador, el desarrollo web, la seguridad informática, y desde luego, las aplicaciones en el ámbito científico y de ingeniería.

2. Tipos de datos

Enteros

En matemáticas se clasifican como enteros aquellos números en el intervalo **(-oo, oo)**, que no contienen una parte decimal o fraccionaria. Dadas las limitaciones de los ordenadores es evidente que no podrá representarse un entero demasiado "grande", generalmente el límite de representación es dado por las características del procesador.

En Python existen 2 tipos de enteros: el tipo entero ordinario (int) y el entero largo (long). La diferencia es precisamente el límite de representación de cada uno. Puede obtener información acerca del máximo entero tecleando lo siguiente en el intérprete de Python:

```
>>> import sys
>>> max_int=sys.maxint
>>> max_intl=sys.maxsize
```

Siendo `max_int` el entero ordinario máximo representado, generalmente este valor equivale a **(2^31)-1**. El valor de `max_intl` corresponde al máximo entero largo.

Para definir un tipo entero no hace falta declararlo de forma explícita, por ejemplo:

```
>>> a=1
>>> b=100
>>> type(a)
<type 'int'>
>>> type(b)
<type 'int'>
>>> c=-100
>>> type(c)
<type 'int'>
```

De coma flotante

Los números de coma flotante son aquellos que normalmente conocemos como números reales. En Python para que un valor numérico sea "reconocido" como real, debe utilizarse la función `float` o bien añadir la parte decimal del número, aun cuando esta sea nula:

```
>>> a=3
>>> b=float(3)
>>> c=3.0
>>> type(a),type(b),type(c)
(<type 'int'>, <type 'float'>, <type 'float'>)
```

Booleanos

También conocidos como valores lógicos, son un tipo de dato fundamental, cuyo valor sólo puede adquirir dos estados: verdadero y falso. Se utilizan comunmente para la toma de decisiones en conjunto con las estructuras de control de flujo. Las constantes booleanas en Python se nombran mediante `True` y `False`.

Por ejemplo, si comparamos dos números enteros cualesquiera para ver si son iguales:

```
>>> a=10
>>> b=15
>>> a==b
False
>>> a>b
False
>>> a<b
True
```

Cadenas de caracteres

Listas

Tuplas

Esto es una tupla

```
    >>> print "HOLA"
```

```
print 1
import os
os.system('pause')
```

Diccionarios

3. Estructuras de control

Sentencia if

La sentencia `if` es una estructura de control flujo simple, utilizada para la toma de decisiones en el curso de ejecución que sigue el programa. En casi todos los algoritmos existen porciones de *código* que sólo deben ejecutarse cuando se cumpla una condición establecida, es ahí donde la sentencia `if` tiene su utilidad. Vea el ejemplo dado a continuación:

```
a = 10
if a > 0:
    print "Número positivo"
```

Veamos: primero se define una variable **a** con valor 10, posteriormente se hace la comprobación que si **a** es mayor a cero, lo cual implicaría imprimir en pantalla un mensaje, de lo contrario no se realizará acción alguna. Para el ejemplo mostrado, puesto que **a** es 10, entonces es un número positivo, y por ende se ejecutará la porción de código incluido en la sentencia `if`.

En muchas situaciones la sentencia `if` se utiliza en conjunto con la sentencia *complementaria* `else`, para formar una bifurcación doble, que permite seleccionar una opción especificada o bien otra por default. A manera de ejemplo vamos a desarrollar un programa que determinará si un número ingresado es par o impar:

```
n = input("n = ")
if (-1)**n == 1:
    print "Par"
else:
    print "Impar"
```

Si la condición propuesta se cumple entonces se ejecutará el bloque incluido dentro de la sentencia if, en cualquier otro caso se ejecutará el bloque definido por la cláusula `else`.

Además de las anteriores, existe otra forma más *general* definida por las sentencias `if-elif-else` cuya estructura general es:

```
if cond1:
    # ...
elif cond2:
    # ...
elif cond3:
    # ...
.
.
.
elif condN:
    # ...
else:
    # ...
```

La anterior es un tipo de bifurcación múltiple, que permite seleccionar entre varias condiciones definidas en la cláusula `if` y en las subsecuentes `elif`, y una opción por defecto establecida en la cláusula `else`. Un ejemplo:

```
n = input("n = ")
if n>0:
    print "Positivo"
elif n<0:
    print "Negativo"
else:
    print "Cero"
```

Bucle for

Bucle while

4. Funciones nativas

5. Funciones definidas por el usuario

6. Introducción a la POO

7. Ficheros

8. Vectores y matrices con NumPy

9. Gráficas con Matplotlib