EE 551 A
Programming in Python
Fall 2018

Lecture 1

08/31/2018

Assistant Prof. Sergul Aydore

Department of Electrical and Computer Engineering

STEVENS
INSTITUTE *of* TECHNOLOGY
THE INNOVATION UNIVERSITY®

1870

About Me

- **Assistant Prof** at ECE department at **SIT** since *August 2018*
- **Machine Learning Scientist** at **Amazon** *2016 – 2018*
- **Data Scientist** at **JP Morgan** *2015-2016*
- **Postdoctoral Researcher** at **Columbia University** *2014 – 2015*
- **PhD Student** at Signal and Image Processing Institute of **University of Southern California** *2009 – 2014*
- **BS and MS student** at Electrical Engineering from **Bogazici University** *2002 - 2009*

My Interests

- Machine Learning
- Signal Processing
- Software Development
- Brain Imaging
- Teaching

- http://www.sergulaydore.com/

LOOKING FOR STUDENTS WHO WANT TO DO RESEARCH ON MACHINE LEARNING WITH ME!

About the Course

- Will use canvas for communication and grades
- Everything will be shared in public Github repo:

  https://github.com/sergulaydore/EE551A_Fall_2018

- Tools that students will learn how to use are:
  - Git, github, Pycharm, terminal
- Tools that I will be using to organize assignments are:
  - Github education: https://classroom.github.com/
  - Travis CI: https://travis-ci.com/

Survey

- Distribute the survey
- Students make groups of 4 and select a representative

General Information

- **Meeting Times**: Fridays at 6.15pm-8.30pm
- **Classroom Location**: Main Campus, M (Morton) 103
- **Instructor**: Assistant Prof. Sergul Aydore
- **Contact Info**: Burchard Building 211, saydore@stevens.edu
- **Office Hours:** Fridays at 3.30pm-6.00pm
- **Course Web Address:** https://github.com/sergulaydore/EE551A_Fall_2018
- **Cross-listed with:** CPE 551A

Course Description

This course presents tool, techniques, algorithms, and programming techniques using the Python programming language for data intensive applications and decision making. The course formally introduces techniques to: (i) gather,(ii) store, and (iii) process large volumes of data to make informed decisions. Such techniques find applicability in many engineering application areas, including communications systems, embedded systems, smart grids, robotics, Internet, and enterprise networks, or any network where information flows and alters decision making.

STUDENT LEARNING OUTCOMES

- After successful completion of this course, students will be able
  - Program basic algorithms in Python
  - Extract and analyze data in Python
  - Prepare for their future career in Technology related fields

FORMAT AND STRUCTURE

- The classes will include dynamic illustrations of the concepts. Students are expected to bring their laptops and run the programs in their own laptops.

- There will be pop quizzes at the beginning/end of some classes.

- There will be weekly coding assignments. The students will return assignments via github which will be tested using continuous integration tool Travis-CI.

- Final exam will be handwritten! Because tech companies will interview you on a white board!

Course Materials

- **Textbook(s):** Learning Python (5E) by Mark Lutz
  - **Other Readings:** How to think like a Computer Scientist by Downey, Elkner, Meyers

Course Requirements

- **Attendance** Attendance is crucial for an effective learning but will not be graded. However, there will be pop quizzes.
- **Participation** During the lectures, randomly selected students might be asked questions about the concepts taught in the lectures, but these will not be graded.
- **Homework** The assignments will be submitted via github and graded and tested in the cloud.
- **Quizzes** Pop quizzes will be given at the end of the lectures.
- **Project(s)** This is optional. Students are welcome to work on a project they desire.
- **Exams** There will be a single final hand-written exam.

Grading Procedures

- Grades will be based on:
  - Homework (40 %)
  - Quizzes or Individual Project (20 %)
  - Final Exam (40 %)

- Late Policy:
  - Assignments submitted after the deadline will not be graded.
  - There will be no make-ups for the quizzes.

Graduate Student Code of Academic Integrity

- *All Stevens graduate students promise to be fully truthful and avoid dishonesty, fraud, misrepresentation, and deceit of any type in relation to their academic work. A student's submission of work for academic credit indicates that the work is the student's own. All outside assistance must be acknowledged. Any student who violates this code or who knowingly assists another student in violating this code shall be subject to discipline.*

- All graduate students are bound to the Graduate Student Code of Academic Integrity by enrollment in graduate coursework at Stevens. It is the responsibility of each graduate student to understand and adhere to the Graduate Student Code of Academic Integrity. More information including types of violations, the process for handling perceived violations, and types of sanctions can be found at www.stevens.edu/provost/graduate-academics.

Special Provisions for Undergraduate Students in 500-level courses

- The general provisions of the Stevens Honor System do not apply fully to graduate courses, 500 level or otherwise. Any student who wishes to report an undergraduate for a violation in a 500-level course shall submit the report to the Honor Board following the protocol for undergraduate courses, and an investigation will be conducted following the same process for an appeal on false accusation described in Section 8.04 of the Bylaws of the Honor System. Any student who wishes to report a graduate student may submit the report to the Dean of Graduate Academics or to the Honor Board, who will refer the report to the Dean. The Honor Board Chairman will give the Dean of Graduate Academics weekly updates on the progress of any casework relating to 500-level courses. For more information about the scope, penalties, and procedures pertaining to undergraduate students in 500-level courses, see Section 9 of the Bylaws of the Honor System document, located on the Honor Board website.

Learning Accomodations

- Stevens Institute of Technology is dedicated to providing appropriate accommodations to students with documented disabilities. For more information about Disability Services and the process to receive accommodations, visit https://www.stevens.edu/office-disability-services. If you have any questions please contact: Phillip Gehman, the Director of Disability Services Coordinator at Stevens Institute of Technology at pgehman@stevens.edu or by phone (201) 216-3748.

- See https://sit.instructure.com/courses/27243/pages/syllabus for the rest

Why Python?

- Software Quality
  - Simple and readable syntax; hence reusable and maintainable
  - Python seems to `fit your brain` - features of the language interact in consistent and limited ways and follow naturally from a small set of core concepts.
  - Easier to learn, understand and remember
  - Python adopts a somewhat minimalist approach. Explicit is better than implicit, and simple is better than complex.

Why Python?

- Developer Productivity
  - Python code is typically one-third to one-fifth the size of equivalent C++ or Java code.
    - Less to type, less to debug and less to maintain
  - Python programs run immediately without the lengthy compile steps.
  - Python allows programmers to get done with less effort.

Why Python?

- Program Portability
  - Most Python programs run unchanged on all major computer platforms.
  - Porting Python code between Linux and Windows, for example, is usually just a matter of copying scripts between machines.

Why Python?

- Support Libraries
  - Python comes with a large collection of prebuilt and portable functionality, known as the standard library.
  - In addition, can be extended with both homegrown libraries and a vast collection of third-party software.
  - e.g. NumPy: a third party extension is a free and more powerful equivalent to the Matlab numeric programming system.

Why Python?

- Component Integration
    - Python scripts can easily communicate with other parts of an application.
    - Python code can invoke C and C++ libraries, can be called from C and C++ programs, can integrate with Java components, etc.

Why Python?

- Enjoyment
  - Because of Python's ease of use and built-in toolset, it can make the act of programming more pleasure than chore.

Is Python a "Scripting language"?

- Python is a general-purpose programming language that is often applied in scripting roles.

- The term "scripting" often implies languages that don't require an expilict compilation step.

- Python is often used without a compilation step but can be compiled, too.

- So, in general, the term "scripting" is probably best used to describe the rapid and flexible mode of development that Python supports.

OK, but What's the Downside?

- The only significant universal downside to Python is that, as currently implemented, its execution speed may not always be as fast as that of fully compiled as lower-level languages such as C and C++.

- If your domain requires optimal execution speeds, you can still use Python – simply split off the parts of the application that require optimal speed into compiled extensions, and link those into your system for use in Python scripts.

Who uses Python today?

- Roughly 1 million users around the world
- Generally considered top 5 most widely used programming languages in the world.
- Companies: Google, Amazon, JP Morgan Chase, Netflix, Dropbox, NASA, Intel, Cisco, etc.

Projections of future traffic for major programming languages
Future traffic is predicted with an STL model, along with an 80% prediction interval.
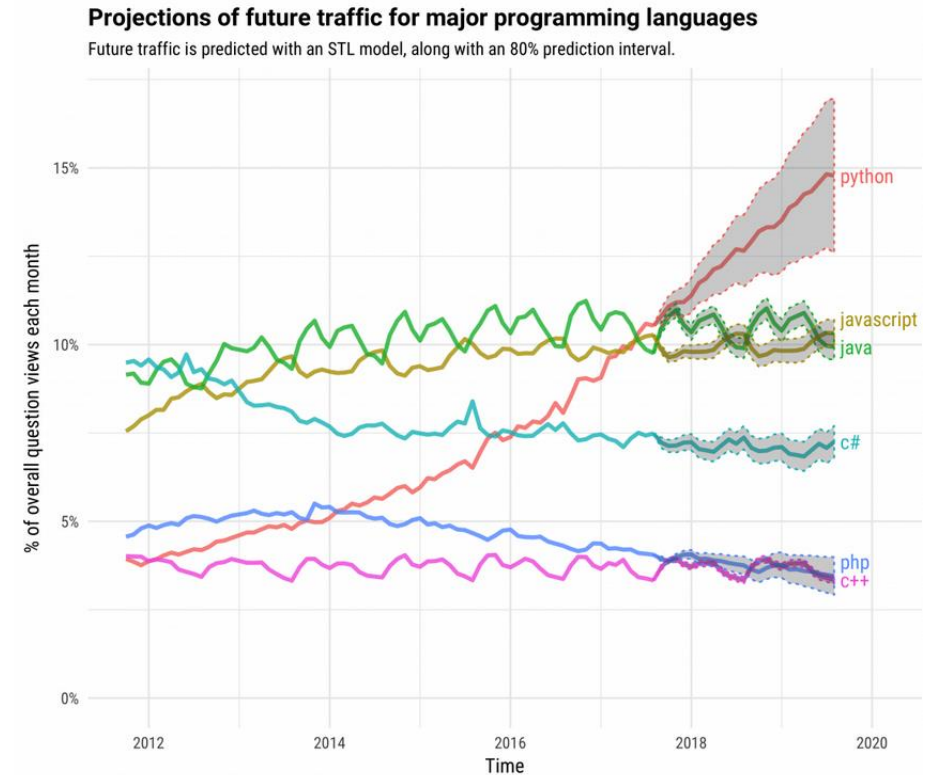
What can I do with Python?

- System Programming
- GUIs
- Internet Scripting
- Component Integration
- Database Programming
- Data Mining
- Rapid Prototyping
- Numeric and Scientific Programming
- Data Science and Machine Learning
- Gaming, Robots, and more…

What are Python's Technical Strengths?

- It's Object-Oriented and Functional
- It's free
- It's portable
- It's powerful
  - Dynamic typing
  - Automatic memory management
  - Programming-in-the-large support
  - Built-in object types
  - Built-in tools
  - Library utilities
  - Third-party utilities
- It's mixable
- It's relatively easy to use
- It's relatively easy to learn

Know your Tools for Programming!

- Command Line:
  - Read this tutorial

  https://philchodrow.github.io/cos_2017/1_terminal_and_git/Introduction%20to%20terminal.pdf
  - Windows users:
    - Install Linux - recommended!
    - OR use Cygwin to use linux command line

    https://www.youtube.com/watch?v=uTeH7vm8JZU

Know your Tools for Programming!

- Git – version control
- Github - "Facebook" for coders

https://github.com/scikit-learn/scikit-learn/pull/10020

- If you are not familiar with git and Github,
  - Watch Git essential training course by Kevin Skoglund at Lynda

classroom.github.com

A private repository will be created for each student
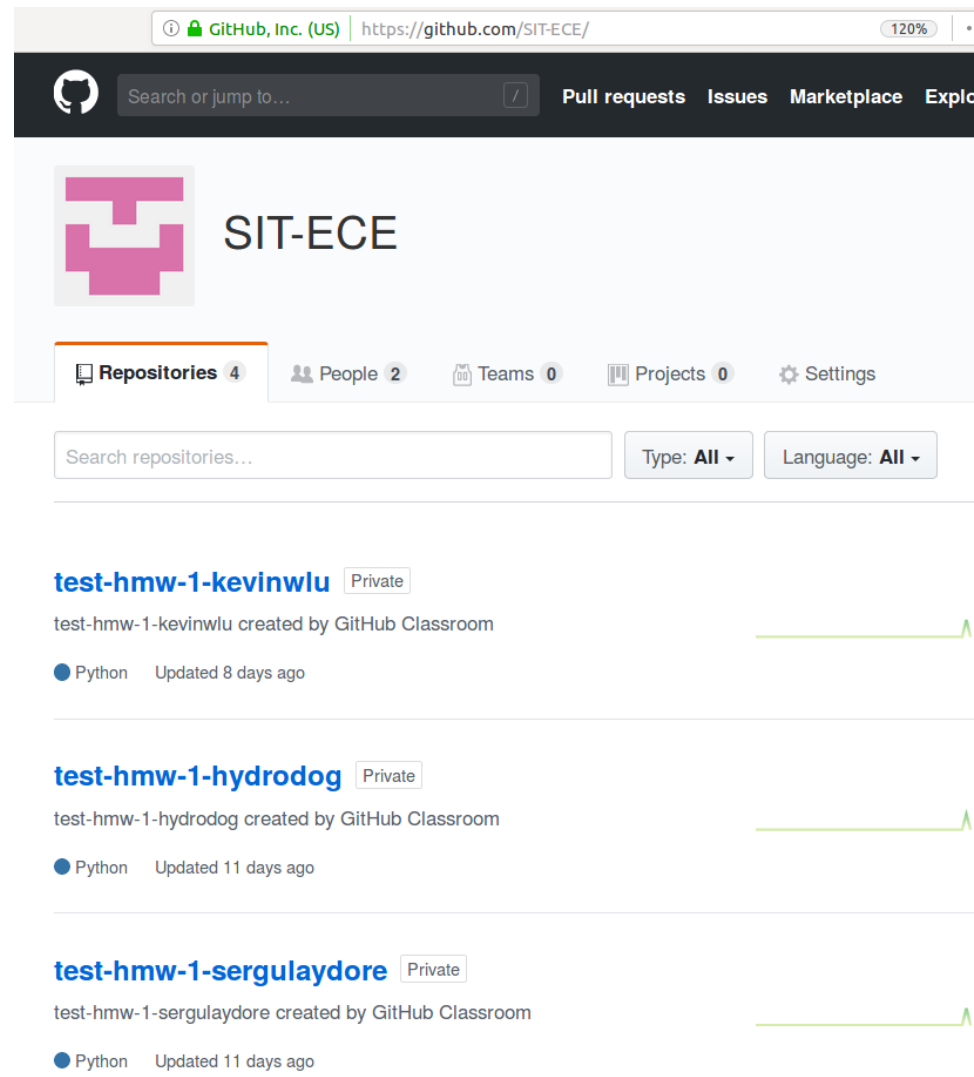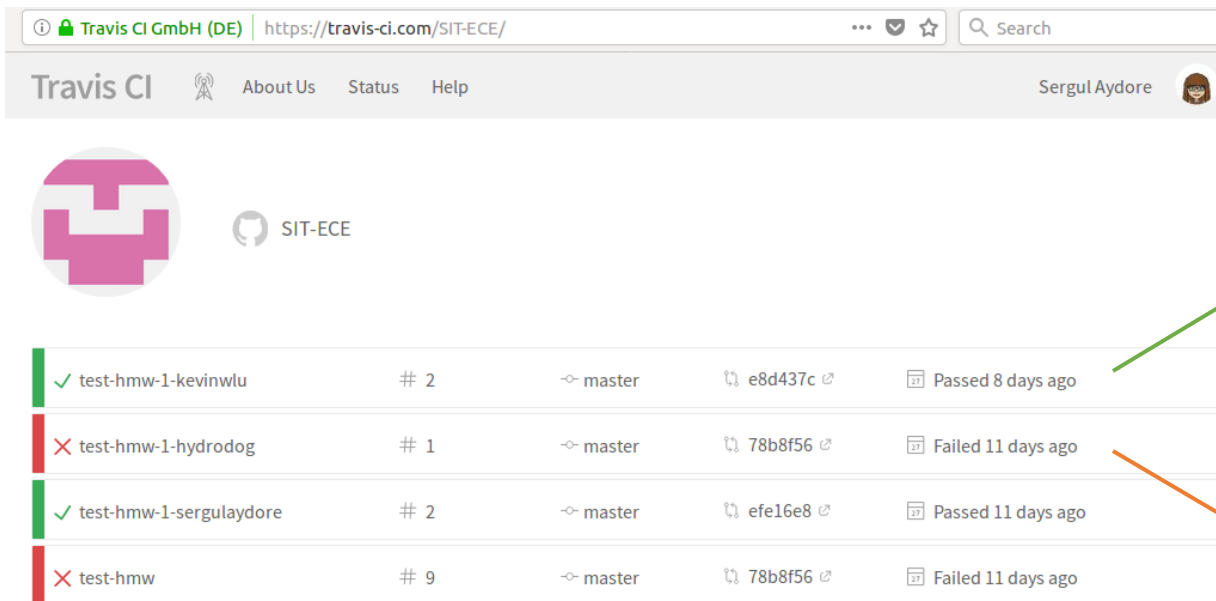
I can see all these!
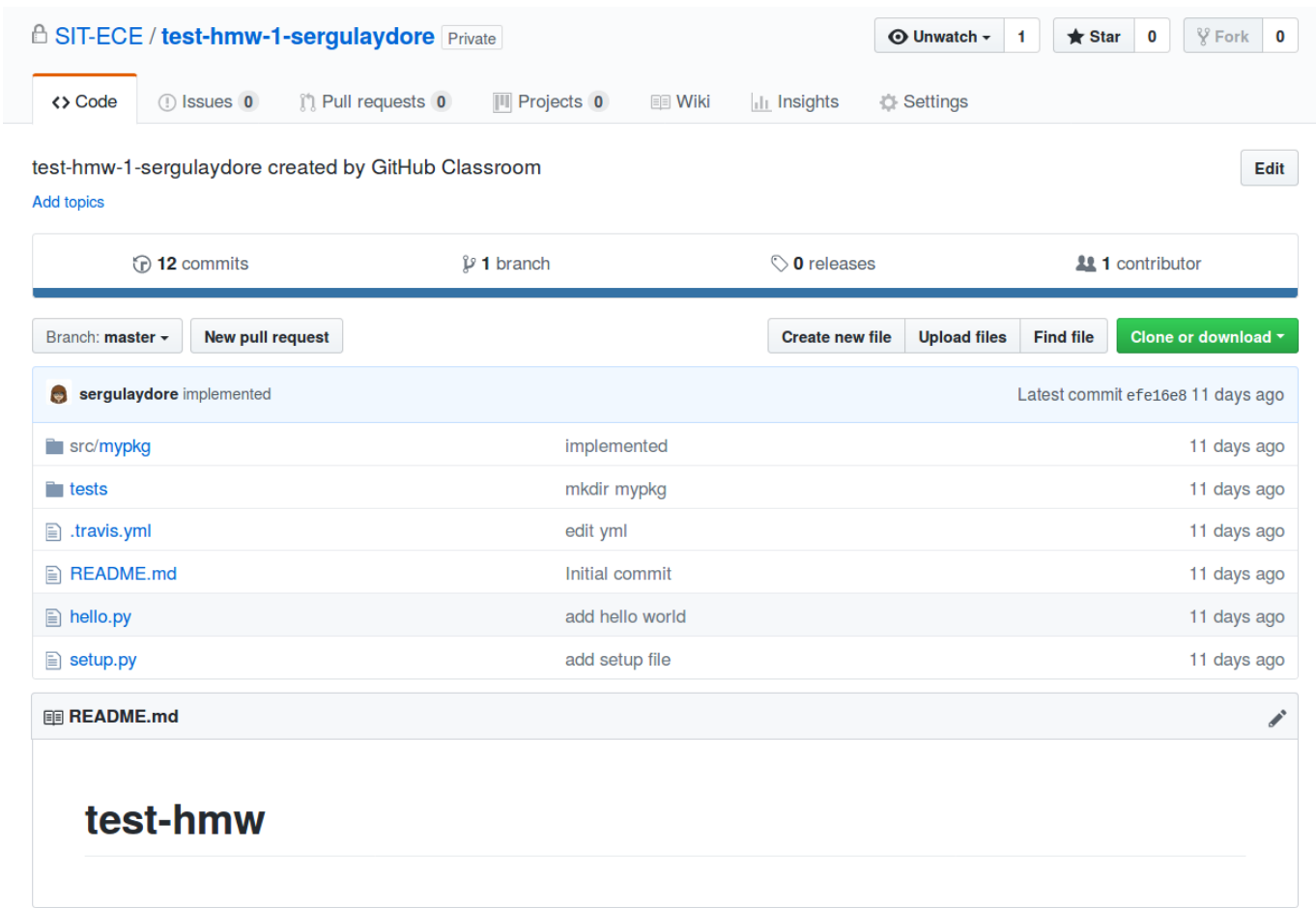
This is what you see (student's github id is sergulaydore).
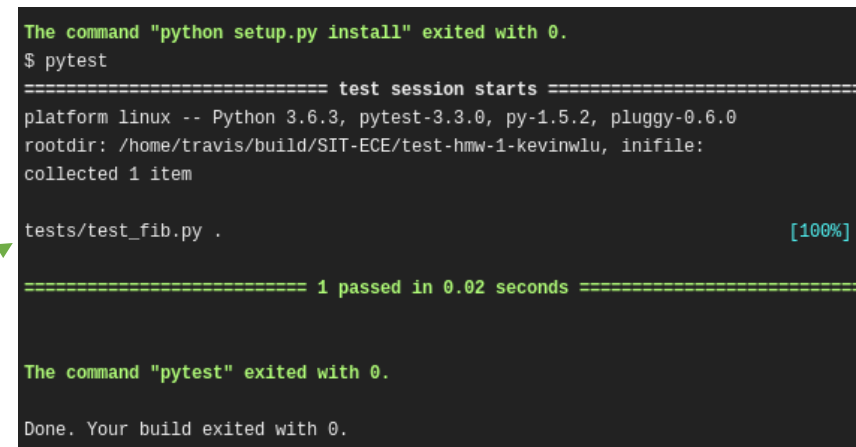
Travis CI

- Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub. (wikipedia)

When you "push" your code, it will automatically run the test I wrote for you. And I can see whose code passed the tests or not

Local Testing

- You can test your code locally before pushing it to github.
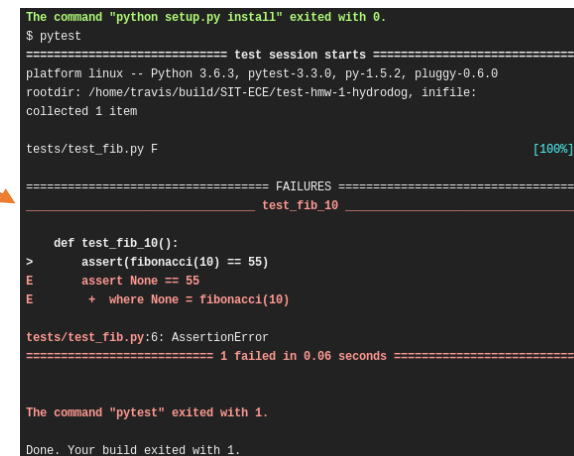
```
e$ pytest
============================ test session starts ============================
platform linux -- Python 3.5.2, pytest-3.4.0, py-1.5.2, pluggy-0.6.0
rootdir: /home/sergul/Documents/courses/EE 551A Fall 2018/github/test-hmw-1-sergulaydore, inifile:
plugins: rerunfailures-4.1
collected 1 item

tests/test_fib.py .                                                    [100%]

========================== 1 passed in 0.01 seconds ==========================
```
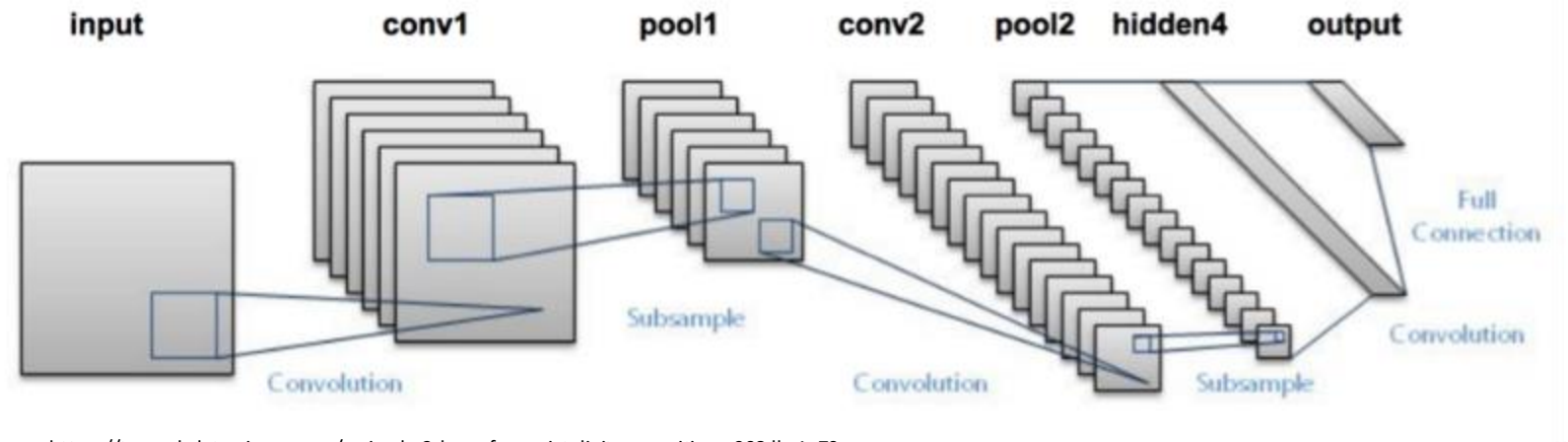
Demo with Python for Deep Learning

- MNIST classification using Pytorch

https://towardsdatascience.com/a-simple-2d-cnn-for-mnist-digit-recognition-a998dbc1e79a

Introducing the Python Interpreter

- An interpreter is a kind of program that executes other programs.
- To enable that, you must install a Python interpreter to your computer.
- Make sure to have Python 3.4 or later installed in your system by typing python3 –V in your shell.
- If you don't have it, install the recent version from

https://www.python.org/getit/ for your platform.

Setting Up a Virtual Environment

- pip is a tool for installing Python packages. It comes with Python 3.x installation but make sure you have have upgraded it https://pip.pypa.io/en/stable/installing

- A Virtual Environment enables us to keep the dependencies required by different projects in separate places, by creating virtual Python environments.

- virtualenvwrapper is a set of extensions to virtualenv tool. The extensions include wrappers for creating and deleting virtual environments and otherwise managing our development workflow, making it easier to work on more than one project at a time without introducing conflicts in their dependencies.

- Setup an environment with virtual environment wrapper based on your system: http://virtualenvwrapper.readthedocs.io/en/latest/

Interactive Prompt

- Activate your Python environment
- Type python in the shell
- Type print("Hello world!")
- Type print(2 ** 8)
- Install even more sophisticated interactive shell ipython by running pip install ipython
- Type ipython
- Type exit() to exit
- Type jupyter notebook in the shell to start a browser based environment.

Setting up a Text Editor

- Interactive environments are useful for fast prototyping but a better text editor is required for more professional development.

- You can write your Python code in any text editor.

- There are special editors designed for Python software development: IDE (Integrated Development Environment)

- One of the most popular IDEs for Python is PyCharm

- Download PyCharm

https://www.jetbrains.com/pycharm-edu/download

Configure PyCharm

- Change the appearances in Settings to Darcula
- Change the interpreter to the location where your virtual Environment is located.

Summary

- We learned the reasons for popularity of Python
- We set up our environment for our Python software development including
  - The interpreter
  - Virtual Environment
  - PyCharm as an IDE
- We wrote our first Python program

Next

- Python Object Types

- Be prepared for a pop quiz on git/github and command line!