# Introduction to Programming for Public Policy Week 2

Eric Potash

March 3, 2018

# Introduction to Python

# Python

For the remainder of class we will be learning a high-level programming language called **Python**.

# What is a Programming language?

# Assembly

- Computers run programs written in low-level language called *assembly*
- Assembly code is very fast and efficient but:
  - The code is not portable between different operating systems
  - The code is difficult to write and read

# Assembly "Hello, World"

This is a program for printing the text "Hello, World" in assembly:

```
global    _start

section   .text
_start:   mov       rax, 1
mov       rdi, 1
mov       rsi, message
mov       rdx, 13
syscall
mov       rax, 60
xor       rdi, rdi
syscall

section   .data
message:  db        "Hello, World", 10
```

# Programming languages

- People created "higher level" programming languages to make our (programmers') lives easier:
  - Stata, SAS, SPSS
  - R
  - Shell
  - Python
  - Java
  - C, C++

# Interpretter vs Compiler

- There are two basic ways of translating high-level languages into low-level languages:
  - Interpretter: Code is read and translated and executed line by line (e.g. shell, Python, R)
  - Compiler: Code is read all at once and translated before it is executed
- Generally interpretted languages are easier to read and write but may be slower

# Interpretted "Hello, World"

- Shell:

```
echo Hello, World
```

- Python:

```
print("Hello, World")
```

- R:

```
print("Hello, World')
```

# Compiled "Hello, World"

- C:

```c
#include <stdio.h>
int main()
{
   printf("Hello, World!");
   return 0;
}
```

# Language choice

- People have preferences for programming languages just like they do for anything else

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria
  - e.g. speed or efficiency or portability

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria
  - e.g. speed or efficiency or portability
- But often they're subjective

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria
  - e.g. speed or efficiency or portability
- But often they're subjective
  - e.g. many people think that python code is very easy to read and write

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria
  - e.g. speed or efficiency or portability
- But often they're subjective
  - e.g. many people think that python code is very easy to read and write
- And there are many myths

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria
  - e.g. speed or efficiency or portability
- But often they're subjective
  - e.g. many people think that python code is very easy to read and write
- And there are many myths
  - e.g. you'll hear people say that python is faster than R

# Language choice

- People have preferences for programming languages just like they do for anything else
- Sometimes these preferences are based on objective criteria
  - e.g. speed or efficiency or portability
- But often they're subjective
  - e.g. many people think that python code is very easy to read and write
- And there are many myths
  - e.g. you'll hear people say that python is faster than R
  - it's not exactly true

# Why Python?

In this class we'll use python because it has proven over time to be:

- Easy (relatively) to learn
- Works well for many policy tasks (data analysis, text mining, visualization, modeling, etc.)
- Scales to large applications/datasets
- Has a large developer community

# Python

# Interactive Interpretter

- Simple way of running python
- Interactively enter text commands like the command line itself
- Like a calculator

```
$ python3.6
Python 3.6 (default, Sep 16 2015, 09:25:04)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more i
>>>
```

# Using python as a calculator:

TODO: take it from here
https://docs.python.org/3/tutorial/introduction.html

# String manipulation

# Lists, Dictionaries

# Executing a script

# Jupyter notebook

# More Python