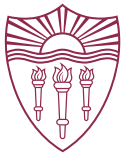
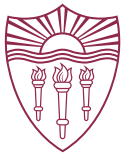


DSCI 510
PRINCIPLES OF PROGRAMMING FOR DATA SCIENCE

Itay Hen

Supervising Computer Scientist/Principal Scientist,
Information Sciences Institute, University of Southern California

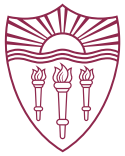
Class Sections



DSCI 510: Principles of Programming for Data Science (4.0 units)

SECTION	UNITS	TYPE	SCHEDULE	LOCATION	INSTRUCTORS	SYLLABUS	REGISTERED	DETAILS
32404	4.0	**Lecture**	Tue, 10:00-11:50 am	SLH 100	Itay Hen	Not Available	65 / 80	**View More**
FULL 32427	0.0	**Lab**	Thu, 10:00-11:50 am	SLH 100		Not Available	**65 / 65**	**View More**
32433	4.0	**Lecture**	Tue, 4:00-5:50 pm	OHE 122	Alexey Tregubov	View Syllabus	33 / 100	**View More**
32434	0.0	**Lab**	Thu, 4:00-5:50 pm	OHE 122		Not Available	33 / 95	**View More**
32405 (D)	4.0	**Lecture**	Tue, 4:00-5:50 pm	DEN@Viterbi	Alexey Tregubov	View Syllabus	16 / 30	**View More**
32428	0.0	**Lab**	Thu, 4:00-5:50 pm	DEN@Viterbi		Not Available	16 / 30	**View More**

Syllabus



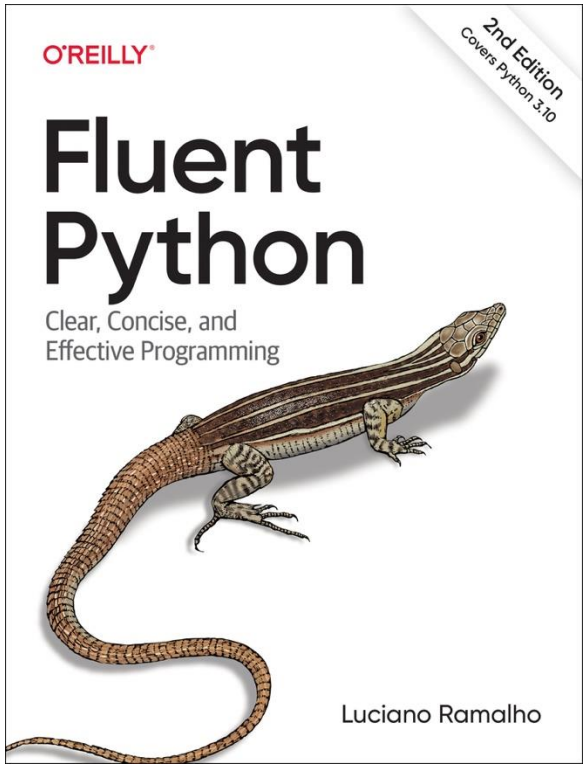
- Course Objective
- Office Hours
- Textbook
- USC Libraries
- Class Structure and Schedule
- Grading Scheme
- Academic Conduct

Section Details (1)

- Lecture
  - **Who**: Itay Hen
    - hen@usc.edu Please put DSCI 510 in the subject line!
    - Office Hours:
      - Tuesday – 1:00 pm – 2:00 pm, TBD
      - Otherwise, send me an email with DSCI 510 in the subject line
  - **When**: Tuesday, 10:00 am – 11:50 am, with a break at some point.
  - **Where**: SLH 100
- Lab
  - **Who**: Emily Jia
    - eyjia@usc.edu Please put DSCI 510 in the subject line
    - **When**: Thursday 10:00 am – 11:50 am
- Graders:
  - TBD

Section Details (2)

- Platform: BrightSpace
  - https://brightspace.usc.edu/d2l/home/219847
- Class Textbook:
  - Python for Everybody, Exploring Data Using Python 3 by C. R. Severance
  - https://www.py4e.com/book
  - Free to download as PDF
  - Other resources are available on the site: slides, videos, etc.
  - You may also consult: Fluent Python (2nd Edition)
  - Available on USC Libraries
- The Internet is out there: Use it! Check out:
  - https://stackoverflow.com/: question/answer forum
  - https://www.python.org/: official site, documentation
  - https://www.google.com/: A good place to start



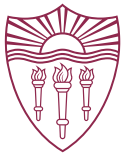
Section Details (3)

- Software
  - Brightspace
  - BrightSpace/Github Classroom (for assignment submission)
  - Zoom
  - Python
  - Jupyter Lab
  - Code Editors:
    - Up to you.
    - I am used to IDLE (stands for Integrated Development and Learning Environment)

Lectures hybrid format (?)

- I will most likely be delivering the lectures in a hybrid mode, with both in-person and Zoom participation. However:
  - Precedence will be given to in-person attendees.
  - You must attend lab! In many ways, more important than lecture.
  - Practice! Practice! Practice!

Acknowledgements

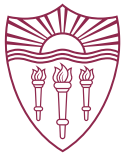


- Many of the slides we use were created by **Dr. Chuck Severance**, the author of our book.
  - You'll typically recognize them by their black backgrounds .
  - They are Copyright 2010–2017 Charles R. Severance and are made available under a Creative Commons Attribution 4.0 License.
- Most of the slides we use were created by a former instructor of DSCI-510, **Amandeep Singh,** and updated by me.

About Me

- Academic background:
  - Computational Physicist (PhD in Physics, 2009)
  - At USC since 2013
- Research and Programming Interests:
  - Physical simulation of quantum many-body systems
  - Algorithm development; mostly scientific algorithms (C++ mostly, Mathematica, some Python)
  - Quantum Monte Carlo, Discrete optimization
  - Quantum Computing

About You



- Who is here?
  - How many not in Data Science?
- Who has any Programming Experience?
  - Expert/Novice
- In what Programming Language(s)?
  - Java/R/Go/Python/Rust/C
- What do you expect from class?

Grading*

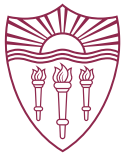
- Lab Assignments: 30%
- Mid-Term: 30%
- Final Project: 40%
  - No Final Exam!
  - Project Guidelines: obtain, analyze and visualize data from multiple (web) sources

*subject to change

The Class

- I will try to stick to the published syllabus, but may deviate from it somewhat, based on class progress and interests
- I expect people to read the relevant book chapters and come prepared, also with questions
  - *Please raise your hand anytime if you have a question! Questions are not a disruption!*
- If you feel that you're falling behind, speak up, come to my or the TA's office hours.
- Feel free to use AI only for learning, not assignments
- No cheating, please

What is Cheating



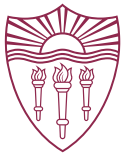
Good Practice

- Searching the web for ideas
  - Check out StackOverflow!
- Using “packages” that perform useful analysis (as long as you acknowledge)
- Discussing homework problems and solutions with the instructor or the TA
- Discussing final project ideas with anyone in general terms

Cheating

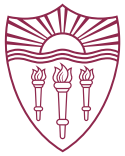
- Asking for, or receiving specific solutions from classmates or friends
- Copying code you found elsewhere, whether on the web, or from other students or friends, and presenting it as your own
- Looking at someone else’s exam or homework
- Submitting assignment solutions from ChatGPT or other AI software

What We'll Do in Class



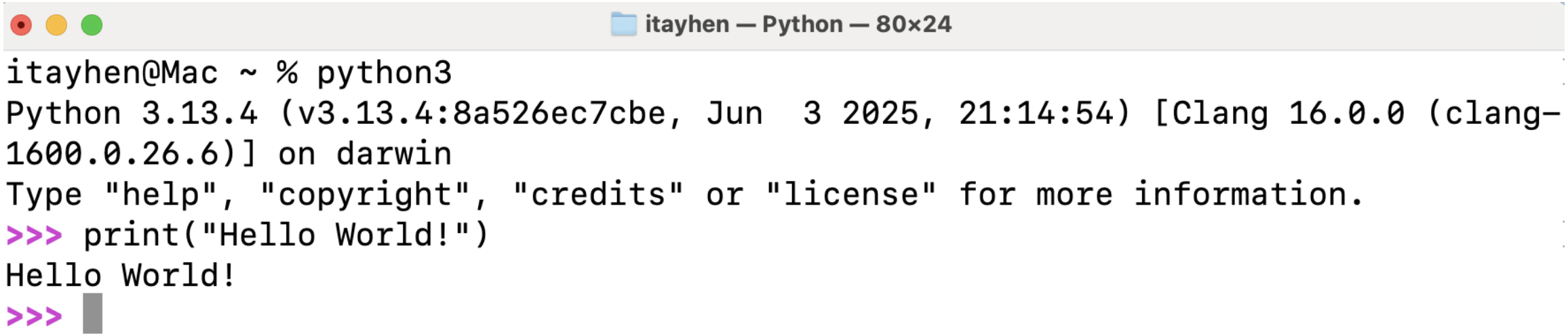
- I'll go over the basic material that the book covers
- I'll answer questions
- I'll illustrate material with snippets of code, so
  - Have your laptops, so you can try things out!
  - You'll get to install Python and setup your programming environment in Lab 1

Getting Started with Python



- README: https://github.com/usc-isi-i2/DSCI-510-Fall-2024/blob/main/README.md
- Recommended to complete before the lab session
- Get a good, simple **text editor**, preferably one that "understands" Python for syntax highlighting
  - I use IDLE; it works well on macOS.
  - SublimeText4 works on all platforms (license needed for long-term use) https://www.sublimetext.com/
  - Visual Studio Code https://code.visualstudio.com/
  - PyCharm Edu https://www.jetbrains.com/edu-products/download/#section=pycharm-edu

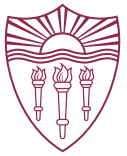
The "Terminal"



- A reference to those old boxy things you see in old movies for communicating with the computer
- Lets you query the basic "operating system" of the computer and send it commands to run programs
- Who has never seen/used the Terminal?

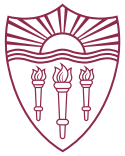
A Crash Course in Unix

https://learnpythonthehardway.org/book/appendixa.html



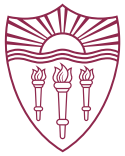
- `pwd` — path, folders, directory
- `mkdir` — make a directory
- `cd` — change directory
- `ls` — list directory
- `rmdir` — remove a directory
- `cp` — copy a file
- `mv` — move/rename a file
- `less`, `more` — view a file
- `cat` — stream a file
- `rm` — remove a file
- `exit` — exit the terminal

If you are lost: `cd ~` or simple `cd`



Questions Please!!

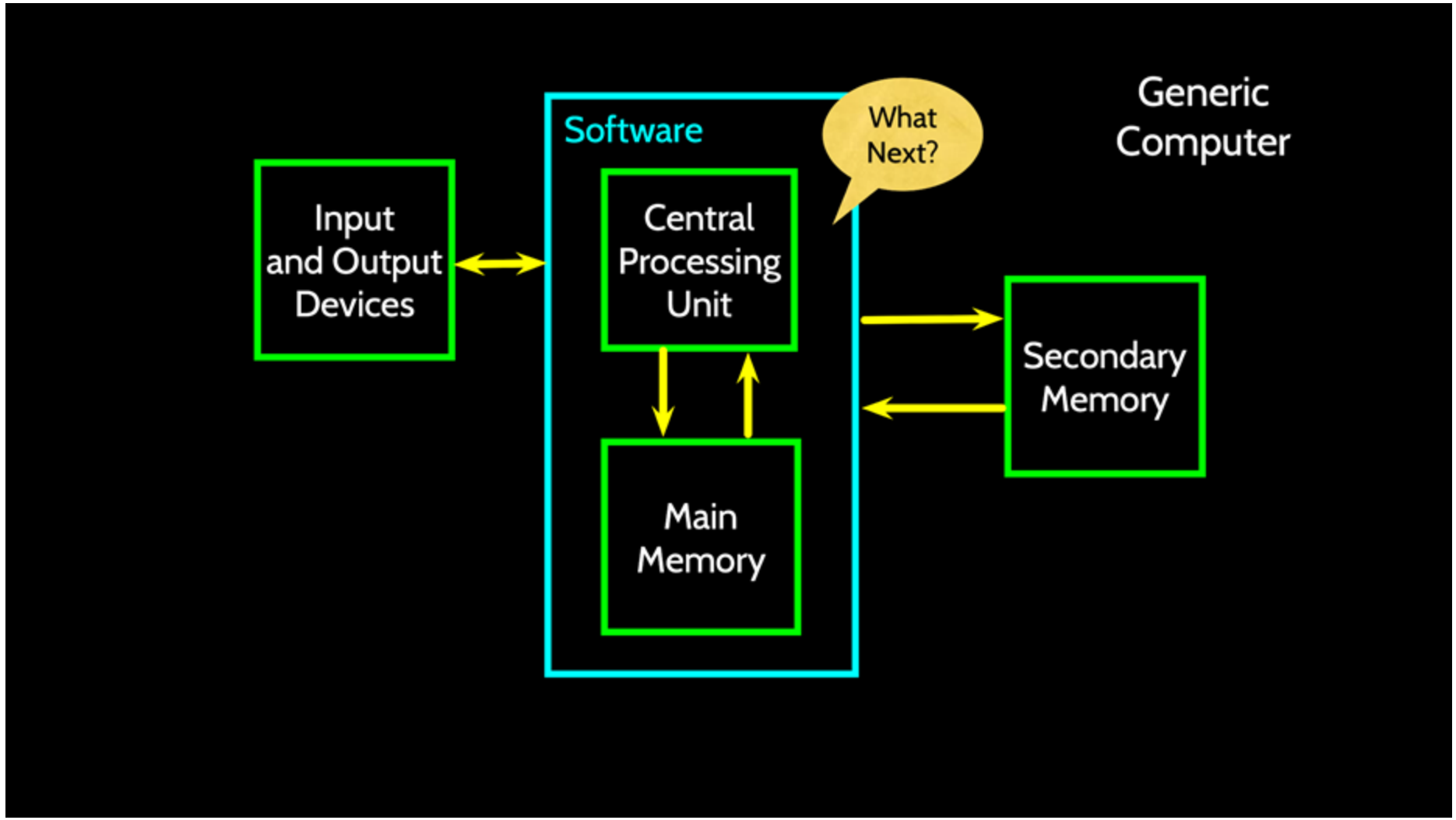
SO, LET'S GET STARTED



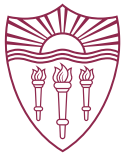
Computers

PYTHON: VARIABLES, EXPRESSIONS, STATEMENTS

This, like many other slides are from Charles Severance's "Python for Everybody"

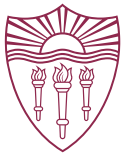


Definitions



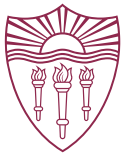
- **Central Processing Unit**: Runs the program – The CPU is always wondering "what to do next?" Not the brains exactly, as smart as the programmer.
- **Input Devices**: Keyboard, Mouse, Touch Screen
- **Output Devices**: Screen, Speakers, Printer, DVD Burner
- **Main Memory**: Fast small temporary storage – lost on reboot – aka RAM
- **Secondary Memory**: Slower large permanent storage – lasts until deleted – disk drive/memory stick

The Python Programming Language



- Python was created by Guido van Rossum in the late 1980s, while at Google
  - Python 2.0 released in 2000 (end-of-life in 2020)
  - Python 3.0 in 2008, (latest **Python 3.13.7**, released Aug 14, 2025)
- It's how we provide instructions to the CPU
- Python, like all computer languages, has a syntax
- If you don't obey it, you will get a syntax error
  - Don't despair
  - You can always try things out in your terminal window until you're sure you understand how Python works

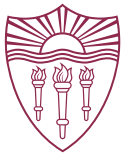
Zen Of Python - https://peps.python.org/pep-0020/



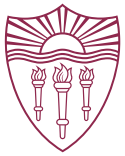
```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Programming in Python



- Programming used to be much more tedious
- Python is an interpreted language (as opposed to compiled)
  - You can type instructions, and they're executed on the spot
  - Or you can load a file with a script, and it will then be run — that is, the commands will be executed in order
- Everything complicated that you'll need has probably already been written: Libraries
- **Pay attention to indentations — they matter!**
- Let's try things

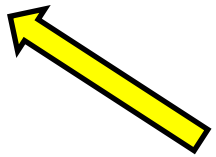


Let's try things!

GO TO TERMINAL/PYTHON

Python Terminal

```
Last login: Sun Aug 17 09:40:52 on ttys005
itayhen@Mac ~ % python
zsh: command not found: python
itayhen@Mac ~ % python3
Python 3.13.4 (v3.13.4:8a526ec7cbe, Jun  3 2025, 21:14:54) [Clang 16.0.0 (clang-
1600.0.26.6)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



What next?