



IISTA

Instituto Interuniversitario de Investigación
del Sistema Tierra en Andalucía

INTRODUCCIÓN PRÁCTICA A PYTHON

Seminario interno

21-22 de Enero 2016

Pedro J. Magaña Redondo

pmagana@ugr.es

Manuel Cobos Budia

mcobosb@ugr.es

Información de ayuda

- Librería
- Instalar paquetes adicionales
- Sesiones en ipy de Robert Johansson (algunas traducidas por Pedro Rodelas)
- Ejemplos numpy, sympy, scipy y matplotlib
- Presentaciones

Disponibles en <http://gdfa.ugr.es/python>

Bibliografía

- Think Python. How to Think Like a Computer Scientist
- Python para todos
- Scientific Computing with Python
- Guía de estilo para el código Python – PEP 8 en Español
- Instalación de Python en Windows
- <https://github.com/jrjohansson/scientific-python-lectures>
- Python para usuarios de Matlab y R
- Pandas , powerful Python data analysis toolkit
- Rapid GUI programming with Python and Qt. The definitive guide t
- <http://matplotlib.org/>
- <http://www.scipy.org/>
- <http://www.numpy.org/>
- <http://pandas.pydata.org/>
- <https://docs.python.org/2/library/logging.html>

Python

¿Por qué usar Python?

Entornos científicos

MATLAB

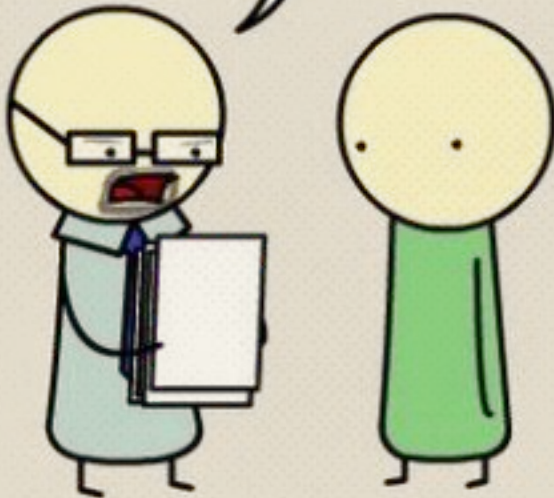
- No es de propósito general
- Grandes problemas para generar ejecutables
- Uso de recursos
- Completamente cerrado
- De pago

R

- No es de propósito general
- Limitaciones para generar ejecutables

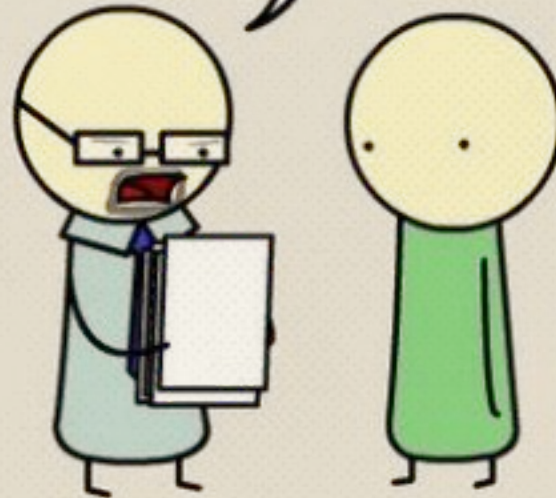
PYTHON

THIS IS PLAGIARISM.
YOU CAN'T JUST "IMPORT ESSAY."



JAVA

I'M TWO PAGES IN AND I STILL
HAVE NO IDEA WHAT YOU'RE SAYING.



Comparativa completa: <http://i.imgur.com/ZyeCO.jpg>

iHola mundo!

JAVA

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hellold!");
    }
}
```

PYTHON

```
print "Hello, world!"
```

Complex object sort

RC4 encryption



Swift

220x

Objective-C

127x

Python

1x

TOP ACTIVE LANGUAGES

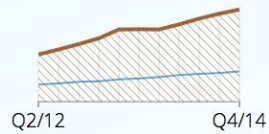
A split by language view of active repositories

%

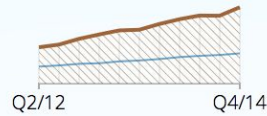
GitHub Average
REPOS

100k
1k
Q2/12 Q4/14

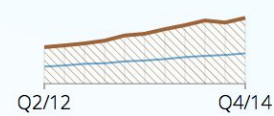
1. JavaScript



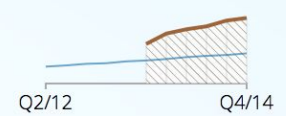
2. Java



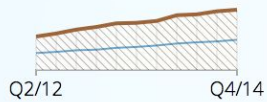
3. Python



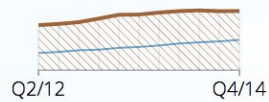
4. CSS



5. PHP



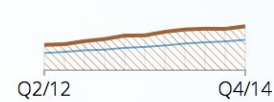
6. Ruby



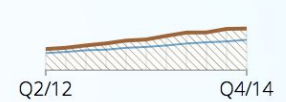
7. C++



8. C



9. Shell



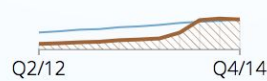
10. C#



11. Objective-C



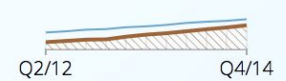
12. R



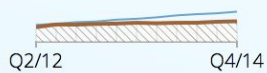
13. VimL



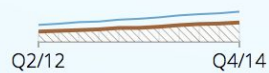
14. Go



15. Perl



16. CoffeeScript



17. TeX



18. Swift



19. Scala



20. Emacs Lisp

21. Haskell

22. Lua

23. Clojure

24. Matlab

TOOLBOX

PICK UP PYTHON

A powerful programming language with huge community support.

ILLUSTRATION BY THE PROJECT TWINS



BY JEFFREY M. PERKEL

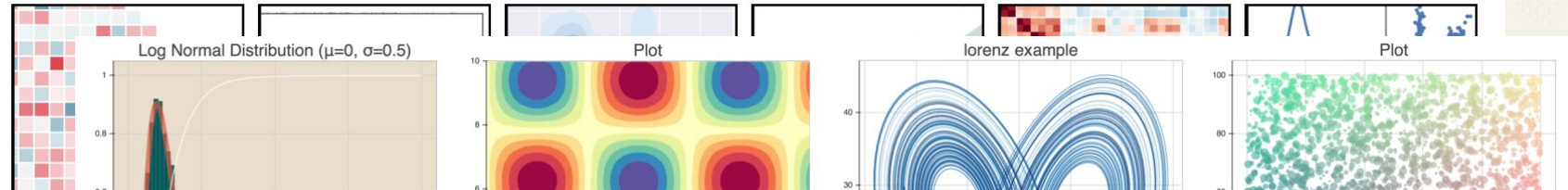
Last month, Adina Howe took up a post at Iowa State University in Ames. Officially, she is an assistant professor of agricultural and biosystems engineering. But she works not in the greenhouse, but in front of a keyboard. Howe is a programmer, and a lea-

Brown specializes in bioinformatics and uses computation to extract meaning from genomic data sets, and Howe had to get up to speed on the computational side. Brown's recommendation: learn Python.

Among the host of computer-programming languages that scientists might choose to pick up, Python, first released in 1991 by Dutch pro-

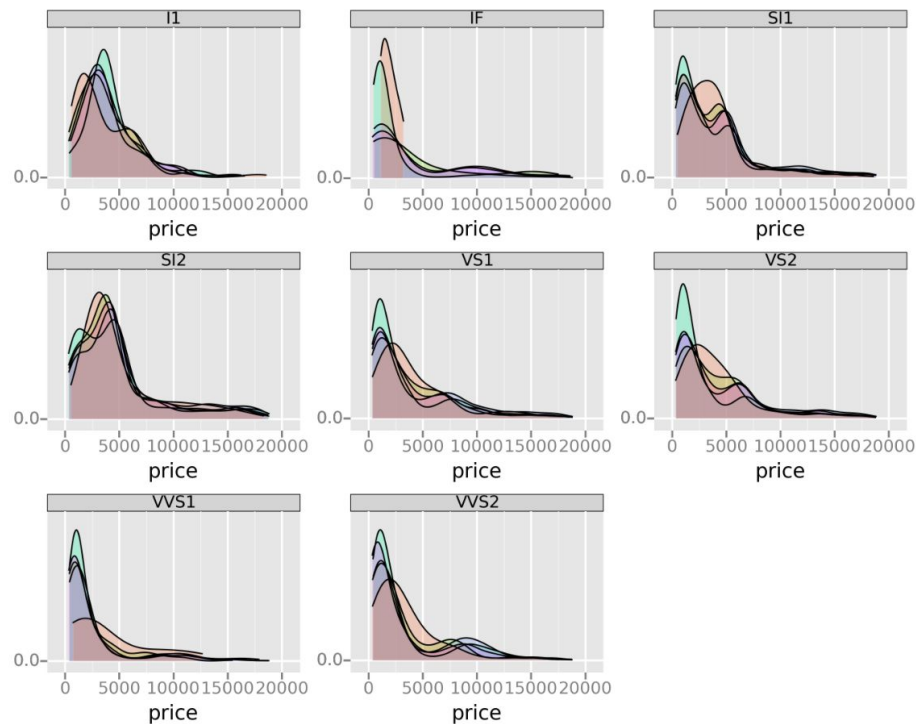
is becoming ever more crucial. Researchers who can write code in Python can deftly manage their data sets, and work much more efficiently on a whole host of research-related tasks — from crunching numbers to cleaning up, analysing and visualizing data. Whereas some programming languages, such as MATLAB and R, focus on mathematical and statis-

Nature 518, 125–126 05
February 2015 doi:
10.1038/518125a



ggplot is powerful

```
ggplot(diamonds, aes(x='price', fill='cut')) +\
  geom_density(alpha=0.25) +\
  facet_wrap("clarity")
```



Entorno de desarrollo

¿Por dónde empezar?

Versiones de Python



Intérprete de Python

Compilation

Interpretation

Compiled		Interpreted	
PROS	CONS	PROS	CONS
ready to run	not cross platform	cross-platform	interpreter required
often faster	inflexible	simpler to test	often slower
source code is private	extra step	easier to debug	source code is public

File Edit Source Refactoring Navigate Search Project Run Window Help

Debu

__init__.py - [~/PycharmProjects/Softpediatest]

~/Documents/repos/tutorials/intro-angular-flask/part3/app.py — intro-angular-flask

FOLDERS

- intro-angular-flask
 - part1
 - templates
 - app.py
 - requirements.txt
 - part2
 - part3
 - static
 - templates

app.py

```
1 from flask import Flask, abort, jsonify, render_template
2 import requests
3 import when
4
5 app = Flask(__name__)
6
7
8 @app.route('/')
9 def index():
10     return render_template('index.html')
11
12
13 @app.route("/api/forecast/")
14 @app.route("/api/forecast/<lat>/<lng>/")
15 def forecast(lat=None, lng=None):
16     """ Returns the weather for a specific location"""
17
18     if not lat or not lng:
19         abort
20
21     root = "543c0f05cb394d6ece5a8b5fb3f8002f"
22     key = "https://api.forecast.io/forecast"
23     timestamp = when.format(when.now(), '%Y-%m-%dT%H:%M:%S')
24
25     url = root + key + "/" + lat + "," + lng + ".json"
26
27     r = requests.get(url)
28
29     return jsonify(r.json()) if r.ok else {}
30
31
32 if __name__ == "__main__":
33     app.run(debug=True)
34
```

Terminal — bash — 80x24

```
[ daz Bert ~ ] ll
total 0
drwx----- 10 daz staff 340 2008-01-04 18:43 Desktop/
drwx----- 15 daz staff 510 2008-01-04 18:44 Documents/
drwx----- 6 daz staff 204 2008-01-04 19:47 Downloads/
drwx----- 35 daz staff 1.2K 2008-01-03 20:28 Library/
drwx----- 3 daz staff 102 2008-01-02 19:33 Movies/
drwx----- 5 daz staff 170 2008-01-02 20:41 Music/
drwxr-xr-x 22 daz staff 748 2007-08-14 17:48 Na-Box/
drwx----- 14 daz staff 476 2008-01-03 21:41 Pictures/
drwxr-xr-x 5 daz staff 170 2008-01-02 19:33 Public/
drwxr-xr-x 5 daz staff 170 2008-01-02 19:33 Sites/
[ daz Bert ~ ]
```

Line 11, Column 1

Spaces: 4 Python

Spyder (Python 2.7)

Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Herramientas Ver Ayuda

Explorador de archivos

Name	Size
about_box	
combine_bits	
quit_prog	
show_licence	
truss	
truss3	
tuts4pyside	
wiki-src	
.gitignore	351 byt
LICENSE	201
README.md	996 byt

Editor - C:\Users\m_cob\https\github.com\OldAI\tuts4pyside\truss\truss.py

```
1 #!/usr/bin/env python
2 # truss.py
3
4 # Copyright (c) 2010-2011, 2013 Algis Kabaila. All rights reserved.
5 # This work is made available under the terms of the
6 # Creative Commons Attribution-ShareAlike 3.0 license,
7 # http://creativecommons.org/licenses/by-sa/3.0/.
8
9 import sys
10 import os
11 import platform
12
13 import PySide
14 from PySide.QtCore import QRect, QMetaObject, QObject
15 from PySide.QtGui import (QApplication, QMainWindow, QWidget,
16                             QGridLayout, QTabWidget, QPlainTextEdit,
17                             QMenuBar, QMenu, QStatusBar, QAction,
18                             QIcon, QFileDialog, QMessageBox, QFont)
19
20 import qrc_truss
21 import ncrunch
22
23 __version__ = '0.1.6'
24
25 class Truss(QMainWindow):
26     def __init__(self, parent=None):
27         super(Truss, self).__init__(parent)
```

Inspector de objetos

Uso

En este panel es posible obtener la ayuda de cualquier objeto al oprimir **Ctrl+I** estando al frente del mismo, bien sea en el Editor o en la Terminal.

Esta ayuda también se puede mostrar automáticamente después de escribir un paréntesis junto a un objeto. Este comportamiento puede activarse en *Preferencias > Inspector de objetos*.

Nuevo en Spyder? Lee nuestro [tutorial](#)

Inspector de objetos Explorador de variables

Terminal

Python 1 IP Núcleo 1

To read more about this, see <https://github.com/ipython/ipython/issues/2049>

To connect another client to this kernel, use:
--existing kernel-3696.json

Terminal de IPython

```
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra
details.
%gui -> A brief reference about the graphical user
interface.

In [1]:
```

Historial de comandos Terminal de IPython

Permisos: RW Fin de línea: CRLF Codificación: UTF-8-GUESSED Línea: 1 Columna: 1 Memoria: 52 %

Distribuciones científicas de Python

- Anaconda: multiplataforma
- Canopy
 - No incluye Spyder. Tiene entorno propio
- Python(x,y)
 - Solo para Windows
- Spyder
 - Es difícil instalar paquetes adicionales

Cómo empezar

Primeros pasos

Importación de paquetes

```
import os  
import numpy as np  
from numpy import linspace as linspace  
from sympy import *
```

Este nombre por legibilidad sigue unos estándares.

```
import BAS
```

¡Nuestros paquetes también debemos importarlos!

Se puede importar en cualquier momento que lo necesites pero es más apropiado hacerlo al principio.

Para encontrar algún módulo o paquete que cubra una cierta necesidad, puedes consultar la lista de PyPI (Python Package Index) en <http://pypi.python.org/>, que cuenta con más de 4000 paquetes distintos.

PyPI - the Python Package Index

<https://pypi.python.org/pypi?%3Aaction=index>

Tipos y estructuras de datos

Listas, diccionarios y tuplas


Tipos de datos

Los tipos de datos hacen referencia al tipo de información que se trabaja. Esto incluye imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar.

Enteros	int	# L: long
Flotantes	float	# double: con mayor precisión
Cadena	str	
Booleano	bool	# True (1) or False (0)

 **Python**

```
var = 1/2  
>>> 0
```

 **Matlab**

```
var = 1/2  
>>> 0.5
```

Estructuras de datos

Las distintas formas en las que se pueden organizar los datos.

<https://docs.python.org/2/tutorial/datastructures.html>

Listas

Es un tipo de colección ordenada (arrays o vectores). Las listas pueden contener cualquier tipo de dato: números, cadenas, booleanos, etc. y también listas. Son mutables.

Diccionarios

Los diccionarios (matrices asociativas) son colecciones que relaciona una clave y un valor. No son secuencias, si no mappings (mapeados, asociaciones).

Tuplas

Similar a la lista con varias limitaciones: son inmutables y tienen un tamaño fijo. A cambio, son más ligeras que las listas (ahorran memoria).



Python

```
lista = [True, 'una lista', [1, 2]]
```



Python

```
tel = {'jack': 4098, 'sape': 4139}  
tel['guido'] = 4127  
print tel['guido']  
>>> 4127
```




Python

```
t = (1, 2, True, "python")
```


Estructura de datos

El aliasing

Cuando se asigna una variable a a otra b , lo que Python hace es referenciar '**b**' a '**a**'. Está ahorrando memoria asignándole a la variable b la misma dirección de memoria que a . Cualquier cambio en b , modificará a .

 Python

```
a = [1, 2, 3]
b = a
b is a
>>> True
```

 Python

```
a = [1, 2, 3]
b = list(a)
```


Acceso y particionado de listas

El particionado es el acceso y la extracción de los elementos de una lista.

El primer índice es el 0.

Desde un determinado elemento hasta el final

[5:]

Extracción de elementos de x en x

[::x]

Extracción de elementos desde el final

[-1:10:-3]

Extracción de un elemento de una lista dentro de otra

[0][0]



Python

```
lista = [22, True, 'una lista', [1, 2]]
```

```
lista[0:2] = [0,1]
```

```
print lista
```

```
>>> [0, 1, 'una lista', [1, 2]]
```

Métodos para listas

Python incluye una serie de métodos (o funciones) propias para operar con los distintos tipos de datos y estructuras (incluye algunas más, denominadas built-in functions).



Python

```
t = ['a', 'b', 'c']  
t.append('d')
```



Python

```
s = 'Disfruta con python'  
t = s.split()
```



Python

```
t = ['Disfruta', 'con', 'Python']  
delimiter = '  
delimiter.join(t)
```

<https://docs.python.org/2/tutorial/datastructures.html>

Asignación múltiple con tuplas

Un aspecto muy importante a considerar es la posibilidad de realizar asignación múltiples de números, cadenas, funciones, etc. mediante tuplas.



Matlab

```
PI = 3.14; G = 9.81; name = 'Matlab';
```




Python

```
(PI , G, name) = (3.14, 9.81, 'Python')
```

Las estructuras de datos en los paquetes

La creación de **vectores** (arrays) se realiza a partir de los distintos paquetes, como por ejemplo, los más habituales, numpy (numerical) o scipy (scientific).

 **Python**

```
import numpy as np

# Elementos previamente definidos
serie = np.array([0, 5, 10])

# Serie equiespaciada
serie = np.linspace(0, 100, 101)
```

Aspectos básicos de la programación con Python

Primeros pasos

Bucles, condiciones y máscaras

Existen diversas opciones para acceder de forma iterativa a los elementos de una estructura (vector o lista)



Python

```
for i in range(10):  
    print i
```



Python

```
for i in [0, 5, 20]:  
    print i
```



Python

```
for i in a:  
    print a[i]
```

Bucles , condiciones y máscaras



:

Se inicializa con su respectiva palabra clave y : al final de la sentencia



Break

La salida del bucle se consigue con el comando *break*.



Inicio/fin

El intervalo de valores a recorrer en el bucle comienza con cero (si no asigna otro) y termina en el inmediatamente anterior al que se asigne.



Definición previa del tamaño

Es una muy buena costumbre (y muy cómoda en Python) definir tanto el tamaño de la estructura como del tipo de datos a soportar antes de asignarle valores.



Indentado

Es necesario indentar la siguiente línea tras una palabra clave como: for, if, while, try. Por comodidad y legibilidad se debe usar un indentado de 4 espacios.



Cierre

Deshacemos el indentado.

Bucles, condiciones y máscaras



Python

```
if x == 5:  
    x = x + 10  
elif x >= 6:  
    x = x - 1  
else :  
    x += 1
```

Python reconoce una variable vacía o un cero como **False** y un uno como **True**.



Python

```
mask = y > 2  
mask = np.array(len(y), dtype= 'bool')
```

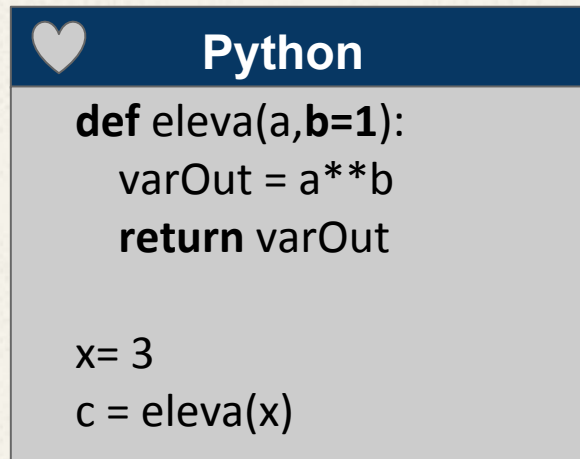
Programa funcional

Creación y uso de funciones

Programación funcional

Una función consta del encabezado, el cuerpo y el cierre.






El **encabezado** de la función se define con: *def*, sigue con el nombre de la función, entre paréntesis sus variables de entrada y cerrando dos puntos. Después le sigue el **cuerpo**, donde aparece el código con lo que la función hace. La función **cierra** con un *return x*, donde *x* define las variables de salida

A code block with a dark blue header containing a white heart icon and the word "Python". The body has a light gray background and contains Python code for a function and its usage.

```
def eleva(a,b=1):  
    varOut = a**b  
    return varOut  
  
x= 3  
c = eleva(x)
```


Programación funcional

```
def x(a,b=2,*c):  
    if len(c) == 1:  
        return (a + b, a - b)  
    else:  
        return (a, b)
```

 Python	print x(2)	[2, 2]
 Python	print x(2, 1)	[2, 1]
 Python	print x(2, 1, 1)	[3, 1]
 Python	print x(2, 2, 1, 1)	[2, 2]
 Python	print x(2, 2, [1, 1])	[4, 0]

Programación funcional

La llamada a las funciones se realiza de dos maneras: (1) si la función está en el mismo script; (2) si está en otro paquete.

Tengo un paquete llamado **misFunciones** que dentro tiene dos funciones, una denominada, **misCosas**, y otra, **algoMas**.



Python

```
import misFunciones as myF  
  
varOut1 = myF.misCosas(x)  
varOut2 = myF.algoMas(x)
```



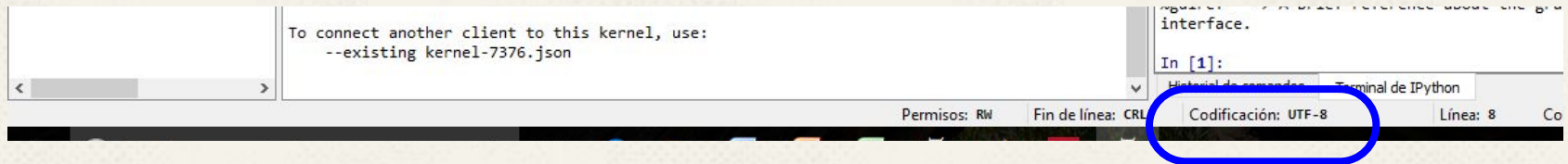
- *Indentación*
- *Salida múltiple en formato lista o tupla*
- *Argumentos clave*

Algunos detalles más

De Spyder y de Python

Codificación

```
# -*- coding: utf-8 -*-
```



Es posible que también aparezca en algunos casos

```
#!/usr/bin/python
```

A esta línea se le conoce en el mundo Unix como shebang, hashbang o sharpbang. El par de caracteres `#!` indica al sistema operativo que dicho script se debe ejecutar utilizando el intérprete especificado a continuación. Por supuesto además de añadir el shebang, tendremos que dar permisos de ejecución al programa.

Comentarios

```
"""
```

```
Created on Wed Jan 13 08:52:07 2016
```

```
@author: m_cob
```

```
"""
```

```
"""
```

```
Esto puede ser un comentario  
de variables líneas
```

```
"""
```

Comentario de línea

```
# Esto también es un comentario
```

```
var = 1 # Y ahora, un comentario en la misma línea
```

Comentario de bloque

```
#=====
```

```
#
```

```
#=====
```

Docstrings

CTRL + 1

CTRL + 4

Palabras reservadas

By convention, variable names start with a lower-case letter, and Class names start with a capital letter.

In addition, there are a number of Python keywords that cannot be used as variable names. These keywords are:

```
and, as, 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, with, yield
```

Note: Be aware of the keyword `lambda`, which could easily be a natural variable name in a scientific program. But being a keyword, it cannot be used as a variable name.

Lectura y escritura de archivos

Funciones precompiladas y el uso de paquetes

Lectura de archivos

De esta forma tan sencilla leemos todas las líneas de un archivo, las separamos por espacios y las guardamos en una lista.



Python

```
numList= list()
with open('numeros.txt') as open_file:
    for line in open_file:
        numList.extend(line.split())
```

Lectura de archivos

El paquete numérico **numpy** contiene una serie de clases para leer archivos **.txt**

np.loadtxt(fname, dtype=<type 'float'>, comments='#', delimiter=None, converters=None, skiprows=0, usecols=None)

<http://docs.scipy.org/doc/numpy/reference/generated/numpy.loadtxt.html>



Python

```
import numpy as np
```

```
dataSimar = np.loadtxt('SIMAR_2006008', skiprows=90)
```

Nombre	Tipo	Tamaño	Valor
dataSimar	float64	(166444L, 18L)	array([[1.95800000e+03, 1.00000000e+0... 1.3 ...

El paquete **csv** implementa una serie de clases para leer y escribir datos tabulados en formato **csv** y **excel**.

<https://docs.python.org/2/library/csv.html>

Lectura de archivos

Ficheros **.mat**



Python

```
from scipy.io import loadmat as ldmat

data = ldmat('T170_6_1.mat',
struct_as_record=False, squeeze_me=True)

for i in range(0, 8):
    signal_i = data['ScopeData'].signals[i].values
```

Escritura de archivos

Usando las funciones precompiladas



Python

```
fileOut=open('input_main.swn','w')
fileOut.write('PROJ \'Duck\' \'94\' \n')
fileOut.write('$Caso ' + id + '\n')
fileOut.close()

fileOut = open('prueba.txt','w')
fileOut.write("{0} entero {1} decimal {2} cadena".format(4,2.5,'python
rocks!'))
fileOut.close()
```

Escritura de archivos

El paquete numérico **numpy** contiene una serie de clases para escribir archivos **.txt**

- **np.savetxt**(*fname, var, fmt='%.18e', delimiter=' ', newline='\n', header='', footer=''*)
- **np.savez_compressed**('dataSimar.npz', x=dataSimar)

El paquete **csv** implementa una serie de clases para escribir datos tabulados en formato **csv** y **excel**.

Las cadenas de documentación

Uso sencillo y útil

Docstrings

Las cadenas de documentación se colocan al comienzo de una función y la definen y explican. Se encierran entre comillas triples.

```
def PreprocesoSL(signal, freq, nel, wavetype, T, nt):  
    """
```

Lee la señal de la superficie libre de los sensores acústicos y:

- 1. Elimina los transitorios**
- 2. Coloca la señal a nivel 0**
- 3. Devuelve una matriz con la posición de la onda de cada oscilación**

Las entradas son:

- Signals: las señales de los sensores [mm].**
- freq: frecuencia del sensor [hz]**
- nel: nº de elementos del transitorio**
- wavetype: oleaje regular (por ahora)**
- T: periodo**
- nt: nº de elementos por periodo de salida**

Las salidas de la función son:

- tsig: registro temporal de la señal tratada**
- esig: posición de la superficie libre**

```
v.1 15/12/2015 Autor  
    """
```

Un ejemplo de paquete web

La urllib2

Urllib2

urllib2 puede leer datos de una URL usando varios protocolos como HTTP, HTTPS, FTP, o Gopher.



Python

```
from urllib2 import urlopen
import calendar

def downFile(aa,mm):
    d, n = calendar.monthrange(aa,mm)
    for dd in range(1,n+1):
        for hh in range(0,24):
            url = 'http://opendap.puertos.
es/thredds/fileServer/RADAR_GIBRALTAR/%d/%02d/CODAR_GIBR_%d_%
02d_%02d_%02d00.nc.gz' %(aa,mm,aa,mm,dd,hh)
            r = urlopen(url)
            output = open(url[69:], 'wb')
            output.write(r.read())
            output.close()

for aa in range(2012,2015):
    for mm in range(1,13):
        downFile(aa,mm)
```

Algunas pautas sobre legibilidad

“El código se lee muchas más veces de lo que se escribe”

- Guido Van Rossum -

El PEP20 y los buenos hábitos



Indentación

Usar siempre cuatro espacios.



Tamaño máximo de línea

Limitar el tamaño máximo de línea a 79 caracteres.



Importación de paquetes

Importar cada paquete en una línea diferente. Varios módulos de un mismo paquete si se pueden importar en la misma línea.



Comentarios

Prioriza su actualización.

Peor es tener comentarios que contradigan al código que no tener comentarios.



Uso de espacios

Escribe como de manera habitual. Dentro de funciones, los espacios se eliminan.



Nombrando funciones

Utiliza palabras en minúscula para nombrar funciones y usa el guión bajo para mejorar la legibilidad.

El PEP20 y los buenos hábitos



Importación de paquetes

Utilizar un alias cuando se puedan producir coincidencias



Comando *startswith()*

No apliquéis el slicing para chequear sufijos.



Constantes

Escribir con letras mayúsculas y con guiones bajos separando palabras.

MAX_OVERFLOW



Python

```
from numpy import array  
serie = array([0, 1])
```



Python

```
if lista.startswith('nx'):  
    print 'OK'
```



Directorio de carpetas

Proyecto

- src
- DATA
- IM
- programa principal

Los paquetes más utilizados en el entorno científico

Numpy, scipy, pandas, sympy, matplotlib

Buscando para el entorno científico

jrjohansson Fix grammar error. Closes #32		Latest commit ed57653 9 days ago
images	Fix mispring in hyposthesis word on theory-experiment-computation.svg...	2 years ago
scripts	added missing extra files	3 years ago
.gitignore	Initial commit	3 years ago
Lecture-0-Scientific-Computing-with-P...	conda/miniconda as preferred installation method	4 months ago
Lecture-1-Introduction-to-Python-Prog...	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago
Lecture-2-Numpy.ipynb	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago
Lecture-3-Scipy.ipynb	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago
Lecture-4-Matplotlib.ipynb	Fix grammar error. Closes #32	9 days ago
Lecture-5-Sympy.ipynb	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago
Lecture-6A-Fortran-and-C.ipynb	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago
Lecture-6B-HPC.ipynb	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago
Lecture-7-Revision-Control-Software.i...	convert to newer ipython notebook format. minor cleanups. header updates	4 months ago



<https://github.com/jrjohansson/scientific-python-lectures>



**PROGRAMACIÓN
SENCILLA, RÁPIDA Y
VERSÁTIL**

**COMUNIDAD MUY
GRANDE Y
CRECIENDO**

¿Te unes?

¡GRACIAS!

¿Preguntas?

pmagana@ugr.es

mcobosb@ugr.es