Introducci?n al ecosistema SciPy

January 31, 2018

```
In [1]: # Inline interactive widgets
        %matplotlib nbagg


        ## Cython
        %load_ext Cython
```

1 Ecosistema SciPy

2 Python Scientific Computing Environment

Un típico entorno para cómputo científico en Python se compone de distintas herramientas *dedicadas*. Por ejemplo: - Optimización - Plotting (gráficas) y visualización - Análisis de datos - Shell interactiva - Symbolic mathematics (Cálculo simbólico/cálculo algebraico/álgebra computacional/) - Bases de datos - Extensiones especializadas (scikits)

3 £Quién usa SciPy?

- Autodesk
- Bloomberg
- Amazon
- Microsoft
- Intel

4 scipy.org

5 NumPy

- Provee métodos para arreglos (arrays) y la clase: numerical N-dimensional array
- Soporte básico del ecosistema
- Disintinto de `array`

```
In [2]: import numpy as np
        np.arange(10)
```

```
Out[2]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

- Implementado mayormente en C
- Se puede extender a otras herramientas de análisis númerico
- Rapidez

```
In [3]: a_pythonic = list(range(10000))
        %timeit [v ** 2 for v in a_pythonic]
```

```
100 loops, best of 3: 3.57 ms per loop
```

```
In [4]: b_numpy = np.array(a_pythonic)
        %timeit b_numpy ** 2
```

```
The slowest run took 16.35 times longer than the fastest. This could mean that an intermediate
100000 loops, best of 3: 9.1 ţs per loop
```

5.1 Array Indexing

```
In [5]: x = np.arange(100).reshape(5, 20)

        # Simple indexing
        print(x[2])
```

```
[40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59]
```

```
In [6]: # Slicing
        print(x[2:5])
```

```
[[40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99]]
```

```
In [7]: # Boolean indexing
        print(x[(x % 2) ==0])
```

```
[ 0  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48
 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98]
```

```
In [8]: # Fancy indexing
        print(x[[1, 4, 2]])
```

```
[[20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39]
 [80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99]
 [40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59]]
```

5.2 Polinomios

$$p(x) = 1 + 2x + 3x^2$$

```
In [9]: import numpy as np

        from numpy.polynomial import Polynomial as P

        # p(x) = 1 + 2*x + 3*(x)^2
        polinomio = P([1,2,3])
        print(polinomio)

        # Suma
        print(polinomio + polinomio)

        # Resta
        print(polinomio - polinomio)
```

```
poly([ 1.  2.  3.])
poly([ 2.  4.  6.])
poly([ 0.])
```

Para aprender más: An Introduction to Scientific Python (and a Bit of the Maths Behind It) – NumPy

6 Matplotlib

- Una imagen vale más que mil palabras
- Las tablas no lo son todo
- Herramienta multiplataforma
- Con soporte interactivo

Como crear una gráfica en tres líneas de código:

```
In [10]: import matplotlib.pyplot as plt
         plt.plot([1, 5, 3])
         plt.show()
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>
```

- Amigable y no obstrusivo
- Control de detalles
- Variedad de estilos para gráficas
- Interactive Applications using Matplotlib

7 SciPy

- Un paso adelante de algoritmos básicos de Matemáticas (`import math`)
- Contiene algoritmos para diferentes dominios
- `constants`
- `cluster`
- `fftpack`
- `integrate`

```
In [14]: from scipy import constants
         import pint
         ureg = pint.UnitRegistry()
         m = 10 * ureg.kg
         c = constants.c * ureg.meters / ureg.second
         E = m*c**2
         print(E)
```

```
8.987551787368177e+17 kilogram * meter ** 2 / second ** 2
```

- `interpolate`

```
In [15]: from scipy import interpolate
         x, y = np.linspace(-5, 5, 25), np.linspace(-5, 5, 25)
         xx, yy = np.meshgrid(x, y)
         z = np.sin(xx**2+yy**2)
         f_rect = interpolate.RectBivariateSpline(x, y, z)

         xnew, ynew = np.linspace(-4.5, 4.5, 1000), np.linspace(-4.5, 4.5, 1000)
         znew = f_rect(xnew, ynew)
```

```
In [16]: fig, axes = plt.subplots(1, 2)
         axes[0].imshow(z, vmin=-1, vmax=1)
         axes[1].imshow(znew, vmin=-1, vmax=1)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
Out[16]: <matplotlib.image.AxesImage at 0x11a738588>
```

- `io` - Soporte para formatos especificos usandos en investigación
- matlab, fortran, netcdf3, arff, IDL, wav
- `ndimage` - Filtros, transformaciones, convoluciones
- `measurements` (estadística en pixeles con una categoría)

- morphology (erosión, dilatación)
- Para funciones avanzadas: scikit-image

p.e.

```
In [17]: n_stores = 5
         n_customers = 500
         stores = np.random.random((n_stores, 2)) * 400
         customers = np.random.random((n_customers, 2)) * 400
         threshold = 200
```

```
In [18]: d = np.hypot(stores[:, 0][:, np.newaxis] -
                      customers[:, 0][np.newaxis, :],
                      stores[:, 1][:, np.newaxis] -
                      customers[:, 1][np.newaxis, :])
         indexes = np.argmin(d, axis=0)
         d = d[indexes, np.arange(len(indexes))]
         d[d > threshold] = np.inf
         indexes[np.isinf(d)] = len(indexes)
         rev_indexes = [np.where(indexes == store_i)
                        for store_i in range(len(stores))]
```

```
In [19]: fig, ax = plt.subplots()
         for inds, in rev_indexes:
             ax.plot(customers[inds, 0], customers[inds, 1], '.')
         ax.plot(stores[:, 0], stores[:, 1], '*y', markersize=25,
                 mec='k', mew=3)
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


Out[19]: [<matplotlib.lines.Line2D at 0x11b14c5f8>]
```

- spatial

```
In [20]: from scipy.spatial import cKDTree as KDTree
```

```
In [21]: %%timeit
         store_tree = KDTree(stores)
         d, indexes = store_tree.query(customers,
                                       distance_upper_bound=threshold)
         rev_indexes = [np.where(indexes == store_i)
                        for store_i in range(len(stores))]
```

```
The slowest run took 18.31 times longer than the fastest. This could mean that an intermediate
1000 loops, best of 3: 382 µs per loop
```

- `stats` - Distribuciones estadsticas
- Generación de datos aleatorios en base a una distribución
- Para más funciones de Estadistíca, tenemos el paquete statsmodels
- patsy - Describing statistical models in Python

- `special`
- Funciones Lambda, pero no `lambda`

8 Resto del ecosistema SciPy

8.1 Visualización

- bokeh - seaborn - cartopy - ggplot - descartes - mayavi - vispy & glumpy

```
In [22]: import numpy as np
         import pandas as pd

         from bokeh.plotting import figure, show, output_file
         from bokeh.palettes import brewer

         N = 20
         categories = ['y' + str(x) for x in range(10)]
         data = {}
         data['x'] = np.arange(N)
         for cat in categories:
             data[cat] = np.random.randint(10, 100, size=N)

         df = pd.DataFrame(data)
         df = df.set_index(['x'])

         def stacked(df, categories):
             areas = dict()
             last = np.zeros(len(df[categories[0]]))
             for cat in categories:
                 next = last + df[cat]
                 areas[cat] = np.hstack((last[::-1], next))
                 last = next
             return areas

         areas = stacked(df, categories)

         colors = brewer["Spectral"][len(areas)]

         x2 = np.hstack((data['x'][::-1], data['x']))

         p = figure(x_range=(0, 19), y_range=(0, 800))
```

```
p.grid.minor_grid_line_color = '#eeeeee'

p.patches([x2] * len(areas), [areas[cat] for cat in categories],
          color=colors, alpha=0.8, line_color=None)

output_file("brewer.html", title="brewer.py example")

show(p)
```

8.2 PyData y la evolución de los arrays

- pandas
- xarray
- pytables
- ibis
- odo
- blaze
- pint

8.3 Cython / python with rockets

- cython
- numba
- numexpr
- dask
- theano

```
In [23]: def pyprimes(kmax):
             p = [0] * 1000
             result = []
             if kmax > 1000:
                 kmax = 1000
             k = 0
             n = 2
             while k < kmax:
                 i = 0
                 while i < k and n % p[i] != 0:
                     i = i + 1
                 if i == k:
                     p[k] = n
                     k = k + 1
                     result.append(n)
                 n = n + 1
             return result
```

```
In [24]: %%cython
         def cprimes(int kmax):
             cdef int n, k, i
```

```
        cdef int p[1000]
        result = []
        if kmax > 1000:
            kmax = 1000
        k = 0
        n = 2
        while k < kmax:
            i = 0
            while i < k and n % p[i] != 0:
                i = i + 1
            if i == k:
                p[k] = n
                k = k + 1
                result.append(n)
            n = n + 1
        return result
```

```
In [25]: %timeit pyprimes(5000)
         %timeit cprimes(5000)

10 loops, best of 3: 82.3 ms per loop
100 loops, best of 3: 2.21 ms per loop
```

8.4 Symbolic math

- sympy
- sage
- patsy

8.5 Geographic Processing

- GDAL/OGR (`osgeo.ogr` & `osgeo.osr`)
- fiona
- geopandas
- shapely
- pyproj
- geopy

```
In [28]: from shapely.geometry import Point
         a = Point(1, 1).buffer(1.5)
         b = Point(2, 1).buffer(1.5)
         c = a.intersection(b)

         print(c.area)
         print(c.length)
         print(c.bounds)
```

```
4.1161985901396605
7.380229156672531
(0.5, -0.4123059204384314, 2.5, 2.412305920438431)
```

```
In [32]: from descartes import PolygonPatch
         fig, ax = plt.subplots()
         ax.set_xlim(-1, 4); ax.set_ylim(-1, 3)
         for p, color in zip([a, b, c], 'rcy'):
             ax.add_patch(PolygonPatch(p, fc=color))
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

8.6 Formatos de archivo para Ciencia/Investigación

- netCDF4
- pytables
- h5py
- PyNIO
- GDAL (osgeo.gdal)
- rasterio

9 Scikits / Dominio especial

- scikit-image - scikit-learn - astropy - statsmodels - metpy - pyart - Lista Completa

9.1 Documentación / Guías de estudio

- Documentation for core SciPy Stack projects
- Lectures on scientific computing with Python, as IPython notebooks.
- Scipy Lecture Notes. One document to learn numerics, science, and data with Python
- A Crash Course in Python for Scientists

9.2 Libros

- Python for Data Analysis: Wes Mckinney
- Cython - A Guide for Python Programmers: Kurt Smith
- Effective Computation in Physics - Field Guide to Research with Python: Anthony Scopatz & Katy Huff
- Mastering Matplotlib: Duncan McGreggor

9.3

- Vistazo general del ecosistema
- Omitimos Bio y visualización 3D
- Scientific Computing with Python | Austin, Texas July 10-16, 2017