

Programación Técnica y Científica

Grado en Ingeniería Informática

Tema 1

Introducción a la programación en Python

Esquema

- Introducción a la PTC
- Cuestiones básicas de Python
- Funciones
- Tipos de datos básicos
- Estructuras de control
- Tipos de datos más complejos
- Ficheros

© Prof.Miguel García Silvente

3

Programación técnica y científica

- La programación científica está relacionada con la **construcción de modelos matemáticos** y **técnicas de análisis cuantitativo** usando ordenadores para **analizar y resolver** problemas **científicos y de ingeniería**.
- Es habitual la aplicación de **simulaciones** por ordenador y otras formas de computación a partir de **análisis numéricos** e **informática teórica** para resolver problemas de distintas disciplinas científicas y técnicas.

© Prof.Miguel García Silvente

4

Aplicaciones

- Simulaciones numéricas:
 - reconstrucción y comprensión de eventos (terremotos, tsunamis, etc),
 - Predicciones de tiempo o del comportamiento de partículas subatómicas.
- Ajustes de modelos y análisis de datos:
 - Ajustar modelos o resolver ecuaciones (exploración de pozos petrolíferos),
 - Teoría de grafos para modelar redes.
- Optimización computacional.

© Prof.Miguel García Silvente

5

Ámbitos (I)

- Ingeniería aeroespacial e ingeniería mecánica.
- Biología y medicina.
- Química.
- Ingeniería civil.
- Ingeniería informática, ingeniería eléctrica e ingeniería en telecomunicaciones.
- Ingeniería industrial.
- Ciencia de los materiales.

© Prof.Miguel García Silvente

6

Ámbitos (II)

- Ingeniería nuclear: modelado de explosiones nucleares, simulación del proceso de fusión.
- Ingeniería petrolífera: modelado de reservas petrolíferas, exploración de petróleo y gas.
- Predicción meteorológica, investigación climática, computación geofísica (datos sísmicos).
- Simulación de campos de batallas y juegos militares.
- Sistemas astrofísicos.

© Prof.Miguel García Silvente

7

Cuestiones a considerar

- Algunos aspectos son muy importantes, como la precisión:

¿Cual es el resultado?

$$0.3 - (0.1 + 0.1 + 0.1)$$

$$-5.551115123125783e-17$$

- Generación de números aleatorios.
- Aproximación de funciones.
- Enteros de gran tamaño

© Prof.Miguel García Silvente

8

Métodos y algoritmos

- Análisis y visualización de datos.
- Modelado y simulación.
- Análisis numérico (aproximación numérica).
- Series de Taylor.
- Cálculo de derivadas (diferenciación automática, diferencias finitas).
- Métodos de integración.
- Resolución de ecuaciones diferenciales.
- Métodos probabilísticos.
- etc

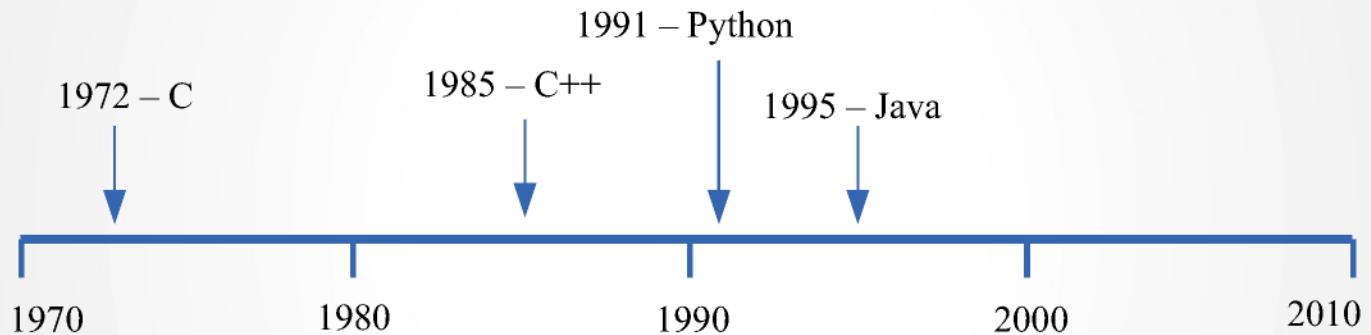
© Prof.Miguel García Silvente

9

Lenguajes para Programación Científica

- Python
- Matlab, Mathematica, SciLab, Octave
- R (computación estadística y gráfica)

História de los Lenguajes de Programación



© Prof.Miguel García Silvente

11

Lenguajes Script

- Un script es un programa de alto nivel que suele ser corto y que se guarda habitualmente en un fichero de texto.
- Python es un lenguaje de tipo script.
- Otros lenguajes de tipo script: Unix shells, Tcl, Perl, Python, Ruby, Scheme, Rexx, JavaScript, VisualBasic, ...

© Prof.Miguel García Silvente

12

Ventajas de los lenguajes Script

- Es interpretado, no es necesario compilar (¿también es una desventaja?).
- Desarrollo del software más rápido.
- No hay que declarar las variables (¿es una ventaja?).
- Gran cantidad de bibliotecas y herramientas.

© Prof.Miguel García Silvente

13

Programas más cortos

Supongamos que queremos leer los siguientes datos (de la entrada estandar):

1.1 9 5.2
1.762543E-02
0 0.01 0.001
9 3 7

El código en python sería:

```
f = open(filename, 'r')  
n = f.read().split()
```

© Prof.Miguel García Silvente

14

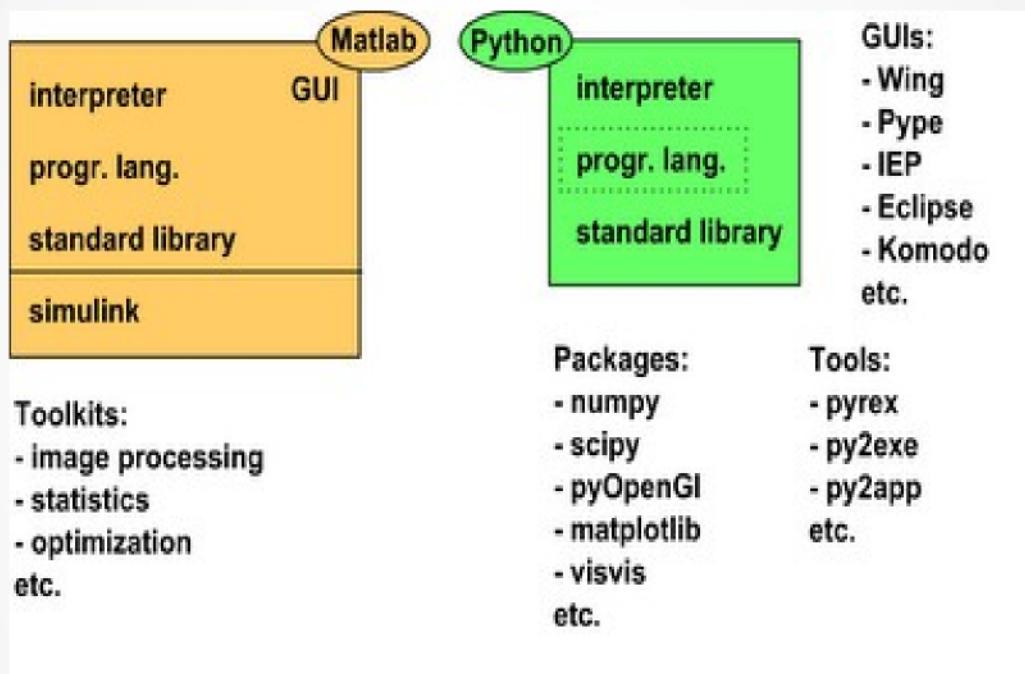
Algunas características de Python

- Multiplataforma.
- Manejo de matrices: **numpy**
- Representación de datos: **matplotlib**
- Funciones de cálculo científico: **scipy**
- Cálculo simbólico: **sympy**
- Evaluar expresiones: **eval.**
- Módulos para crear interfaces de usuario: **tkinter**

© Prof.Miguel García Silvente

15

Matlab vs Python (I)



<https://sites.google.com/site/pythonforscientists/python-vs-matlab>

<https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html>

© Prof.Miguel García Silvente

16

Matlab:

- Código propietario.
- Caro.
- Portabilidad más difícil de realizar.
- Espacios de nombres.

Código Matlab vs Python (I)

Python:

- No es necesario usar **end**
- No se usa () para las matrices, sino []
- Código un 10-20% más corto
- Los índices empiezan en 0
- Hay menos restricciones, por ej., se puede hacer:

>>> [1,2,4,7,11,14,16,17][n]

Código Matlab vs Python (II)

Python:

- Incluye el tipo de dato diccionario.
- POO más clara que en Matlab (y que en C++).
- Se puede combinar “main” con otras funciones en un mismo fichero.
- Organización del código.
- Más opciones para programar GUIs
http://phillipmfeldman.org/Python/Advantages_of_Python_Over_Matlab.html
<https://web.archive.org/web/20190403064043/https://www.stat.washington.edu/~hoytak/blog/whypython.html>

Código Matlab vs Python (III)

Python usa “propagación automática”

Dados A (20x1x15), B (20x12x1), C (1x12x15):

- En matlab
$$D = \text{bsxfun}(@times, \text{bsxfun}(@minus, B, C), A);$$
- En Python/NumPy
$$D = A * (B - C)$$

- Historia
- Instalación y ejecución
- Tipos de datos

Breve historia de python

- Fue creado en Holanda, en el año 1991, por Guido van Rossum.
- Su nombre se debe a los Monty Python.
- Open source desde el inicio.
- Considerado un lenguaje script.
- Escalable, orientado a objetos y funcional desde su diseño inicial.
- Mayoritariamente usado por muchas empresas: youtube, Google, NASA, etc.
- Implementado en C.

The screenshot shows the Python 3.9.7 Documentation page in Spanish. The left sidebar contains links for 'Descarga' (Downloads), 'Documentos por versión' (Documents by version) including links for Python 3.11 (in development) through 2.7, 'Otros recursos' (Other resources) like the PEP index and developer guide, and 'Índices y tablas' (Indices and tables). The main content area features a welcome message: 'Welcome! This is the official documentation for Python 3.9.7.' It includes sections for 'Áreas de la documentación:' (Areas of the documentation) such as '¿Qué hay de nuevo en Python 3.9?' (What's new in Python 3.9?), 'Tutorial' (Tutorial), 'Referencia de la Biblioteca' (Library Reference), 'Referencia del lenguaje' (Language Reference), 'Configuración y uso de Python' (Python Configuration and Usage), 'Códigos (HOWTOs) de Python' (Python HOWTOs), 'Instalación de módulos de Python' (Module Installation), 'Distribuir módulos de Python' (Distributing Python Modules), 'Extender e incrustar' (Extending and Embedding), 'Python/C API' (Python/C API), and 'Preguntas frecuentes' (FAQ). Below these are sections for 'Índice Global de Módulos' (Global Module Index), 'Página de búsqueda' (Search Page), 'Índice General' (General Index), 'Índice de contenidos completo' (Full Contents Index), and 'Glosario' (Glossary). A 'Meta información:' (Metadata) section at the bottom includes links for reporting errors, contributing to the documentation, and viewing the history and license.

© Prof.Miguel García Silvente

23

Tutorial de Python

The screenshot shows the Python Tutorial page in Spanish. The left sidebar provides navigation links for 'Tema anterior' (Previous topic), 'Próximo tema' (Next topic), and 'Esta página' (This page). The main content area is titled 'Tutorial de Python'. It begins with a paragraph about Python's strengths and its use as a scripting and development language. It then describes the Python interpreter and its extensibility. The tutorial aims to introduce basic concepts and practical usage. It mentions the standard library, C/C++ extensions, and other resources like the glossary. A note states that the tutorial is informal and focuses on common uses. The sidebar also includes links for reporting bugs and viewing the code.

24

Incompatibilidad entre versiones

- Las versiones 2 y 3 son incompatibles en muchos aspectos.
- Por ejemplo:
 - v3:

```
>>> 2/3
```

0.6666666666666666

```
>>> print x # Error, hay que usar ()
```
 - v2:

```
>>> 2/3
```

0

© Prof.Miguel García Silvente

25

Ejecución interactiva

>>> es el prompt de Python e indica que Python está preparado para que le demos una orden.

```
>>> "Hello, world"  
Hello, world  
>>> print("Hello, world")  
Hello, world  
>>> 2+3  
5  
>>> print(2+3)  
5  
>>> print("2+3=", 2+3)  
2+3= 5  
>>>
```

© Prof.Miguel García Silvente

26

Ejecución interactiva en UNIX

```
% python  
>>> 3+3  
6  
% python3
```

Para salir:

- En Unix, escribe CONTROL-D
- En Windows, escribe CONTROL-Z + <Enter>
- También con exit()

© Prof.Miguel García Silvente

27

Instalación

- Python está preinstalado en muchos sistemas unix, incluyendo Linux y Mac OS X
- Las versiones preinstaladas pueden no ser las más recientes (2.6 es de Nov 2008)
- Descargar de <http://python.org/download/> Python 3.9.7
- Distribuciones windows: Python(x,y), Winpython, Anaconda
- Python incluye una biblioteca con módulos estandar.
- IDEs:
 - Ninja IDE, Spyder
 - Eclipse with Pydev (<http://pydev.sourceforge.net/>)

© Prof.Miguel García Silvente

28

Cuestiones básicas (I)

- No existe un **main()** o función principal.
- Se puede solicitar ayuda con **help(x)**
- Los bloques se inician con : y se delimitan usando la indentación.
- Es sensible a mayúsculas y minúsculas.
- Gestión de memoria: usa un recolector de basura (basado en referencias).

Cuestiones básicas (II)

- Se ejecuta un archivo desde el inicio hasta el fin, de forma consecutiva.
- No es necesario usar ; al final de cada sentencia.
- Las variables no se declaran. Se crean al asignarle un valor.
- El tipo de dato es dinámico y se puede conocer con **type()**

Cuestiones básicas (III)

- La extensión para los archivos suele ser .py
- Las extensiones .pyc .pyo son versiones “compiladas” (en bytecode)
- Lo que va después de # se ignora, son comentarios
- Los módulos se incorporan con **import**
- Posee una gran cantidad de módulos.
- Existe una guía de estilo **PEP-8**:
<https://www.python.org/dev/peps/pep-0008/>

© Prof.Miguel García Silvente

31

Uso como calculadora

- Se usan directamente las operaciones aritméticas:
`>>> 3 + 2`
`5`
- Se puede reutilizar el último resultado:
`>>> 3 + 2`
`5`
`>>> 7 * _`
`35`
- **import math** para incorporar funciones matemáticas

© Prof.Miguel García Silvente

32

Ejecución de programas en UNIX

- Ejecutar el script con el intérprete
% **python fact.py**
- Convertir el fichero en ejecutable:
 - Añadiendo el path de python como la primera línea del archivo

```
#!/usr/bin/python3
```

 - Darle permisos de ejecución

```
% chmod a+x fact.py
```

 - Invocar el fichero desde la línea de órdenes

```
% ./fact.py
```

© Prof.Miguel García Silvente

33

Eficiencia

- Python es un lenguaje interpretado:
 - Más lento que un lenguaje compilado como C/C++.
 - Más flexible en el diseño y la codificación.
- Las bibliotecas están escritas en C/C++.
- Tiene módulos para medir la eficiencia.
 - Profile.
 - Cprofile.
- Es posible mejorar la eficiencia, por ej. usando Cython.

© Prof.Miguel García Silvente

34

Asignación (I)

- Se asocia un nombre a una referencia a un objeto
 - *Las asignaciones crean referencias, no copias.*
- Una variable se crea la primera vez que aparece en la parte izquierda de una asignación:

```
>>> x = 5          >>> x = 7  
>>> id(x)        >>> y  
505910928        ???  
>>> y = x  
>>> id(y)  
505910928
```

© Prof.Miguel García Silvente

35

Asignación (II)

Asignar valores a más de una variable:

```
>>> x, y = 2, 3
```

```
>>> x
```

```
2
```

```
>>> y
```

```
3
```

Facilita el intercambio de valores entre variables:

```
>>> x, y = y, x
```

Las asignaciones devuelven valores

```
>>> a = b = x = 2
```

© Prof.Miguel García Silvente

36

Las variables deben inicializarse

Si no se le asigna un valor, se genera un error al intentar usarla

```
>>> y
```

Traceback (most recent call last):

File "<pyshell#16>", line 1, in <toplevel->

y

NameError: name 'y' is not defined

```
>>> y = 3
```

```
>>> y
```

```
3
```

Nombres de variables

- Son sensibles a mayúsculas y minúsculas y no pueden empezar por un número. Pueden contener letras, números y el carácter de subrayado.
bob Bob _bob _2_bob_ bob_2 BoB
- Palabras reservadas:
and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while

Convenciones

La comunidad Python recomienda:

- Para funciones, métodos y atributos usar letras **minúsculas** y usar el carácter **subrayado_**
- Usar **sólo mayúsculas** para las constantes.
- **Primera letra en mayúscula** para las clases
- Atributos: interface, _internal, __private

Funciones (I)

- Se usa la palabra reservada **def**
- La definición de una función es muy simple:

```
def saludo(nombre):
    print("Hola", nombre)
    print("¿Cómo estás?")
```
- Se podría usar con datos de distinto tipo:
`saludo("Pepe")`
`saludo(3452)`

Funciones (II)

- Puede devolver algo usando `return`. Si no hay `return`, la función devuelve `None`
- Los parámetros se pasan como referencias por valor.
- No es necesario especificar tipos:
`def sumar(x, n) :`
 `return x+n`
- Se pueden devolver varios valores al mismo tiempo:
`def cociente_resto(a, b) :`
 `return a/b, a%b`

`x, y = cociente_resto(12, 5)` © Prof.Miguel García Silvente

41

Funciones: documentación

Se documenta usando “””

`def fact(n):`

“””fact(n) asume que n es un entero positivo y devuelve el factorial de n.“””

...

Se puede consultar la documentación con

`fact.__doc__`

'fact(n) asume que n es un entero positivo y devuelve el factorial de n.'

Funciones: swap

- Correcta

```
def swap(a, b) :  
    return b, a
```

- Incorrecta

```
def swap_incorrecto(a, b) :  
    temp = a  
    a = b  
    b = temp
```

Funciones: alias y parámetros por defecto

- Se pueden usar alias (variables tipo función)

```
f = swap  
a, b = f(a,b)
```

- Parámetros con valores por defecto

```
def func(a, b = 10) :  
    return a + b
```

```
print(func(6))
```

- Uso de nombres de parámetros

```
func(b=12, a= 10)  
func(10,12)
```

Funciones lambda

- Una forma abreviada
- <función> = lambda <params> : <valores devueltos>

```
>>> suma = lambda x, y : x+y  
>>> suma(10, 15)  
>>> f = lambda : 1
```

Tipos de datos básicos

- Entero: int/long
- Lógico: bool
- Real: float
- Complejo: complex
- Cadena de caracteres: str

Tipos de datos: int

- Precisión ilimitada en decimal, octal (`0o`) y hexadecimal (`0x`)
- Operaciones: `= + - / // * % ** divmod abs int`
- Operaciones bit a bit: `| & ^ ~ << >>` `bin x.bit_length`
- `long` se ha eliminado en la versión 3 de python

```
>>> x = 34
```

```
>>> x
```

```
34
```

```
>>> x / 5
```

```
6.8
```

© Prof.Miguel García Silvente

47

Tipos de datos: bool

- Valores lógicos: `True False`
- `False` se asocia al 0 y `True` a los demás valores.
- Operadores: `= and or not`

```
>>> x = True
```

```
>>> x
```

```
True
```

```
>>> x = true
```

Traceback (most recent call last):

 File "<stdin>", line 1, in <module>

NameError: name 'true' is not defined

© Prof.Miguel García Silvente

48

Tipos de datos: float (I)

- Se representan en base 2. Se pierde precisión si su representación no es exacta en base 2, por ejemplo, 0.1
- ```
>>> 0.1 + 0.1 + 0.1 == 0.3
```
- Operaciones: `= + - / // * ** divmod abs, int`
  - Operaciones bit a bit: `| & ^ ~ << >>` float bin  
`x.bit_length()`
  - `abs(0.1 + 0.1 + 0.1 -0.3) < 1e-10`
  - `x = float("123")`

# Tipos de datos: float (II)

- Las funciones `exp, log, sin, cos`, etc están en el módulo `math`
- Para usarlas hay que importar el módulo `math`.

```
import math
```

```
x = math.exp(12)
```

- O también:

```
from math import exp
```

```
x = exp(12)
```

- Y la forma menos aconsejable:

```
from math import *
```

```
x = exp(12)
```

## Tipos de datos: complex

- La parte imaginaria se indica con j  
`>>> x = 3 + 2j`
- Operaciones: `= + - / * % ** abs imag real`  
`//` (python2)  
`>>> x.conjugate()`  
`(3-2j)`

## Tipos de datos: str (l)

- Son tipos **inmutables**.
- Se pueden usar ' o "
- Operadores: [] +
- Se usa """ cuando la cadena ocupa varias líneas.  
`x = """cadena`  
`multilínea"""`

## Tipos de datos: str (II)

- Funciones: len, lower, lstrip, replace, split, swapcase, upper, count, find, join, partition, str (o repr)
- La barra \ hay que “escaparla” con otra \  
cad = “\\windows”
- Una forma de no tener que escaparla  
cad = r”\windows”

© Prof.Miguel García Silvente

53

## Tipos de datos: str (III)

### Operador []

```
>>> cad = "Hola mundo"
```

```
>>> cad [0:3]
```

'Hol'

```
>>> cad[3:-1]
```

'a mund'

```
>>> cad[-1:0:-1]
```

'odnum alo'

© Prof.Miguel García Silvente

54

## Tipos de datos: str (IV)

### Operador []

```
>>> cad[:]
```

'Hola mundo'

```
>>> cad[::-1]
```

'odnum aloH'

## Tipos de datos: str (V)

### Conversión de tipo

```
>>> cad = str(1/5)
```

```
>>> cad = "El resultado de dividir " + str(x) + "
entre " + str(y) + " es " + str(x/y)
```