INTRODUCCION A LA PROGRAMACIÓN EN PYTHON

Facultad de Estudios Ambientales y Rurales
Departamento de Ecología y Territorio
john.chavarro@javeriana.edu.co

05 de Agosto de 2014

CONTENIDO

Definición

Las secuencias en Python son una colección de objetos que puede ser incluso de distinto tipo. Entre las secuencias se encuentran las listas, las Tuplas y Arreglos o matrices.

Las listas se definen como una colección de valores entre paréntesis cuadrados por ejemplo:

```
mi_lista = [1, 'MARIA', 3.151416, TRUE, FALSE]
```

Las listas pueden ser multidimensionales, o lo que es lo mismo una lista unidimensional puede embeber elementos que son listas:

```
otra_lista = [0, ['PERA', 'MANZANA', 'MIEL'], 3.141516]
```

Definición

Para acceder al elemento de una lista basta con acceder a él a través del nombre de la lista y el **índice** del elemento, por ejemplo:

```
W = otra_lista[0]
```

En este caso `W` será igual a 0. Hay que recordar que el primer elemento de una lista tiene el índice `0` **[Python Style]**.

```
W = otra_lista[1]
```

Devolverá:

```
W = ['PERA', 'MANZANA', 'MIEL']
```

Definición

Si se desea acceder a un elemento de una lista embebida después del índice que señala al elemento que es la lista embebida, se señala en corchetes cuadrados el índice del elemento de esa lista que es necesario devolver:

```
W = otra_lista[1][2]
print W

[1] MIEL
```

Definición

Es una de las estructuras de datos mas importante en cualquier lenguaje de programación.

Consiste en una estructura que ordena eficientemente cualquier tipo de información.

Python no tiene una estructura nativa de arreglos (array), pero si una lista mucho mas general y fácil de utilizar, aun con arreglos multidimensionales.

Array Creation

```
import numpy as np
```

NumPy es el paquete fundamental para la computación científica en Python. Es una librería de Python que proporciona objetos que pueden ser entre otras arreglos multidimensionales, objetos derivados (tales como matrices enmascarados y/o matrices), y por supuesto una variedad de rutinas para operaciones rápidas sobre matrices, incluyendo matemáticas, lógica, reorganizado e indexado, clasificación, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.

Tomado de http://docs.scipy.org/doc/numpy/user/whatisnumpy.html

Array Creation

```
import numpy as np

Q = np.array([[1, 2],[3, 4],[5, 6]])
print q

# Quiero extraer el número 5

A = Q[2]
print A
A = Q[2][0]
print A
print Q[2, 0]

# Ahora con cadenas de texto

B = np.array([[1, 2], [3, 4], [5, 'wok']])
```

Array Creation

```
# Ahora arreglos 3D

m3d = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[10, 11, 12], [13, 14, 15], [16, 17, 18]]])

'''
Este arreglo de tres dimensiones puede tratarse cómo un arreglo unidimensional que contiene como
elementos arreglos bidimensionales
'''

# Cual es la estructura de este arreglo ?

print m3d.shape

# Creando ndarray

A = np.arange(start=1, stop=4, step=1, dtype=float) ó A = np.arange(1, 4, 1, float)

B = range(1, 5)
```

Array Creation

Nótese que el arreglo obtenido es un **espacio semi-abierto** que no contiene el valor de “2”. El intervalo creado nunca contiene al valor definido por el argumento “fin”.

El **tipo de dato** del arreglo se puede especificar directamente desde el comando “arange”. Si este no se específica explícitamente, **el tipo de dato es asignado por el interpretador cómo el más complejo de los tipos presentados** entre los argumentos pasados al comando “arange”.

```
# En forma implicita

A = np.arange(start=1.0, stop=4., step=1)
```

Array Creation

Algo un poco incómodo en la función np.arange(…) es que siempre devuelve un **arreglo unidimensional**, sin embargo otra utilidad de la función **shape** es que permite reorganizar la estructura del un arreglo.

```
# Ejemplo
A = np.arange(start=1, stop=13)
print A
print A.shape
# Reorganizar a una matrix de 3 filas por 4 columnas
A.shape = (3, 4)
print A
# y si la queremos 3D ?
… ?
# Otros usados para creación de array son range, linspace, np.zeros, np.ones
```

Leyendo archivos externos …. Con NumPy

Utilizando arreglos numpy se puede aplicar el método **numpy.loadtxt** para leer archivos de texto que contengan el mismo número de columnas en cada fila. Para guardar se utiliza el método **numpy.savetxt**.

```
# la matriz se carga en el arreglo x,
# todas las filas de “x” deben tener el mismo número de columnas
import numpy as np

X = np.loadtxt(‘logistica.txt’)
m = 5
B = 10
Y = m * X + B
np.savetxt(‘recta.txt’, Y)
np.savetxt('recta.txt', Y, fmt='%.18e', delimiter=' ')
np.savetxt('recta.txt', Y,  fmt='%.2f', delimiter=' ')
```

Leyendo y guardando archivos de Excel®

Para leer archivos de Excel se utiliza la libreria **xlrd**, esta se puede descargar de http://www.python-excel.org

```
import xlrd

wb = xlrd.open_workbook('myworkbook.xls')
print wb.sheet_names()

# Seleccionar una hoja especifica

worksheet = wb.sheet_by_name('QL')

# Nos vemos en PANDAS !
```

Python Fancy Slicing (Rebanado de matrices)

```
# ¿Como puedo seleccionar una parte de mi matriz?
# Generar una matriz de dimensiones (5, 2)
import numpy as np
# Recordemos el indexado en un arreglo
# ¿Como obtengo un valor de mi matriz?
A = np.arange(1, 11)
A.shape = (5, 2)
print nombre_matriz[posición fila, posición columna]
X = np.loadtxt('logistica.csv', delimiter=',')
m = 5
B = 10
Y = m * X[:, 1] + B

np.savetxt('recta.txt', Y,  fmt='%.2f', delimiter=' ')
```

POSICION		COLUMNAS	
		0	1
FILAS	0	1	2
	1	3	4
	2	5	6
	3	7	8
	4	9	10

Python Fancy Slicing (Rebanado de matrices)

```
print A

[1]   A[:, :]
[2]   A[1:3, :]
[3]   A[2, :]
[4]   A[:, 1]
[5]   A[1:] # de la fila uno hasta el final
[6]   A[:2] # todas hasta la fila 2
[7]   A[:, :1] # todas las columnas hasta la 1
[8]   A[:, 1:2] # las columnas entre la columna 1 y 2 (sin incluir 2)
[9]   A[:-1] # todas menos la última fila
[10]  A[::2] # una tercera bandera (step)
```

POSICION		COLUMNAS	
		0	1
FILAS	0	1	2
	1	3	4
	2	5	6
	3	7	8
	4	9	10

Generación de graficas con matplotlib (http://matplotlib.org/)

John Hunter (1968–2012)

On August 28 2012, John D. Hunter, the creator of matplotlib, died from complications arising from cancer treatment, after a brief but intense battle with this terrible illness. John is survived by his wife Miriam, his three daughters Rahel, Ava and Clara, his sisters Layne and Mary, and his mother Sarah.

If you have benefited from John's many contributions, please say thanks in the way that would matter most to him. Please consider making a donation to the John Hunter Memorial Fund.

```
from pylab import *

t = arange(0.0, 2.0, 0.01)
s = sin(2*pi*t)

plot(t, s)
xlabel('time (s)')
ylabel('voltage (mV)')
title('About as simple as it gets, folks')
grid(True)
savefig("test.png")
show()
```

Mi primer *.py

```
import numpy
from pylab import *
from enthought.mayavi.mlab import *

'''
De este modo se habilitan varios tipos de gráficos 3D implementados en la biblioteca mayavi,
esta última queda habilitada con la instalación de Enthought Python Distribution.
'''


X = linspace(-100 * pi, 100 * pi, 1000)
Y = 100 * sin(pi * X)
Z = 100 * cos(pi * X)
plot3d(x, y, z, arange(1000,2000), tube_radius=5, colormap='Spectral')
```

Mi primer *.py

```
# http://journals.ametsoc.org/doi/abs/10.1175/1520-0469%281963%29020%3C0130%3ADNF%3E2.0.CO%3B2

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def lorenz(x, y, z, s=10, r=28, b=2.667) :
    x_dot = s*(y - x)
    y_dot = r*x - y - x*z
    z_dot = x*y - b*z
    return x_dot, y_dot, z_dot

dt = 0.01
stepCnt = 10000
# Need one more for the initial values
xs = np.empty((stepCnt + 1,))
ys = np.empty((stepCnt + 1,))
zs = np.empty((stepCnt + 1,))
# Setting initial values
xs[0], ys[0], zs[0] = (0., 1., 1.05)
# Stepping through "time".
for i in range(stepCnt) :
    # Derivatives of the X, Y, Z state
    x_dot, y_dot, z_dot = lorenz(xs[i], ys[i], zs[i])
    xs[i + 1] = xs[i] + (x_dot * dt)
    ys[i + 1] = ys[i] + (y_dot * dt)
    zs[i + 1] = zs[i] + (z_dot * dt)

fig = plt.figure()
ax = fig.gca(projection='3d')

ax.plot(xs, ys, zs)
ax.set_xlabel("X Axis")
ax.set_ylabel("Y Axis")
ax.set_zlabel("Z Axis")
ax.set_title("Lorenz Attractor")

plt.show()
```

No dejen de visitar …

Sitio web oficial de Python
Http://www.python.org

Descripción de la biblioteca ScyPy
http://www.scipy.org/more_about_SciPy

Descripción de la biblioteca NumPy
http://numpy.scipy.org/

Recetas de programación numérica en Python
http://www.scipy.org/Cookbook

Recetas de programación gráfica con Matplotlib
http://www.scipy.org/Cookbook/Matplotlib

Biblioteca/Módulo para series de tiempo en Python
http://pytseries.sourceforge.net/contents.html

Página comparativa NumPy vs. MatLab
http://www.scipy.org/NumPy_for_Matlab_Users

Entornos de desarrollo para Python
http://wiki.python.org/moin/IntegratedDevelopmentEnvironments

Guía para utilizar texto LaTex para escribir ecuaciones en los ejes y títulos de las figuras creadas con Matplotlib
http://matplotlib.sourceforge.net/users/mathtext.html

Página para Mayavi
http://code.enthought.com/projects/mayavi/docs/development/html/mayavi/index.html