Análisis de datos ambientales con Python I

Marvin J. Quispe Sedano

Mayo 2021

Contenido

- Introducción y conceptos generales de Python
- Instalación de Anaconda con Python 3.8
- Uso de Python en la nube con Google colab
- Operaciones básicas

¿Por qué muchos usan Python?

- El código en Python está diseñado para ser legible y, por lo tanto, reutilizable y mantenible.
- El código desarrollado en Python suele ser 1/3 o 1/5 del tamaño del código escrito en C++ o Java.
- Python se ejecuta sin problemas en las principales plataformas informáticas y sistemas operativos (Windows, GNU/Linux, OSX).
- Python viene con una gran colección de funcionalidades precompiladas y portátiles, conocidas como biblioteca estándar y también admite ampliar la biblioteca con aplicaciones o librerías de terceros.
- Python es "open source".

¿Quiénes usan Python?

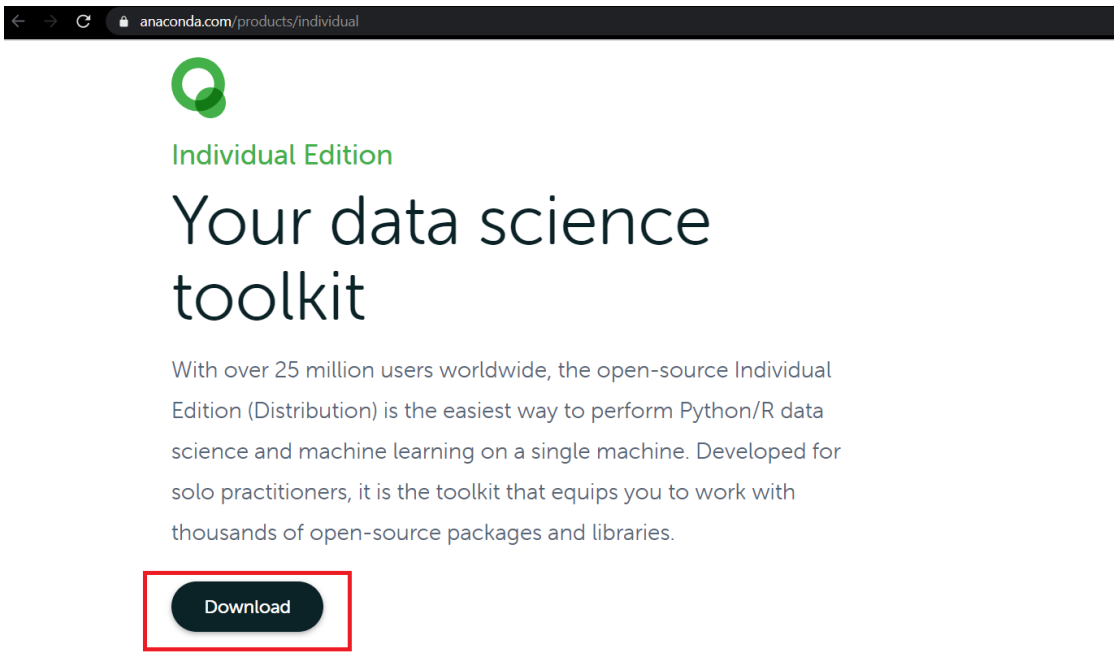
¿Qué puedo hacer con Python?

- Los roles de Python son prácticamente ilimitados: puede usarlo para todo, desde el desarrollo de sitios web, videojuegos, pronósticos meterólogicos, modelado climático, robótica, etc.

¿Cómo instalo Python en mi PC?

- Se recomienda instalar Python desde Anaconda, aunque se puede instalar directamente desde:
  `https://www.python.org/downloads/`

- Anaconda es una distribución open source de Python y R que incluye más de 250 paquetes orientados a la ciencia de datos.



Operaciones aritméticas

- Los operadores aritméticos se utilizan con valores numéricos para realizar operaciones matemáticas comunes.

Operator	Meaning	Example
+	Addition	4 + 7 ⟶ 11
-	Subtraction	12 - 5 ⟶ 7
*	Multiplication	6 * 6 ⟶ 36
/	Division	30 / 5 ⟶ 6
%	Modulus	10 % 4 ⟶ 2
//	Quotient	18 // 5 ⟶ 3
**	Exponent	3 ** 5 ⟶ 243

Tipos de objetos (Python's Core Data Types)

- En Python se denomina "objeto" a una estructura de datos definida para representar diversos componentes en el dominio de su aplicación.
- Por defecto, Python proporciona varios tipos de objetos como parte intrínseca del lenguaje.

Object type	Example literals/creation
Numbers	`1234, 3.1415, 3+4j, Decimal, Fraction`
Strings	`'spam', "guido's", b'a\x01c'`
Lists	`[1, [2, 'three'], 4]`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`
Tuples	`(1, 'spam', 4, 'U')`
Files	`myfile = open('eggs', 'r')`
Sets	`set('abc'), {'a', 'b', 'c'}`
Other core types	Booleans, types, None
Program unit types	Functions, modules, classes (Part IV, Part V, Part VI)
Implementation-related types	Compiled code, stack tracebacks (Part IV, Part VII)

Números (Numbers)

- Los números en Python pueden ser enteros (int), decimales (float) y complejos (complex).
- Además, con la ayuda de los "módulos" o librerías de python podemos realizar diversas operaciones e importar constantes numéricas.

Texto (Strings)

- Las **secuencias** mantienen un orden de izquierda a derecha entre los elementos que contienen.

```
String = "PYTHON"
```

P	Y	T	H	O	N
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

- Las cadenas o "strings" se usan para registrar información textual, así como colecciones arbitrarias de bytes.
- Las cadenas se concatenan usando "+" y se multiplican usando "*".

Listas (List)

- Las listas son **secuencias** ordenadas de objetos y a diferencia de los "strings" son de tipo **mutable**.

String = "PYTHON"

P	Y	T	H	O	N
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

List = ["TEMP", 26.5, "HUM", 85, "DIRV", "NE"]

List = ["TemC", 26.5, "Hum", 85, "DirV", "NE"]

Diccionarios (Dict)

- Los diccionarios son **asignaciones**, por ello no mantienen un orden confiable de izquierda a derecha.
- Son útiles siempre que necesitemos asociar un conjunto de valores con claves. Por ejemplo:

```
Dict = {"TEMP": 26.5, "HUM": 85, "DIRV": "NE"}

Dict["DIRV"]

'NE'
```

Tuplas

- Los tuplas son muy parecidas a las listas. Al igual que las listas son **secuencias**, con la diferencia que son de tipo **inmutable**.
- Las tuplas proporcionan una especie de restricción de integridad que es conveniente en programas más grandes.

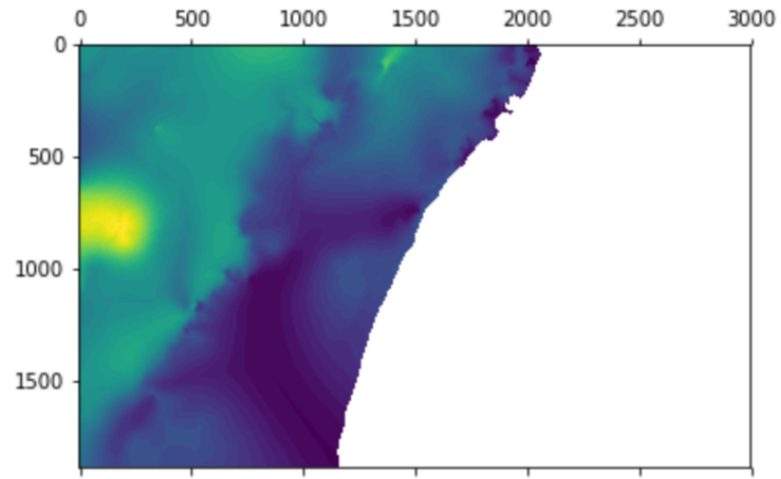
Archivos (Files)

- Este tipo de objeto sirve para importar y exportar diferentes tipos de archivos en Python.

- Python es compatible en la importación/exportación de la mayoría de extensiones de archivos, para ello se usan librerías de la biblioteca y/o de terceros.

```
In [3]: import georasters as gr
        demFile = gr.from_file('../InputData/RasterElevacion.tif')
        demFile.plot()
```

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x10a05ba8>
```



GRACIAS