Resumen de clase 04

Emilio López Sotelo

10/01/2019

La cuarta clase se enfocó en trabajar con Python y familiarizarse con el funcionamiento de este lenguaje de programación. El primer paso fue abrir "Idle" para poder empezar a trabajar, para hacer esto basta con ejecutar el comando "idle-python2.7" desde el bash y posteriormente se abrirá una ventana nueva de "Idle". En esta ventana de "Idle" nos encontramos en un shell de python y desde aquí podemos:

- crear archivos nuevos
- abrir y modificar archivos existentes
- correr programas de python
- escribir directamente en el shell como si lo hubieramos abierto desde bash, entre otras cosas.

Estando en el shell pudimos de primera mano observar algunos de los comportamientos natos que ocurren dentro de Python. Un ejemplo de esto fue darnos cuenta que Python por default toma los números ingresados sin punto decimal como enteros y de la misma manera una operación entre ellos da como resultado un entero. Es decir que para el shell no es lo mismo escribir 5 que 5.0 y de la misma manera no es lo mismo hacer la división $\frac{5}{2}$ a la división $\frac{5.0}{2}$. En el primer caso el resultado es 2 ya que como es una operación entre enteros el resultado debe ser un entero. En el segundo caso el resultado es 2.5 ya que como escribimos 5.0 como operando Python automáticamente lo reconoce

como un numero flotante, esto quiere decir una representación hecha por la computadora de un numero real, y por lo tanto también reconoce al 2 como un numero flotante y esto tiene como resultado que como es una operación entre flotantes el resultado debe ser un flotante. También aprendimos que al asignarle un valor a una variable esta conserva su valor hasta que se le asigna otro valor a la misma variable, cuando esto ocurre se sobrescribe el segundo valor sobre el primero y este ultimo deja de existir. Por supuesto también vimos varios de los operadores y funciones básicas de Python, por ejemplo "**" es la forma de utilizar una potencia de manera que "2**3" en Python es igual a 2^3 mientras que "print" es la función que muestra algo en la pantalla, por ejemplo si yo escribo "l+h" en el shell Python me va a marcar un error ya que "l" y "h" no están definidas pero si yo escribo "print 'l+h'" en mi pantalla lo que va a aparecer en una nueva linea es "l+h" ya que Python no lo esta leyendo como una suma de variables si no como una cadena de caracteres que debe de mostrar por instrucción de la función "print".

Finalmente otra de las cosas indispensables vistas en esta clase fue que no solo existen palabras reservadas de Python si no también símbolos reservados de Python, en particular uno muy importante es "#". Este símbolo tiene como finalidad el poder hacer uso de comentarios dentro del código para por ejemplo explicar que se esta haciendo en alguna linea específica o para que sirve el programa en su totalidad. Como funciona "#" es de hecho una idea muy sencilla, si se utiliza el caracter "#" todo lo que se escribe después del símbolo Python lo omite o, dicho de otra forma, no lo lee. Un ejemplo de esto seria que si yo ingreso "5+2" en el shell este me mostraría en una nueva linea el resultado de la operación el cual es "7", por otro lado si yo lo ingreso como #5+2 en el shell esto tendría el mismo efecto a que si yo hubiera presionado "Intro" sin escribir nada en el shell, es decir Python no lee nada asi que simplemente me muestra de nuevo el prompt. Es importante destacar que en este caso no se muestra una linea nueva en blanco si no que simplemente un prompt inmediatamente abajo del prompt anterior. "#" también tiene este mismo efecto aun con palabras reservadas de Python, es decir que mientras que "print 'a" me mostrará "a" en una linea nueva "#print 'a'" tendra el mismo efecto que "#5+2" en el ejemplo anterior.

Por último para concluir la clase intentamos solucionar el problema de tiro parabólico de la clase anterior utilizando Python. Con este ejercicio se vieron varias cosas. En primer lugar fuera del problema en si aprendimos como crear

un documento nuevo en "Idle" y lo importante que es guardarlo con la extensión .py y dentro de nuestro repositorio local para conseguir dos cosas. Por un lado con la extensión nos aseguramos que el shell de Python reconozca al archivo como un archivo de Python y de esta manera lo pueda ejecutar. Por otro lado el guardarlo en nuestro repositorio local nos permite usar lo que vimos en la clase pasada para poder subir este archivo a Github en la nube. En segundo lugar fue un buen primer acercamiento a como se definen variables en Python y también como ocupar la función "print" para mostrar un resultado. En tercer y ultimo lugar vimos como ejecutar nuestro programa desde el archivo .py en el shell, esto es muy poderoso y de gran utilidad ya que en lugar de tener que reescribir las instrucciones del programa en el shell cada vez que se quiera ejecutar basta con escribirlas una vez en el archivo .py y de esta manera se puede ejecutar el programa cada vez que se corre el archivo .py.