QuantEcon–RSE Intensive Course on Computational Modeling

Introduction

December 2019

Introduction

Personel

- Fedor Iskhakov (RSE, lecturer)
- Matt McKay (RSE, lecturer + technical assistance)
- Nicole Millar (RSE, organization and administration)
- Varun Satish (U Syd, lecturer + teaching assistant)
- John Stachurski (RSE, co-organizer)
- Sebastian Wende (Treasury, co-organizer)

Thanks

- Alfred P. Sloan Foundation, Research School of Economics

Resources and Timeline

Wifi TBA

Course homepage

- `https://github.com/QuantEcon/summer_course_2019`
    - please sign up to GitHub and watch this repo

Timeline

- Morning and late afternoon lectures — see course homepage
- Early afternoons are for exercises, best done jointly
- TA will be available during that time

Prereqs / Aims / Outcomes

Assumptions:

- econ/computer/maths/stats literate
- some basic familiarity with Python

Aims:

- Overview of scientific computing and Python
- Learn some elements of advanced economic modeling
- Show how to solve such models with Python
- Interact with and learn from policy professionals
- Resources for further study

Background — Language Types

Proprietary

- Excel
- MATLAB
- STATA, etc.

Open Source

- Python
- Julia
- R

closed and stable vs open and fast moving

Background — Language Types

Low level

- C/C++
- Fortran
- Java

High level

- Python
- Ruby
- Javascript

Low level languages give us fine grained control

Example. $1 + 1$ in assembly

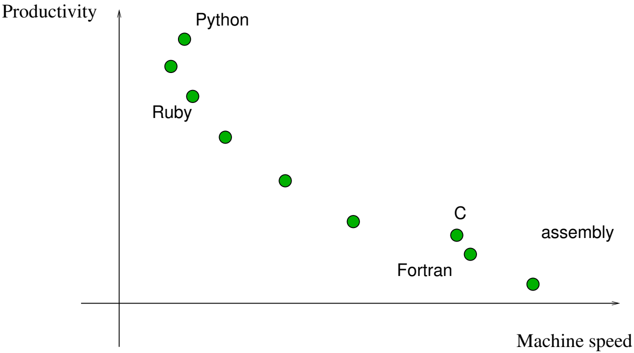
```
pushq    %rbp
movq     %rsp, %rbp
movl     $1, -12(%rbp)
movl     $1, -8(%rbp)
movl     -12(%rbp), %edx
movl     -8(%rbp), %eax
addl     %edx, %eax
movl     %eax, -4(%rbp)
movl     -4(%rbp), %eax
popq     %rbp
```

High level languages give us abstraction, automation, etc.

Example. Reading from a file in Python

```
data_file = open("data.txt")
for line in data_file:
    print(line.capitalize())
data_file.close()
```

Trade-Offs

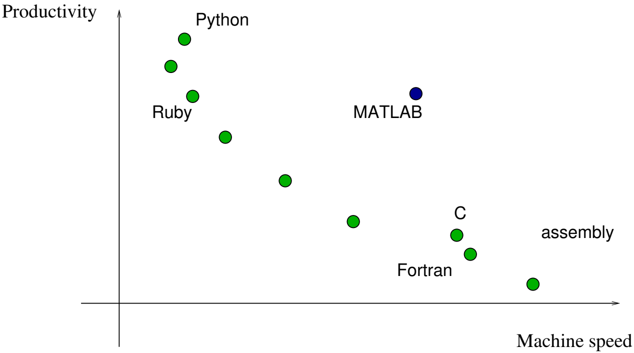

Productivity
Python
Ruby
C
assembly
Fortran
Machine speed

But what about scientific computing?

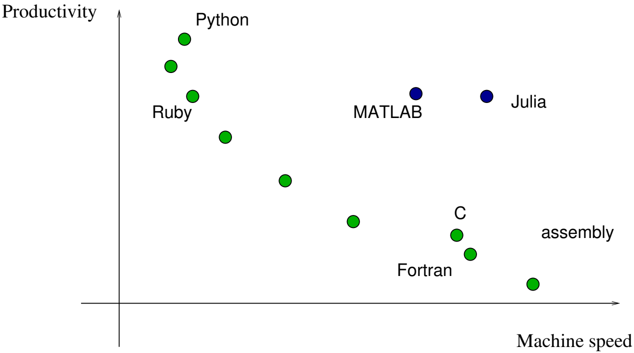
Requirements

- Productive — easy to read, write, debug, explore
- Fast computations

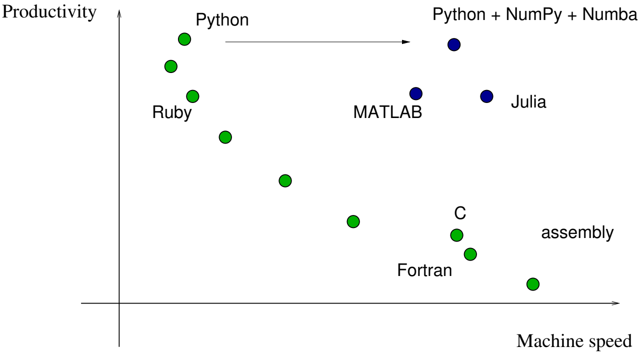
Trade-Offs


Productivity
Python
Ruby
MATLAB
C
assembly
Fortran
Machine speed

Trade-Offs

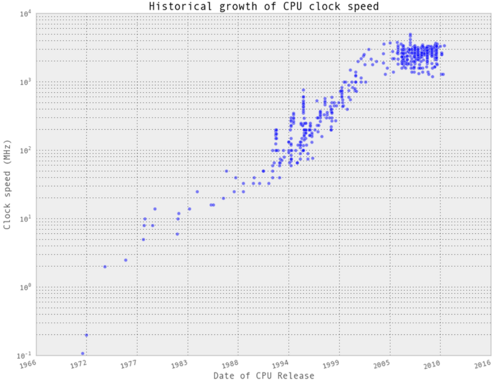

Productivity
Python
Ruby
MATLAB
Julia
C
assembly
Fortran
Machine speed

Trade-Offs

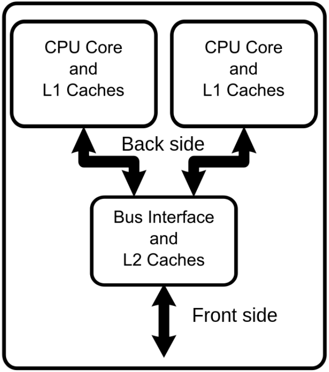

Productivity
Python
Python + NumPy + Numba
Ruby
MATLAB
Julia
C
assembly
Fortran
Machine speed

Trend 1: Parallelization

CPU frequency (clock speed) growth is slowing

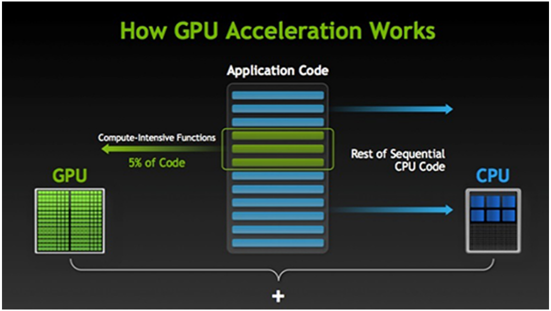


Chip makers have responded by developing multi-core processors



Source: Wikipedia

GPUs / ASICs are also becoming increasingly important


How GPU Acceleration Works
Application Code
Compute-Intensive Functions
5% of Code
GPU
Rest of Sequential CPU Code
CPU
+

Applications: machine learning, deep learning, etc.

Trend 2: Distributed Computing

Advantages:

- run code on big machines we don’t have to buy
- customized execution environments
- circumvent annoying internal IT departments

Options:

- University machines
- AWS
- Google Colab, etc.

Why Python?

- Easy to learn, well designed
- Massive scientific ecosystem
- Open source
- Huge demand for tech-savvy Python programmers

Scientific Computing

Python has strong tools in vectorization / JIT compilation / parallelization / visualization / etc.

Examples:

- SciPy, NumPy, Matplotlib, pandas
- Numba (JIT compilation, multithreading)
- Tensorflow, PyTorch (machine learning, AI)
- JAX, NetworkX, etc., etc.

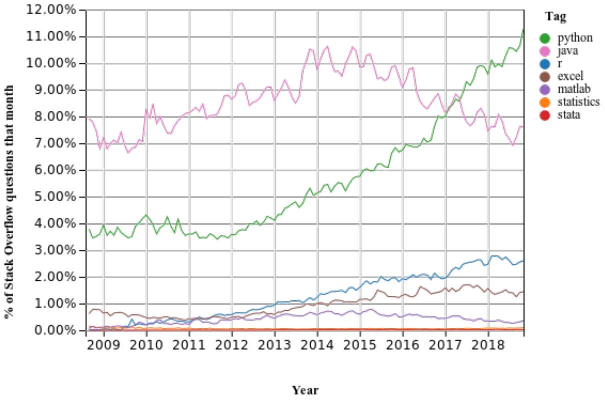
Python is convenient because it covers so many bases

- web dev, databases, system admin, GUIs

Chris Wiggins, Chief Data Scientist at The New York Times:

> Python has gotten sufficiently weapons grade that we don't descend into R anymore. Sorry, R people. I used to be one of you but we no longer descend into R.

As a result of these advantages:



Downloads / Installation / Troubleshooting

Install Python + Scientific Libs

- Install Anaconda from `https://www.anaconda.com/downloads`
    - Select latest Python version (3.x)
    - For your OS!
- Not plain vanilla Python

Remote options

- `https://colab.research.google.com`
- `https://notebooks.azure.com/`

Jupyter notebooks

A browser based interface to Python / Julia / R / etc.

Step 1: Open a terminal

- on Windows, use `Anaconda Command Prompt`

Step 2: type `jupyter notebook`

Workshop Resources

Lectures are at the course homepage:

https://github.com/QuantEcon/summer_course_2019

Get a copy

- via `git` or the Download button

Try running `day1/test.ipynb` using Jupyter notebook/lab