ANÁLISIS NUMÉRICO I / ANÁLISIS NUMÉRICO – 2020
Guía básica de `Python`

Recomendaciones

- No usar espacios en los nombres de los archivos o carpetas

Comenzar a trabajar con Python

- Abrir una terminal: `Crtl+Alt+T` o Aplicaciones....
- Ir al directorio donde están guardados los archivos: ∼`$ cd Documentos/Numerico_I`
- Abrir un editor de texto: ∼`$ gedit &`
- Abrir la terminal de `Python`: ∼`$ ipython` o ∼`$ python`
- Escribir una lista de números v:
  `In[n]:#>v = [1, 2.56, -3.43, 0]`.
- Conseguir un número aleatorio entre 0 y 1:
  `In[n]:#>import random`
  `In[n+1]:#>c = random.random()`.

Condicionales y bucles

- La estructura `if ...`
  La **estructura del if** simple es la siguiente:

```
if condicion:
  Acciones a realizar si es cierta la condicion
else:
  Acciones a realizar si es falsa la condicion
```

La indentación (tabs) marca el final de la estructura. Hagamos una función que verifica si un número n es divisible por m y devuelve True si lo es, False en caso contrario.

```
def multiplo(n,m):

    if n \% m == 0:
        print('Es divisible')
        return True
    else:
        print('No es divisible')
return False
```

(Nota: Llegar a un `return` desemboca en el final de la función.)

Operaciones lógicas

`<=`	menor o igual a	`<`	menor que
`>`	mayor que	`>=`	mayor o igual a
`==`	igual a	`!=`	distinto

- El bucle (loop) `for ...`

  En el bucle `for ...`, la ejecución de uno o varios comandos se repite un número fijo y predeterminado de veces.

```
for variable_contador in range(INICIO, FIN):
   Acciones_bucle_externo
   for variable_otro_contador in range(OTRO_INICIO, OTRO_FIN):
       Acciones_bucle_interno
```

  Por ejemplo, si se quiere imprimir los valores de una matriz $A \in \mathbb{R}^{m\times n}$, triangular superior, tal que $A_{ij} = 1/(i+j)$ para $i \leq j$, se debe hacer

```
def print_vander(m,n):

    for idx in range(0, m):
        for jdx in range(idx, n):
            print("A({},{}) = {}".format(idx, jdx, 1/(idx+jdx)))
```

- El bucle `while ...`

  Este bucle se usa cuando no se conoce el número de veces que debe repetirse la ejecución de cierto comando. Supongamos que queremos sumar números aleatorios hasta superar una tolerancia `Tol` y que, al finalizar, queremos saber cuántas veces hemos iterado. Se debe hacer:

```
import random

def suma_aleatorio(Tol):

    s = 0
    contador = 0

    while s <= Tol:
        s = s + random.random()
        contador = contador + 1

    return s, contador
```