WiFi
INDUSTRY-CORP
U: ga-guest
P: yellowpencil

2019-02-19

Intro to Python

Please install Python 3.7 from
https://anaconda.com/download

GENERAL ASSEMBLY

Intro to Python Agenda

What We'll Cover Today

In this class, we'll explore the following topics:

Time	Topic
15 min	Introductions
15 min	Python Overview
15 min	Software Install
40 min	Programming Basics
20 min	Example Program
15 min	Create a Learning Plan

Steve Iannaccone

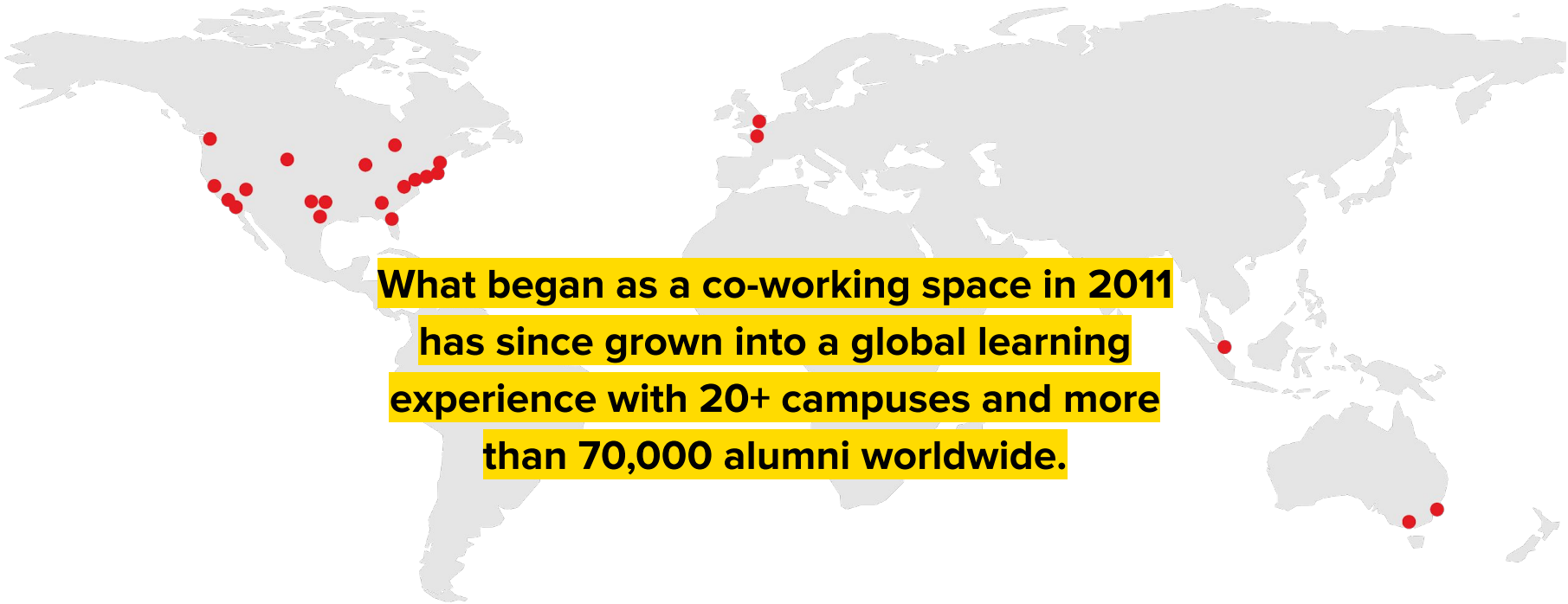
Business Intelligence at Ibotta

Data Scientist at Gaia Inc.

Business Intelligence at Univ. of Colorado

What Is General Assembly?

General Assembly is a pioneer in education and career transformation, specializing in today's most in-demand skills. We foster a flourishing community of professionals pursuing careers they love.



What began as a co-working space in 2011 has since grown into a global learning experience with 20+ campuses and more than 70,000 alumni worldwide.

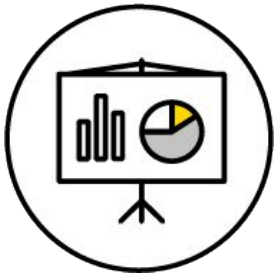
Atlanta	Chicago	Houston	Melbourne	Paris	San Diego	Singapore	Toronto
Austin	Dallas	London	New York City	Phoenix	San Francisco	Stamford	Washington, D.C.
Boston	Denver	Los Angeles	Orlando	Providence	Seattle	Sydney	

General Assembly is a pioneer in education and career transformation, specializing in today's most in-demand skills. We foster a flourishing community of professionals pursuing careers they love.

What We Teach

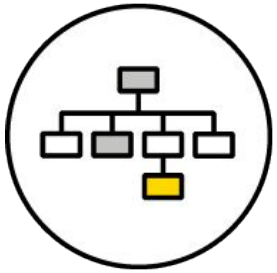
Coding

UX & Design



Data

Marketing



Business

Career Development

Intro to Python Agenda

What We'll Cover Today

In this class, we'll explore the following topics:

Time	Topic
15 min	Introductions
15 min	Python Overview
15 min	Software Install
40 min	Programming Basics
20 min	Example Program
15 min	Create a Learning Plan

About this course

Learning Objectives

- Discuss the history of Python & how it's used in different industries
- Describe the benefits of a Python workflow when looking at data
- Demonstrate basic Python programming fundamentals to solve a real world problem
- Create a custom learning plan to build your data science skills after this workshop!

About this course

Getting the most out of this course

- Make sure you have the tools you need running smoothly
- Think / ask how Python could fit into your workflow
- The exercises are guidelines, pursue your interests in during practice
- Plan how you will continue your learning

“

Today dozens of Google engineers use Python, and we're looking for more people with skills in this language.

Peter Norvig,

Director of search quality at Google, Inc.

Intro to Python

What Is Python?

What is Python

- Created by Guido Van Rossum in 1991
- Emphasizes **productivity** and code **readability**
- **Easy** to pick up and learn
- Easier for many to contribute to **production level code**
- **Readable** code means that almost anyone can read and **understand what code is doing**

Why is Python readable

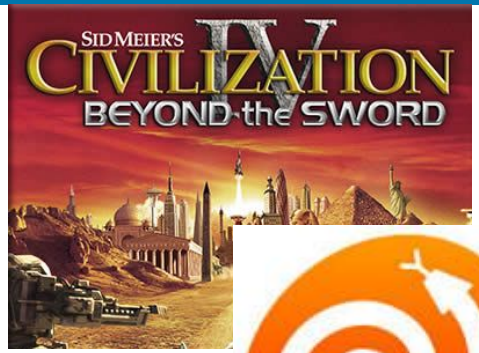
- **Interpreted language**:
  - Step by step execution for easier programming ideation
  - Write once, run anywhere
  - Performance tradeoff
- **Object-oriented (OO)**
  - Code with objects that contain data and functions to manipulate it in predefined ways
- **High-level programming**
  - Use natural language syntax where possible

Why Python?

- **Free, Flexible,** and **Open Source**
- Rapid **prototyping** and **full-stack** commercial applications
- **Extensible** with easy to install **libraries**
- Great **documentation**
- Established and growing **community**
- **Scripts** can be run many times
    - When data changes
    - On different machines
    - At different times

Real Cases:
Who uses Python?

- **Industry & Academia**
  - AstroPy
  - BioPython
- **Web Development**
  - Youtube
  - DropBox
- **Game Development**
  - Civilization IV
- **Standalone Applications**
  - BitTorrent

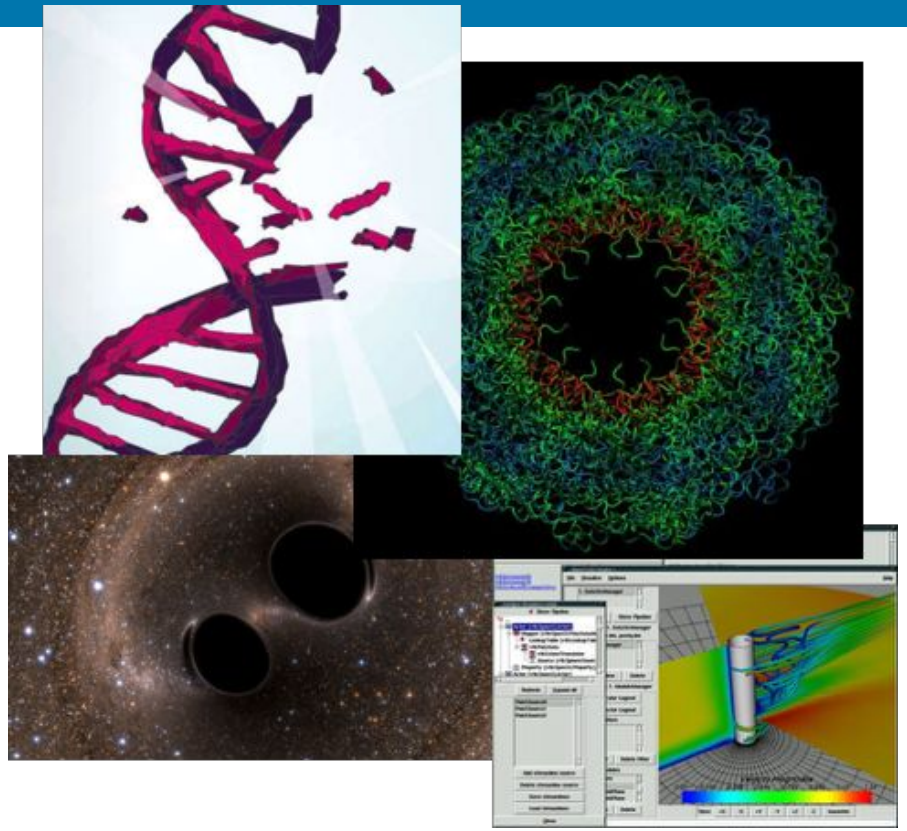


BitTorrent™

Real Cases:

Examples

- **Industry**
  - Drug discovery
  - Financial services
  - Films and special effects
- **Academia**
  - Gravitational waves
  - Scientific visualisation
  - Biomolecule simulation
- **More**
  - Success Stories



Discussion:
Storytime

How will you use python?

- Job or Industry
- Workflow Improvements
- Dream Projects

“

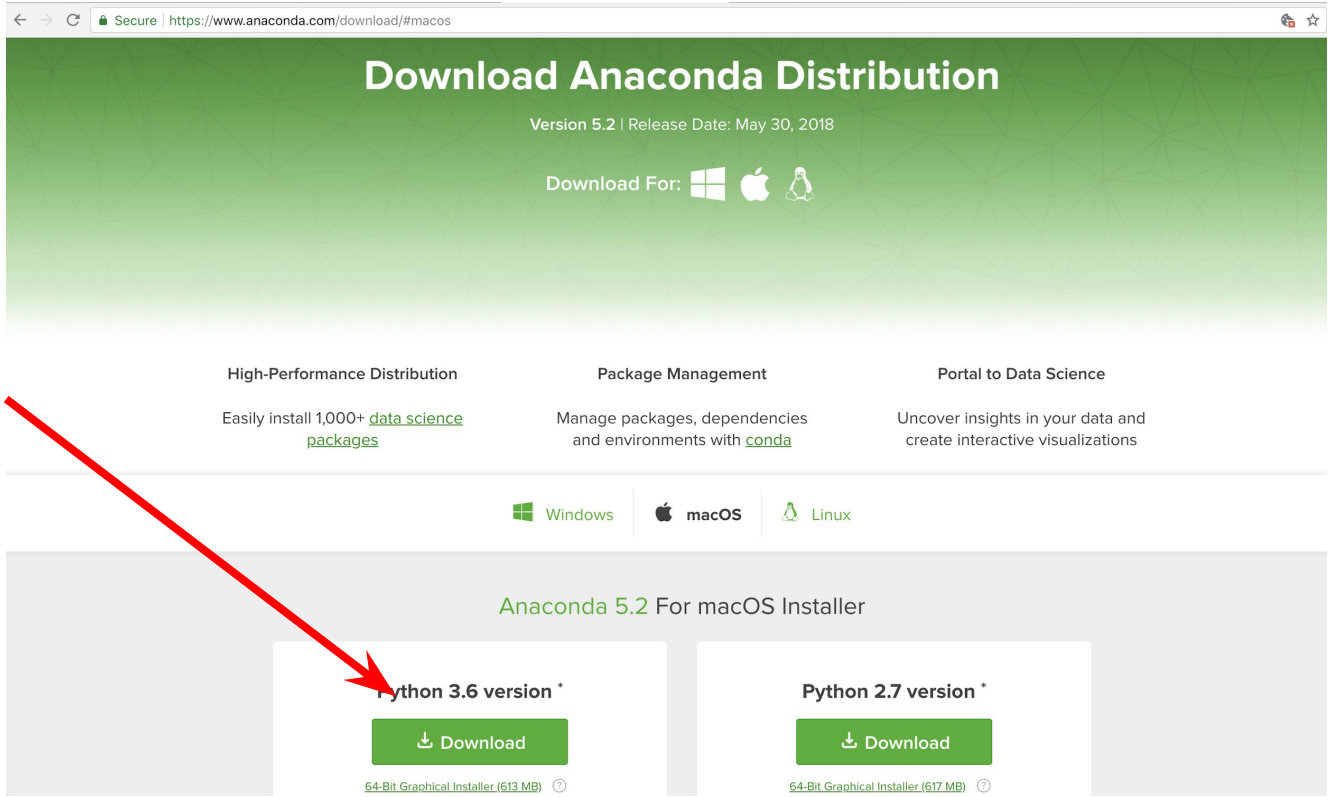
No great marketing decisions have ever been made on qualitative data.

John Sculley,
Former VP PepsiCo + CEO Apple

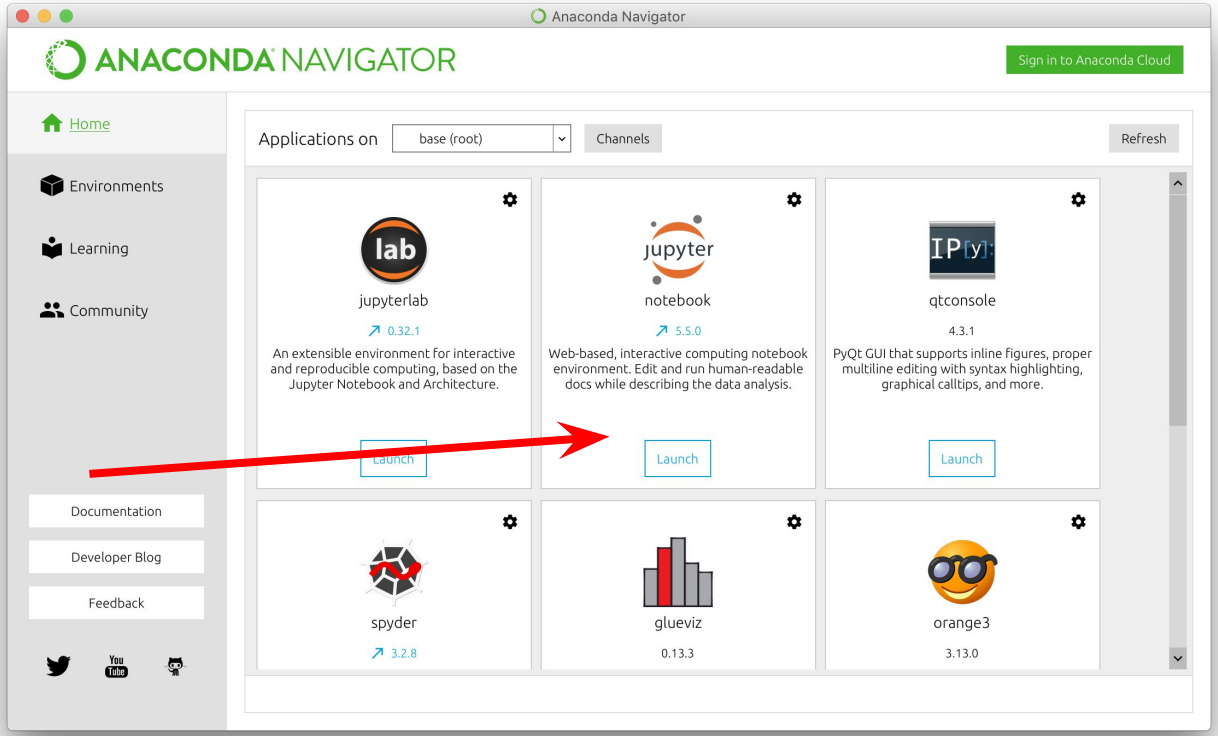
Intro to Python

Exploring Python With Anaconda

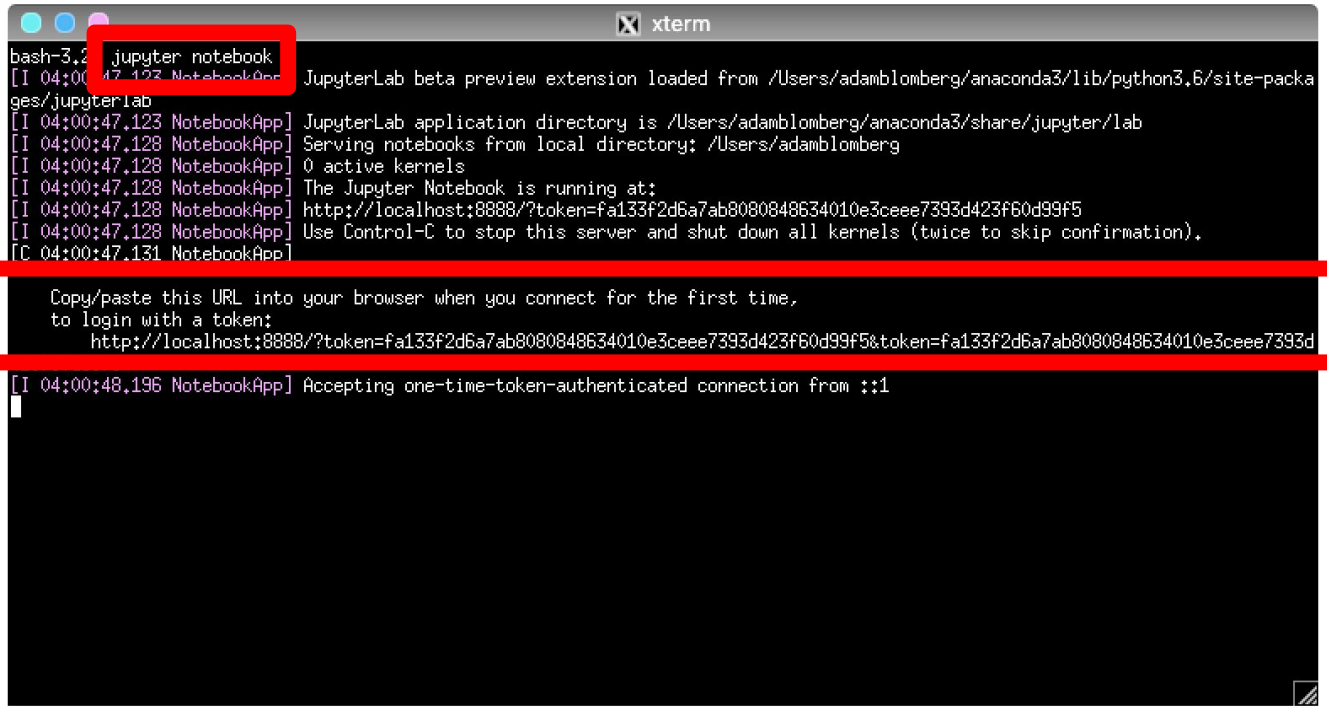
Download Anaconda https://anaconda.com/download



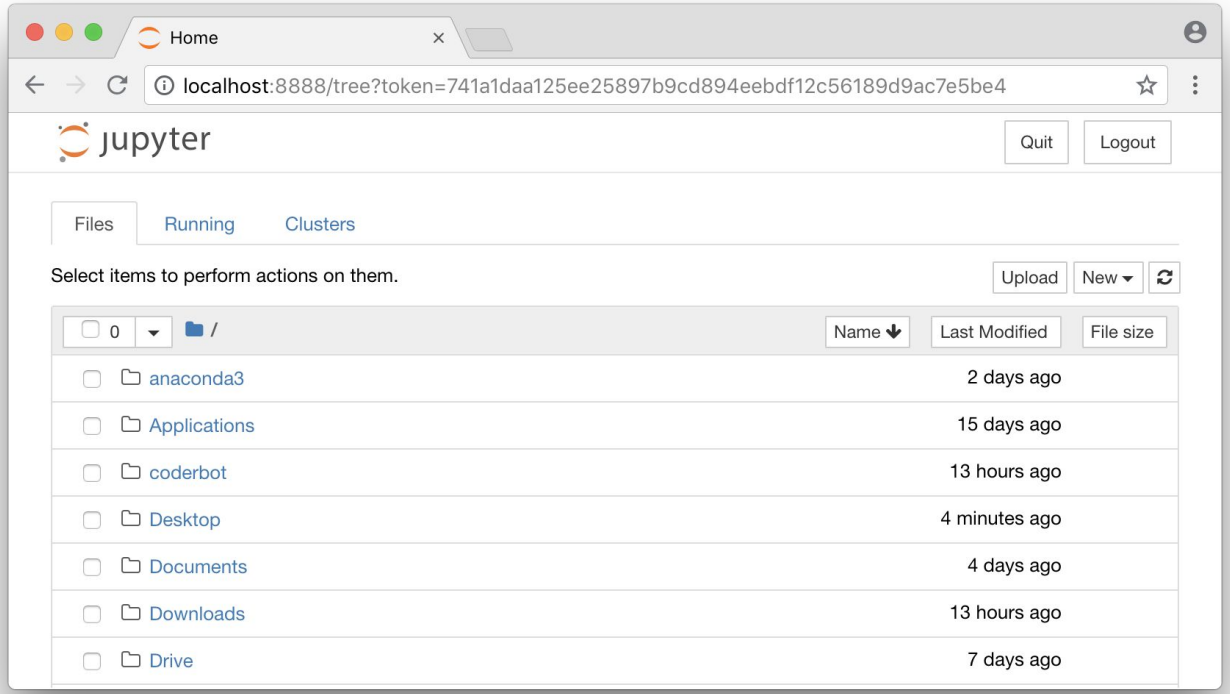
Anaconda Navigator



Terminal / Anaconda Prompt

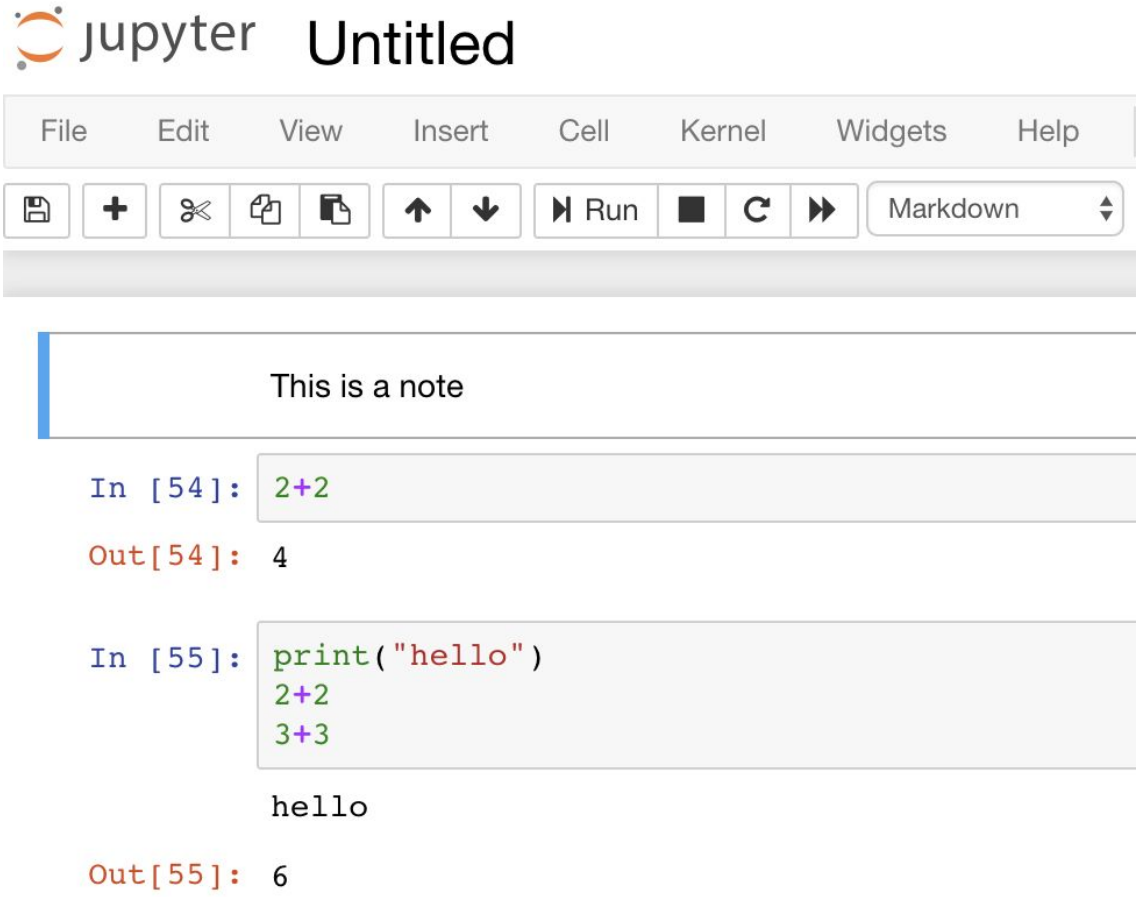


New Web Browser window with personal files



Jupyter Notebook

- **Cells**:
  - **Markdown** for notes
  - **Code** for Python
- **Modes**
  - **Blue** for commands
  - **Green** for editing
- **Execution**
  - Shift + return
- **Output**
  - Print (all)
  - Return values (last)



Jupyter Notebook errors

Mistakes happen! Here's what they look like:

```
just some code
```

```
  File "<ipython-input-56-2516a36d8922>", line 1
    just some code
              ^
SyntaxError: invalid syntax
```

1. **Try to understand what went wrong**
2. **Attempt to fix the problem**
3. **Execute the cell again**

Solo Exercise:

Try out Jupyter notebook

5 minutes

Take some time to try out Jupyter notebook.

Make sure you can:

1. Convert cells between Markdown and Code
2. Edit and execute a note cell
3. Edit and execute a code cell
4. Try to do some math
5. Make an error

“

We wrote more code in python than we were expecting, including all [the] in-game screens and the main interface.

Soren Johnson,
Lead designer on Civilization IV, Firaxis Games

Intro to Python

Python Programming Fundamentals

Programming Fundamentals Framework

Every programming language can be broken down into core components. Having a general framework for this context will help us learn specifics for Python.

- **Syntax**
  - The structure of the commands given to the computer
- **Variables**
  - How computers store information
- **Control Structures**
  - Sets the hierarchy/priorities of programming logic
- **Data Structures**
  - How computers organize data

Syntax

The **syntax** of a programming language is the set of rules that define the combinations of symbols that are considered to be correctly structured programs in that language

Just like our spoken languages have structures like paragraphs and sentences, programming languages have **code blocks** and **statements**.

Variables

Variables are symbolic names that store a specific piece of information. Variables come in different types in order to hold different kinds of information.

- For example: **r = 3**
  - Defines a variable: named “**r**”
  - This holds a numerical integer: **3**

- Other types of variables:
  - float (3.14)
  - string (“Hello”)

Control Structures

A block of programming that analyses variables and chooses a direction in which to go based on given parameters is a **control structure.**

- The term **flow control** details the direction the program takes (how program logic “flows”). It determines how a computer will respond when given certain conditions and parameters. Some typical structures include:
  - **If** statements
  - **For** loops
  - **Functions**

Data Structures

A **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently.

- Some examples in Python include:
  - Lists
  - Tuples
  - Dictionaries
  - DataFrames

Python programming

- Let’s see what a Python program looks like.
- We’ll start with the classic "*Hello World*!" program:
    - This code will print the message "Hello World!" on the screen.

```
1 | print("hello world")
```

What did we just see?

- Data
  - “hello world”
  - string
  - Denoted by quotation marks
    - Single ‘…‘
    - Double “…”
    - Triple “””…”””
- Function
  - Print
- Aside: # makes comments

```
1 | print("hello world")
```

Variables

- **Types**
  - **Bool** - 1 or 0, True or False
  - **Int** - number w/o decimal point
  - **Float** - number w/ decimal
  - **String** - text
- **Assignment**
  - = operator defines variable
- **Variable names**
  - snake_case
  - Lowercase Letters, Numbers, Underscores
- **type() function** - shows data type

```
1  # variable assignments
2  x = 1.0
3  my_variable = 12.2
4  type(x)
5
6  y = 1
7  type(y)
8
9  b1 = True
10 type(b1)
11
12 s = "String"
13 type(s)
```


Variables

- **Operators -** combine data
  - **+, -, *, /** (add, subtract, multiply, divide)
  - **+=, -=, etc** (perform operation and save result)
  - ** (power)
  - **//, %** (floor division, remainder – *or modulus*)
  - **>, <, ==, !=, <=, >=** (greater than, less than, equal, not equal, etc)
- **Functions and methods** – saved instructions to manipulate data
  - **abs(**my_int**) -** function_name(data)
  - **len(**my_string**)**
  - my_string**.lower() –** data.method_name()
  - my_int**.__add__(5)**
    - **"Dunder" -** double underscore (__) = important Python built in item

Partner Exercise:

Practice in Jupyter notebook

5 minutes

Now you try! Pair up with a partner to attempt the following in your notebooks. Help each other out!

1. Create variables
    a. Bool
    b. Int
    c. Float
    d. String
2. Check their class with type()
3. Try out some operations on your variables
    a. What works? What causes errors?

Data Structures

- **Lists**
  - A **collection** of objects
    - Mixed types are okay
  - Defined with **square brackets** [ ]
  - They can be modified
    - my_list**.append()**
    - my_list**.remove()**
  - **Slicing**
    - Access elements
    - my_list[start **:** end **:** step]

```
1  l = [1,2,3,4]
2  print(type(l))
3  print(l)
4  print(l)
5  print(l[1:3])
6  print(l[::2])
7
8  # Python starts counting from 0
9  print(l[0])
```


Data Structures

- **Tuples**
  - very similar to lists, but:
    - They are defined with parentheses ( ) instead of square brackets
    - They cannot be changed
      - No append() method
      - No remove() method
  - Slicing works the same way

```
1 | point = (10, 20)
2 | print(point, type(point))
3 |
4 | x, y = point
5 | print("x =", x)
6 | print("y =", y)
```


Data Structures

- **Dictionaries**
  - Collections of **key/value pairs**
  - Defined by curly brackets { }
  - Slicing uses **keys**
  - **Order** is not preserved

Python

```
1 params = {"parameter1" : 1.0, "parameter2" : 2.0,
2 "parameter3" : 3.0,}
3 print(type(params))
4 print(params)
```

Control Structures

- **If / elif / else**
  - Check conditions with boolean operators (i.e. <, >, ==)
  - Execute a single code block depending on result
  - If True: code runs
  - Can be:
    - if alone
    - if/else
    - if/elif(s)/else
  - Indentation controls end of **if** block
  - Data controls which code runs

Python

```
1  if age_person > 18:
2      return "They can drive"
3  else:
4      return "They cannot drive"
```

Control Structures - if

Python

```
1  A = 10
2  B = 100
3  if A>B:
4      print("A is larger than B")
5  elif A==B:
6      print("A is equal to B")
7  else:
8      print("A is smaller than B")
```


Control Structures

- **for** loop
  - Repeat operations
  - Loop variable takes each value from list in turn
  - Indentation controls end of loop
  - Watch out for infinite loops!
    - Interrupt or restart kernel when this happens

Python

```python
users = ["Jeff", "Jay", "Theresa"]

for user in users:
    print("Hello %s" % user)
```


Control Structures

- **Functions**
  - Groups of instructions repeat / create abstractions of common tasks.
  - Divide our code into useful blocks
  - Provide order, making the code more readable and reusable
  - def name**(input1, input2, ...)** :
  - First line is “Document String” describing how function works
  - Definition saves instructions only – no execution!

Python

```
1  def square(x):
2      """
3      Return the square of x.
4      """
5      return x ** 2
```

Control Structures

- **Functions**
  - Run functions with parentheses after the name
  - Needs to be defined before it can be executed!
  - The return value is saved in var2
  - Other functions can be run on the result!

Python

```
1  var1 = 7
2
3  var2 = square(var1)
4
5  print(var2)
```

Expanding python

- **Packages**
  - Install new libraries to add functionality to Python:

    conda install <name>

    or

    pip install <name>
  - Use packages by importing them into your Python scripts

```
1 import math
2 x = math.cos(2 * math.pi)
3 print(x)
4
```


Real Cases: Expanding python

Common Packages

- Data manipulation: pandas, Numpy, scipy
- Machine Learning: scikit-learn, nltk
- Databases: psycopq2, sqlalchemy
- Visualizations: matplotlib, plotly, bokeh
- API calls / web scraping: requests, BeautifulSoup, Scrapy
- Web development: Django, Flask, Twisted, Scapy
- Game Development: Pygame, Pyglet
- Desktop App: pyQt, Tkinter

More

“

I have students learn Python in our undergraduate and graduate Web courses. Why? Because there's nothing else with the flexibility or as many pre-built libraries...

Prof. James A. Hendler, Univ. of Maryland

Intro to Python

Python Programming Practice!

Guided Walk-Through:

Data Analysis Datafile

We are going to use a Jupyter Notebook to classify some flowers!

- Go to: https://github.com/whsky/intro_to_python
- Click on the `iris_classification.ipynb` file
- Click on the `Raw` button and save the file to your computer (make sure you don't add an extension like '.txt' to the file when you save it!)

“

The goal is to turn data into information, and information into insight.

Carly Fiorina,

Former CEO, Hewlett-Packard

Intro to Python

Let's Review

Review and Recap

In this workshop, we've covered the following topics:

- Python as a popular, flexible programming language
- Python has applications in many different areas
- Python is particularly great for data manipulation
- Python programming basics include: types, variables, functions, and more!

Finish That Sentence

What are your biggest takeaways from this lesson?

"Something that really got me thinking is..."

"The best thing I got out of this unit is..."

"I discovered..."

"I still want to learn about..."

"I was surprised to learn that..."

Check your inbox!

We sent you a survey.

Please take 5 minutes to tell us what you thought of this class.

We Want Your Feedback!

Let us know what you liked about this class and what we can improve.

Complete a quick survey at:

ga.co/introclass

Intro to Python

Next Steps

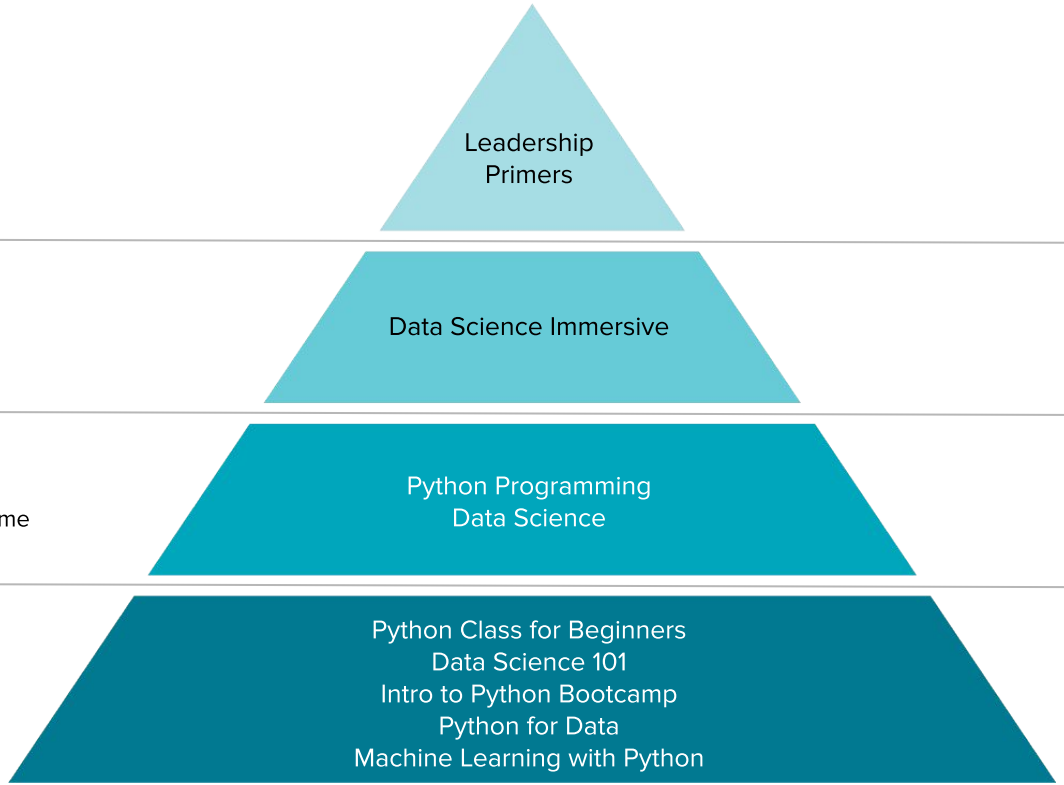
Our Python Products: Learn With Structure

Level IV
Customized Training for Leaders
Contact us for pricing

Level III
Switch Careers
3 Months | Full-time
$16,000

Level II
Advance Your Career
1 or 10-week courses | Part-time
$1,000–$4,000

Level I
Master the Basics
2 hours–2 days
$0–$500



Take Your Next Step with GA

	Python Programming	Data Science Immersive
Format	Part-time/evenings	Full-time/Monday–Friday
Next Course	Nov 18th	Dec 9th
Outcome	Learn python programming skills, and translate new knowledge into career gains. Working with experts, you'll build your own custom web application.	Make smarter decisions by gaining the data analytics, data modeling, programming and statistics skills you need to start a career in data.
Upcoming	**Upcoming Data & Coding Workshops:** SQL Bootcamp (10/18) Crafting Stories with Data (11/5)	

**Financing options available.*

Questions?

See you next time!

THANK YOU!

GENERAL ASSEMBLY


GA