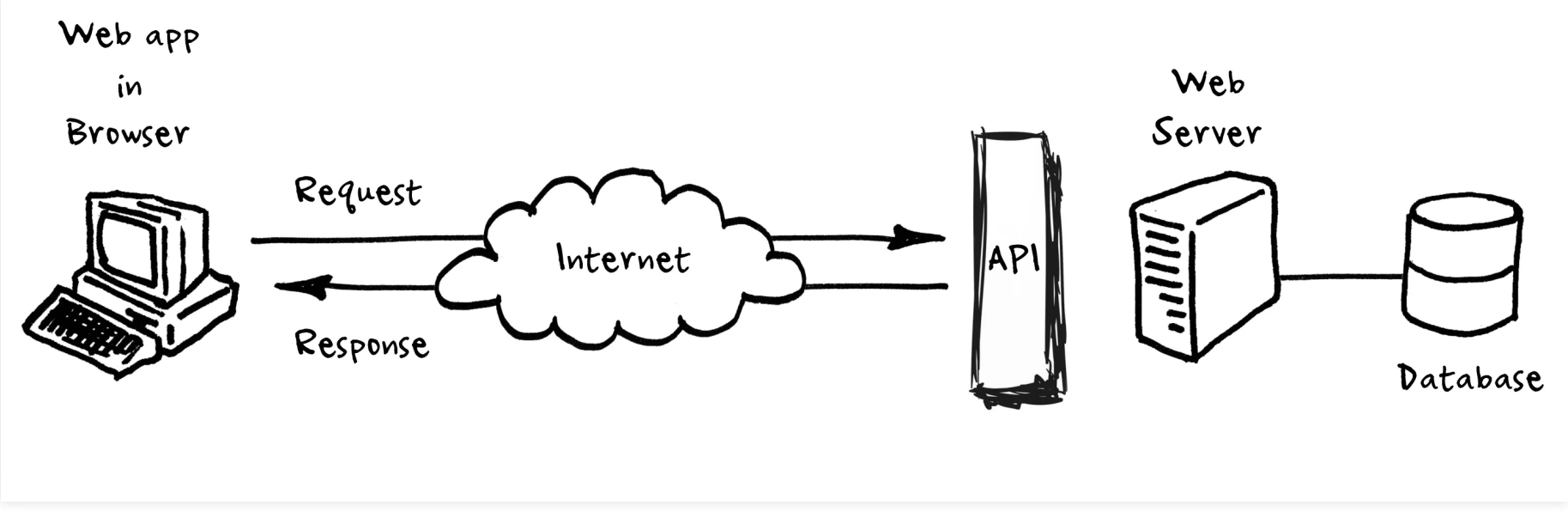
Storytelling with Data

February 5, 2018

Programming Fundamentals

You already got a peek...

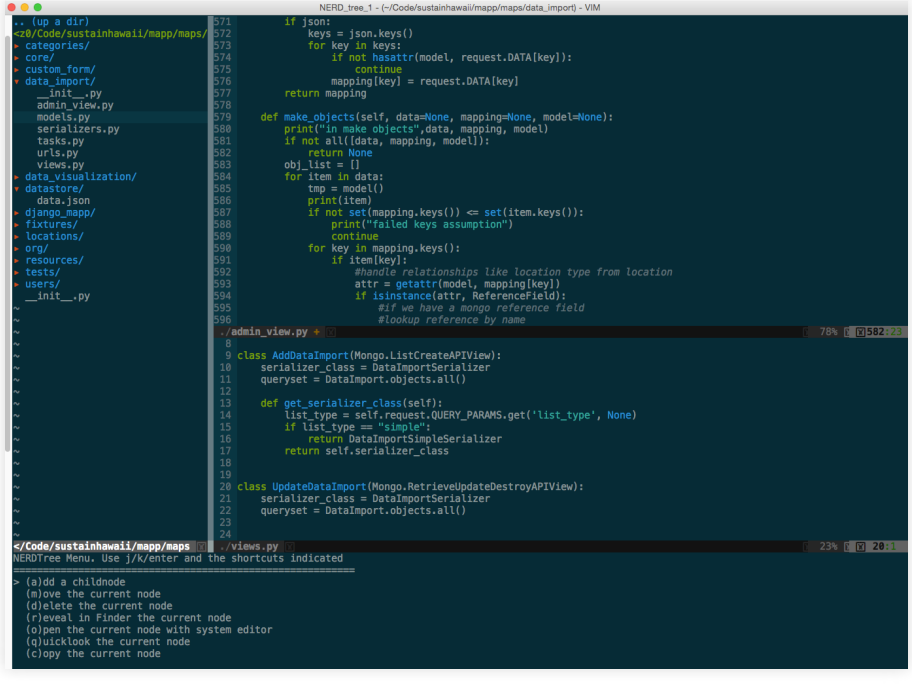
In the second lecture, we showcased a particular task that is well served by use of a programming language: accessing data via an API



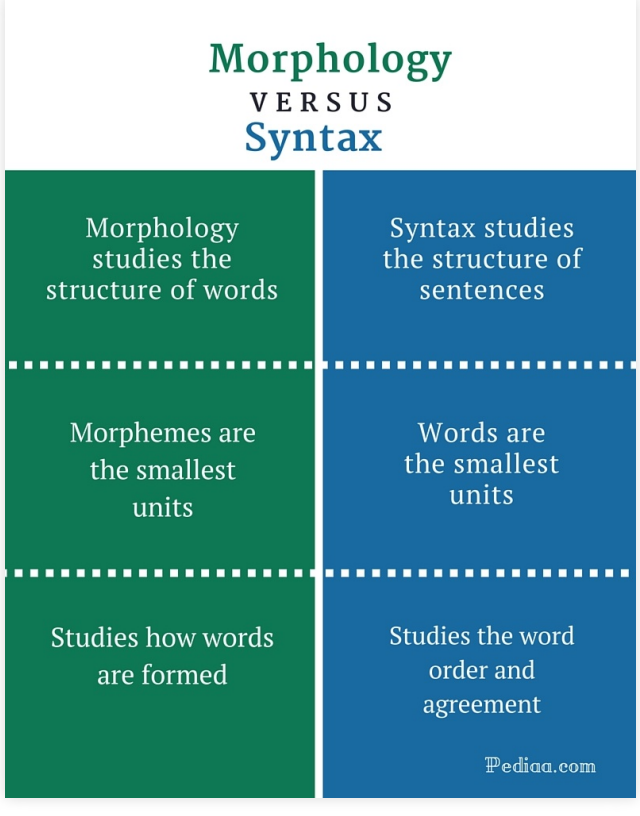
Now let's take some time with the basics.

What is a programming language?

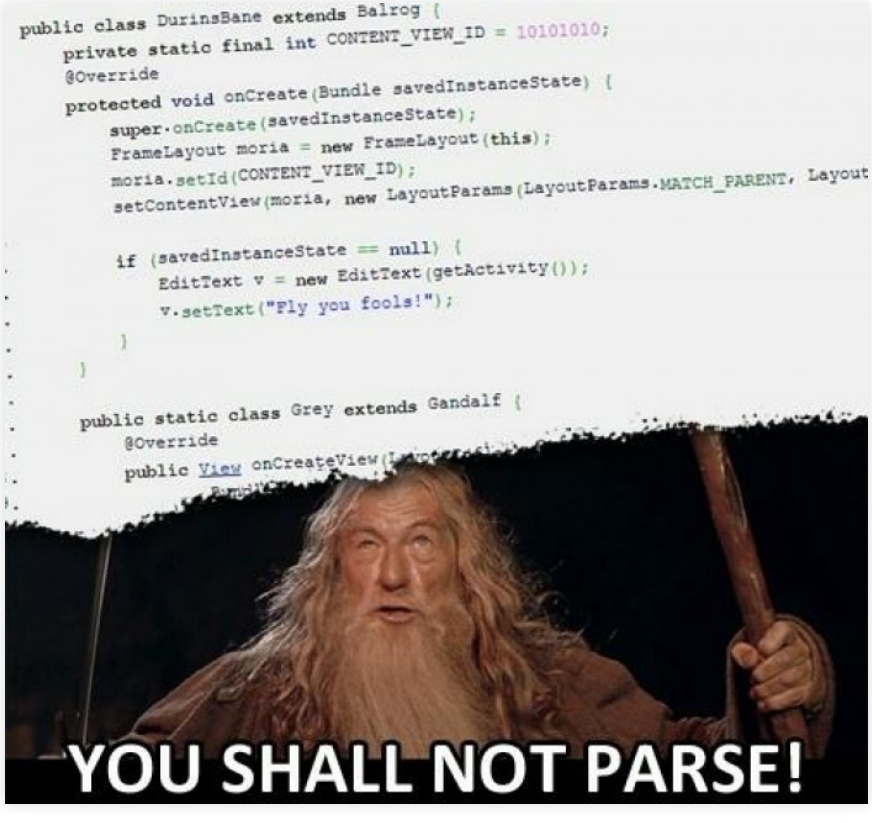
To execute tasks, computers "consume" very low-level binary instructions. It is hard, however, to write instructions as they are consumed because who can keep track of what they mean in a sea of 0s and 1s. Programming languages were developed to provide varying levels of abstractions around certain kinds of low-level tasks. These languages allow us to write prose (of sorts) to communicate our objectives with a computer.



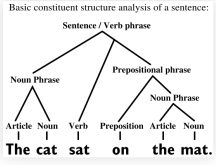
Programming languages are *easier* than spoken languages



Grammar can be decomposed into morphology and syntax.



Syntax analysis still works...



Why bother learning to code?

While we will provide you with the option to submit assignments using visualization tools of your choosing, a major goal of this course is to expose you to a programming language. Why?

- Writing code promotes reproducibility because it provides an unambiguous record of decisions
- It promotes iterative development/refinement because duplication costs are low
- Done well, writing code can provide durable tools that can be used for different projects
- Using a general computing language will give you a better tacit sense of how computers operate
- Exposure to programming increases the range of tasks you can accomplish

Why choose Python?

IEEE Spectrum Rating

Take this with a grain of salt...

Language Rank	Types	Spectrum Ranking
1. Python		100.0
2. C		99.7
3. Java		99.5
4. C++		97.1
5. C#		87.7
6. R		87.7
7. JavaScript		85.6
8. PHP		81.2
9. Go		75.1
10. Swift		73.7

1. Python is very popular
2. There are a ton of pedagogical tools available
3. Python helps you do things other than data analysis
4. Learning Python doesn't require a CS background

What's the difference between Python and IPython?

Python is the base language itself. It is what you are actually writing (in most cases), and what is ultimately converted into instructions for the computer.

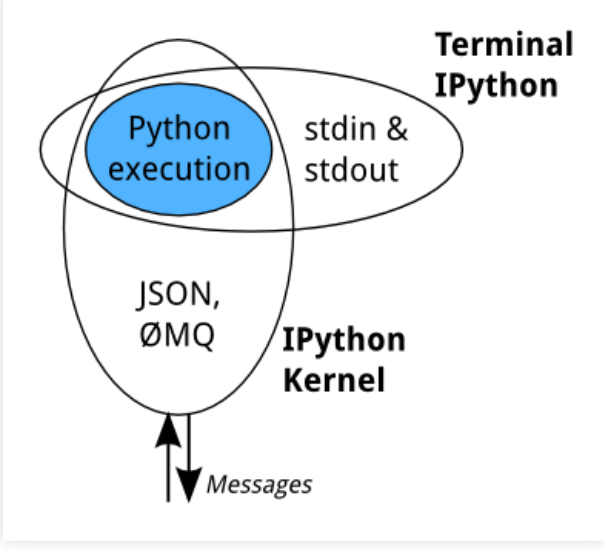
The "I" in IPython stands for *interactive*. IPython is an interface that extends the functionality of Python, while also improving the user experience. Among other things, it provides:

- Tab autocompletion
- Clearer and more comprehensive error messages
- Improved command history management
- UNIX shell integration (more on this later)
- A whole mess of graphical user interfaces

Maybe it's just a few GUIs...

... but how does it offer more than one?

The legit genius of the IPython project was to separate the kernel (the thing that executes code) from the user interface (the thing into which you input code).



Why on earth do I care?

That separation makes Jupyter Lab possible...

...and it has implications for your workflow.

The Jupyter project is an offshoot of the IPython project

The Jupyter Project supports a wide range of languages

On to the live examples...