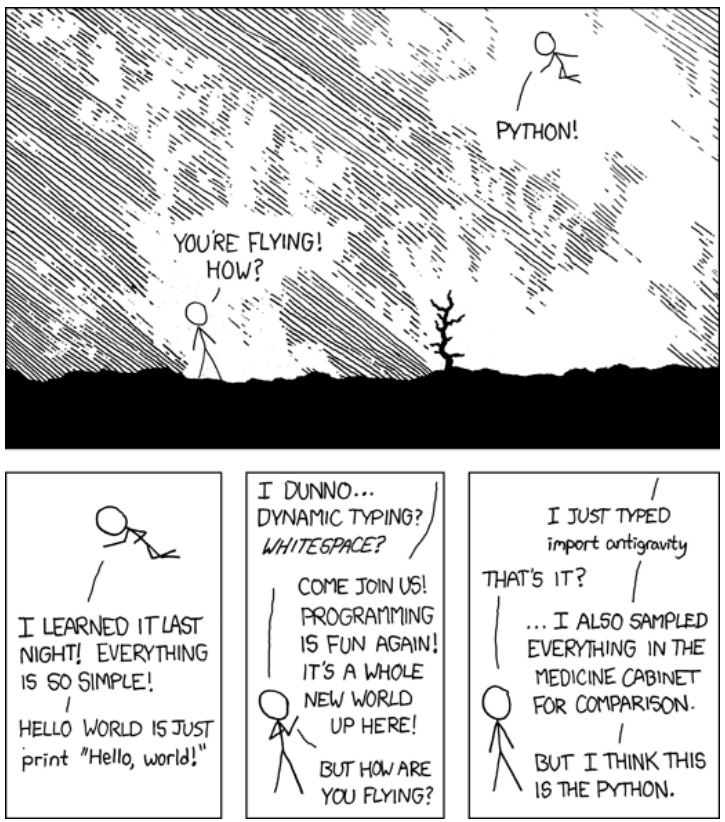
Métodos Numéricos (Quarantine Edition)

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

24 de Abril de 2020

Menú del día

- Introducción a Python
- Introducción a Numpy
- Introducción a herramientas de experimentación



Introducción

- Usaremos el lenguaje `Python` (específicamente `Python3`) para realizar los talleres
- Nos enfocaremos en la librería `Numpy` y algunas otras útiles para experimentación
- Ambas pueden ser muy útiles en la experimentación
- `Python` es un lenguaje interpretado que no requiere compilación como `C++`
- No entraremos en detalle técnicos sobre el lenguaje

Ciclos

Para escribir ciclos tenemos varias opciones:

```
for variable in iterable:
    # iteration
```

```
# Ejemplos
for n in range(2):
    print(n)
# 0
# 1
```

```
suma = 0
for n in range(10, 0, -1):
    suma = suma + n
print(suma) # 55
```

```
while condition:
    # iteration
```

```
# Ejemplos
n = 0
while n < 2:
    print(n)
    n = n + 1
```

```
n = 1
suma = 0
while (n < 11):
    suma = suma + n
    n = n + 1
```

Listas

Para escribir listas hacemos:

```
lista = [] # Lista vacia
lista = [elem1, elem2, ...]
# Pueden ser de distinto tipo

b = [1,2,3]
# Operaciones
if (len(b) > 0) : print(b.pop(0)) # 1
if b == [1,2,3]: print('z') # No hace nada
b = b + [4,4]
b.append(0.2)
print(b[-1]) # 0.2
del b[0]
print(b) # [3,4,4,0.2]

# Sublistas
print(b[1:3]) # [4,4,0.2]
print(b[:2]) # [3,4]

# Listas definidas por comprension
c = [2*e for e in b if e > 1]
print(c) # [6,8,8]
```

Funciones

Para definir funciones hacemos:

```
def fun(param1, param2, ...):
    # code
    return # result

# Ejemplos
def sumaLista(lista):
    suma = 0
    for n in lista: suma = suma + n
    return suma
# sum(lista)

def quitarRepetidos(lista):
    sin_repes = []
    for e in lista:
        if not (e in sin_repes): sin_repes.append(e
    return sin_repes
```

Formas de correr Python

- Python se puede correr de muchas maneras
- Al tener tantas cosas disponibles, es común instalar muchas librerías que pueden generar incompatibilidades
- Veremos virtualenvwrapper y Jupyter

virtualenvwrapper

- Si vamos a trabajar en un proyecto, como los tps de métodos, la idea es no instalar librerías a nivel sistema operativo sino hacerlo en un ambiente más controlado.
- virtualenvwrapper es una herramienta que nos deja crear un ambiente en donde las librerías que instalemos nos quedarán ahí dentro.
- Se instala corriendo *sudo apt install virtualenvwrapper*.
- Se crea un nuevo ambiente corriendo *mkvirtualenv nombreambiente*.
- Para acceder al ambiente se corre *workon nombreambiente*.
- Para salir del ambiente se corre *deactivate*.

Matrices y vectores

Vamos a usar la librería `Numpy` para manejar matrices y vectores. También podemos definir arreglos multidimensionales (no los recomendamos para hacer operaciones de Álgebra Lineal)

```
import numpy as np
import numpy.linalg as lng

# Distintas maneras de inicializar una matriz
A = np.array([[1,2],[3,4]])
B = np.array(np.mat('9 10; 11 12')) # Subclase
C = np.array([[5,6],[7,8]], dtype=float)

# Para los vectores usamos matrices por columnas
v = np.array([[11],[12]])
w = np.array(np.mat('17; 18')) # Subclase
x = np.array([21,22,23,24], ndmin=4)
```

Matrices particulares y operaciones

```
# Matrices especiales
I = np.eye(3) # Identidad de 3x3
D = np.diag([77,90]) # Matriz diagonal
N = np.zeros((3,4)) # Matriz nula de 3x3

# Operaciones basicas entre matrices y vectores
A + B                 # Suma
A - B                 # Resta
A @ B                 # Producto de matrices
A @ v                 # Producto de matriz por vector
3.2 * A               # Producto por escalar
A ** 2                # Potencia
A.T                   # Traspuesta (transpose())
lng.inv(A)            # Inversa
```

Ejercicios

1. Dados $x_1, \ldots, x_n$ una muestra de una variable aleatoria, implementar rutinas que calculen la media y la varianza utilizando operaciones vectoriales

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

2. Sea $A \in \mathbb{R}^{m \times n}$
   1. Demostrar que $A^t A$ y AA^t son simétricas (implementativamente)
   2. Implementar una rutina que dada una matriz cuadrada verifique si la misma es simétrica

Ejercicios (cont.)

3. Analizar la función implementada en el ítem anterior con la matriz B generada de la siguiente forma:

```
>> from numpy import *
>> A = random.rand(4)
>> B = A.transpose()*A*0.1/0.1
```

   En base al resultado, revisar la implementación y (de ser necesario) reimplementar la función

4. Sean $A, B \in \mathbb{R}^{n \times n}$, con n par y B triangular inferior:
   1. Realizar la multiplicación AB por bloques, partiendo ambas matrices en bloques de tamaño $n/2$
   2. Implementar una rutina que realice la multiplicación por bloques, evitando cuentas innecesarias

Sistemas de ecuaciones

La librería `Numpy` también nos permite realizar operaciones sobre sistemas de ecuaciones lineales

```
import numpy as np
import numpy.linalg as lng

# Inicializaciones
A = np.array([[1,2],[3,4]], float) # matriz 2x2
B = np.array([[5,6],[7,8]], float) # matriz 2x2
b = np.array([[1],[2]], float) # vector columna en l

k = 1
# Distintas partes de una matriz
A[0,:]      # primera fila de A (indexa desde cero)
A[:,0]      # primera columna de A
A[0:k,0:k] # k-esima submatriz principal de A
np.triu(A)     # parte triangular superior de A
np.tril(A)     # parte triangular inferior de A
```

Sistemas de ecuaciones (cont.)

```
# Resolucion de sistemas y determinantes
lng.solve(A,b) # solucion del sistema Ax = b
lng.solve(A,B) # matriz solucion del sistema AX = B
lng.det(A)     # determinante de la matriz A

# Normas vectoriales
c = np.array([1,2,3,4], float) # usamos array
lng.norm(c,2)   # norma 2
p = 3
lng.norm(c,p)   # norma p, con p entero
lng.norm(c,np.inf) # norma infinito

# Numero de condicion segun la norma matricial
lng.cond(A,2)
lng.cond(A,np.inf)
lng.cond(A,'fro')
```

Ejercicios I

1. Describir e implementar un algoritmo que calcule un vector no nulo $z \in \mathbb{R}^n$ tal que $Uz = 0$, donde $U \in \mathbb{R}^{n \times n}$ sea una matriz triangular superior con $u_{n,n} = 0$ y $u_{1,1} \dots u_{n-1,n-1} \neq 0$
2. Consideremos una familia de matrices $A_n \in \mathbb{R}^{n \times n}$, $n \geq 2$ con una estructura particular que depende de n. Para el caso $n = 5$, la matriz en cuestión es la siguiente:

$$A_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

    1. Analizar qué sucede al aplicar el método de eliminación gaussiana con pivoteo parcial a A_6. Generalizar el resultado para n genérico

Ejercicios II

2. Implementar un algoritmo que resuelva el sistema de ecuaciones $A_n x = b$
3. Variando el n, considerar vectores $b \in \mathbb{R}^n$ para los cuales la solución al sistema $A_n x = b$, sea conocida. Llamemos x^* a la solución exacta del sistema y $\bar{x}$ a la solución obtenida por el algoritmo del punto anterior. ¿Es $\bar{x}$ una buena aproximación?
4. Graficar cómo evoluciona el $\|x^* - \bar{x}\|_2$ en función del tamaño de la matriz

Experimentación

- Python es simple para realizar experimentación y manipular datos.
- Jupyter sirve para experimentar gradualmente e ir viendo resultados parciales.
- *Pandas* es una librería que sirve para manejar datos y realizar análisis.
- *Matplotlib* y *Seaborn* son librerías que nos sirven para graficar resultados.
- *Subprocess* es una librería que nos sirve para correr código externo.