Introducción al desarrollo de software

Tecnicatura Universitaria en Software Libre

Emiliano P. López

Mayo de 2015

UNIVERSIDAD NACIONAL DEL LITORAL
Facutlad de Ingeniería y Ciencias Hídricas

A todos los que están mirando,
por el amor

Índice general

Contents:

CAPÍTULO 1

Introducción

Docente: Emiliano López (emiliano [dot] lopez [at] gmail [dot] com)

En el presente capítulo introduciremos los conceptos necesarios para desarrollar los primeros algoritmos computacionales. Además, se explican las herramientas necesarias para llevar a cabo el desarrollo y sus diferentes alternativas.

1.1 Motivación

que lindo es programar

1.1.1 ¿Por qué Python?

lo lindo que es python y quienes lo usan, su crecimiento, su ámbito de aplicación: web, científico, etc.

1.2 Instalando Python

Actualmente existen dos versiones de Python comúnmente utilizadas, la versión 2 y 3, ambas son completamente funcionales y muy utilizadas. En este curso nos basaremos en la versión 3.

1.2.1 Windows

Para instalar Python en una máquina con Windows, debemos seguir los siguientes pasos:

- Apuntar el navegador a: https://www.python.org/downloads/windows/
- Ir al link de la última versión disponible (por ej: latest python 3 relase)
- En la sección Files, descargar el instalador correspondiente a su arquitectura (64/32 bits), por ej: https://www.python.org/ftp/python/3.4.3/python-3.4.3.msi
- Ejecutar el instalador (por ej: python-3.4.3.msi) aceptando las opciones por defecto

1.2.2 GNU/Linux

En los sistemas operativos serios, es muy probable que ya contemos con el intérprete instalado, incluso en sus dos versiones. Para instalarlo utilizando los administradores de paquetes debemos ejecutar los siguientes comandos desde una terminal:

Para sistemas basados en Debian (como Ubuntu o sus derivados):

```
apt-get install python3
```

1.3 Entornos de programación

1.3.1 El intérprete interactivo

Ya con el intérprete de Python podemos comenzar programar. Si ejecutamos en una terminal `python3`, ingresaremos al intérprete en modo interactivo y veremos una salida similar a la siguiente:

```
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Podemos ejecutar algunas operaciones matemáticas para corroborar su funcionamiento.

```
>>> 2*5
10
>>> 2*5+10
20
>>> -3*19+3.1415
-53.8585
>>>
```

1.3.2 El intérprete interactivo mejorado

IPython[1] es una interfaz mejorada del intérprete nativo. Se lo puede utilizar en modo consola o a través de una interfaz web. La instalación en GNU/Linux es: `apt-get install ipython3`.

La ejecución de ipython desde una terminal nos arroja una pantalla similar a la siguiente:

```
emiliano@pynandi:~ $ ipython3
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
Type "copyright", "credits" or "license" for more information.

IPython 2.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
```

[1] http://ipython.org

```
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Otra alternativa muy interesante son los notebooks de ipython, una interfaz que permite programar utilizando el navegador web como entorno. No entraremos en detalle ya que posteriormente analizaremos su funcionamiento. Se debe ejecutar en una terminal `ipython3 notebook` y esto abrirá el navegador por defecto con el entorno cargado.

1.3.3 Entorno integrado de desarrollo (IDE)

Un IDE es un entorno que nos facilita las tareas a la hora de programar. Consiste en la integración de un editor de texto, con características de resaltado de sintaxis autocompletado -entre otras-, y el intérprete de Python. Existen cientos de entornos muy buenos, como por ejemplo Spyder[2], PyCharm[3] o Ninja-IDE[4]. Para el presente curso, nos basaremos en Ninja-IDE, software libre que ha sido desarrollado por la comunidad de Python Argentina, PyAr[5].

Una lista bastante completa sobre las IDEs disponibles pueden encontrarse en la wiki oficial de Python[6]

1.4 El primer programa "Adiós mundo!"

El acercamiento inicial a un lenguaje de programación suele ser con el archiconocido programa "Hola mundo". Consiste simmplemente en un programa que muestra en pantalla ese mensaje.

Renunciando a cualquier pretensión de originalidad comenzaremos del mismo modo, pero despidiéndonos. Para esto utilizaremos la instrucción *print()* pasando el mensaje de despedida entre comillas, a continuación la instrucción.

```
print("Adios mundo cruel!")
```

Podemos probar la intrucción directamente desde el intérprete, creando con un editor de texto plano un archivo guardado como `chau.py` y luego ejecutándolo desde la terminal haciendo `python3 chau.py`, o bien utilizando un IDE y haciendo todo desde ahí mismo.

[2] https://github.com/spyder-ide/spyder
[3] https://www.jetbrains.com/pycharm
[4] http://ninja-ide.org
[5] http://python.org.ar
[6] https://wiki.python.org/moin/IntegratedDevelopmentEnvironments

Ahora bien, es muchísimo más lo que podemos hacer programando además de saludar cordialmente. Veamos los elementos de un programa que nos permitirán realizar tareas más complejas y entretenidas.

1.5 Algoritmos computacionales

En forma simplificada, un programa o software es un conjunto de instrucciones que la computadora puede ejecutar. Este procedimiento formado por un conjunto de instrucciones es lo que denominamos algoritmo computacional. Una analogía a un algoritmo computacional es una receta de cocina, por ejemplo:

```
Prender el fuego
Salar la carne
Controlar cada 5 minutos hasta que haya brasas
Poner la carne a la parrilla
Cocinar hasta que esté la carne, controlar cada 5 minutos
Dar vuelta la carne
Cocinar hasta que esté la carne, controlar cada 5 minutos
Si falta sal al probar, salar
```

En esta receta se ven una serie de instrucciones que deben ser seguidas en un determinado orden, en algunos casos contamos con ingredientes, intrucciones, decisiones y acciones que se repiten. No muy distinto a un programa de computación, comencemos con algunos *ingredientes* simples de Python y veamos lo que podemos hacer con ellos.

1.6 Elementos de un programa

A continuación veremos los ingredientes fundamentales de un lenguaje de programación como Python, para llevar a cabo los ejemplos utilizaremos el intérprete interactivo mejorado ipython.

1.6.1 Números y expresiones

Frecuentemente requerimos resolver cálculos matemáticos, las operaciones aritméticas básicas son:

- adición: +
- sustracción: -
- multiplicación: *
- división: /
- módulo: %
- potencia: **
- división entera: //

Las operaciones se pueden agrupar con parentesis y tienen precedencia estándar. Veamos unos ejemplos.

```
In [9]: 1/3
Out[9]: 0.3333333333333333

In [10]: 1//3
Out[10]: 0

In [11]: 10%3
Out[11]: 1

In [12]: 4%2
Out[12]: 0
```

El caso de la potencia, también nos sirve para calcularraices. Veamos una potencia al cubo y luego una raíz cuadrada, equivalente a una potencia a la 1/2.

```
In [13]: 5**3
Out[13]: 125

In [14]: 2**(1/2)
Out[14]: 1.4142135623730951
```

Los datos numéricos que obtenidos en las operaciones previas se clasifican en reales y enteros, en python se los clasifica como float e int respectivamente, además existe el tipo complex, para números complejos.

Utilizando la función type() podemos identificar el tipo de dato. Veamos:

```
In [15]: type(0.333)
Out[15]: float

In [16]: type(4)
Out[16]: int
```

1.6.2 Cadenas de carateres

Además de números, es posible manipular texto. Las cadenas son secuencias de caracteres encerradas en comillas simples (‘...’) o dobles (”...”), el tipo de datos es denominado *str*. Sin adentrarnos en detalles, que posteriormente veremos, aquí trataremos lo indispensable para poder desarrollar los primeros programas. Veamos unos ejemplos:

```
>>> 'huevos y pan'          # comillas simples
'huevos y pan'
```

Los operadores algebraicos para la suma y multiplicación tienen efecto sobre las cadenas:

```
>>> 'eco '*4                # La multiplicación repite la cadena
'eco eco eco eco '
```

```
>>>'yo y '+ 'mi otro yo'   # La suma concatena dos o mas cadenas
'yo y mi otro yo'
```

Es posible utilizar cadenas de más de una línea, anteponiendo **triples comillas** simples o dobles al inicio y al final, por ejemplo (fragmento del poema de Fortunato Ramos *Yo jamás fui un niño*):

```
'''
Mi sonrisa es seca y mi rostro es serio,
mis espaldas anchas, mis músculos duros
mis manos partidas por el crudo frío
sólo ocho años tengo, pero no soy un niño.
'''
```

```
`nMi sonrisa es seca y mi rostro es serio,nmis espaldas anchas, mis múscu
```

1.6.3 Variables

Las variables son contenedores para almacenar información. Por ejemplo, para elevar un número al cubo podemos utilizar 3 variables, para la base (*num1*), para el exponenete (*num2*) y para almacenar el *resultado*:

```
num1 = 5                                # A num1 se le asigna el valor numérico 5.
num2 = 3                                # A num2 se le asigna 3.
resultado = num1**num2                  # A resultado se le asigna num1 elevado a
print('El resultado es', resultado)
```

El operador igual (=) sirve para asignar lo que está a su derecha, a la variable que se encuentra a su izquierda. Implementemos la siguiente ecuación para dos valores de x, 0.1 y 0.2.

$$y = (x - 4)^2 - 3$$

```
x1 = 0.1
y1 = (x1-4)**2-3

x2 = 0.2
y2 = (x2-4)**2-3

print(x1,y1)
print(x2,y2)
```

Veremos la siguiente salida por pantalla:

```
0.1 12.209999999999999
0.2 11.44
```

Otros ejemplos utilizando variables que contengan **cadenas de caracteres**:

```
cadena1 = 'siento que '
cadena2 = 'nací en el viento '

cadena3 = cadena1 + cadena2

print(cadena3)
```

Los nombres de las variables (identificador o etiqueta) pueden estar formados por letras, dígitos y guiones bajos, teniendo en cuenta ciertas restricciones, no pueden comenzar con un número y ni ser algunas de las siguientes palabras reservadas:

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

Se debe tener en cuenta que las variables diferencian entre mayúsculas y minúsculas, de modo que juana, JUANA, JuAnA, JUANa son variables diferentes. Esta característica suele denominarse como *case-sensitive*.

1.6.4 Entrada y salida de datos

De los ejemplos que vimos, los valores que almacenan las variables fueron ingresadas en el mismo código, difícilmente sea útil contar con los valores cargados en el programa en forma estática. Por esta razón, generalmente se requiere leer información de diferentes fuentes, puede ser desde un archivo o bien interactuando con un usuario.

La lectura de datos desde el teclado se realiza utilizando la sentencia *input()* del siguiente modo:

```
nombre = input("¿Cómo es su nombre, maestro? ")
print "Hola, " + nombre + "!"
```

El comportamiento es:

```
¿Cómo es su nombre, maestro?
Juan de los palotes
Hola, Juan de los palotes!
```

Es importante tener en cuenta que toda lectura por teclado utilizando la función input() va a almacenar lo ingresado como una variable de tipo *str*, es decir una cadena de caracteres. Veamos el error que obtenemos al intentar operar directamente el valor leído en una ecuación matemática con el siguiente código:

```
x = input("Ingrese x = ")
y = (x-4)**2-3
print(x,y)
```

```
Ingrese x = 3

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-15-3baa5c95d16e> in <module>()
      1 x = input("Ingrese x = ")
----> 2 y = (x-4)**2-3
      3 print(x,y)


TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

El error que obtenido se debe a que estamos realizando operaciones incompatibles entre diferentes tipos, una cadena de caracteres (str) con enteros (int).

1.6.5 Operadores relacionales y lógicos

1.6.6 Funciones

1.6.7 Módulos

1.7 Ejercicios

CAPÍTULO 2

Estructuras de control

2.1 Condicionales

2.1.1 if

2.1.2 else

2.1.3 Estructuras anidadas

2.1.4 elif

2.2 Repeticiones

2.2.1 while

2.3 Ejercicios

```
for i in range(1,10):
    if i%2==0:
        print(i,"es par")
    else:
        print(i, "es impar")
```

```
1 es impar
2 es par
3 es impar
4 es par
5 es impar
6 es par
7 es impar
8 es par
9 es impar
```

CAPÍTULO 3

Más estructuras de datos y control

3.1 Listas

3.2 for

3.3 Manipulando textos con strings

CAPÍTULO 4

Funciones, archivos, diccionarios

4.1 Definiendo funciones

4.1.1 Variables globales y locales

Agrupando el código en módulos

4.2 Números aleatorios

4.3 Lectura y escritura de archivos

4.4 Diccionarios

CAPÍTULO 5

Clases y objetos

CAPÍTULO 6

This is a Title

That has a paragraph about a main subject and is set when the '=' is at least the same length of the title itself.

6.1 Subject Subtitle

Subtitles are set with '-' and are required to have the same length of the subtitle itself, just like titles.

Lists can be unnumbered like:

- Item Foo
- Item Bar

Or automatically numbered:

1. Item 1
2. Item 2

6.2 Inline Markup

Words can have *emphasis in italics* or be **bold** and you can define code samples with back quotes, like when you talk about a command: `sudo` gives you super user powers!

CAPÍTULO 7

Indices and tables

- *genindex*
- *modindex*
- *search*