
Introducción al desarrollo de software

Publicación 1.0

Emiliano López

14 de May de 2015

1. Introducción	3
1.1. Instalando Python	3
1.2. Entornos de programación	4
1.3. Entorno integrado de desarrollo (IDE)	4
1.4. El primer programa “Adiós mundo!”	5
1.5. Elementos de un programa	5
1.6. Ejercicios	5
2. Estructuras de control	7
2.1. Condicionales	7
2.2. Repeticiones	7
2.3. Ejercicios	7
3. Más estructuras de datos y control	9
3.1. Listas	9
3.2. for	9
3.3. Manipulando textos con strings	9
4. Funciones, archivos, diccionarios	11
4.1. Definiendo funciones	11
4.2. Números aleatorios	11
4.3. Lectura y escritura de archivos	11
4.4. Diccionarios	11
5. Clases y objetos	13
6. This is a Title	15
6.1. Subject Subtitle	15
6.2. Inline Markup	15
7. Indices and tables	17

Contents:

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 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.

1.1.1 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

1.1.2 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
```

1.2 Entornos de programación

1.2.1 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
>>>
```

1.2.2 El intérprete interactivo mejorado

IPython¹ 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.
%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 Entorno integrado de desarrollo (IDE)

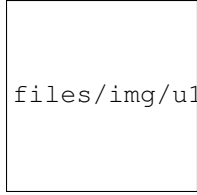
Un IDE es un entorno que nos facilita las tareas a la hora de programar. Consiste en la integración de un editor de texto, con características de resaltado de sintaxis autocompletado -entre otras-, y el intérprete de Python. Existen cientos de entornos muy buenos, como por ejemplo **Spyder**², **PyCharm**³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

¹<http://ipython.org>

²<https://github.com/spyder-ide/spyder>

³<https://www.jetbrains.com/pycharm>

⁴<http://ninja-ide.org>



files/img/ul/ninja-ide.png

Ninja-IDE, software libre que ha sido desarrollado por la comunidad de Python Argentina, [PyAr](http://python.org.ar)⁵.

Una lista bastante completa sobre las IDEs disponibles pueden encontrarse en la [wiki oficial de Python](https://wiki.python.org/moin/IntegratedDevelopmentEnvironments)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 simplemente 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.

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.

⁵<http://python.org.ar>

⁶<https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>

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.

```
1/3
```

```
0.3333333333333333
```

```
1//3
```

```
0
```

```
10%3
```

```
1
```

```
4%2
```

```
0
```

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

```
5**3
```

```
8
```

```
2**(1/2)
```

```
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:

```
type(0.333)
```

```
float  
type(3)  
  
int
```

1.6.2 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 exponente (*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 num2.  
print("El resultado es", resultado)
```

Del código podemos rescatar que el operador igual (=) sirve para asignar lo que está a la derecha a la variable que se encuentra a la izquierda.

1.6.3 Entrada y salida de datos

Comunicarnos con el entorno

1.6.4 Operadores relacionales y lógicos

1.6.5 Funciones

1.6.6 Módulos

1.7 Ejercicios

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
```

Más estructuras de datos y control

3.1 Listas

3.2 for

3.3 Manipulando textos con strings

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

Clases y objetos

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!

Indices and tables

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