Programming

and python/jupyter review

C. Alex Simpkins Jr., Ph.D
RDPRobotics LLC,
UC San Diego, Department of Cognitive Science
rdprobotics@gmail.com
csimpkinsjr@ucsd.edu

Lectures :http://casimpkinsjr.radiantdolphinpress.com/pages/cogs108_ss1_23/index.html

Course announcements

- Zoom
- Discussion Lab: for finding group mates, or tonight we will start assigning, so you can get going on the project
- My OH: by appt. thus far
- https://github.com/drsimpkins-teaching/cogs138/tree/main/lectures/week1

Plan for the lecture

- Programming - what is a program?
- Why python? What is python?
- What packages are needed for data science and why? What do they need to do?
- NumPy
- Pandas
- MatPlotLib
- Seaborn
- SciKitLearn

So what is a program?

What is a program?

- Generally a **program** is a **set of instructions** the programmer defines for a device or entity (usually a computer but not always) to follow
- Regarding computers-> programmer writes a set of instructions (“program”) that tells the computer to perform a set of operations
- When the program is executed, the instructions are carried out
  - How does this work (big picture)?
    - Relates to the speed discussion we are about to get into…

Programming languages

- **Low level** machine language (binary/hex) provides instructions for the processor to execute
- **Mid-level** language is called ‘assembly’ language
- **High-level** languages such as C, C++, Fortran, BASIC, etc
- **Very high-level** languages (‘scripting’ languages) such as Python, MATLAB

Low level programming languages

- Machine language (binary/hex) provides instructions for the processor to execute
- Computer processors operate with binary (1’s and 0’s)
- Not easy for humans to read, write, program
- Error prone, extremely tedious

Mid-level programming languages

- ‘assembly’ language and is more human-readable but still tedious to work with
- Blazing fast, minimal footprint
- Useful for simple tasks operating on tiny processors or that must be very optimized
- Not very re-usable code - tied to the processor and hardware, very specific operations

High level programming languages

- Such as C, C++, Fortran, etc
- Very human readable, much less work to code - a balance between ease of development and speed of execution
- A compiler takes the high level code and converts it into low level machine language
- Portability and reusability of code much improved (functions, classes, libraries)
- Embedded systems, highly efficient numerical computations live here

Very high level programming language

- Scripting languages
- Python, MATLAB
- Interpreted at runtime (when you run the script), uses existing facilities from a compiled language like C
- Much much less effort - one line in Python might be the length of a book in binary
- This is why they tend to be slower - another layer

Seems fairly straightforward right?

Have to be careful what you tell it to do!

- NASA example
  - 1999 Climate Orbiter spacecraft,
  - 286 days to Mars.
  - Miscalculations due to a conversion error sent the craft off course gradually,
  - so this $125M piece of technology smashed itself into the surface of Mars.

Have to be careful what you tell it to do!

- NASA example
- **A program *must* have reasonable inputs and outputs**
- **Just because a function works, it does *not* mean nothing can go wrong!**

Why write a program?

- Many reasons you may want to write a program
- This can be anything:
  - Processing data
  - Making a robot walk
  - Controlling traffic lights to meter traffic during rush hour in an optimal way
  - Displaying a photo, etc

What is python?

- A high-level (sometimes called ‘very high level) programming language (scripting/interpreted)
- Emphasizes readability
- Highly extensible via ‘modules’
- First released in 1991, written by Guido van Rossum

Guido van Rossum

source: https://en.wikipedia.org/wiki/Python_(programming_language)#/media/File:Guido_van_Rossum_OSCON_2006_cropped.png

What is python?

- Simple is better than complex philosophy
- Not speed critical
- Links:
  - *Main page: http://www.python.org*
  - *Documentation: https://docs.python.org/3/*
  - *Tutorials: https://docs.python.org/3/tutorial/index.html*

Guido van Rossum

source: https://en.wikipedia.org/wiki/Python_(programming_language)#/media/File:Guido_van_Rossum_OSCON_2006_cropped.png

Why python?

- It’s free
- Tremendous library support
- Easy interpreted language, quick for prototyping
- Highly optimized computational libraries
- Cross platform/portability
- Strong user community for answering questions/knowledgebase

When python?

- Web app development
- Data science
- Scripting
- Database programming
- Quick prototyping

Python’s extensibility

- The extensible core of python is where the true power lies
- Python is great, but without expansion it is not useful for scientific computing - not originally designed for numerical computing
  - Lacks matrix and linear algebra operations
  - No scientific visualization in 2d and 3d
  - Slow, memory intensive

Modules to the rescue!

- You will learn and gain experience with:
  - ***NumPy***
  - ***Pandas***
  - ***Matplotlib***
  - ***Seaborn***
  - ***SciKitLearn***
- And learn how to acquire new module skills as needed

Modules to the rescue!

- **NumPy** - fast numerical computation and multi-dimensional array operations
  - https://numpy.org
- **Pandas** - “a fast, powerful, flexible and easy to use open source data analysis and manipulation tool”
  - https://pandas.pydata.org
- **MatPlotLib** - 2d and 3d visualization and plotting functions, very customizable, can create animated and interactive figures
  - https://matplotlib.org
- **Seaborn** - package built on top of MatPlotLib to produce beautiful scientific visualizations with less ‘tuning’ and includes some simple analysis
  - https://seaborn.pydata.org
- **scikit-learn** - “Simple and efficient tools for predictive data analysis”
  - https://scikit-learn.org/stable/

Modules to the rescue!

- **NumPy** is the successor to two other numerical packages ***Numeric*** and ***Numarray***
  - https://numpy.org
  - https://numpy.org/doc/stable/
  - Travis Oliphant ported features from Numarray to Numeric, releasing V1.0 in 2006, which was part of ***SciPy***
  - Separated in order to allow for smaller installs if you just want numerical computation on array objects

Speed and relevance to data science

- NumPy provides similar array operations to MATLAB
    - Both are interpreted
    - Anybody who has coded in MATLAB or Python (with NumPy) knows it's powerful, but not all realize it's not just about convenience
    - The functions are highly optimized
        - Array operations execute orders of magnitude faster than coding with a loop or alternative methods
    - ***Why does this matter for Data Science?***

How we will bring these in…

- We will introduce each module as we go through the course
- Remember though you'll get proficient with each module, this course is a beginning not an end
- You are likely already familiar with some of these modules, but let's expand that knowledge

Why Jupyter Notebooks

- Mixed media is excellent for data exploration and communication
- Don’t have to write a separate program from your notes, results, etc
- Easy to experiment in nonlinear and compartmentalized ways

JN use cases

- Prototyping
- Data ingestion
- Exploratory data analysis
- Feature engineering
- Model comparison
- Final model

An overview of python and jupyter

- To the notebook!