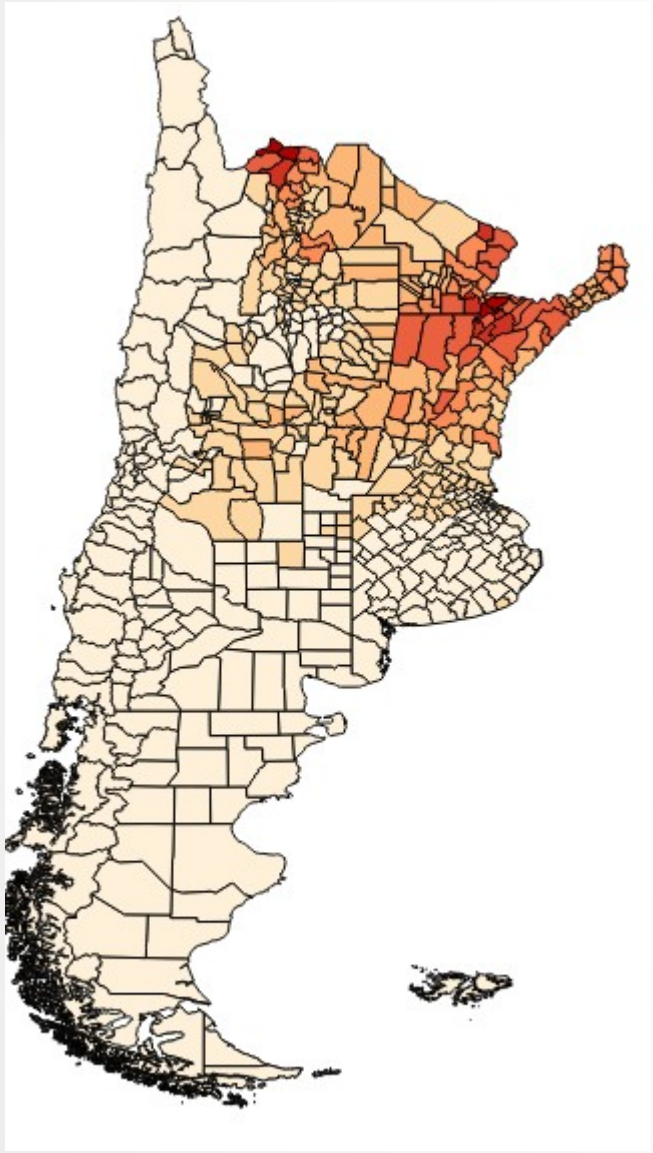
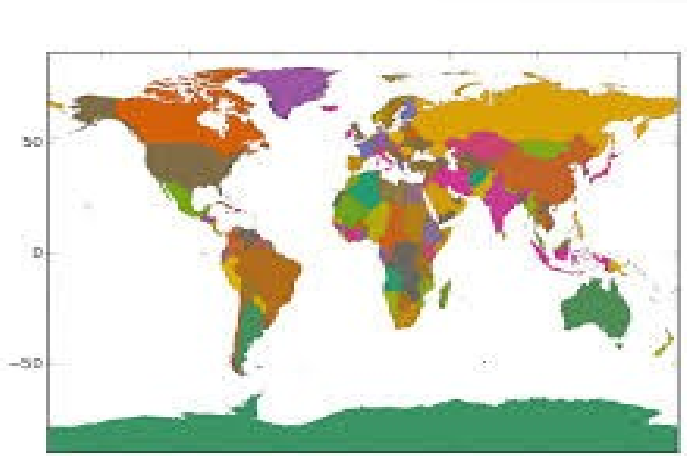
Análisis Geoespacial con Python





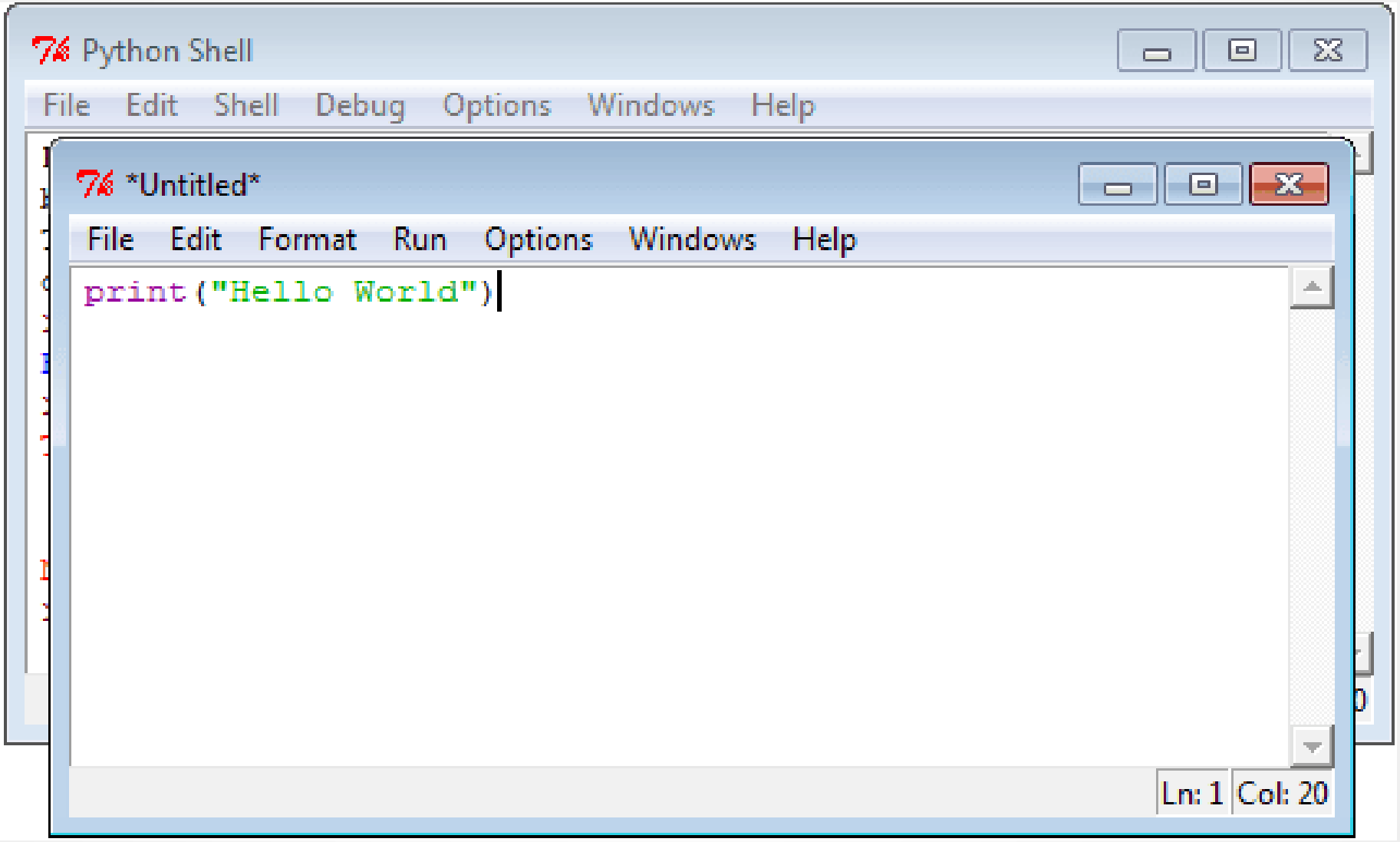
python™

3Geo 2017
WORKSHOP INTERNACIONAL

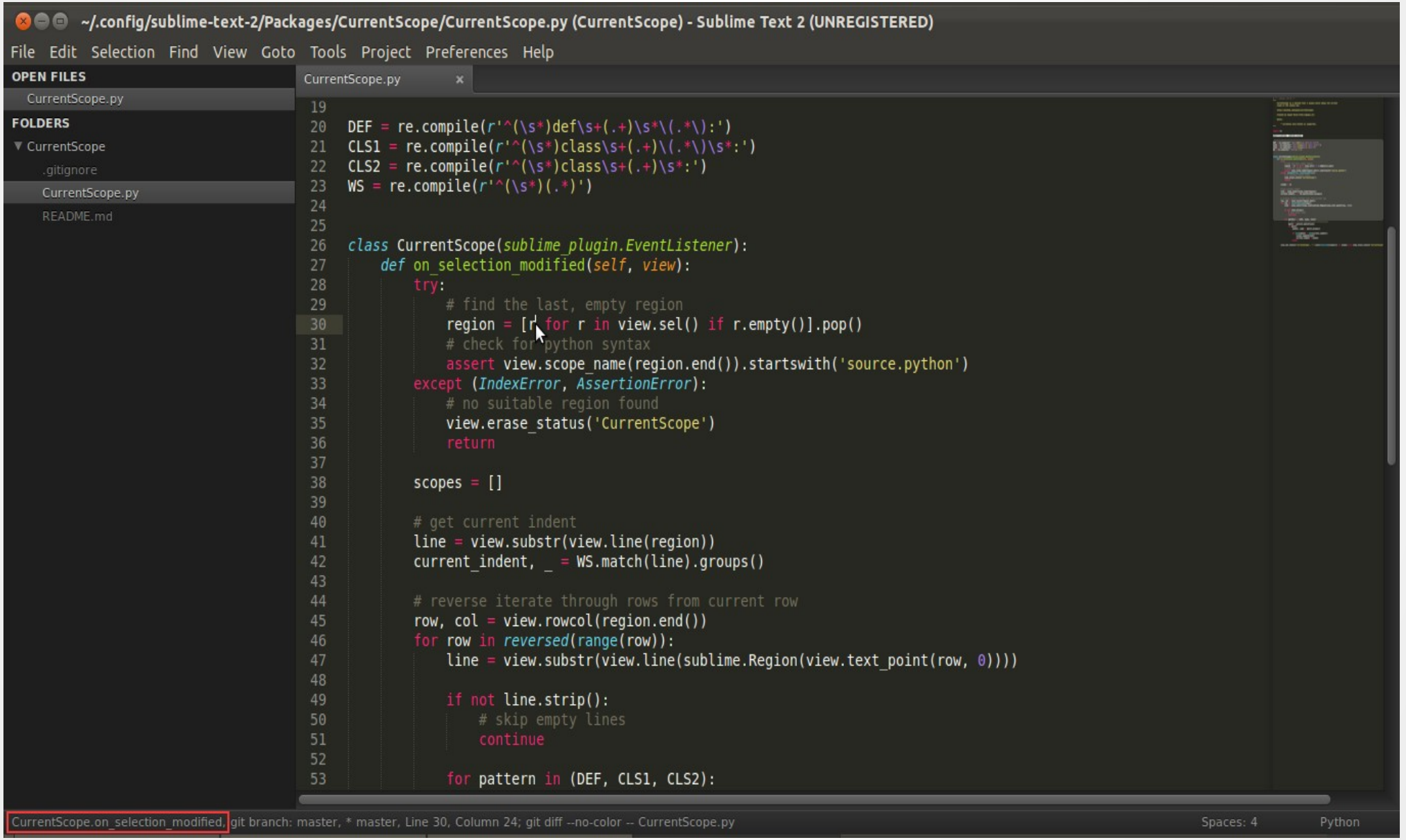
Características de Python

- Lenguaje de propósito general de amplia difusión.
- Multiparadigma: orientado a objetos, procedural, algunas características funcionales.
- Promueve el desarrollo de código “sintético” y “legible”.
- Dos versiones en curso: Python 2 y Python 3.

Python IDEs: IDLE



Python IDEs: Sublime Text

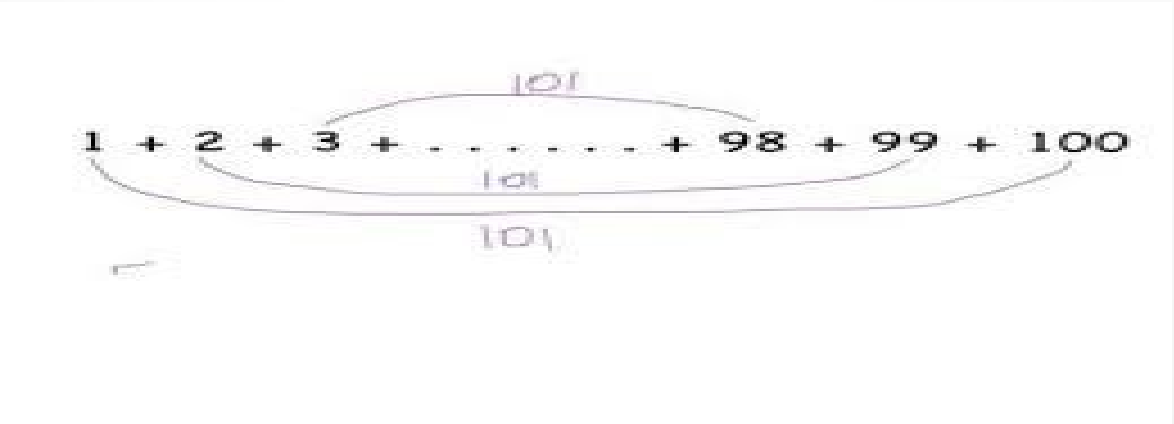


Python básico

- Uso desde la consola:

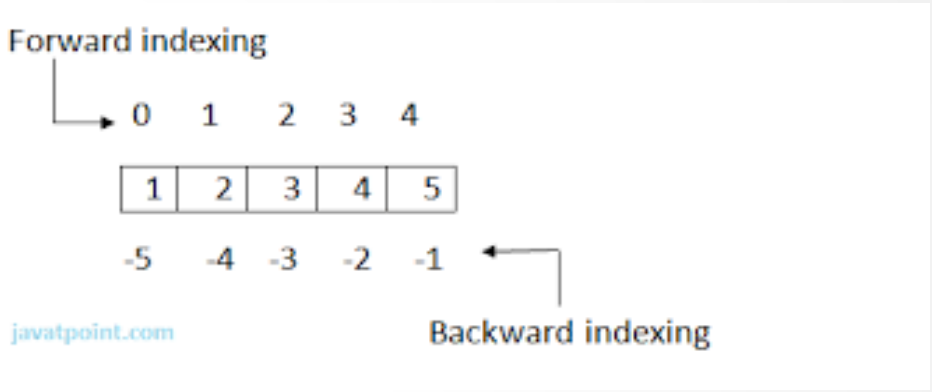
```
accum = 0.0
for index in range(0,100):
    accum = accum + index
print “Acumulador: “, accum
```

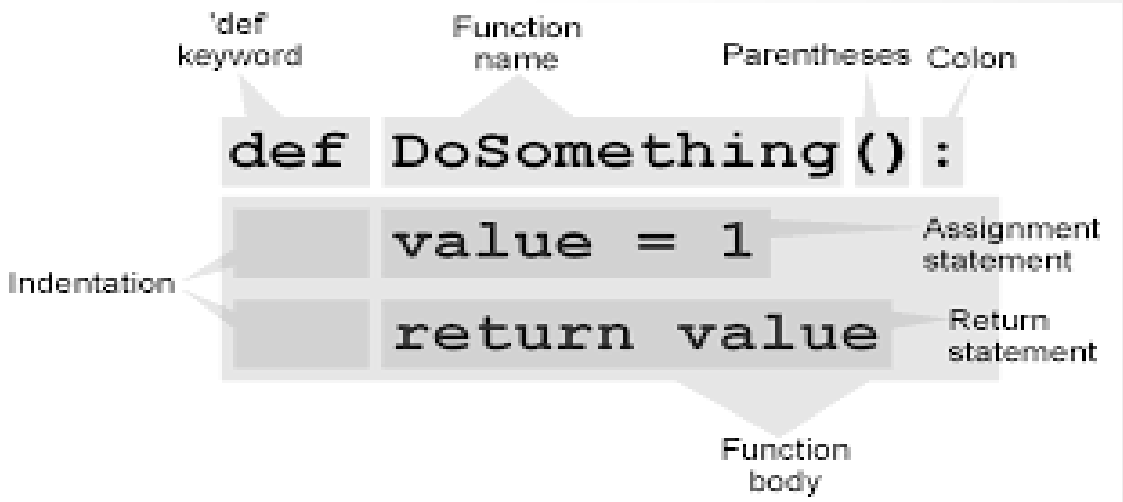
- La indentación define el alcance y la inclusión.



Python básico II

```
# Listas:
lista = [56, 12, 20]
accum = 0.0
for e in lista:
    accum = accum + e
print “Suma: “, accum
# Funciones:
def sumar(n):
    accum = 0.0
    for I in range(0,n):
        accum = accum + I
    return accum
print “Suma de Gauss: “, sumar(100)
```





Python básico III

- Diccionarios (arreglos asociativos)
- Funcionan como listas, pero permiten indexación genérica
- Ejemplo:

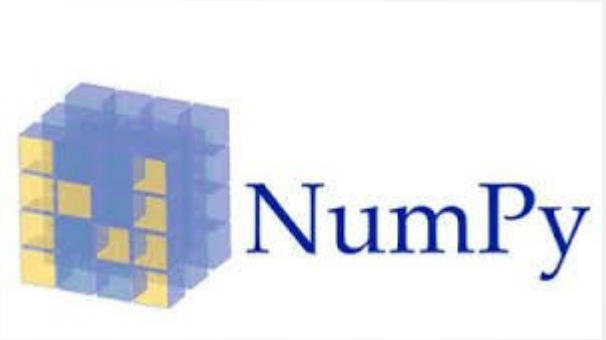
  persona = {“nombre”: “Jose”, “edad”: 43}

  persona[“nombre”]

NumPy

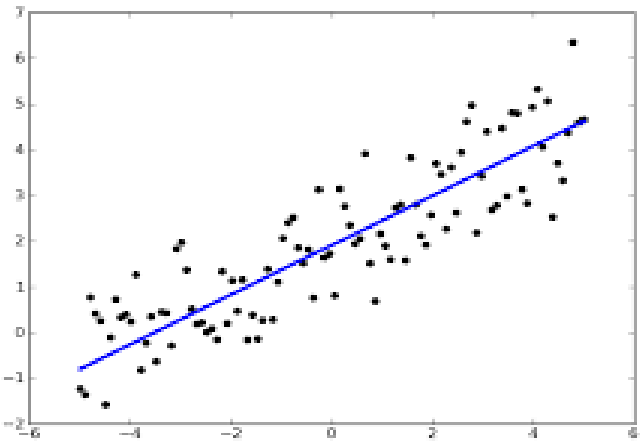
- Librería básica para cómputo científico.
- Incorpora el objeto “Array” en contraposición de la lista para almacenamiento contiguo.
- Soporta también matrices.

```
x = np.array([4,10,1])

A = np.array([[1.5,2,3], [4,5,6]])

b = A.dot(x) ← multiplicación

x = np.linalg.solve(A,b)
```



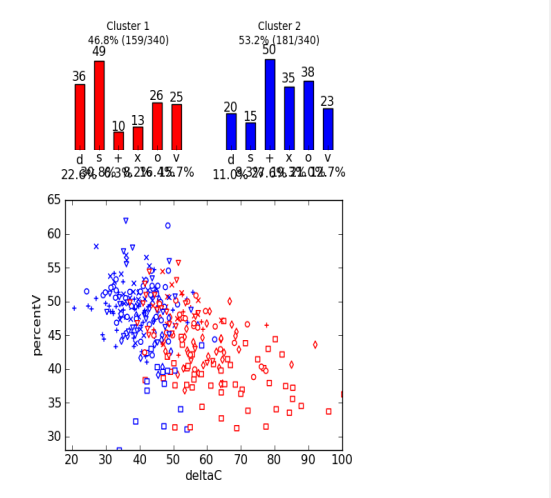
SciPy

- ¡Soporte para MÁS cómputo científico!
- Integración Numérica.
- Optimización.
- Interpolación.
- Álgebra Lineal
- Estadística (básico)
- Procesamiento de señales



Librerías clásicas de Python

- Amplia difusión de librerías:
  - Cómputo numérico (NumPy, SciPy)
  - Análisis de Datos (Pandas)
  - Gráficos (Matplotlib, basemap)
  - Diseño de Interfaces (PyQT)
  - Soporte de bases de datos (PostgreSQL, MySQL, SQLite, MongoDB, etc)
  - Desarrollo Web (Django, Flask, Tornado)
  - Manejo de paquetes (pip)



Herramientas geoespaciales

Librerías:

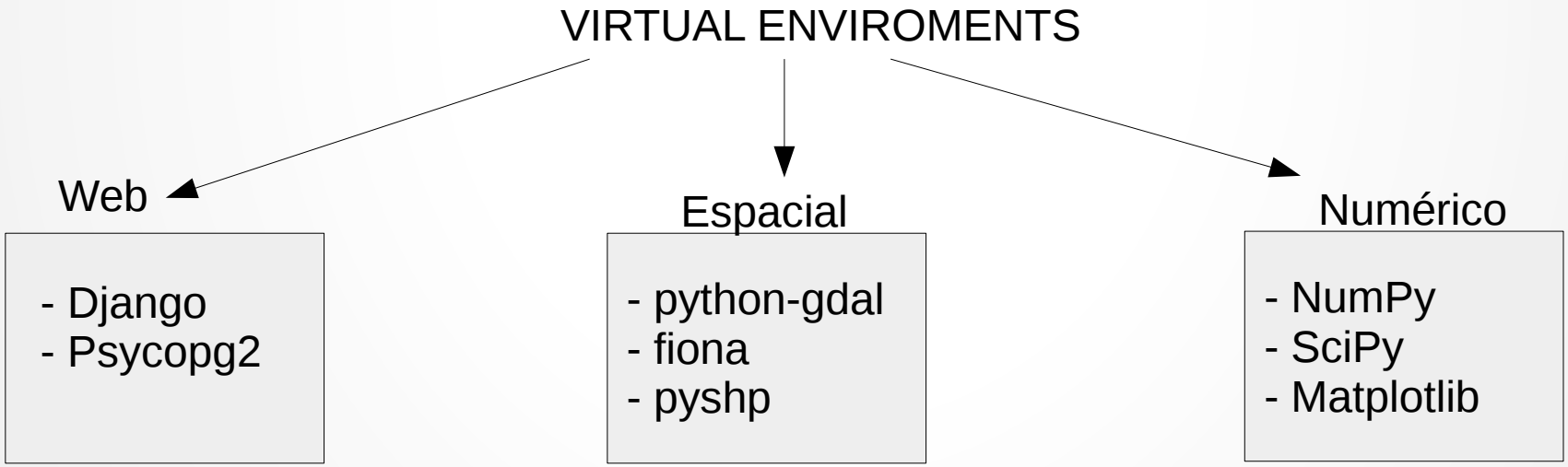
- OGR/GDAL (trabajo con formatos geoespaciales)
- Fiona (alternativa más amigable)
- Pyshp (alternativa vectorial en Python puro)
- Shapely (manipulación de geometrías)

Almacenamiento

- Shapefiles (.dbf + .shp + .shx + extras)
- PostGIS (extensión de PostgreSQL)
- SpatialLite (extensión de SQLite)

Virtual Environments (opcional)

- Forma prolija de mantener separadas las dependencias de cada proyecto.
- Más ligero que implementar una máquina virtual.



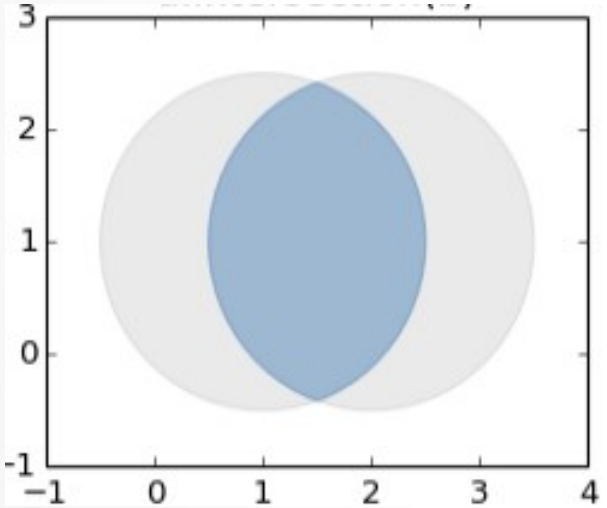
Herramientas: pip + virtualenv

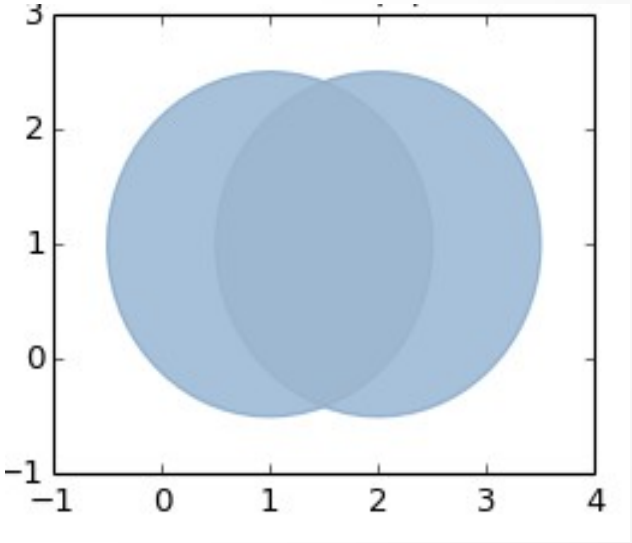
Típicas tareas “Geoespaciales”

- Conversión de formatos:
  - Almacenamiento de shapefile en PostGIS (ogr2ogr)
  - Conversión de PostGIS a shapefile (ogr2ogr)
  - Shapefile a raster (ogr/gdal)
  - Raster (GeoTiff) a Shapefile (ogr/gdal)
  - CSV a Shapefile (pyshp)

Típicas tareas “Geoespaciales”

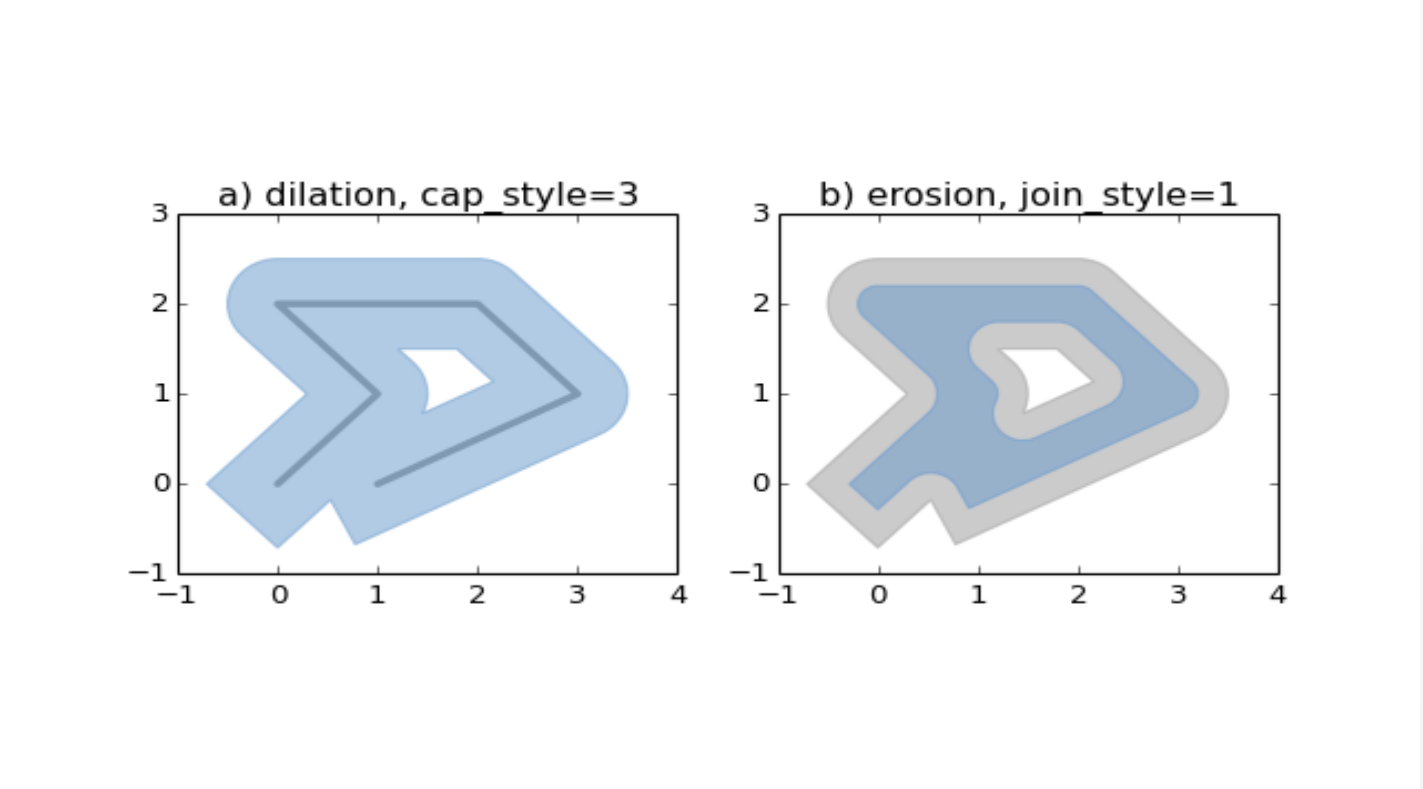
- Análisis en capas vectoriales (shapely):
  - Álgebra de geometrías (unión, diferencia, etc)





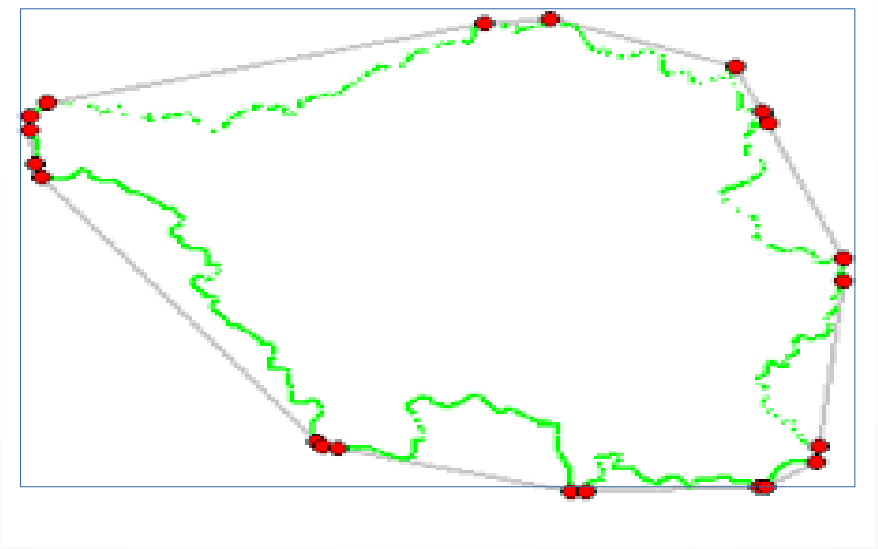
Típicas tareas “Geoespaciales”

- Análisis en capas vectoriales (shapely):
  - “Ensanchado” y “erosión” de geometrías (Buffer)



Típicas tareas “Geoespaciales”

- Análisis en capas vectoriales (shapely):
  - Cápsula convexa y bounding box



Tratamiento de Shapefiles: Pyshp

+ Puramente hecho en Python, código accesible
+ Fácil de instalar (pip)
+ Todo el código de Pyshp entra en un archivo
- No es el más eficiente
- De a ratos puede ser difícil de operar:
  - Complicado para acceder a los campos
  - En algunos casos, también para la geometría

Pyshp

- Apertura de shapefile y lectura básica

```
import shapefile

sf = shapefile.Reader("deptos_Argentina") # DBF, SHX o SHP

sf.fields           # Atributos

[('DeletionFlag', 'C', 1, 0), ['GADMID', 'C', 4, 0], ['NAME_LOCAL', 'C', 54, 0], …

shapeRecs = sf.shapeRecords()

print "Atributos del feature 14: ", shapeRecs[13].record

[None, 'ARG', 'Pilar', ……

print “Atributos del feature 14: “, shapeRecs[13].shape.points

[(-26.396727999999968, -58.39007499999997), (-26.26781499999985,
-58.39347799999996), …..
```

Pyshp

- Generación de un Shapefile puntual

```
import shapefile
w = shapefile.Writer(shapefile.POINTM)
w.field("Nombre", "C")
w.field("Poblacion", "N")
w.record("Pilar",296826)
w.record("La Quiaca", 14753)
w.point(-58, -34)
w.point(-65, -22)
w.save(“Localidades”)
```

Fiona

- Un poco más difícil de instalar, requiere OGR.
- Más amigable para la lectura que Pyshp.
- Tiene un “encaje natural” con Shapely.
- Generar shapefiles es un poco más laborioso que leerlos.

Ejemplo simple de Fiona

```
import fiona

c = fiona.open('municipios/Sudamerica_adm2_simpl.shp', 'r')

geometrias = list(c)

print “Hay “, len(geometrias) , “ geometrias.”
```

Hay 8510 geometrias.

```
print "Geometria 430: ", geometrias[430]
```

Geometria 430: {'geometry': {'type': 'Polygon', 'coordinates': [[(-68.33526699999993, ….

Ejemplo simple de Fiona

```
w = fiona.open('municios/Geometria430', 'w', source_driver = c.driver,
        source_crs = c.crs, # Proyeccion.
        source_schema = c.schema) # Esquema de la geometria y la proyeccion.
rec = geometrias[430]
w.write(rec)
w.close()
```

Así, ponemos en un shapefile nuevo la geometría 430 con todos los campos que tenía antes.

Ejemplo práctico: Rayos

- Tenemos archivos con información de descargas mundiales de rayos/relámpagos.

Fecha,Hora,Latitud,Longitud, Deteccion, Antenas

2015/01/01,00:00:01.049218, 3.0943, 100.6329, 10.8, 5

2015/01/01,00:00:01.168400,-13.4643, 138.9822, 25.3, 7

2015/01/01,00:00:01.316063,-18.2103, 42.2718, 12.0, 6

2015/01/01,00:00:01.780632, 35.2894, 11.6543, 12.0, 6

……..

Aproximadamente 500.000 descargas detectadas por día.

Rayos

- Buscamos asociar a cada municipio la cantidad de rayos que le cayeron,
- y otra magnitud que es la “densidad” de rayos, o sea la cantidad dividida por el área del municipio.
- Habrá que hacer cuentas con inclusión de puntos en polígonos
- Se creará un nuevo shapefile con dos campos nuevos (#rayos y densidad de rayos).

Rayos

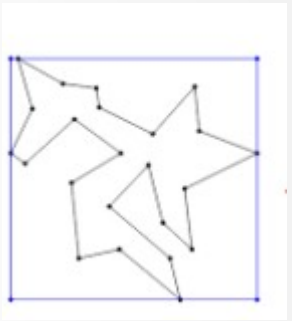
- Fue resuelto usando la estrategia de pyshp.
- El código también tiene la versión hecha en fiona + shpely.
- Consideraciones:
  - Son muchos municipios (8510)
  - Son MUCHOS rayos (488791)
- Es importante tener en cuenta optimizaciones o no terminará efectivamente nunca...

Rayos

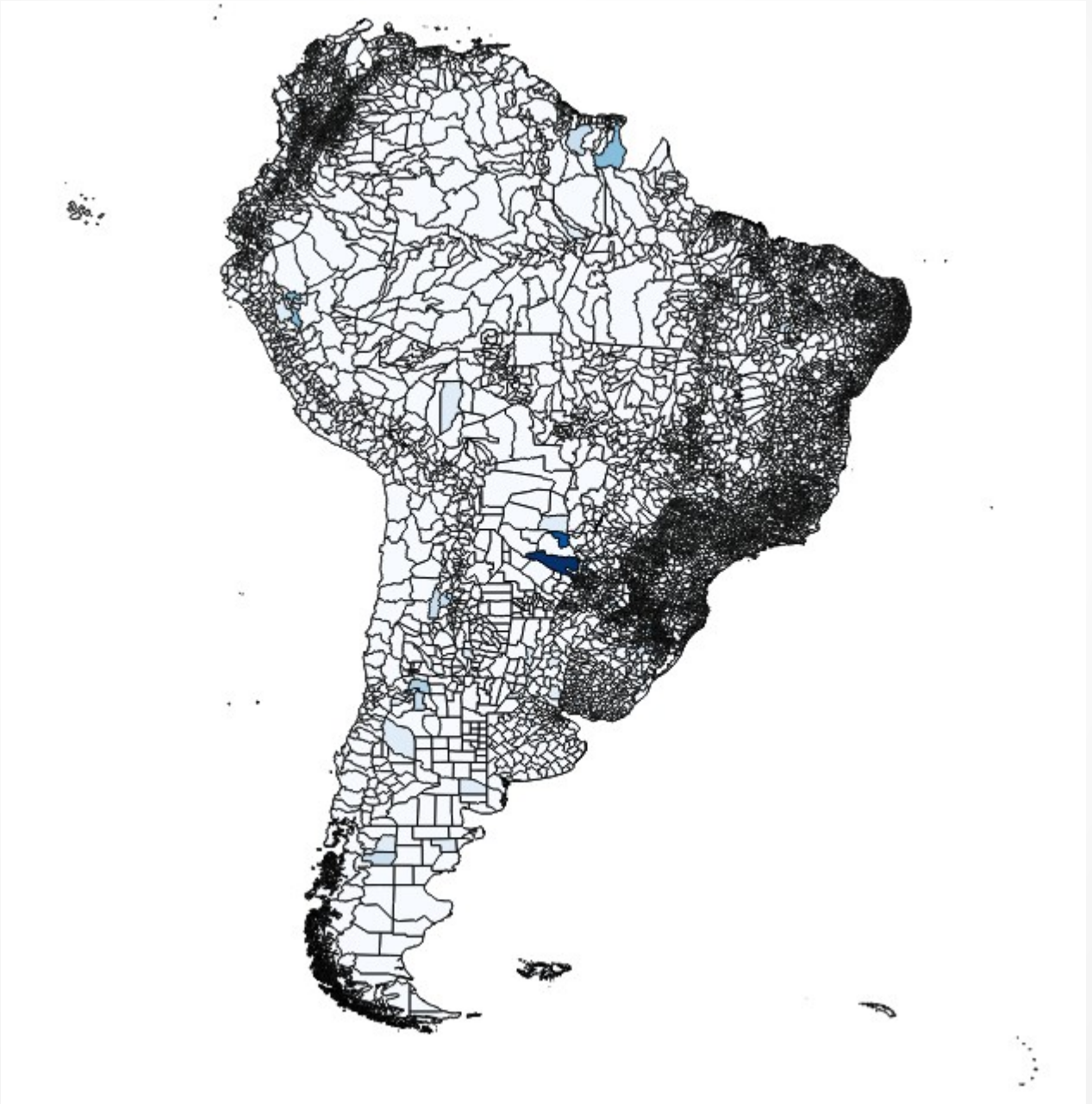
- En la medida de lo posible, precalcular todo lo que se pueda para evitar cómputos redundantes.
- Por ejemplo, precálculo de los bounding boxes ayudó mucho en términos de performance.
- (“Limpieza de datos”) Siempre mirar el archivo en su interior
  - Por ejemplo, están mezclados los países con los municipios...

Estrategia

- Recorrer los municipios del *shapefile* y precalcular el bounding box y almacenarlos en una lista.
- Recorrer cada rayo:
  - Si el rayo está fuera de sudamérica, saltearlo.
  - Recorrer los municipios:
    - Si el rayo está fuera del BB del municipio, ir al próximo municipio (costo bajo).
    - Si el rayo está dentro del BB, verificar si está dentro del municipio (costo alto). Caso afirmativo, incrementar en uno el contador de rayos del municipio.
- Grabar el nuevo *shapefile.*



Resultado de corrida: 1 de enero 2015



¿Qué más hago con el shapefile?

- Tomar el shapefile para procesarlo con:
  - QGIS para visualización y análisis.
  - GeoServer para compartición como capa geográfica web.
  - Por ejemplo, en formato WMS.

¡GRACIAS POR SU ATENCIÓN!

¿Preguntas?