Unidad 02
Conceptos Básicos de Python I

Curso de geoprocesamiento de datos con Python
2016

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

Unidad 02 - Sumario de contenidos

The Zen of Python

También conocido como PEP 20,
"The guiding principles for Python's design".

Si ejecutamos lo siguiente, podremos leerlo:

```
>>> import this
```

The Zen of Python

(by Tim Peters)

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one-- and preferably only one --obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
*- Although never is often better than *right* now.*
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea -- let's do more of those!

PEP8

Style Guide for Python Code

Es la guía oficial de estilo de la Python Software Foundation, por lo que es importante leerla y seguir sus prescripciones.

https://www.python.org/dev/peps/pep-0008

Una guía, en la que se explica de forma muy amigable el PEP8:

http://pep8.org/

PEP8

Style Guide for Python Code

La guía es larga, sólo remarcar varios puntos clave:

- Indentar los bloques de código con 4 espacios (por nivel).

- Usar espacios mejor que tabulaciones.

- No superar los 80 caracteres por línea de código.

Tipos de datos básicos: numéricos

`Int`

Integer numbers

2, -3, 5432121

`float`

Floating point numbers

2.13, 31e-3

`complex`

Complex numbers

2 + 3j

Tipos de datos básicos: numéricos

Aparte de los operadores matemáticos (+, -, *, /, %, etc.) y las funciones propias asociadas a estos tipos numéricos (round, abs, int, float, etc.),

es clave la librería (de la PSL) math

```
>> import math
```

y la librería (de la PSL) cmath para números complejos

```
>> import cmath
```

Tipos de datos básicos: numéricos

`bool`

booleans

True, False

El tipo `bool` es realmente un subtipo de `int`.

Falso es igual a 0 y verdadero es igual a 1.

Las operaciones booleanas y las comparaciones devuelven verdadero o falso.

Tipos de datos básicos: numéricos

`bool`

booleans

comparaciones

Operation	Meaning
`<`	strictly less than
`<=`	less than or equal
`>`	strictly greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity

operaciones booleanas

Operation	Result
`x or y`	if *x* is false, then *y*, else *x*
`x and y`	if *x* is false, then *x*, else *y*
`not x`	if *x* is false, then `True`, else `False`

Tipos de datos básicos: numéricos

La Python Standard Library implementa otros tipos numéricos:

`decimal`

Decimal fixed point and floating point arithmetic

True, False

`fractions`

Rational numbers

True, False

Tipos de datos básicos: numéricos

Para entender mejor los tipos de datos numéricos en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_01.

Tipos de datos básicos: strings

El dato textual se maneja en Python a través del tipo string `str`.
Es un tipo de dato inmutable.

Hay diferentes maneras de generar strings, pero básicamente son cadenas de texto delimitadas entre:

– single (' '), double (" "), triple single (''' '''), or triple double (""" """) quotations –

Como veremos a continuación, hay numerosas fuciones built-in para el manejo de strings.

Tipos de datos básicos: strings

Para entender mejor el tipo de dato `string` en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_02.

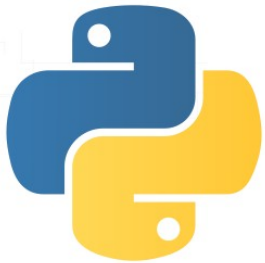
Listas

La lista (`list`) es uno de los tipos de dato propios de Python más interesantes y flexibles. Es un tipo de dato mutable.

Una lista puede contener otros tipos de datos (¡incluso listas!). En una misma lista pueden incluirse diferentes tipos de datos.

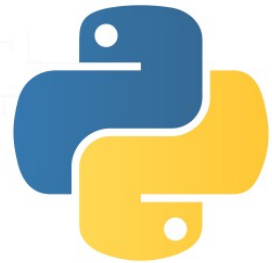
Su sintaxis es:

```
>> emptylist = []
>> mylist = [“hello”, 1, 4.5, “world”]
```



Para entender mejor el tipo de dato `list` en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_03.

Tuplas

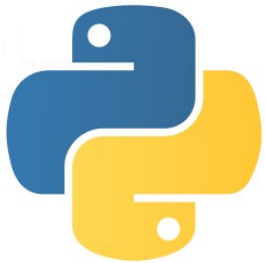


Una tupla (`tuple`) es una secuencia de datos muy similar a la lista, pero inmutable.

No pueden ser modificadas.

Pueden contener diferentes tipos de datos.

```
>> emptytuple = ()
>> mytuple = (37.5, -6.03)
```



Para entender mejor el tipo de dato `tuple` en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_04.

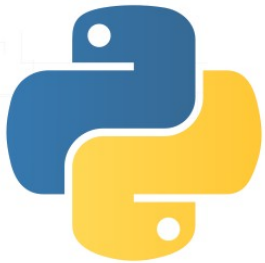
Diccionarios

Un diccionario (`dict`) es una estructura de datos basada en el concepto de *hash table* o matriz asociativa (asociación de pares clave/valor). Sus elementos no están indexados por su posición en el array, sino por las claves.

Son estructuras en las que las búsquedas de valores, en base a su clave, son muy eficientes.

Los valores pueden ser modificados y contener diferentes tipos de datos. Las claves son inmutables.

```
>> emptydict = {}
>> mydict = {'key01': 'value01', 'key02':
   'value02'}
```



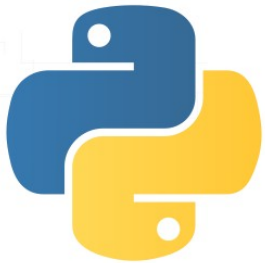
Para entender mejor el tipo de dato `dict` en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_05.

Sets

Un set (`set`) es un conjunto desordenado de elementos *"hashables"* distintos (recordemos los diccionarios).

Son tremendamente eficientes para manejar listas de elementos diferentes. Su uso tiene muchas ventajas, como el poder llevar a cabo con las listas operaciones de intersección, unión, diferencia, diferencia simétrica, etc.

```
>> a = set(["Red", "Green", "Blue"])
```



Para entender mejor el tipo de dato `set` en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_06.

Control de flujo: estructuras condicionales

Una estructura de control de flujo permite controlar cómo se ejecutan las órdenes de un determinado programa.

En el caso de las sentencias IF, las más importantes, las órdenes se ejecutan o no en función del valor de una condición.

En Python, la sintaxis de una sentencia IF es:

```
>> if x > 5:
        A = True
        print()
```

Control de flujo: estructuras condicionales

Para entender mejor el funcionamiento de las estructuras de control de flujo condicionales en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_07.

Control de flujo: estructuras iterativas

Este tipo de estructuras de control de flujo inician o repiten un determinado conjunto de instrucciones mientras se cumpla una condición.

La estructura iterativa más importante, y con diferencia la más usada en Python, es la declaración FOR:

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

Control de flujo: estructuras iterativas

La otra estructura iterativa importante en Python, es la declaración WHILE:

```
>>> a = 0
>>> while a < 10:
        print(a)
        a += 1
```

Control de flujo: estructuras iterativas

Para entender mejor el funcionamiento de las estructuras de control de flujo iterativas en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_08.

Funciones

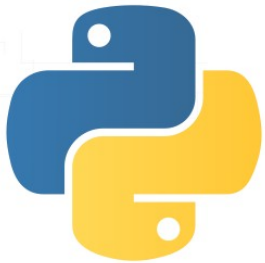
Una función, llamada en otros lenguajes subrutina, rutina, procedimiento, etc.,

"(...) se presenta como un subalgoritmo que forma parte del algoritmo principal, el cual permite resolver una tarea específica."

(http://es.wikipedia.org)

Las funciones se definen en Python con la palabra clave `def` :

```
>>> def circPerim(r):
        return 2*math.pi*r
```



Para entender mejor cómo construir y utilizar funciones en Python, realizaremos un ejercicio práctico siguiendo el Notebook unit02_09.

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.