# GUIA DE EJERCICIOS 1.
# MATEMÁTICAS DISCRETAS

SERGIO HERNÁNDEZ
*SHERNANDEZ@UCM.CL*

## 1. Introducción a Python

Python es un poderoso lenguaje de programación en cuanto a portabilidad, flexibildiad, estilo y extensibilidad. De modo de definir una función o iniciar un ciclo, Python usa identación en vez de corchetes. Al igual que Matlab, Python es un lenguaje interpretado por lo tanto es lento comparado con otros lenguajes compilados como C o Java. Sin embargo, es posible utilizar interfaces a estos lenguajes de manera de ganar eficiencia y al mismo tiempo mantener la versatilidad y rapidez de prototipado.

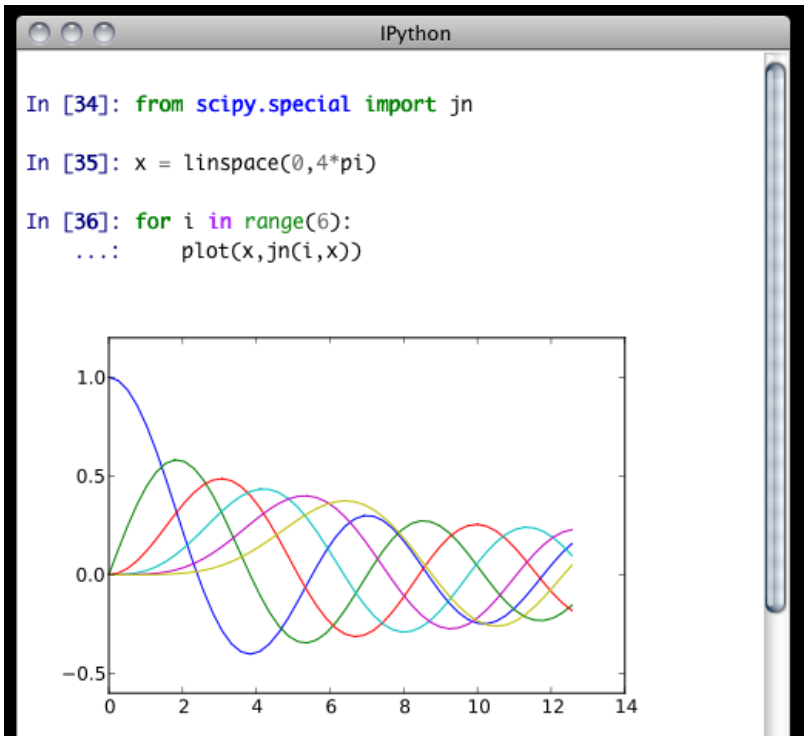


Figure 1. Entorno de desarrollo IPython

Aparte de proveer orientación a objetos y operadores matemáticos básicos, Python por una serie de paquetes (`https://docs.python.org/2/tutorial/modules.html#packages`) que extienden la capacidad del entorno. En particular Numpy y Scipy se usan para para desarrollar funciones matemáticas y gráficas de alta calidad.

1.1. **Ventana de comandos.** Cuando el usuario inicia el IDE, se encuentra con una ventana de comandos y un historial de comandos. Es posible inicializar cualquier variable sin necesidad de declararla para luego utilizarla para realizar cálculos.

Por ejemplo, si necesitamos calcular el volúmen de una esfera de radio $r = 2$:

$$V = \frac{4}{3}\pi r^3$$

En IPython se escribe en la ventana de comandos:

**Program 1** Volumen de una esfera en Python

```
>> import numpy
>> r=2
>> vol=(4./3.)*numpy.pi*(r**3)
>> vol =
33.510321638291124
```

Los vectores de NumPy son mucho más eficientes que las listas de Python puro. Por ejemplo si queremos hacer la siguiente operación $x^2 + x^3$ sobre el vector $x = [0, 1, 2]$.

**Program 2** Suma de vectores en Python

```
def pythonsum(n):
    a = range(n)
    b = range(n)
    c = []
    for i in range(len(a)):
        a[i] = i ** 2
        b[i] = i ** 3
    c.append(a[i] + b[i])
    return c
```

**Program 3** Suma de vectores en NumPy

```
def pythonsum(n):
    a = numpy.arange(n) ** 2
    b = numpy.arange(n) ** 3
    c = a+b
    return c
```

| Operadores Aritméticos | |
|---|---|
| suma | + |
| resta | − |
| multiplicación | ∗ |
| división | / |
| potencia | ∗∗ |
| modulo | % |

| Operadores Relacionales | |
|---|---|
| igual | == |
| no igual | != |
| menor | < |
| mayor | > |
| menor igual | <= |
| mayor igual | >= |

| Operadores Lógicos | |
|---|---|
| AND | & |
| OR | \| |
| OR exclusivo | ∧ |

1.2. **Operadores.** Los siguientes operadores están disponibles:

1.3. **Funciones.** Las funciones en Python se guardan en archivos con extensión .py y equivalen a subrutinas que pueden ser ejecutadas desde la línea de comandos o desde otros programas.

Por ejemplo, la siguiente función:

$$f(x) = \frac{2x^3 + 7x^2 + 3x - 1}{x^2 - 3x + 5\,\mathrm{e}^{-x}}$$

Puede ser escrita en el archivo *demo_func.py* de la siguiente forma:

**Program 4** Función de una sola variable

```
import numpy as np

def y(x):
  dato=float((2*x**3) + (7*x**2) + (3*x)-1)/((x**2)-(3*x)+5*np.e**(-x))
  return  dato
```

**Ejercicio 1**

*1.1 Escriba una función que devuelva la media y la desviación estándar de un vector de entrada.*
*tip : Para un vector de entrada* $x = [x_1, x_2, \ldots, x_n]$ *utilice las funciones* n=length(x) *y* x.sum() *para determinar la media:*

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

*Luego utilice la media para calcular la desviación estándar:*

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

*1.2 Compare los resultados usando la función* x.mean() *y* x.std(). *Grafique los histogramas para distintos valores de* mu,sig *y números de muestras* nsamples

**Program 5** Generación de variables aleatorias

```
>> import numpy as np
>> import matplotlib.pyplot as plt
>> mu=3
>> sig=.5
>> nsamples=1000
>> x=np.random.normal(mu,sig,nsamples)
>> plt.hist(x)
>> plt.show()
```

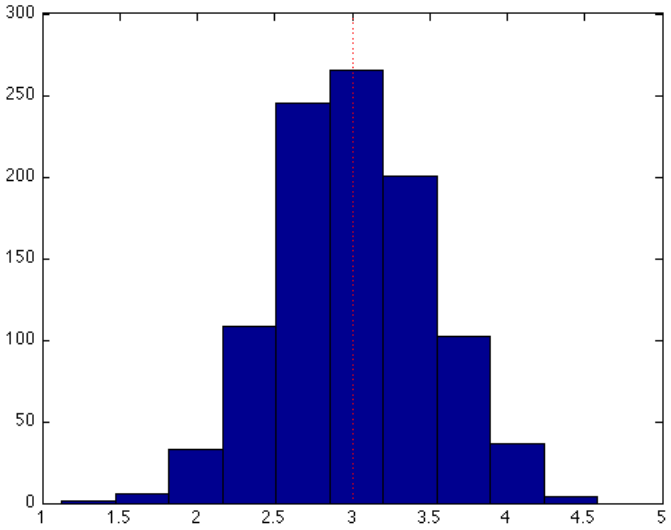


FIGURE 2. Histograma de una distribución Gaussiana

## 2. Comprensiones

Una forma de construir un conjunto es mediante una propiedad, por ejemplo el conjunto $A$ podría ser construido como

$$A = \{2 * x : x \in \{1, 2, 3\}\}$$

En Python, es posible construir el conjunto $A$ mediante comprensiones:

**Program 6** Comprensiones

```
>>> {2*x for x in {1,2,3}}
set([2, 4, 6])
>>> {2*x for x in [1,2,3]}
set([2, 4, 6])
>>> {2*x for x in (1,2,3)}
set([2, 4, 6])
>>> [2*x for x in {1,2,3}]
[2, 4, 6]
>>> [2*x for x in (1,2,3)]
[2, 4, 6]
>>> [2*x for x in [1,2,3]]
[2, 4, 6]
```

De la misma forma, el producto cartesiano entre dos conjuntos definido como:

$$A \times B = \{(a, b) : a \in A \text{ y } b \in B\}$$

Puede ser evaluado como:

**Program 7** Producto cartesiano

```
>>> A={1,2,3}
>>> B={'a','b','c'}
>>> {(a,b) for a in A for b in B}
set([(1, 'c'), (3, 'c'), (1, 'b'), (3, 'b'), (3, 'a'),
 (2, 'a'), (2, 'b'), (2, 'c'), (1, 'a')])
```

Aparte de usar operaciones (concatenación,unión,intersección, etc) sobre las estructuras de datos donde se itera, es posible filtrar los resultados de acuerdo a un criterio:

Tambien es hacer comprensiones sobre diccionarios:

**Program 8** Condiciones

```
>>> S={-4,-2,1,2,5,0}
>>> {x*x for x in S | {5,7}}
set([0, 1, 4, 16, 49, 25])
>>> {x*x for x in S | {5,7} if x>0}
set([1, 4, 49, 25])
```

**Program 9** Condiciones

```
>>> [2*x for x in {4:'a',3:'b'}.keys()]
[6, 8]
>>> [x for x in {4:'a',3:'b'}.values()]
['b', 'a']
```

**Ejercicio 2**

*2.1 Dado un conjunto de enteros (por ejemplo* $S = \{-4, -2, 1, 2, 5, 0\}$*), escriba una comprensión que devuelva una tripleta* $(i, j, k)$ *tal que la suma de los tres valores sea cero.*
*2.2 Escriba una comprensión que devuelva los enteros impares mayores o iguales a cero (tip: usar relación de congruencia).*
*2.3 Suponga que tenemos un diccionario d que mapea un ID de empleado hacia su salario. Si tenemos una lista ordenada de acuerdo al ID con los nombres, escriba una comprensión que crea un nuevo diccionario que reemplaza el ID por el nombre. Por ejemplo:*

$$d = \{0 : 1000.0, 3 : 990, 1 : 1200.0\}$$
$$names = [\text{'Larry'}, \text{' Curly'}, \text{''}, \text{' Moe'}]$$