introduccion

February 2, 2017

Figure 1: taller_python

1 Programar ¿con qué se come?

La computadora es una gran gran calculadora que permite hacer cualquier tipo de cuenta de las que necesitemos dentro de la Física (y de la vida también) mientras sepamos cómo decirle a la máquina qué cómputos hacer.

La computadora para hacer cuentas tiene que almacenar los números que necesitemos y luego hacer operaciones con ellos. Nuestros valores numéricos se guardan en espacios de memoria, y esos espacios tienen un nombre, un rótulo con el cual los podremos llamar y pedirle a la computadora que los utilice para operar con ellos, los modifique, etc. Ese nombre a cada espacio de memoria se asigna, al menos en Python, con el símbolo = que significa de ahora en más: "asignación".

Pero no sólamente guardaremos valores numéricos. Además de haber distintos tipos de valores numéricos, como veremos ahora, podemos guardar otros tipos de datos, como texto (strings) y listas (lists) entre muchos otros. Todos los tipos de valores que podremos almacenar difieren entre si el espacio en memoria que ocupan y las operaciones que podremos hacer con ellos.

Veamos un par de ejemplos

```
In [2]: x = 5
        y = 'Hola mundo!'
        z = [1,2,3]
```

Aquí hemos guardado en un espacio de memoria llamado por nosotros "x" la información de un valor de tipo entero, 5, en otro espacio de memoria, que nosotros llamamos "y" guardamos el texto "Hola mundo!". En Python, las comillas indican que lo que encerramos con ellas es un texto. x no es un texto, así que Python lo tratará como variable para manipular. "z" es el nombre del espacio de memoria donde se almacena una lista con 3 elementos enteros.

Podemos hacer cosas con esta información. Python es un lenguaje interpretado (a diferencia de otros como Java o C++), eso significa que ni bien nosotros le pedimos algo a Python, éste lo ejecuta. Así es que podremos pedirle por ejemplo que imprima en pantalla el contenido en y, el tipo de valor que es x (entero) entre otras cosas.

```
In [3]: print y
        print type(x)
        print type(y), type(z), len(z)
```

```
Hola mundo!
<type 'int'>
<type 'str'> <type 'list'> 3
```

Vamos a utilizar mucho la función type() para entender con qué tipo de variables estamos trabajando. type() es una función predeterminada por Python, y lo que hace es pedir como argumento (lo que va entre los paréntesis) una variable y devuelve inmediatamente el tipo de variable que es.

1.0.1 Ejercicio 1

En el siguiente bloque cree las variables "dato1" y "dato2" y guarde en ellas los textos "estoy programando" y "que emocion!". Con la función type() averigue qué tipo de datos se almacena en esas variables.

```
In [2]: # Realice el ejercicio 1
```

Para las variables integers(enteros) y floats (flotantes) podemos hacer las operaciones matemáticas usuales y esperables. Veamos un poco las compatibilidades entre estos tipos de variables.

```
In [3]: a = 5
        b = 7
        c = 5.0
        d = 7.0
        print a+b, b+c, a*d, a/b, a/d, c**2
```

```
12 12.0 35.0 0 0.714285714286 25.0
```

1.0.2 Ejercicio 2

Calcule el resultado de

$$\frac{(2+7.9)^2}{4^{7.4-3.14*9.81}-1}$$

y guárdelo en una variable

```
In [4]: # Realice el ejercicio 2. El resultado esperado es -98.01
```

1.1 Listas

Las listas son cadenas de datos de cualquier tipo, unidos por estar en una misma variable, con posiciones dentro de esa lista, con las cuales nosotros podemos llamarlas. En Python, las listas se enumeran desde el 0 en adelante.

Estas listas también tienen algunas operaciones que le son válidas.

Distintas son las tuplas. Las listas son editables, pero las tuplas no. Esto es importante cuando, a lo largo del desarrollo de un código donde necesitamos que ciertas cosas no cambien, no editemos por error valores fundamentales de nuestro problema a resolver.

```
In [5]: lista1 = [1, 2, 'saraza']
        print lista1, type(lista1)
        print lista1[1], type(lista1[1])
        print lista1[2], type(lista1[2])
        print lista1[-1]
```

```
[1, 2, 'saraza'] <type 'list'>
2 <type 'int'>
saraza <type 'str'>
saraza
```

```
In [6]: lista2 = [2,3,4]
        lista3 = [5,6,7]
        print lista2+lista3
        print lista2[2]+lista3[0]
```

```
[2, 3, 4, 5, 6, 7]
9
```

```
In [7]: tupla1 = (1,2,3)
        lista4 = [1,2,3]
        lista4[2] = 0
        print lista4
        #tupla1[0] = 0
        print tupla1
```

```
[1, 2, 0]
(1, 2, 3)
```

Hay formas muy cómodas de hacer listas. Presentamos una que utilizaremos mucho, que es usando la función range.

```
In [8]: listilla = range(10)
        print listilla, type(listilla)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] <type 'list'>
```

1.1.1 Ejercicio 3

1. Haga una lista con los resultados de los últimos dos ejercicios y que la imprima en pantalla

- Sobreescriba en la misma variable la misma lista pero con sus elementos permutados e imprima nuevamente la lista

Ejemplo de lo que debería mostrarse en pantalla
['estoy programando', 'que emocion!', -98.01]
['estoy programando', -98.01, 'que emocion!']

```
In [9]: # Realice el ejercicio 3
```

1.1.2 Ejercicio 4

1. Haga una lista con la función range de 15 elementos y sume los elementos 5, 10 y 12

- Con la misma lista, haga el producto de los primeros 4 elementos de esa lista
- Con la misma lista, reste el último valor con el primero

```
In [ ]: # Realice el ejercicio 4
```

1.2 Booleans

Este tipo de variable tiene sólo dos valores posibles: 1 y 0, o True y False. Las utilizaremos escencialmente para que Python reconozca relaciones entre números.

```
In [10]: print 5>4
         print 4>5
         print 4==5 #La igualdad matemática se escribe con doble ==
         print 4!=5 #La desigualdad matemática se escribe con !=
         print type(4>5)
```

```
True
False
False
True
<type 'bool'>
```

1.2.1 Ejercicio 5

Averigue el resultado de 4!=5==1. ¿Dónde pondría paréntesis para que el resultado fuera distinto?

```
In [15]: # Realice el ejercicio 5
```

1.3 Control de flujo: condicionales e iteraciones (if y for para los amigos)

Si en el fondo un programa es una serie de algoritmos que la computadora debe seguir, un conocimiento fundamental para programar es saber cómo pedirle a una computadora que haga operaciones si se cumple una condición y que haga otras si no se cumple. Nos va a permitir hacer programas mucho más complejos. Veamos entonces como aplicar un if.

```
In [13]: parametro = 5
         if parametro > 0: # un if inaugura un nuevo bloque indentado
             print 'Tu parametro es', parametro, 'y es mayor que cero'
             print 'Gracias'
         else:             # el else inaugura otro bloque indentado
             print 'Tu parametro es', parametro, 'y es menor o igual que cero'
             print 'Gracias'
         print 'Vuelva pronto'
         print ' '
```

```
Tu parametro es 5 y es mayor que cero
Gracias
Vuelva pronto
```

```
In [14]: parametro = -5
         if parametro > 0: # un if inaugura un nuevo bloque indentado
             print 'Tu parametro es', parametro, 'y es mayor que cero'
             print 'Gracias'
         else:             # el else inaugura otro bloque indentado
             print 'Tu parametro es', parametro, 'y es menor o igual que cero'
             print 'Gracias'
         print 'Vuelva pronto'
         print ' '
```

```
Tu parametro es -5 y es menor o igual que cero
Gracias
Vuelva pronto
```

1.3.1 Ejercicio 6

Haga un programa con un if que imprima la suma de dos números si un tercero es positivo, y que imprima la resta si el tercero es negativo.

```
In [ ]: # Realice el ejercicio 6
```

Para que Python repita una misma acción n cantidad de veces, utilizaremos la estructura for. En cada paso, nosotros podemos aprovechar el "número de iteración" como una variable. Eso nos servirá en la mayoría de los casos.

```
In [21]: nueva_lista = ['nada',1,2,'tres', 'cuatro', 7-2, 2*3, 7/1, 2**3, 3**2]
         for i in range(10):       # i es una variable que inventamos en el for, y que tomará los valor
             print nueva_lista[i]   #lista que se genere con range(10)
```

```
nada
1
2
tres
cuatro
5
6
7
8
9
```

1.3.2 Ejercicio 7

1. Haga otra lista con 16 elementos, y haga un programa que con un for imprima solo los primeros 7

- Modifique el for anterior y haga que imprima solo los elementos pares de su lista

```
In [ ]: # Realice el ejercicio 7
```

La estructura while es poco recomendada en Python pero es importante saber que existe: consiste en repetir un paso mientras se cumpla una condición. Es como un for mezclado con un if.

```
In [22]: i = 1
         while i < 10: # tener cuidado con los while que se cumplen siempre. Eso daría lugar a los loop
             i = i+1
             print i
```

```
2
3
4
5
6
7
8
9
10
```

1.3.3 Ejercicio 8

1. Hacer una función que calcule el factorial de N, siendo N la única variable que recibe la función (Se puede pensar usando for o usando while).

- Hacer una función que calcule la sumatoria de los elementos de una lista.

```
In [ ]: # Realice el ejercicio 8
```

1.4 Funciones

Pero si queremos definir nuestra propia manera de calcular algo, o si queremos agrupar una serie de órdenes bajo un mismo nombre, podemos definirnos nuestras propias funciones, pidiendo la cantidad de argumentos que querramos.

Vamos a usar las funciones lambda más que nada para funciones matemáticas, aunque también tenga otros usos. Definamos el polinomio $f(x) = x^2 - 5x + 6$ que tiene como raíces $x = 3$ y $x = 2$.

```
In [16]: f = lambda x: x**2 - 5*x + 6
         print f(3), f(2), f(0)
```

```
0 0 6
```

Las otras funciones, las más generales, se las llama funciones def, y tienen la siguiente forma.

```
In [17]: def promedio(a,b,c):
             N = a + b + c # Es importante que toda la función tenga su contenido indentado
             N = N/3.0
             return N
         mipromedio = promedio(5,5,7) # Aquí rompimos la indentación
         print mipromedio
```

```
5.66666666667
```

1.4.1 Ejercicio 9

Hacer una función que calcule el promedio de n elementos dados en una lista.

Sugerencia: utilizar las funciones len() y sum() como auxiliares.

```
In [18]: # Realice el ejercicio 9
```

1.4.2 Ejercicio 10

Usando lo que ya sabemos de funciones matemáticas y las bifurcaciones que puede generar un if, hacer una función que reciba los coeficientes a, b, c de la parábola $f(x) = ax^2 + bx + c$ y calcule las raíces si son reales (es decir, usando el discriminante $\Delta = b^2 - 4ac$ como criterio), y sino que imprima en pantalla una advertencia de que el cálculo no se puede hacer en $\mathbb{R}$.

1.4.3 Bonus track 1

Modificar la función anterior para que calcule las raíces de todos modos, aunque sean complejas.

```
In [24]: # Realice el ejercicio 10
```

1.4.4 Ejercicio 11

1. Hacer una función que calcule el factorial de N, siendo N la única variable que recibe la función (Se puede pensar usando for o usando while).

- Hacer una función que calcule la sumatoria de los elementos de una lista.

```
In [ ]: # Realice el ejercicio 11
```

1.5 Bibliotecas

Pero las operaciones básicas de suma, resta, multiplicación y división son todo lo que un lenguaje como Python puede hacer "nativamente". Una potencia o un seno es álgebra no lineal, y para hacerlo, habría que inventarse un algoritmo (una serie de pasos) para calcular por ejemplo sen(π). Pero alguien ya lo hizo, ya lo pensó, ya lo escribió en lenguaje Python y ahora todos podemos usar ese algoritmo sin pensar en él. Solamente hay que decirle a nuestro intérprete de Python dónde está guardado ese algoritmo. **Esta posibilidad de usar algoritmos de otros es fundamental en la programación, porque es lo que permite que nuestro problema se limite solamente a entender cómo llamar a estos algoritmos ya pensados y no tener que pensarlos cada vez**.

Vamos entonces a llamar a una biblioteca llamada math que nos va a extender nuestras posibilididades matemáticas.

```
In [14]: import math # Llamamos a una biblioteca
         #Es usual usar math, en vez de m
         r1 = math.pow(2,4)
         r2 = math.cos(m.pi)
         r3 = math.log(100,10)
         r4 = math.log(m.e)
         print r1, r2, r3, r4
```

```
16.0 -1.0 2.0 1.0
```

Para entender cómo funcionan estas funciones, es importante recurrir a su documentation. La de esta biblioteca en particular se encuentra en

https://docs.python.org/2/library/math.html

1.5.1 Ejercicio 12

Use Python como calculadora y halle los resultados de

1. $\log(\cos(2\pi))$

- $\text{atanh}(2^{\cos(e)} - 1)$
- $\sqrt{x^2 + 2x + 1}$ con $x = 125$

```
In [15]: # Realice el ejercicio 12
```

1.5.2 Bonus track 2

Ahora que nos animamos a buscar nuevas bibliotecas y definir funciones, buscar la función newton() de la biblioteca **scipy.optimize** para hallar x tal que se cumpla la siguiente ecuación no lineal

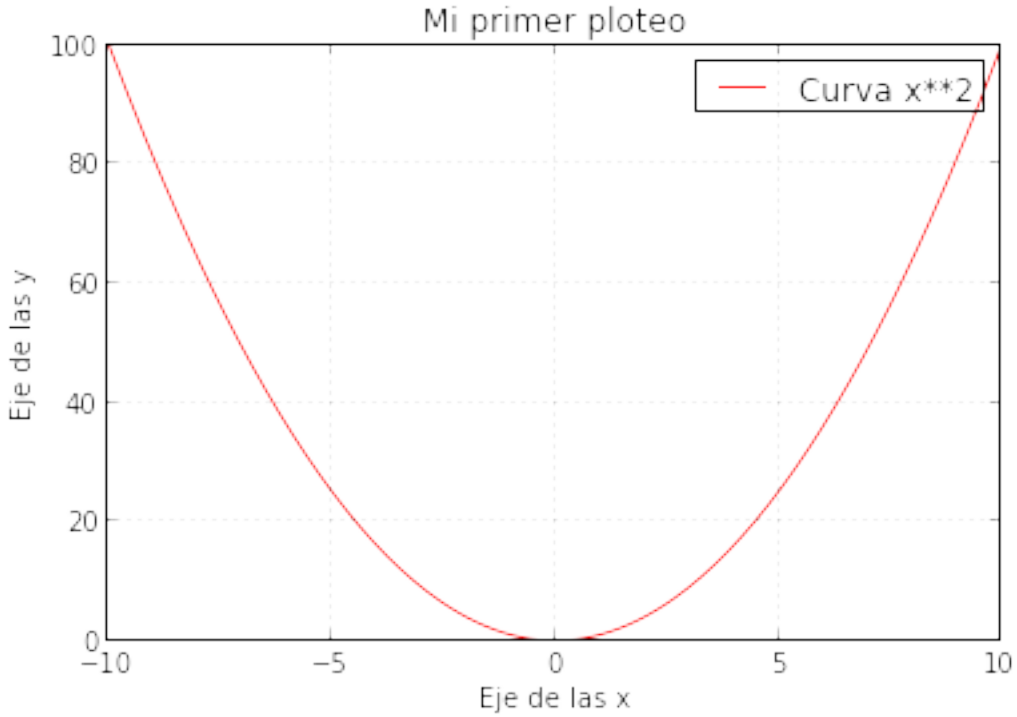
$$\frac{1}{x} = ln(x)$$

1.6 Gráficos

Lo siguiente que Python tiene de interesante para usar son sus facilidades para hacer gráficos. La biblioteca **matplotlib** nos ayudará en este caso. También utilizaremos una biblioteca llamada **numpy** por sus enormes ventajas para el cálculo numérico, motivo de nuestro próximo taller. Primero, definimos un vector que nos hace de dominio, luego, un vector imagen de alguna función, y luego haremos el gráfico. Se muestran aquí algunas de las opciones que tiene matplotlib para presentar un gráfico, pero yendo a la documentación podrán encontrar infinidad de herramientas para hacer esto.

In [25]:
```
from matplotlib import pyplot as plt
import numpy as np
%matplotlib pyplot

x = np.linspace(-10, 10, 200) # con la función linspace generaremos un vector con componentes
y = x**2 # el vector imagen será igual de largo que x
plt.plot(x,y, '-', color = 'red', label = 'Curva x**2') # ver qué pasa con 'r', 'g', '*' entre
plt.title('Mi primer ploteo')
plt.xlabel('Eje de las x')
plt.ylabel('Eje de las y')
#plt.xlim(-5,5)
#plt.ylim(0,4)
plt.legend(loc = 'best')
grid(True)
```



1.6.1 Ejercicio 13

Hallar de manera gráfica la x que resuelve la ecuación del primer bonus track. Recordamos la ecuación.

$$\frac{1}{x} = ln(x)$$

In [26]:
```
# Bloque para el ejercicio 13
```

2 La importancia de las referencias

Para más referencias pueden googlear. Dejamos algunas de referencia:
http://pybonacci.org/2012/06/07/algebra-lineal-en-python-con-numpy-i-operaciones-basicas/

http://relopezbriega.github.io/blog/2015/06/14/algebra-lineal-con-python/
http://pendientedemigracion.ucm.es/info/aocg/python/modulos_cientificos/numpy/index.html
Pero es importantísimo manejarse con la documentación de las bibliotecas que se utilizan
https://docs.python.org/2/library/math.html
http://docs.scipy.org/doc/numpy/reference/routines.linalg.html
http://matplotlib.org/api/pyplot_api.html

3 Recursos

Para seguir profundizando con la programación en Python, ofrecemos distintos recursos

Un tutorial: http://www.learnpython.org/

How to think like a computer scientist (aprendizaje interactivo): http://interactivepython.org/runestone/static/thinkcspy/index.html

Otro tutorial, en inglés, pero muy completo: http://learnpythonthehardway.org/book

Coursera, que nunca está de más: https://www.coursera.org/learn/interactive-python-1

Otro más: https://es.coursera.org/learn/python

Y por fuera del taller, seguimos en contacto. Tenemos un grupo de Facebook donde pueden hacerse consultas y otros chicos que fueron al taller antes o aprendieron por sus medios podrán responderles. El grupo es https://www.facebook.com/groups/303815376436624/?fref=ts

4 Y para seguir manijeando

Tenemos otro taller dentro de una semana donde profundizaremos el uso de algunas herramientas clave para el análisis de datos de Laboratorio. ¡No te lo podés perder!

A parte, en nuestro Github (https://github.com/fifabsas/talleresfifabsas) iremos colgando nuevo material para ejemplificar, con problemas de las materias de Física resueltos numéricamente, herramientas de Labo y mucho más.

5 Agradecimientos

Todo esto es posible gracias al aporte de mucha gente.

- Gente muy copada del DF como Hernán Grecco, Guillermo Frank, Osvaldo Santillán , Martín Elías Costa y Agustín Corbat por hacer aportes a estos talleres de diferentes maneras, desde poner su apellido para que nos presten un labo hasta venir como invitado a un taller.
- El Departamento de Computación que cuatrimestre a cuatrimestre nos presta los labos desinteresadamente.
- Los estudiantes del CODEP de Física, el CECEN y mucha gente que ayuda con la difusión.
- Pibes de la FIFA que prestan su tiempo a organizar el material y llevan a cabo el taller.
- Todos los que se acercan y piden que estos talleres se sigan dando y nos siguen llenando los Labos. Sí ¡Gracias a todos ustedes!