Introduction to Data Science for Public Policy

Class 1: Overview

Thomas Monk

t.d.monk@lse.ac.uk

with thanks to

Frank Muci, LSE; Dr. Melissa Sands, LSE; Dr. Eric Potash, Harris School of Public Policy & Fabian H. C. Raters, Georg-August-Universität Göttingen

30 August 2022

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
  - Programming in a setting of public policy - why should we care?

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
  - Programming in a setting of public policy - why should we care?
  - Thinking algorithmically, taking policy questions to the data via a general purpose programming language.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
  - Programming in a setting of public policy - why should we care?
  - Thinking algorithmically, taking policy questions to the data via a general purpose programming language.
  - Fundamentals of programming: code syntax, libraries, variables, data types, program control, functions and IO.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
  - Programming in a setting of public policy - why should we care?
  - Thinking algorithmically, taking policy questions to the data via a general purpose programming language.
  - Fundamentals of programming: code syntax, libraries, variables, data types, program control, functions and IO.
  - Data science through open-source python libraries. Cleaning, obtaining and analysing structured data.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
  - Programming in a setting of public policy - why should we care?
  - Thinking algorithmically, taking policy questions to the data via a general purpose programming language.
  - Fundamentals of programming: code syntax, libraries, variables, data types, program control, functions and IO.
  - Data science through open-source python libraries. Cleaning, obtaining and analysing structured data.
  - Introducing more advanced applications, such as web scraping and natural language processing.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
    - Programming in a setting of public policy - why should we care?
    - Thinking algorithmically, taking policy questions to the data via a general purpose programming language.
    - Fundamentals of programming: code syntax, libraries, variables, data types, program control, functions and IO.
    - Data science through open-source python libraries. Cleaning, obtaining and analysing structured data.
    - Introducing more advanced applications, such as web scraping and natural language processing.
- The syllabus itself is much subject to change - I want the content to go at the pace and towards the interests of the group.

Overview

- This course is designed as an intensive two-week introduction to programming and data science for public policy students, using Python.
- We'll meet daily in NAB 2.09, 11:00-13:00. Course website: https://tmonk.github.io/dspp/. The syllabus and material is located there.
- We're going to cover:
  - Programming in a setting of public policy - why should we care?
  - Thinking algorithmically, taking policy questions to the data via a general purpose programming language.
  - Fundamentals of programming: code syntax, libraries, variables, data types, program control, functions and IO.
  - Data science through open-source python libraries. Cleaning, obtaining and analysing structured data.
  - Introducing more advanced applications, such as web scraping and natural language processing.
- The syllabus itself is much subject to change - I want the content to go at the pace and towards the interests of the group.
- Let me know as we progress what you'd like to see more of, or something you've seen elsewhere, and I'll try and get this included.

Data Science in Public Policy

- We saw a huge variety of applications of data to causal questions in PP455.
- This course is going to differ in two key ways:
  1. Causality is not going to be our focus. We will care about ingesting and manipulating data, with other meaningful applications, rather than being hung up on identification.
  2. The datasets we will use, and will be able to use, will be bigger.
- There's more to data than causal inference...

Three Modes of Inference

1. Causal Inference: predicting counterfactuals

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
    - Inferring the effect of education on wages.

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
    - Inferring the effect of education on wages.
    - Inferring the effects of ethnic minority rule on civil war onset.
    - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
    - Inferring "preferences" from choice data.

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.
   - Inferring "social networks" from surveys.

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.
   - Inferring "social networks" from surveys.
3. Predictive Inference: forecasting out-of-sample data points

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.
   - Inferring "social networks" from surveys.
3. Predictive Inference: forecasting out-of-sample data points
   - Inferring minimum wage status from demographic data (Cengiz et. al. 2021, NBER).

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.
   - Inferring "social networks" from surveys.
3. Predictive Inference: forecasting out-of-sample data points
   - Inferring minimum wage status from demographic data (Cengiz et. al. 2021, NBER).
   - Inferring partisanship from congressional speech (Gentzkow et. al. 2019, ECMA).

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.
   - Inferring "social networks" from surveys.
3. Predictive Inference: forecasting out-of-sample data points
   - Inferring minimum wage status from demographic data (Cengiz et. al. 2021, NBER).
   - Inferring partisanship from congressional speech (Gentzkow et. al. 2019, ECMA).
   - Inferring exposure to artificial intelligence from job task information (Webb, 2019).

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
   - Inferring the effect of education on wages.
   - Inferring the effects of ethnic minority rule on civil war onset.
   - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
   - Inferring "preferences" from choice data.
   - Inferring "topics" from texts and speeches.
   - Inferring "social networks" from surveys.
3. Predictive Inference: forecasting out-of-sample data points
   - Inferring minimum wage status from demographic data (Cengiz et. al. 2021, NBER).
   - Inferring partisanship from congressional speech (Gentzkow et. al. 2019, ECMA).
   - Inferring exposure to artificial intelligence from job task information (Webb, 2019).

Three Modes of Inference

1. Causal Inference: predicting counterfactuals
    - Inferring the effect of education on wages.
    - Inferring the effects of ethnic minority rule on civil war onset.
    - Inferring whether incumbency status affects election outcomes.
2. Descriptive Inference: summarizing and exploring data
    - Inferring "preferences" from choice data.
    - Inferring "topics" from texts and speeches.
    - Inferring "social networks" from surveys.
3. Predictive Inference: forecasting out-of-sample data points
    - Inferring minimum wage status from demographic data (Cengiz et. al. 2021, NBER).
    - Inferring partisanship from congressional speech (Gentzkow et. al. 2019, ECMA).
    - Inferring exposure to artificial intelligence from job task information (Webb, 2019).

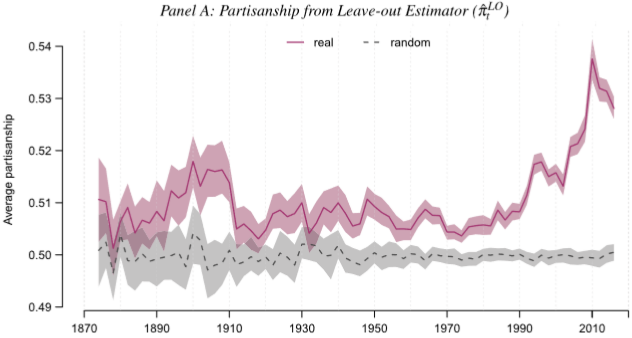
Although the lines between these can be quite blurred, and often one type of inference depends on the conclusions of another.

Predictive/descriptive inference example: Gentzkow (2019)

This paper measures trends in the partisanship of congressional speech in the US House of Representatives and the Senate. The paper is mathematically complex and dense, but delivers an interesting conclusion (non-causally) based on a large quantity of data.

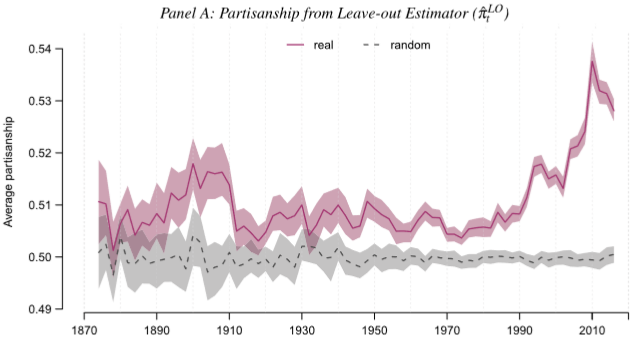
- **Definition.** Partisanship is defined as the ease with which an observer could infer a congressperson's party from a single utterance.
- **Data.** All congressional speech from 1873 to 2016. 508,352 unique phrases spoken a total of 287 million times by 7732 unique speakers.
- **Output**. A partisanship score for each session of congress. When the words (vectors) spoken by Democrats and Republicans are similar, there is low polarisation in Congress. When these vectors are far from each other, the parties speak differently and we say that partisanship is high.

Predictive/descriptive inference example: Gentzkow (2019)



Polarisation in the US congress was roughly stable until the early 1990s.

Predictive/descriptive inference example: Gentzkow (2019)



Polarisation in the US congress was roughly stable until the early 1990s.

Predictive/descriptive inference example: Gentzkow (2019)

Session 110 (2007–2008)						Session 114 (2015–2016)					
Republican	#R	#D	Democratic	#R	#D	Republican	#R	#D	Democratic	#R	#D
tax increas	87	20	dog coalit	0	90	american peopl	327	205	homeland secur	96	205
natur gas	77	20	war iraq	18	78	al qaeda	50	7	climat chang	23	94
reserv balanc	147	105	african american	6	62	men women	123	83	gun violenc	3	74
rais tax	44	10	american peopl	230	278	side aisl	133	93	african american	11	71
american energi	34	3	oil compani	20	65	human traffick	60	26	vote right	2	62
illeg immigr	34	7	civil war	17	45	colleagu support	123	89	public health	24	83
side aisl	132	106	troop iraq	11	39	religi freedom	34	4	depart homeland	48	93
continent shelf	33	8	children health	17	42	taxpay dollar	47	19	plan parenthood	66	104
outer continent	32	8	nobid contract	0	24	mental health	59	32	afford care	40	77
tax rate	26	4	middl class	15	39	radic islam	22	0	puerto rico	42	79

We have an insight into the most polarising phrases of each Congressional session.

Programming and Public Policy: the aim of this course

Why have I shown this paper?

Programming and Public Policy: the aim of this course

Why have I shown this paper? It's doing huge amounts of interesting **data science**.

- It ingests a huge corpus of data, and transforms it into a database ready to be exploited.

Programming and Public Policy: the aim of this course

Why have I shown this paper? It's doing huge amounts of interesting **data science**.

- It ingests a huge corpus of data, and transforms it into a database ready to be exploited.
- It analyses the data and visualises the underlying data in interesting and striking ways.

Programming and Public Policy: the aim of this course

Why have I shown this paper? It's doing huge amounts of interesting **data science**.

- It ingests a huge corpus of data, and transforms it into a database ready to be exploited.
- It analyses the data and visualises the underlying data in interesting and striking ways.
- It uses natural language processing and machine learning techniques at the cutting edge.

Programming and Public Policy: the aim of this course

Why have I shown this paper? It's doing huge amounts of interesting **data science**.

- It ingests a huge corpus of data, and transforms it into a database ready to be exploited.
- It analyses the data and visualises the underlying data in interesting and striking ways.
- It uses natural language processing and machine learning techniques at the cutting edge.

Programming and Public Policy: the aim of this course

Why have I shown this paper? It's doing huge amounts of interesting **data science**.

- It ingests a huge corpus of data, and transforms it into a database ready to be exploited.
- It analyses the data and visualises the underlying data in interesting and striking ways.
- It uses natural language processing and machine learning techniques at the cutting edge.

Course Aims

I want you to come out of this course with an understanding of the **possibilities** (and limitations) that data science holds for us in being able to make informed policy choices and provide social insight.

Programming and Public Policy: the aim of this course

Why have I shown this paper? It's doing huge amounts of interesting **data science**.

- It ingests a huge corpus of data, and transforms it into a database ready to be exploited.
- It analyses the data and visualises the underlying data in interesting and striking ways.
- It uses natural language processing and machine learning techniques at the cutting edge.

Course Aims

I want you to come out of this course with an understanding of the **possibilities** (and limitations) that data science holds for us in being able to make informed policy choices and provide social insight.

The best way for us to do that is to learn this **actively** - which we'll spend the rest of the course doing.

Why not Stata?

- In PP455 we heavily used Stata. Stata is great -

Why not Stata?

- In PP455 we heavily used Stata. Stata is great -

Why not Stata?

- In PP455 we heavily used Stata. Stata is great - but only really excels at a small number of tasks.
- If I want to run regressions (or regression-adjacent work), then I turn to Stata.

Why not Stata?

- In PP455 we heavily used Stata. Stata is great - but only really excels at a small number of tasks.
- If I want to run regressions (or regression-adjacent work), then I turn to Stata.
- If we want to do nearly *anything* else, then we turn to a more general-purpose programming language. Could you code the Gentzkow paper in Stata?

Why not Stata?

- In PP455 we heavily used Stata. Stata is great - but only really excels at a small number of tasks.
- If I want to run regressions (or regression-adjacent work), then I turn to Stata.
- If we want to do nearly *anything* else, then we turn to a more general-purpose programming language. Could you code the Gentzkow paper in Stata?
- Why not Stata?

Why not Stata?

- In PP455 we heavily used Stata. Stata is great - but only really excels at a small number of tasks.
- If I want to run regressions (or regression-adjacent work), then I turn to Stata.
- If we want to do nearly *anything* else, then we turn to a more general-purpose programming language. Could you code the Gentzkow paper in Stata?
- Why not Stata?
    1. Cost. Stata is proprietary - costs $1000 USD per year per license.

Why not Stata?

- In PP455 we heavily used Stata. Stata is great - but only really excels at a small number of tasks.
- If I want to run regressions (or regression-adjacent work), then I turn to Stata.
- If we want to do nearly *anything* else, then we turn to a more general-purpose programming language. Could you code the Gentzkow paper in Stata?
- Why not Stata?
  1. Cost. Stata is proprietary - costs $1000 USD per year per license.
  2. Functionality. Basic programming logic that you'll be exposed to in this course, like *if*, *loops* etc. are far more tedious to write in stata.

Why not Stata?

- In PP455 we heavily used Stata. Stata is great - but only really excels at a small number of tasks.
- If I want to run regressions (or regression-adjacent work), then I turn to Stata.
- If we want to do nearly *anything* else, then we turn to a more general-purpose programming language. Could you code the Gentzkow paper in Stata?
- Why not Stata?
  1. Cost. Stata is proprietary - costs $1000 USD per year per license.
  2. Functionality. Basic programming logic that you'll be exposed to in this course, like *if*, *loops* etc. are far more tedious to write in stata.
  3. Mata. For complex software, Stata forces you to write in a language called Mata. It's fast, but based around matrices and relatively complex for general purpose tasks.

What do we want in a programming language?

- There are a large number of programming languages we could learn in this course:

What do we want in a programming language?

- There are a large number of programming languages we could learn in this course:
  - R, MATLAB, GAUSS, Fortran, C/++ and Julia are all used, to varying extents, in Economics and Public Policy settings.

What do we want in a programming language?

- There are a large number of programming languages we could learn in this course:
  - R, MATLAB, GAUSS, Fortran, C/++ and Julia are all used, to varying extents, in Economics and Public Policy settings.
  - Fortran is used heavily by researchers in macro and public economics, for large, complex models of consumer choice over the lifecycle. See Tony Smith at Yale and his coauthors.

What do we want in a programming language?

- There are a large number of programming languages we could learn in this course:
  - R, MATLAB, GAUSS, Fortran, C/++ and Julia are all used, to varying extents, in Economics and Public Policy settings.
  - Fortran is used heavily by researchers in macro and public economics, for large, complex models of consumer choice over the lifecycle. See Tony Smith at Yale and his coauthors.
  - For context, Fortran was developed by IBM in the 50s on mainframes - my Dad used Fortran punch cards to code as part of his astrophysics doctorate in the 70s.

What do we want in a programming language?

- There are a large number of programming languages we could learn in this course:
  - R, MATLAB, GAUSS, Fortran, C/++ and Julia are all used, to varying extents, in Economics and Public Policy settings.
  - Fortran is used heavily by researchers in macro and public economics, for large, complex models of consumer choice over the lifecycle. See Tony Smith at Yale and his coauthors.
  - For context, Fortran was developed by IBM in the 50s on mainframes - my Dad used Fortran punch cards to code as part of his astrophysics doctorate in the 70s.
- Academics and practitioners have preferences over languages for a variety of reasons.

What do we want in a programming language?

- There are a large number of programming languages we could learn in this course:
  - R, MATLAB, GAUSS, Fortran, C/++ and Julia are all used, to varying extents, in Economics and Public Policy settings.
  - Fortran is used heavily by researchers in macro and public economics, for large, complex models of consumer choice over the lifecycle. See Tony Smith at Yale and his coauthors.
  - For context, Fortran was developed by IBM in the 50s on mainframes - my Dad used Fortran punch cards to code as part of his astrophysics doctorate in the 70s.
- Academics and practitioners have preferences over languages for a variety of reasons.
- One of the main factors is the 'level' of the programming language.

Lowest level programming language: ASM

```
          global    _start

          section   .text
_start:   mov       rax, 1                  ; system call for write
          mov       rdi, 1                  ; file handle 1 is stdout
          mov       rsi, message            ; address of string to output
          mov       rdx, 13                 ; number of bytes
          syscall                           ; invoke operating system to do the write
          mov       rax, 60                 ; system call for exit
          xor       rdi, rdi                ; exit code 0
          syscall                           ; invoke operating system to exit

          section   .data
message:  db        Hello, World, 10        ; note the newline at the end
```

Here we are directly interacting with the processor of the computer, instructing the CPU in the only language it understands.

Lowest level programming language: C

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

The number of lines we have to write has dropped significantly!

What do we think could be happening in the background?

Lowest level programming language: C

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

The number of lines we have to write has dropped significantly!

What do we think could be happening in the background? A program called a **compiler** is translating the more human readable code into a language the computer actually understands.

Higher level programming language: Fortran

```
program hello
  print *, 'Hello, World!'
end program hello
```

We come closer to a more human readable code base.

Higher level programming language: Fortran

```
C This program computes the equilibrium of the stochastic-beta economy in
C "Income and Wealth Heterogeneity in the Macroeconomy" (co-authored with
C Per Krusell of the University of Rochester).
      implicit real*8 (a-h,o-z)

      parameter (nkpts=132,durug=1.5D+00,nmupts=4,
     *           delta=0.025D+00,alpha=0.36D+00,unempg=0.04D+00,
     *           mxloop=12,sfac=0.25D+00,hfix=0.3271D+00,
     *           durgd=8.0D+00,unempb=0.1D+00,kgrid=278,
     *           xkbor=-2.4D+00,mgrid=30,
     *           durbd=8.0D+00,zgood=1.01D+00,zbad=0.99D+00,
     *           npbhat=2,durub=2.5D+00,ntop=9,
     *           xkglow=xkbor,xkghgh=25.0D+00,nlzpts=201,
     *           nbetas=3,nzpts=2,nrspts=132)

      common/cpr/pr(4,4),prbeta(nbetas,nbetas),prob(nbetas)
      common/cgrid/xkgpts(kgrid),xmgpts(mgrid)
      common/clzpts/xlzpts(nlzpts),xlzdat(nlzpts)
      common/cbetas/betas(nbetas)
      common/ctop/toppct(ntop)
      common/ccoef/coefk(kgrid-1,mgrid-1,4,nbetas,4),
```

But things can still get complex.

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be?

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be?

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be? Speed.
- For example, a prime-checking algorithm could run in 1 min in Python. The same logic could take 5 seconds to run in assembly.

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be? Speed.
- For example, a prime-checking algorithm could run in 1 min in Python. The same logic could take 5 seconds to run in assembly.
- Python here is interpreted instead of compiled, this means each instruction can be read line by line by the machine and run. A compiler considers the whole code at once.

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be? Speed.
- For example, a prime-checking algorithm could run in 1 min in Python. The same logic could take 5 seconds to run in assembly.
- Python here is interpreted instead of compiled, this means each instruction can be read line by line by the machine and run. A compiler considers the whole code at once.
- Interpreted code is generally slower than complied code, but easier to write.

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be? Speed.
- For example, a prime-checking algorithm could run in 1 min in Python. The same logic could take 5 seconds to run in assembly.
- Python here is interpreted instead of compiled, this means each instruction can be read line by line by the machine and run. A compiler considers the whole code at once.
- Interpreted code is generally slower than complied code, but easier to write.
- So is Python slow?

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be? Speed.
- For example, a prime-checking algorithm could run in 1 min in Python. The same logic could take 5 seconds to run in assembly.
- Python here is interpreted instead of compiled, this means each instruction can be read line by line by the machine and run. A compiler considers the whole code at once.
- Interpreted code is generally slower than complied code, but easier to write.
- So is Python slow?

High level programming language: Python

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

- This is all we need to do in Python to obtain exactly the same output as the assembly code.
- But what could the the trade offs be? Speed.
- For example, a prime-checking algorithm could run in 1 min in Python. The same logic could take 5 seconds to run in assembly.
- Python here is interpreted instead of compiled, this means each instruction can be read line by line by the machine and run. A compiler considers the whole code at once.
- Interpreted code is generally slower than complied code, but easier to write.
- So is Python slow? No, as we'll be interfacing with pieces of code that are written in C. Python is our gateway to that code.

Why Python?

- **Ease**. It's high level - easy to write, read and learn.

Why Python?

- **Ease**. It's high level - easy to write, read and learn.
- **Resources**. It's used by a huge number of people across a variety of different disciplines. If you have a question, the answer is only a Google search away.

Why Python?

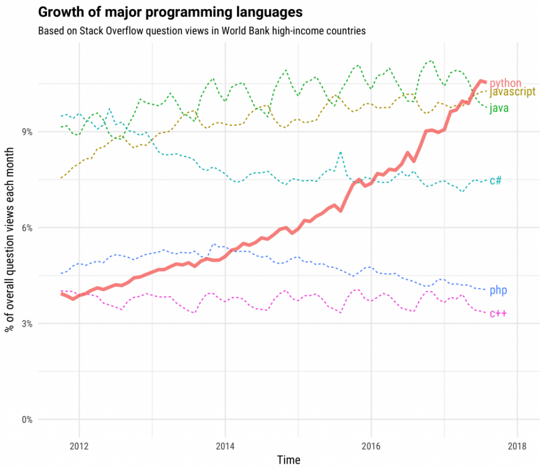
- **Ease**. It's high level - easy to write, read and learn.
- **Resources**. It's used by a huge number of people across a variety of different disciplines. If you have a question, the answer is only a Google search away.
- **Modules**. High quality, fast and feature-packed code has been written for Python which allows us to do data science related tasks with ease, and even progress on to more complex machine learning applications.

Why Python?

- **Ease**. It's high level - easy to write, read and learn.
- **Resources**. It's used by a huge number of people across a variety of different disciplines. If you have a question, the answer is only a Google search away.
- **Modules**. High quality, fast and feature-packed code has been written for Python which allows us to do data science related tasks with ease, and even progress on to more complex machine learning applications.
- **Idiosyncratic preference**. I prefer Python to R, so you'll learn Python.

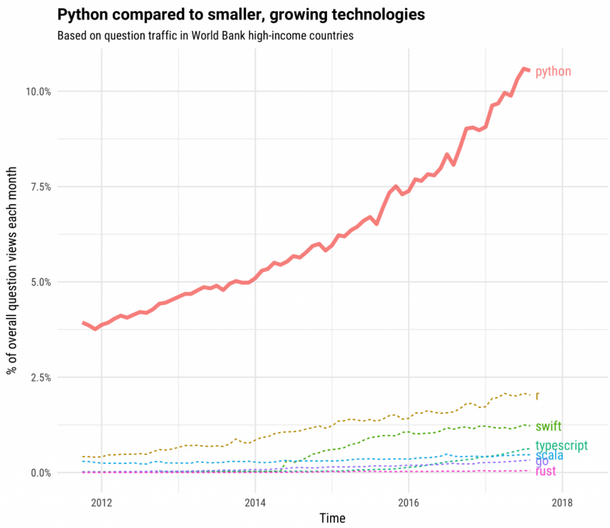
Global preference

Python has become extremely popular.



Global preference

The world prefers Python to R - this makes our life a lot easier.



Thinking beyond Python

- Importantly, learning Python doesn't stop you from picking up any other programming languages.

Thinking beyond Python

- Importantly, learning Python doesn't stop you from picking up any other programming languages.
- In fact, learning how to use Python will give you a fundamental understanding of **programming thinking** - you'll even notice that learning Stata over the last year will have made this easier.

Thinking beyond Python

- Importantly, learning Python doesn't stop you from picking up any other programming languages.
- In fact, learning how to use Python will give you a fundamental understanding of **programming thinking** - you'll even notice that learning Stata over the last year will have made this easier.
- The aim of course course is to teach you language agnostic concepts, that you can apply over your career. We happen to be understand these concepts through the lens of Python.

Installation

Lets take ourselves to the point that we can actually run the example I've shown above!

We're going to spend the rest of the class installing **Anaconda** - the instructions are on these webpages depending on your OS. I'll walk you through.

1. Install the Anaconda distribution (anaconda.com). Anaconda brings together a number of data science tools into one complete package.

Installation

Lets take ourselves to the point that we can actually run the example I've shown above!

We're going to spend the rest of the class installing **Anaconda** - the instructions are on these webpages depending on your OS. I'll walk you through.

1. Install the Anaconda distribution (anaconda.com). Anaconda brings together a number of data science tools into one complete package.
2. Following installation run *Anaconda Navigator*.

Installation

Lets take ourselves to the point that we can actually run the example I've shown above!

We're going to spend the rest of the class installing **Anaconda** - the instructions are on these webpages depending on your OS. I'll walk you through.

1. Install the Anaconda distribution (anaconda.com). Anaconda brings together a number of data science tools into one complete package.
2. Following installation run *Anaconda Navigator*.
3. Open JupyterLab - this will be our main workspace in using Python.

JupyterLab

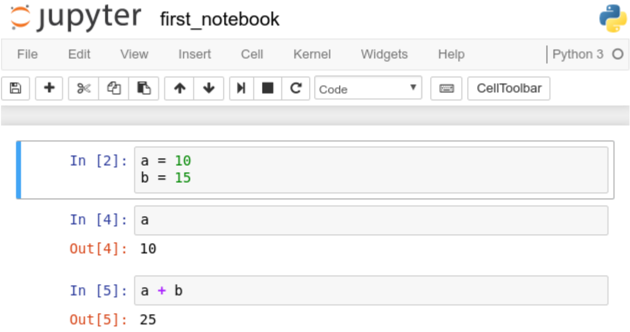
- JupyterLab is our gateway to Python. There are many possible options, but Jupyter provides everything we need.

JupyterLab

- JupyterLab is our gateway to Python. There are many possible options, but Jupyter provides everything we need.
- Jupyter provides us with a **notebook**.

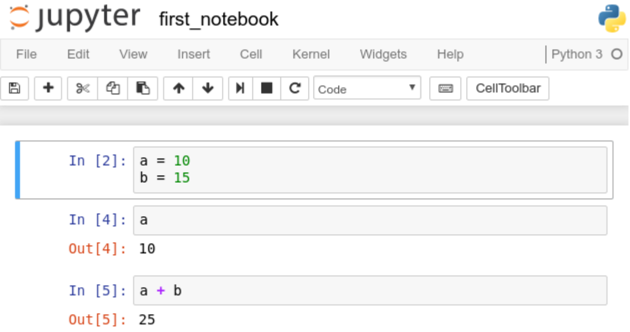
JupyterLab

- JupyterLab is our gateway to Python. There are many possible options, but Jupyter provides everything we need.
- Jupyter provides us with a **notebook**.
- You'll see individual, vertically arranged cells. These can be executed separately.



JupyterLab

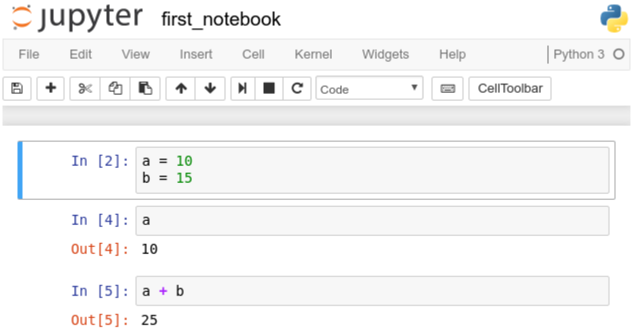
- JupyterLab is our gateway to Python. There are many possible options, but Jupyter provides everything we need.
- Jupyter provides us with a **notebook**.
- You'll see individual, vertically arranged cells. These can be executed separately.



- Type the following into a cell, and press the ▶ button to run (or Control-Enter).

JupyterLab

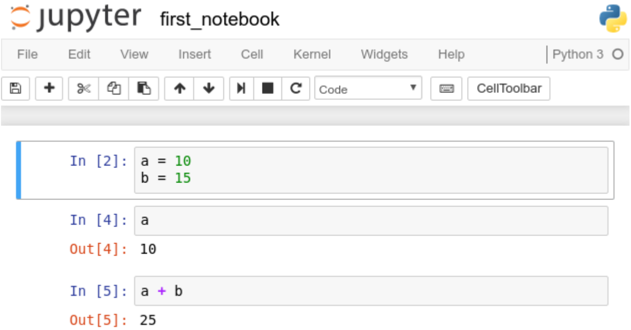
- JupyterLab is our gateway to Python. There are many possible options, but Jupyter provides everything we need.
- Jupyter provides us with a **notebook**.
- You'll see individual, vertically arranged cells. These can be executed separately.



- Type the following into a cell, and press the ▶ button to run (or Control-Enter).

JupyterLab

- JupyterLab is our gateway to Python. There are many possible options, but Jupyter provides everything we need.
- Jupyter provides us with a **notebook**.
- You'll see individual, vertically arranged cells. These can be executed separately.



- Type the following into a cell, and press the ▶ button to run (or Control-Enter).

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