Python in astronomy
+ Monte-Carlo techniques

Michael Ireland (RSAA)

Interpreted languages in astronomy

- For complex data analysis, we want to:

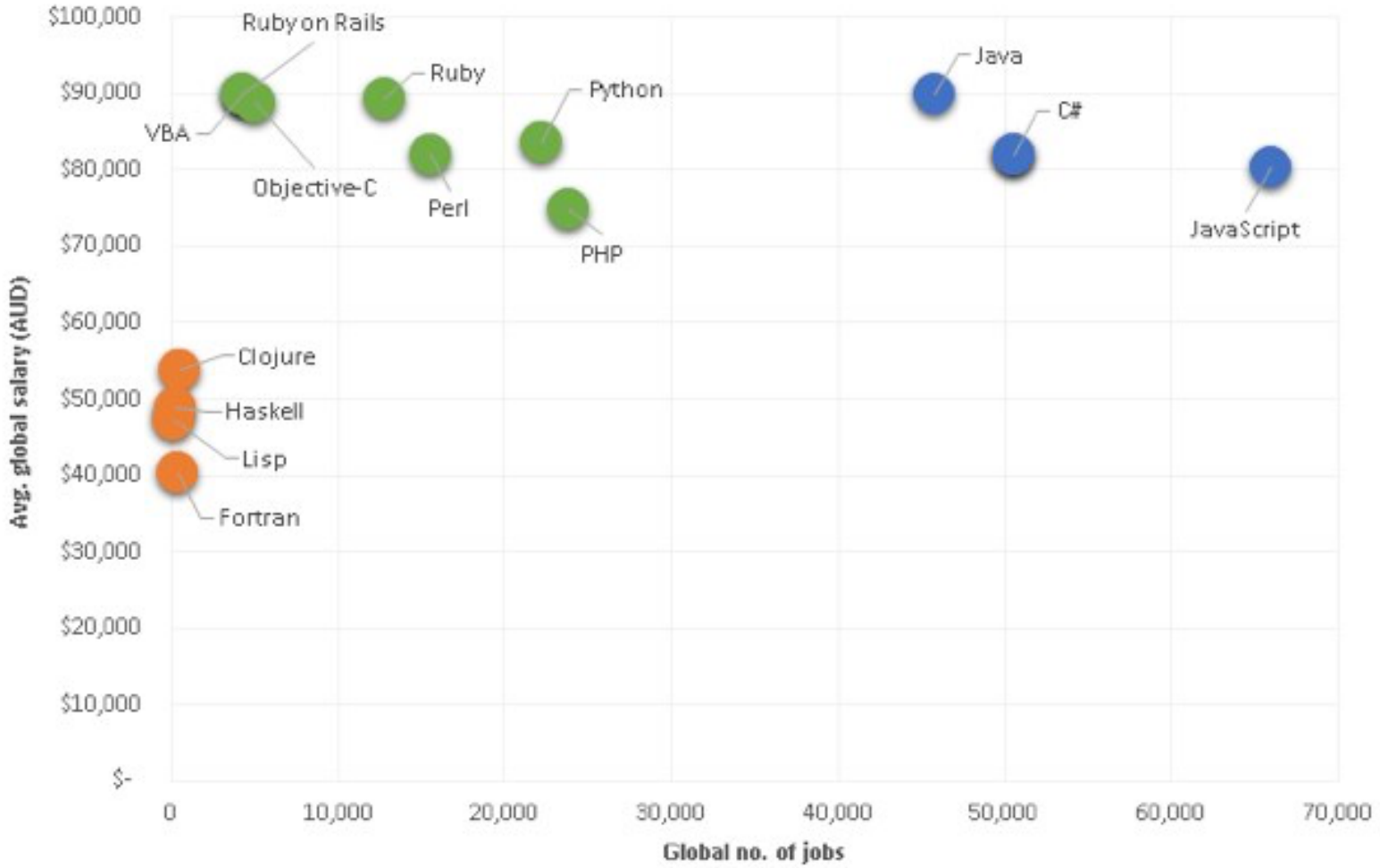
1. Interact with our data – an interpreted language helps.
2. Be moderately fast, and have an option to be very fast.
3. Have the ability to write and test complex code more quickly than our competitors.
4. Have great astronomy libraries.
5. Gain skills relevant for employment.

What are the options and their popularity?

- Yorick (unknown language with some die-hard fans)
- IDL (ranked between #51 and #100)*.
- Julia (ranked #47 and rising)
- R (ranked #18)
- MATLAB (ranked #15)
- Python (ranked #5)

* Rankings from http://www.tiobe.com/tiobe-index/.

Python is the right combination of career-relevant skills for students/postdocs, speed and ease of use.



https://gooroo.io/GoorooTHINK/Article/16225/

Python – the good and bad

- Python is free, with the first version with community support python 2.0 (2000).
- Python is *minimalist* in many aspects of the language. There are only 34 keywords in the global namespace (many of which you'd never use).
- Python has *lightweight object-oriented* programming. Lightweight because there is no explicit C++ style declarations (no *public, private, virtual, overloaded functions, pointers…)*
- Python is designed from the ground up for powerful data structures: lists, dictionaries and tuples (and sets).
- However, Python requires external *modules* to have more than basic functionality. If only all useful modules were packaged and available in a neat way…

Demo python here

Use a Python *distribution* or package manager

- Most popular for Mac and Windows is probably anaconda (https://www.continuum.io/downloads)
- Anaconda comes with *astropy*, which is a great general purpose astronomy package (V1.0 last year).
- Under Ubuntu, you can just use apt-get.
- Most additional packages can be installed with *pip,* e.g. *pip install sick* (Andy Casey's spectroscopic inference crank)
- Given that python is open-source, many packages are on github or other public repositories.

Most Important Packages

- Numpy: From 2006*, a set of structures and routines designed to make python roughly as powerful as Matlab.
- Scipy: Mostly wrappers for powerful libraries such as LAPACK, plus other bits and pieces. From 2001, but still version “0.17”.
- Matplotlib: A great 2D plotting package.
- Astropy: version 1.2.1 – only a few years old in a useable form.

All of these are on github, and you can also contribute to python itself if you are keen (in C)

- numpy example from the command prompt.
- “vectorised” code – style and speed
- N-body example code using the *semi-implicit Euler’s method:*
  - Update velocity to timestep (n+1)
  - Use velocity at timestep (n+1) to update position a timestep (n)
  - Conserves energy.

Exercise (non-assessed)

- If you are not a python expert yet:
  - Learn python for fun, e.g. https://learnpythonthehardway.org/book/
  - Play with the N-body code. Try a different algorithm, better plotting…
  - Read in part of Gaia DR1. Download and plot distributions of your favourite type of star, with:

```
from astropy.table import Table
```

…or…

```
from astroquery.vizier import Vizier
```

- Small assignment given out on Thursday.