Unidad 03
Conceptos Básicos de Python II

Curso de geoprocesamiento de datos con Python
2016

Cayetano Benavent Viñuales
Analista GIS en Geographica
cayetano.benavent@geographica.gs

Unidad 03 - Sumario de contenidos

1. Módulos.
2. Manejo de ficheros.
3. Excepciones.
4. List comprehensions.
5. Dict comprehensions.
6. Iteradores y generadores.
7. Clases y objetos en Python
8. Bibliografía

Módulos

Un módulo no es más que un fichero con extensión .py que contiene un conjunto de definiciones y/o declaraciones de Python.

Para que un módulo pueda utilizar otro módulo, hay que importarlo previamente.

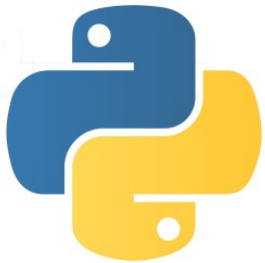
El nombre del módulo para la importación es el nombre del fichero sin la extensión .py.

```
>>> import nombremodulo
```

Un módulo se puede ejecutar como script desde la línea de comandos. Para ello, debemos añadir al módulo cierto código que identificará el script como tal (su comportamiento podemos hacer que sea diferente).

En el ejemplo práctico lo veremos con más claridad. Más información sobre el asunto en:

https://docs.python.org/3/library/__main__.html



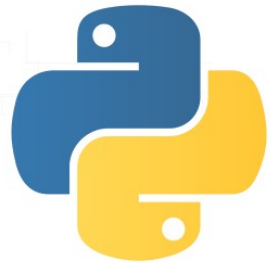
Para entender mejor el funcionamiento de módulos en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit03_01.

Manejo de ficheros

Python contiene una gran cantidad de módulos para lectura y escritura de ficheros. Los ficheros son un tipo especial de objeto.

Así mismo, la gestión de ficheros, carpetas, rutas, etc. es muy sencillo en Python. A continuación, a través de una serie de ejercicios prácticos introduciremos los elementos más importantes.

Algo importante que hay que mencionar es que, dependiendo del Sistema Operativo en el que nos encontremos, ciertas tareas hay que realizarlas de distinta forma (manejo de rutas, especialmente en Windows).



Para entender mejor el manejo de ficheros en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit03_02.

Excepciones

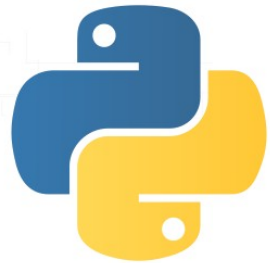
Gestionar adecuadamente las posibles excepciones que surjan en nuestro código es algo de una importancia crucial.

Por suerte, en Python manejar excepciones es muy fácil.

Con el bloque TRY-EXCEPT podemos manejar de forma muy efectiva las excepciones y controlar la ejecución del código.

Para profundizar:

https://docs.python.org/3/tutorial/errors.html



Para entender mejor el manejo de excepciones en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit03_03.

List comprehensions

Definición:

“List comprehensions provide a concise way to create lists.”

Es una manera muy eficiente y rápida de crear listas (mejor que con bucles FOR clásicos + append).

Podemos usar estructuras de control de flujo, incluso anidadas, e introducir expresiones complejas.

List comprehensions

Para entender mejor el funcionamiento de las list comprehensions, realizaremos un ejercicio práctico siguiendo el Notebook unit03_04.

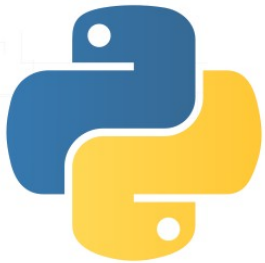
Dict comprehensions

Definición:

Siguen la misma filosofía sintáctica de las list comprehensions, salvo que debes agrupar la expresión con los corchetes propios de un diccionario.

Dentro del bucle FOR se extrae la clave y valor del diccionario separados por coma. Cada clave y valor podemos usarlo en expresiones dentro de estructuras de control de flujo.

Dict comprehensions



Para entender mejor el funcionamiento de los dict comprehensions, realizaremos un ejercicio práctico siguiendo el Notebook unit03_05.

Es importante diferenciar en Python 3 conceptos:

iterables, iteradores y generadores.

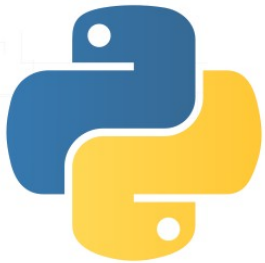
Simplificando, podemos decir que aquellos objetos sobre los que podemos iterar (con un bucle FOR, por ejemplo) son iterables (listas, strings, etc.).

Un iterador (`iterator`) es un objeto que, sin almacenar los elementos sobre los que itera, sabe cuál es el elemento que vendrá a continuación (implementa el método `next`).

Los generadores (`generators`) son una forma de crear iteradores en funciones, haciendo uso de la palabra clave `yield`.

Una variante de los generadores, muy eficiente y en línea con las list comprehensions, son los generator expressions.

Se construyen igual que una list comprehension, pero usando paréntesis para encerrar la expresión.



Para entender mejor el manejo de iteradores y generadores en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit03_06.

Objetos y Clases en Python

Python, como ya hemos mencionado en unidades previas, es un lenguaje de programación orientado a objetos.

“Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. .”

(https://en.wikipedia.org)

Encapsulamiento, herencia y polimorfismo son algunas de las técnicas principales que implementa la POO.

En este curso introductorio no tiene cabida aprender POO, ya que requiere más tiempo y dedicación.

No obstante, es importante que sepamos que en Python todo son objetos. Es por ello que introduciremos algunos conceptos muy básicos para entender mejor el lenguaje y su sintaxis.

Para entender mejor varios conceptos básicos sobre clases y objetos en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit03_07.

Bibliografía

Selección bibliográfica:

- Allen B. Downey (2015): “Think Python. How to think like a computer scientist (2nd Edition)”. O'Reilly. http://www.greenteapress.com/thinkpython2/index.html

- Zed Shaw (2014): “Learn Python the Hard Way (3rd Edition)”. Addison Wesley.

- Alex Martelli (2009): “Python in a Nutshell (2nd Edition)”. O'Reilly.

- Mark Lutz (2015): “Learning Python (5th Edition)”. O'Reilly.

- David Beazley, Brian K. Jones (2013): “Python Cookbook (3rd Edition)”. O'Reilly.

- Luciano Ramalho (2015): “Fluent Python”. O'Reilly.

- David Beazley (2009): “Python Essential Reference (4th Edition)”. Addison Wesley.