Python crash course

02/21/2022

Rodrigo FO Pena

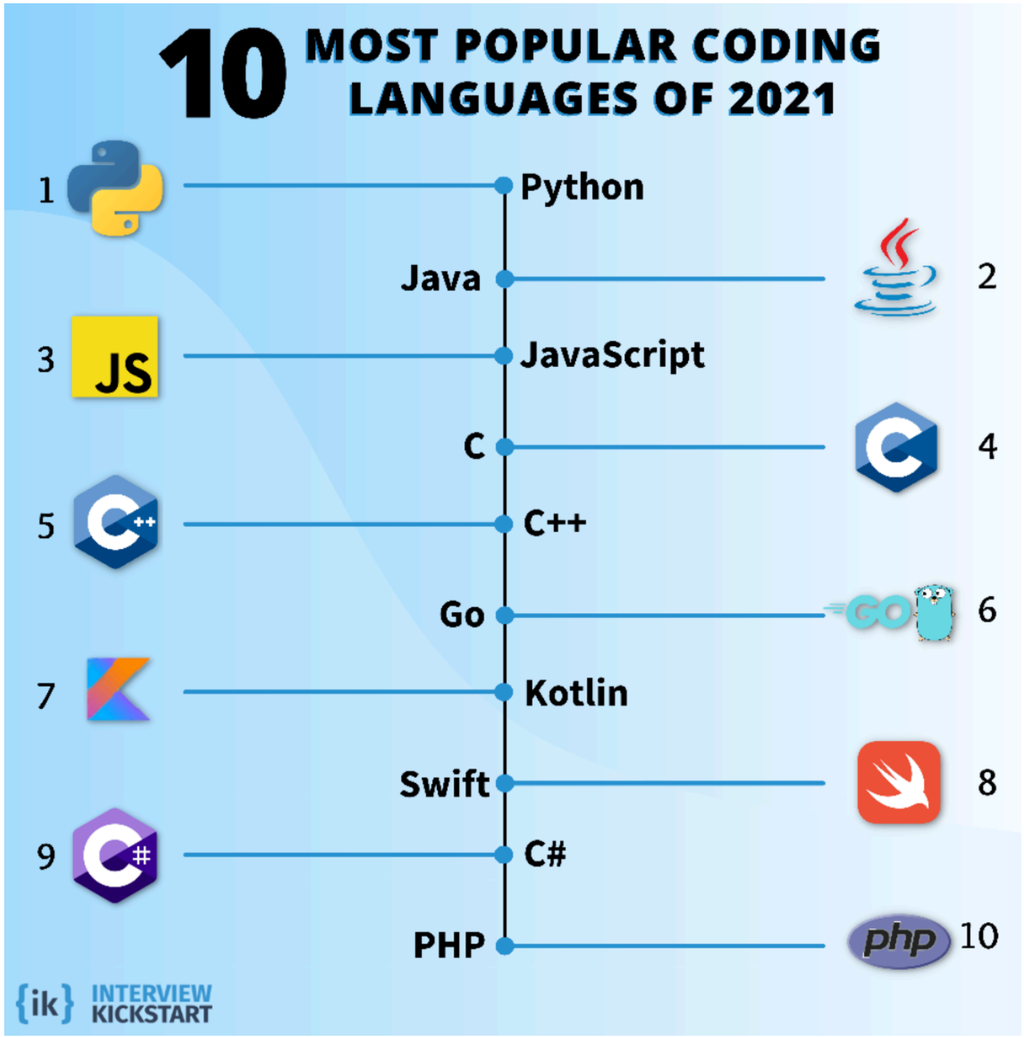
e-mail: pena@njit.edu

BioDatanamics Lab (H. G. Rotstein), Federated Department of Biological Sciences

New Jersey Institute of Technology & Rutgers University

Why python ?

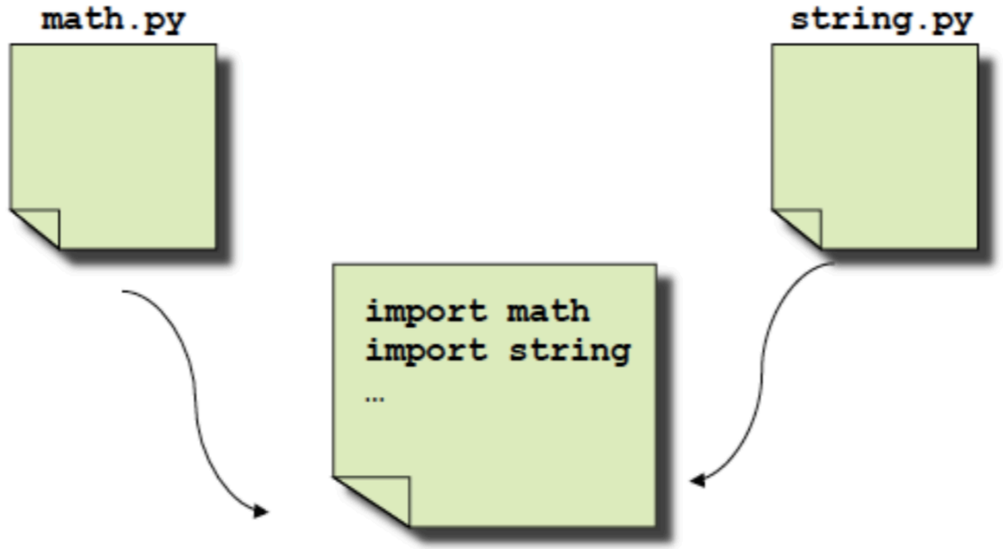
- Python is an **interpreted** object oriented programming language
- **Extensive documentation** and **huge community**
- Modularity with nice modules for scientific computing/data analysis/visualization
  - Data science
  - Machine learning
  - General software development



Modules

A file containing Python definitions and statements

- Modules can be “imported”
- Module file name must end in .py
- Used to divide code between files



”Hello, World”

- C

```
#include <stdio.h>

int main(int argc, char **argv)
{
    print(”Hello, World!\n”);
}
```

- Java

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println(”Hello, World!”);
    }
}
```

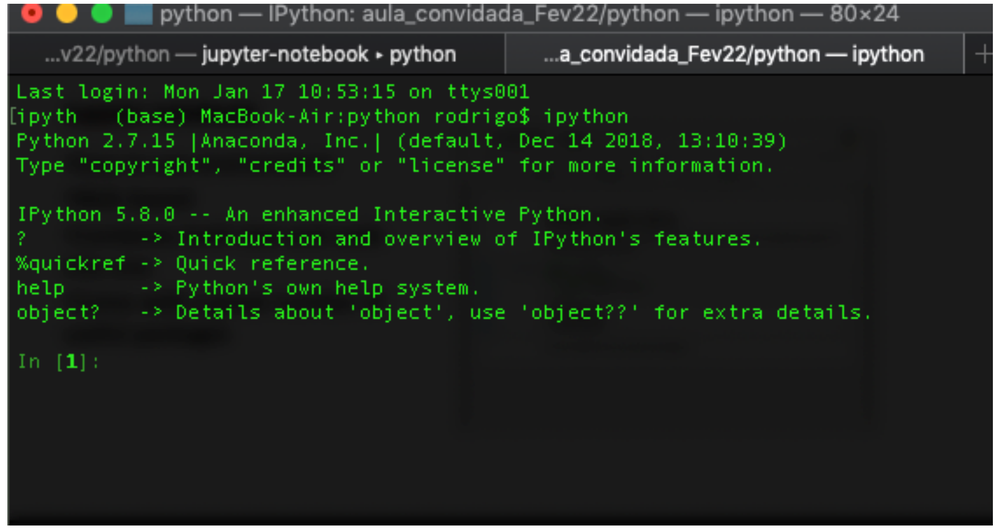
- Python

```
print ”Hello, World!”
```

IPython

Python can be run interactively

Used extensively in research



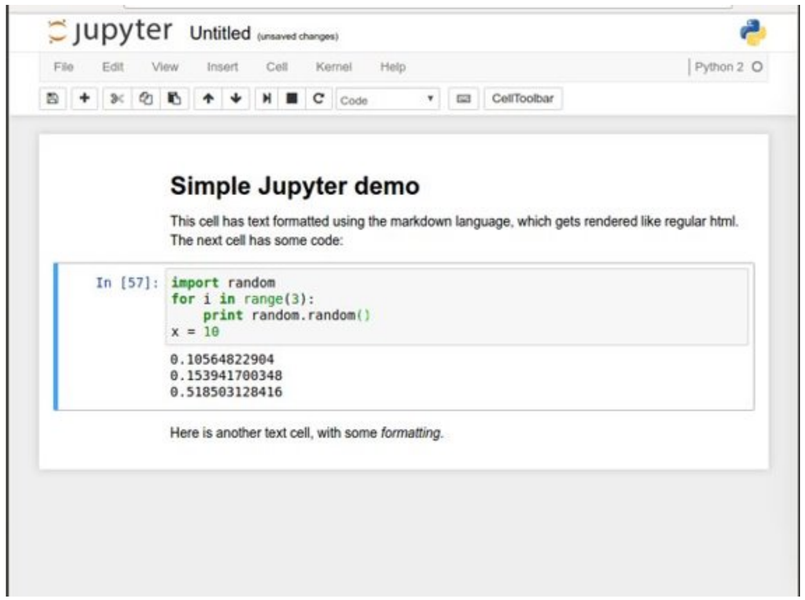
Python scripts

What if we want to run more than a few lines of code?

Then we must write text files in .py

Jupyter notebook

- Easy to use environment
- Web-based
- Combines both text and code into one
- Come with a great number of useful packages



Let's code!
- python_basics

Python libraries

NumPy

- **Numpy**

Support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays;

matplotlib

- **Matplotlib**

Plotting library designed to closely resemble that of MATLAB;

pandas

- **Pandas**

Data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series;

SciPy

- **Scipy**

Used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Let’s code!
- python_libraries

Solving ODEs in python

Using packages

e.g. scipy

```
from scipy.integrate import odeint
```

odeint has a collection of numerical methods in C++

Self-implementation

e.g. Euler Method

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0.$$

$$y_{n+1} = y_n + hf(t_n, y_n).$$

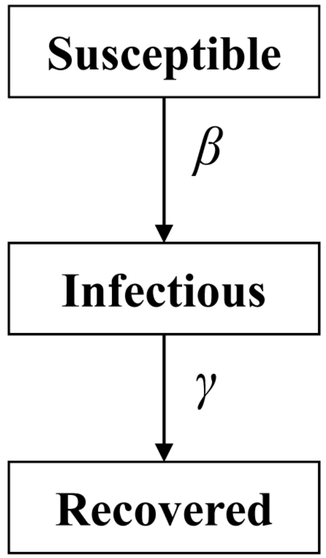
Let's code the predator-prey equations (Lotka-Volterra)

$$\frac{dx}{dt} = x(a - by) \qquad \text{[rabbits]}$$

$$\frac{dy}{dt} = -y(c - dx) \qquad \text{[foxes]}$$

The SIR epidemic model

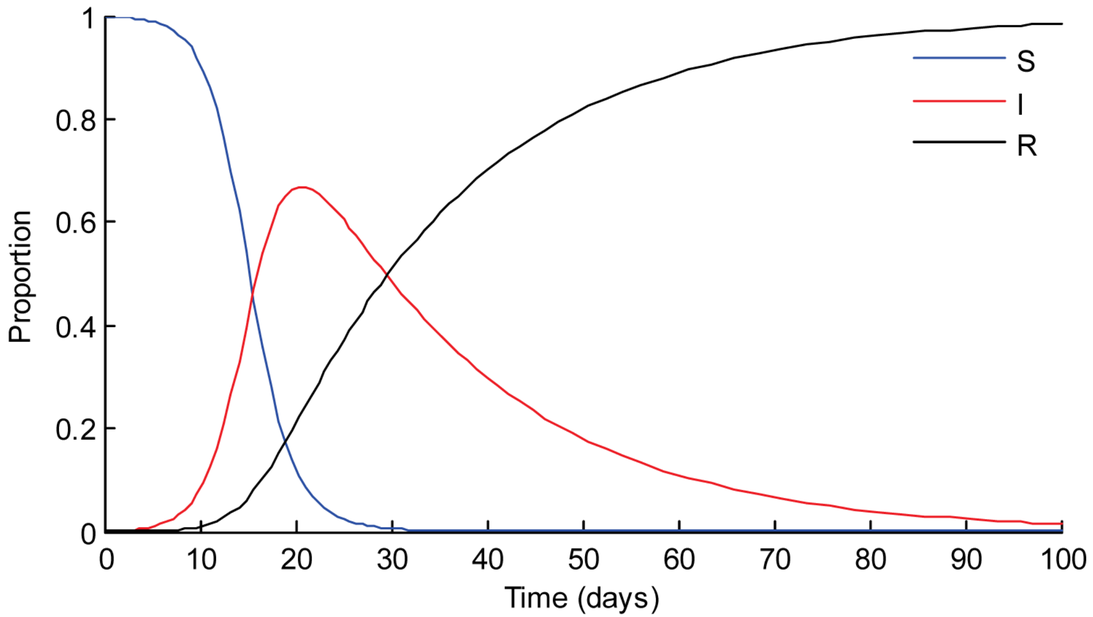
- $S(t)$ are those susceptible to the disease;
- $I(t)$ are the infected individuals;
- $R(t)$ are those recovered.



$$\frac{dS}{dt} = -\beta SI$$

$$\frac{dI}{dt} = \beta SI - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$



doi:10.1371/journal.pntd.0000761.g001

Project

- Implement in Python the SIR model; Use the tools you learned to create your code.
- Look for COVID19 curves of infection (use google, nyt, or any other database).
- Adjust the parameters of your model to reproduce as close as possible the curve. You need to choose the curve of a city (or country) and a period in time.
- Discuss your findings. Discuss the variables and parameters. Discuss how the model could be improved by external factors acting on the COVID19 pandemic. What could have been done differently according to your model?