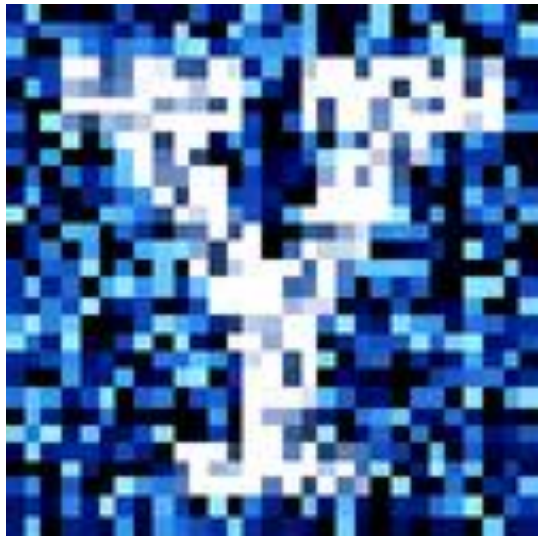
YData: Introduction to Data Science



Class 02: Introduction to Python

Overview

Very quick review of “What is Data Science”, and discussion of the reading from “Everybody Lies”

Intro to Python

- Expressions
- Names
- Call expressions (functions)
- Data types
  - Numbers and strings

- If there is time
  - Lists

Let's test the YCRC Jupyter notebook server…

Before we get started, please log into the YCRC Jupyter notebook server

A link to the server is at the top of the class Canvas page

Can everyone log in?

Announcements

Please fill out two surveys

- 1. Background survey
- 2. Practice session and study group survey

Practice sessions are going to be on:

- **Thursday sessions** in DL 120
  - 4pm to 5pm
  - 5:15pm to 6:15pm
- **Friday sessions**, location TBD
  - 1pm to 2pm
  - 2:15 pm to 3:15pm

Each session is capped at 30 students

Shivam will send an announcement about signing up for a specific session time

Announcement: Homework 1

Homework 1 has been posted!

```
import YData
YData.download_homework(1)
```

It is due on Gradescope on Sunday September 7th at 11pm

- **Be sure to mark each question on Gradescope!**

Review: The history of Data Science

(a very incomplete list)

Data

Ishango bone

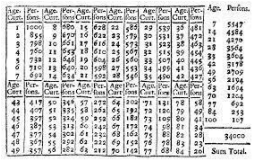
(20,000 BCE)

Cuneiform tablets

(4,000 BCE)

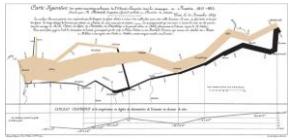
Quipus in South America

(1100-1500)



Demographics

(1600's)



Golden age of data visualization

(1850-1900)

Big data

(now)

Probability

Key Take Away

Probability models dominated data analysis prior to using computational methods

Initial development

(1600's)

"Small data"

Probability in Statistics

(1820's – 1950's)

Math Stats dominates

(1900-1960's)

"Big data"

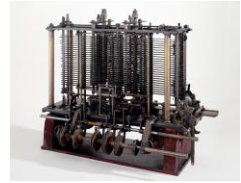
Computers

Abacus

(2400 BCE)

Antikythera mechanism

(100 BCE)



Analytical Engine

(1800's)

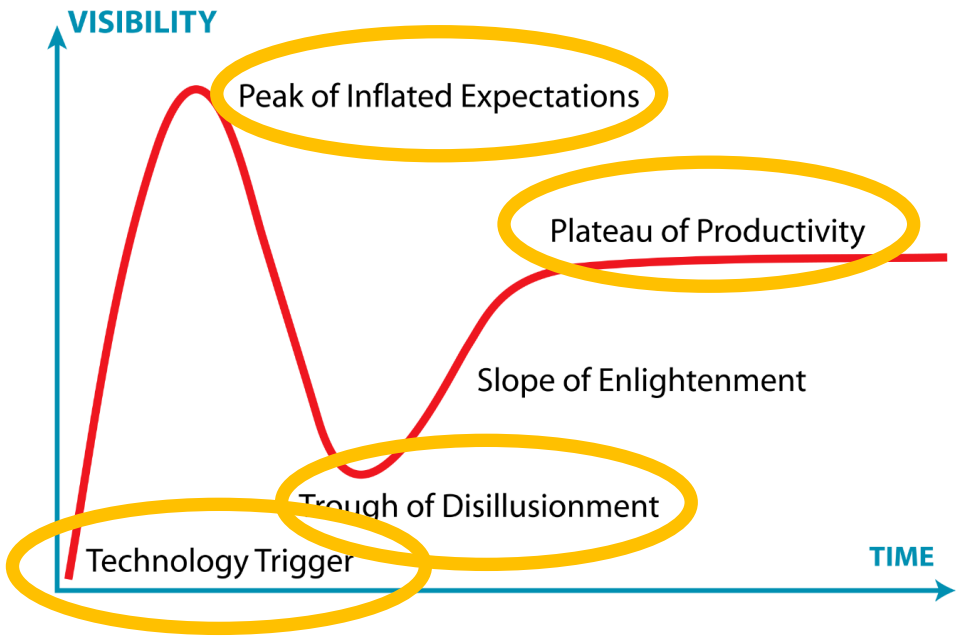
Hollerith Tabulating Machine

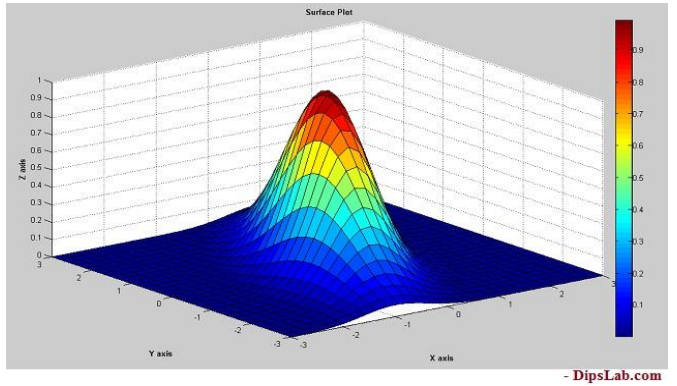
(1890)

Mainframes, PCs, Internet, etc.

(1950-present)

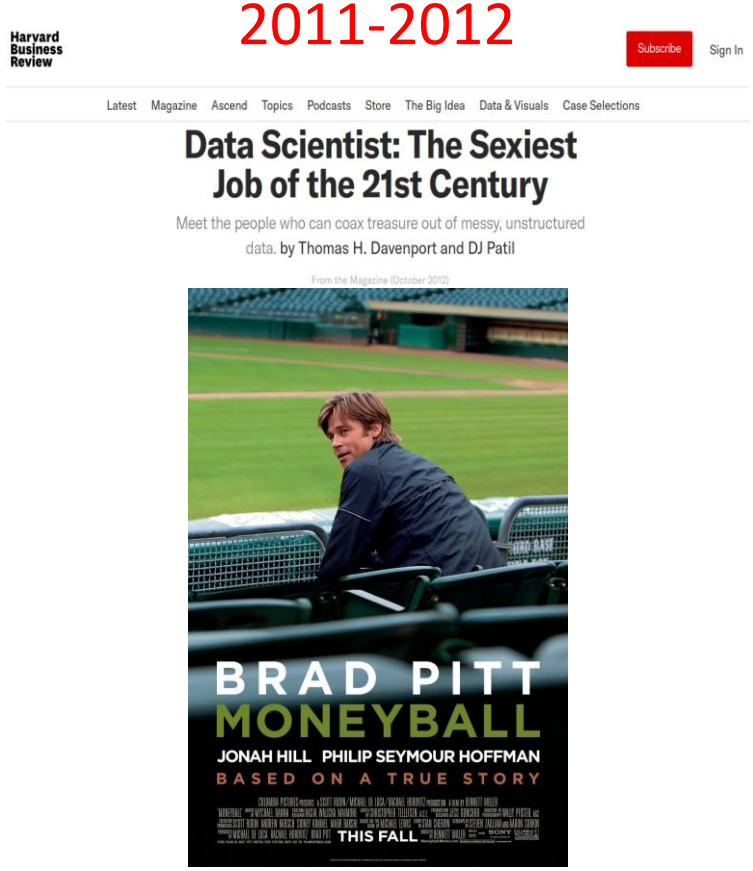
Review: Brief history of Data Science



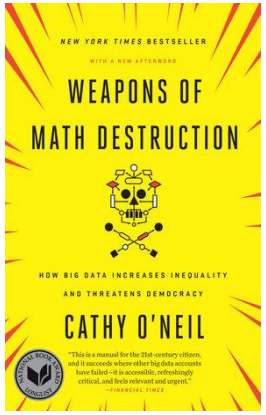


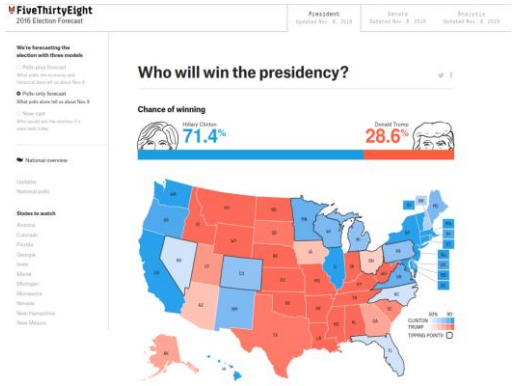
Early 2000's rise of the internet, personal computers and data analysis programming languages

2011-2012



Negative consequences were highlighted circa 2016



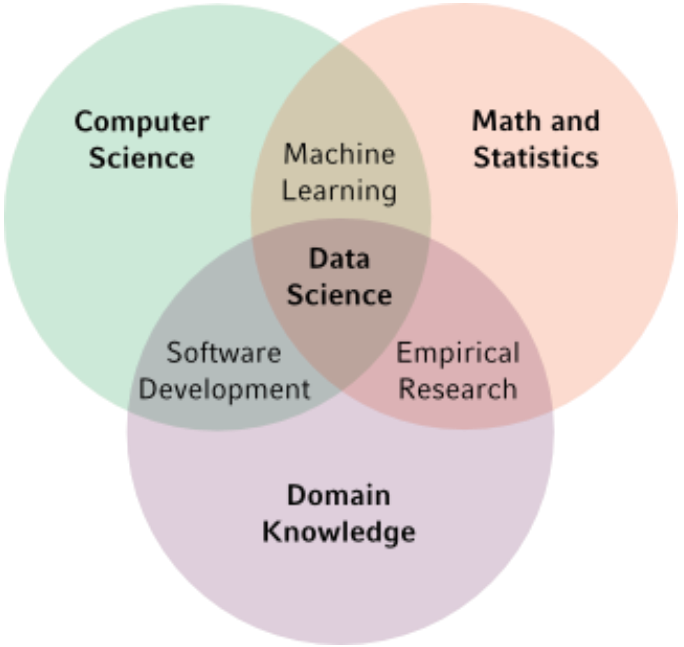


Yale S&DS

In March 2017 Yale renames the Department of Statistics to be the Department of Statistics and Data Science

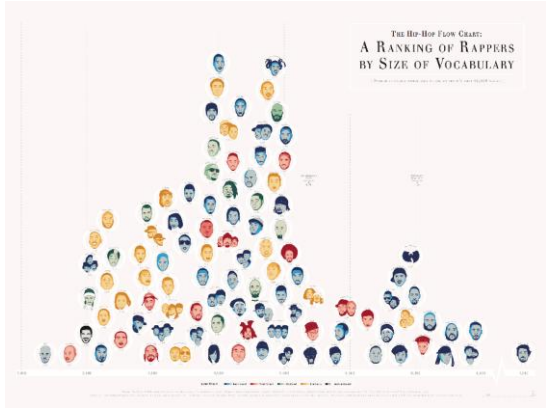
From Downey 2024

What is Data Science?



Data Science is a broadening of data analyses

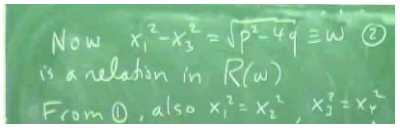
- Goes beyond traditional statistical mathematical/inferential analyses
- Uses more computation/programming
- "Big Data" from Internet, sensors etc., can be mined to transform our understanding in many fields
  - E.g., health, cosmology, social sciences, etc.
- Chooses data analysis methods less on theory and more on what gives clearest insights in particular fields



Examples:

- NYC city bikes
- Wind map visualization

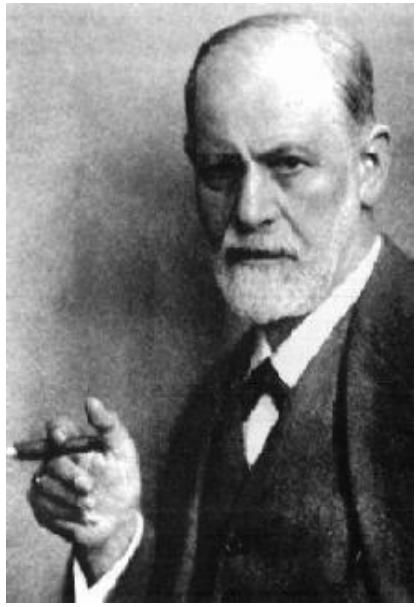
The proof is in the math

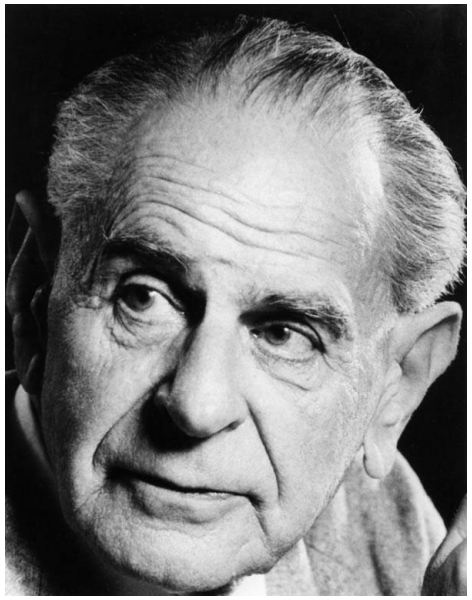


The proof is in the pudding

Data Science vs. Statistician video

Thoughts on the reading from Everybody Lies?





Let's take ~2 minutes to discuss the reading with someone you are sitting next to

- i.e., discuss anything you found interesting about the reading, etc.

Much of Freud's theory dealt with the subconscious

- E.g., Freudian slips

Karl Popper claimed that Freud's theories were unscientific because they couldn't be falsified

- i.e., can come up with any 'just so' story to explain a behavior

New data science analyses might make it possible to actually test Freud's theories

Introduction to Python

Programming languages for Data Science

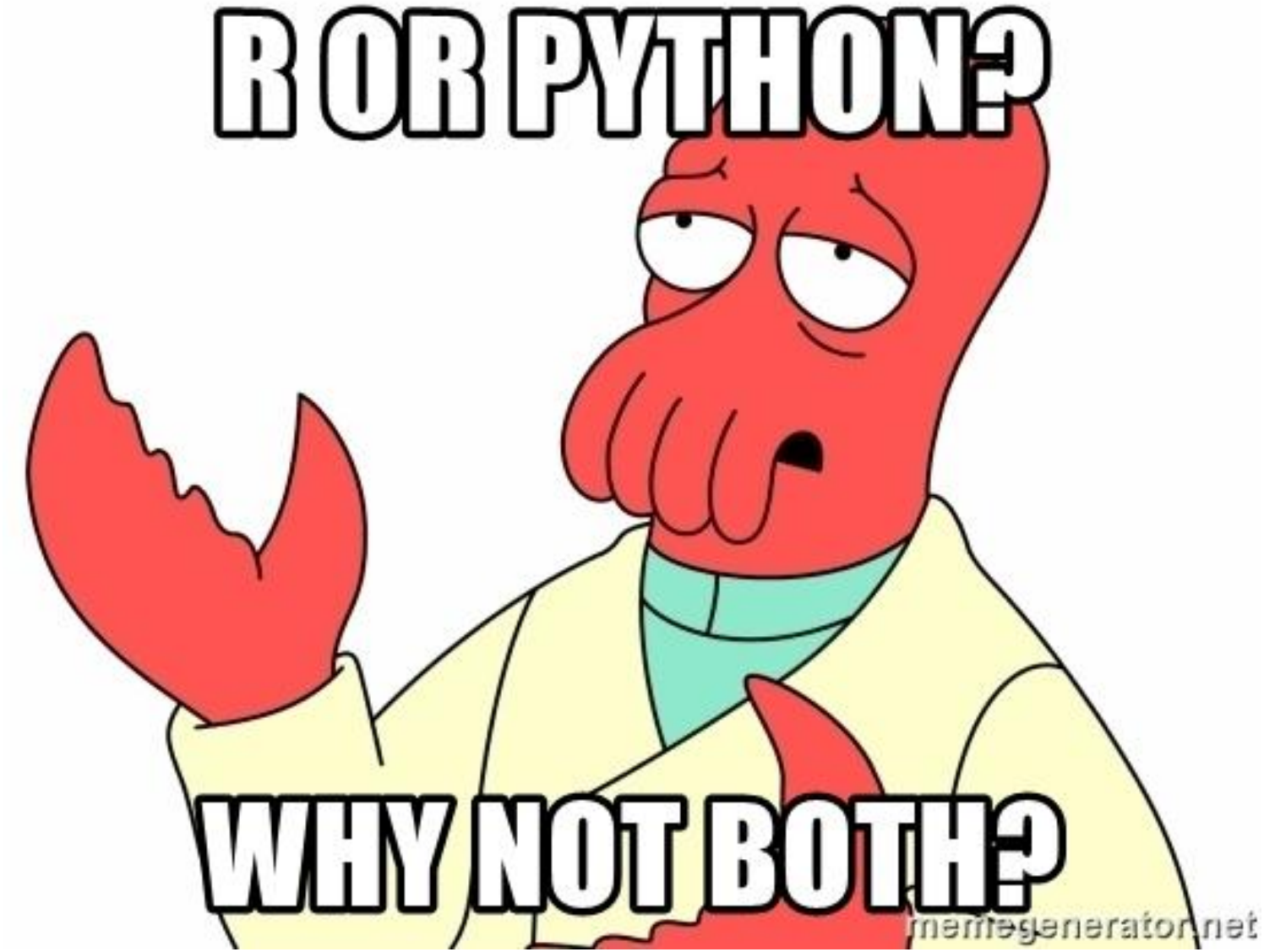
The two most popular languages for Data Science are:

General purpose programming language

- Can do a lot more than data analysis
- Code is easy to read
- Easy to write larger software packages
- Good machine learning package (scikitlearn)

Focused on data analysis

- Better for creating pdf reports
- Easy to create interactive apps
- RStudio created a great IDE and support


R OR PYTHON?
WHY NOT BOTH?
memegenerator.net

Terminology: scripts, modules, and packages

A **script** is a piece of code that is run to accomplish a specific task

- E.g., one could write and run a script to download a set of files

A **module** is a piece of code that reused by different scripts and programs

- Modules are ***imported*** by other programs/scripts to add commonly used functionality.
- E.g., import matplotlib.pyplot as plt

A **package** (library) contains several related modules

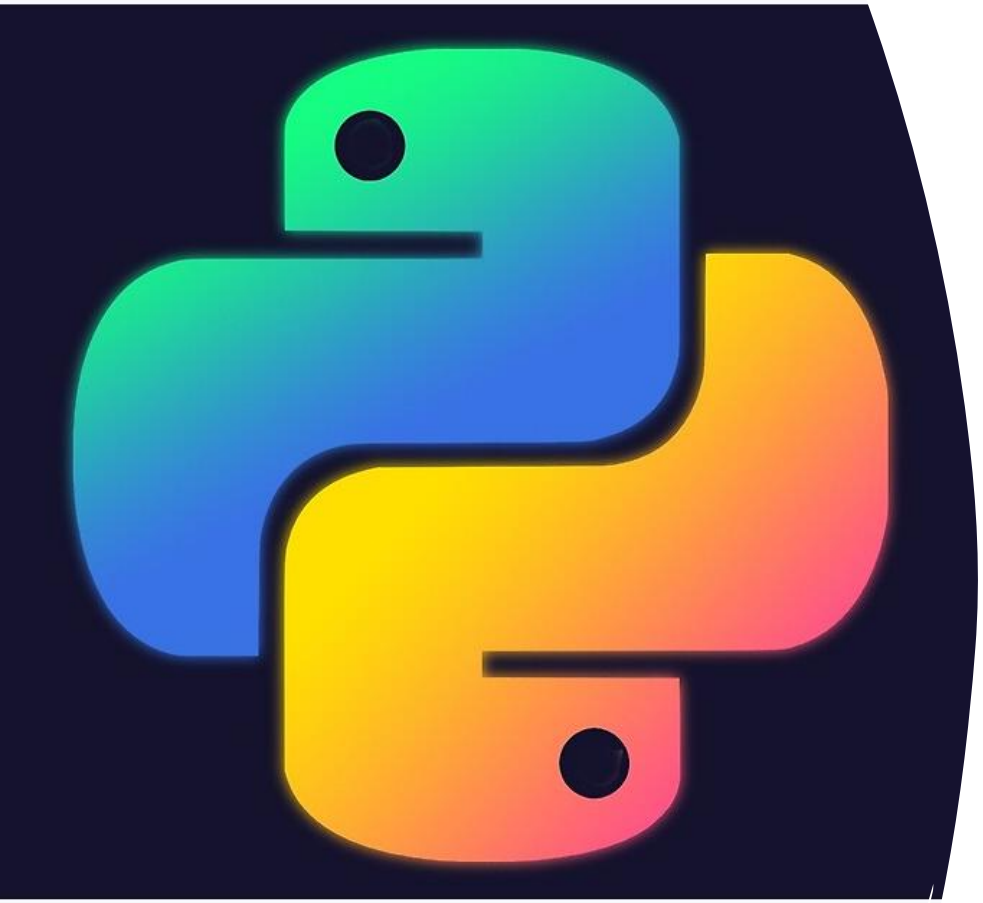
Packages are an essential building block in programming. Without packages, you would waste a lot of time writing code that's already been written

Jupyter notebooks

Jupyter notebooks allow one to create an analysis document that contains text, analysis code, and plotted results

Because one can see all the code used to generate results, it allows one to create reproduce analyses

We will do all our work in this class in Jupyter notebooks!



Programming in Python

Understanding the language fundamentals is important

Learn through practice, not only by reading or watching but by doing

- Like learning to ride a bike

If you need more practice, be sure to attend the Shivam's practice sessions!

Today you can follow along with the class 2 Jupyter notebook which we will download now!

Expressions

Expressions

Expressions describe how a computer should combine pieces of data

- They are evaluated to by the computer and return a value
- E.g., mathematical expressions
  - Multiplication: `3 * 4`
  - Exponentiation: `3**4`

Operation	Operation	Example	**Value**
Addition	+	2 + 3	5
Subtraction	-	2 – 3	-1
Multiplication	*	2*3	6
Division	/	7/3	2.667
Remainder	%	7 % 3	1
Exponentiation	**	2**.05	1.414

Syntax

The *Syntax* of a language is its set of grammar rules for how expressions can be written

- *SyntaxError* indicates that an expression structure doesn't match any of the rules of the language
- E.g., failed attempt at exponentiation: `3 * * 4`

```
  File "<ipython-input-2-012ea60b41dd>", line 1
    3 * * 4
        ^
SyntaxError: invalid syntax
```

Let's explore this in Jupyter!

Names

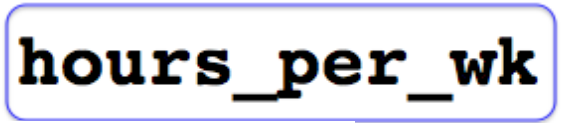
Assignment statements

Names store the values (from an expression)

- i.e., they are like variables in algebra

Names are assigned values using the = symbol

- E.g., my_number = 7



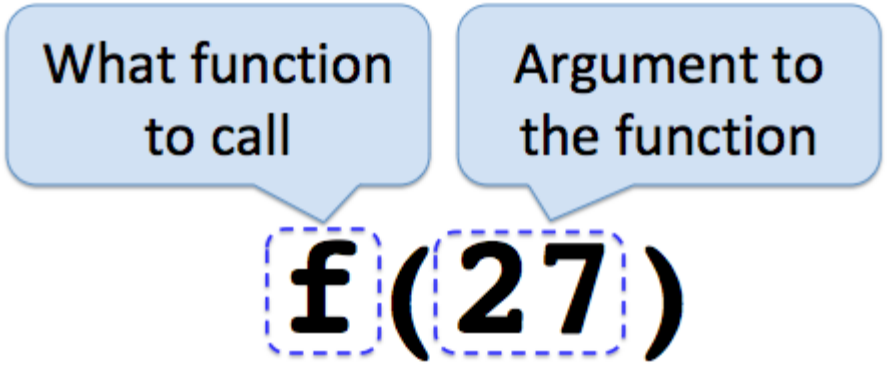
Let's explore this in Jupyter!

Call Expressions

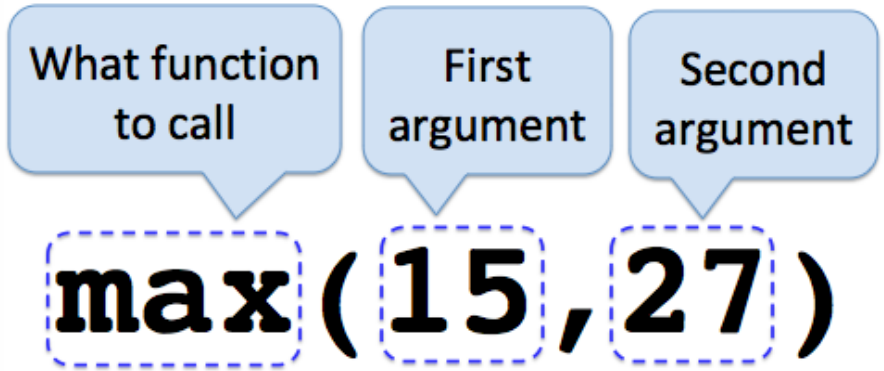
Anatomy of a Call Expression

Call expressions are expressions that call functions

- Functions take in one or more values (arguments) and (usually) return another value



Example: taking the maximum value



Let's explore this in Jupyter!

Numerical data

Arithmetic operations

Operation	Operation	Example	**Value**
Addition	+	2 + 3	5
Subtraction	-	2 – 3	-1
Multiplication	*	2*3	6
Division	/	7/3	2.667
Remainder	%	7 % 3	1
Exponentiation	**	2**.05	1.414

We can store the output of evaluating expression in names

- my_result = 10 * 2

Numbers in Python: Ints and Floats

Python has two basic number types

- **int**: an integer of any size
- **float**: a number with an optional decimal part

An int never has a decimal point, a float always does

- 3 # int of float?
- 2.7 # int of float?

A float might be printed using scientific notation

Notes on Floats

Three limitations of float values:

- They have limited size (but the limit is huge)
- They have limited precision of 15 - 16 decimal places
- After arithmetic, the final few decimal places can be wrong

Let's explore this in Jupyter!

Strings

Text and Strings

A string value is a snippet of text of any length

- 'a'
- "word"
- "there can be 2 sentences. Here's the second!"

Strings start and end with single or double quotes

- If the string starts with a single quote it must end with a single quote, and the same for double quotes

We can convert strings to numbers and numbers to strings

- int('12') # convert a string to an integer
- float('1.2') # convert a string to a float
- str(5) # convert an int to a string

Let's explore this in Jupyter!

Types

Every value has a type

We've seen several types so far:

- int: 2
- Built-in function: abs()
- float: 2.2
- str: 'Red fish, blue fish'

The type function can tell you the type of a value

- type(2)
- type('Red fish')

An expression's type is based on its value, not how it looks

- my_text = 2
- type(my_text)

Conversions

Strings that contain numbers can be converted to numbers

- int('12')
- float('1.2')
- float('one point two') # Not a good idea!

Any numeric value can be converted to a string

- str(5)

Numbers can be converted to other numeric types

- float(1)
- int(1.2) # DANGER: loses information!

Let's explore this in Jupyter!

Lists

Lists

Lists are ways to store multiple items

We can create lists using square brackets []

- my_list = [2, 3, 4]

We can also access list items using square brackets []

- my_list[2]

Lists can contain elements of different types

- my_list2 = [5, 6, 'seven']

Let's explore this in Jupyter!

Next class: Text manipulation!

Homework 1 has been posted

```
import YData
YData.download_homework(1)
```

It is due on Gradescope on Sunday September 7th at 11pm

- **Be sure to mark each question on Gradescope!**