COGS 18: Introduction to Python

Fall 2018: MWF 9:00 - 9:50 am

DRAFT SYLLABUS - SUBJECT TO CHANGE (Dated: Sept. 28th)

Final Version Due: Oct 1st

Course Overview

Welcome to COGS18! The core goal of this class is to teach you introductory, hands-on skills for computer programming, specifically using the Python programming language. We aim to do so in a way that is situated in the Cognitive Science department, and relevant to your potential use cases. The approach is to focus on programming as a tool, and to get you started with the necessary background and basic skill set to get you reading and writing code, so that you have a foundation from which you can continue programming, and apply these skills to your domain or topic of interest.

Course Pre-Requisites

This class has no pre-requisites. You are not expected to already know how to program.

Course Staff

Role	Name	Contact email	OH Time	OH Location
Instructor	Thomas Donoghue	tdonoghue@ucsd.edu	-	-
TA	Rob Loughnan	-	-	-
TA	Daril Brown	-	-	-
TA	Paolo Gabriel	-	-	-
IA	Peilin Chen	-	-	-
IA	Luis (Yinhe) Lu	-	-	-
IA	Brandon Nguyen	-	-	-
IA	Luke (Zehua) Chen	-	-	-

Important Links

Course Website: https://cogs18.github.io
Course Github: https://github.com/COGS18
Course Email: cogs18python@gmail.com
Piazza: https://piazza.com/ucsd/fall2018/cogs18

Feedback About the Class

This is effectively a new course, with a brand new curriculum and design. We will do our best to run this class as smoothly as possible, but please be patient as we get things running. We welcome, and will be soliciting, your feedback about this course.

Course Objectives / Philosophy

The core goal of this class is to teach you how to program, at an introductory level. This is *not* a computer science class - where there is a difference between learning what we need to know to get something working, and learning all the nitty-gritty underlying details, we will focus on getting things working. That is to say, the goal is not to teach you to be computer scientists, but rather to start you on the path of being a productive programmer.

Partially for these reasons, this class is taught in the Python programming language. Python is a general purpose and high-level language, meaning it is relatively abstracted away from the technical details off computer hardware, and instead closer to a human-understandable language. Because of this, it is an extremely powerful and popular language, and has a wide range of applications, including scientific computing, data analysis, rapid prototyping, web development, app creation, robotics and artificial intelligence, game development, and so much more (check out, for example, this list of Python success stories: https://www.python.org/about/success/). Python is of the most popular programming languages in the world, for both academic research, and in industry.

Note that computer programming is a skill, the mastery of which is a potentially never-ending journey. In many ways, we will barely be able to scratch the surface of Python, never mind the broader world of computer programming and technology. There also exists a tremendous amount of available resources for learning Python, though what to learn when, and how everything relates to each other can be rather opaque to new learners. In this context, the purpose of the class is specifically to be a guide through the absolute core of using Python as a tool, such that, by the end of the course, you have a foundation to build on, and an understanding of the ecosystem that allows you to effectively and efficiently continue to use Python, and apply it to your use-cases of interest, should you want to.

In particular, by the end of this course you should be able to:

- 'Think programmatically' about how computers represent and use data and procedures
- Design programmatic solutions to simple problems
- Have a working familiarity of standard library Python
- Read simple Python programs, recognizing the structures they use and how they work
- Write and debug small python programs
- Execute Python programs on local machines, through notebooks and the command line
- Recognize and use best practices, acknowledging that programming is a communicative endeavour done by and for humans
- Apply this knowledge and skills to particular use cases of interest

In order to achieve these goals, with class is mostly an assignment & project based course, with an emphasis on interactive instruction in the course lectures and sections. You will be expected to attend lectures and sections, participate in the class, and complete a series of assignments as well as a small midterm exam, and an individual project. Being housed in the Cognitive Science department, we will use simplified examples and applications related to cognitive science, including data-analysis, artificial intelligence, human-computer interaction, and programmatic thinking.

Grading Outline & Policies

Your grade for this class will be comprised of four components:

Requirements	% of Grade	Grading	Requirement
PI Questions	8%	Pass / Fail	Participate in XX% of in class questions
Coding Labs	12%	Pass / Fail	Complete 6/8 Code Labs
Assignments	40%	Graded	Complete 5 assignments (8% each)
Midterm	15%	Graded	Written Midterm
Project	25%	Graded	Submit Final Project

PI Questions - 8%

To get full credit for clicker questions, you must participate in answering clicker questions, using the peer instruction (PI) approach, in at least XX out of YY class lectures. Each lecture class will have a minimum of N clicker questions, and participation is counted if you answer at least two thirds of the questions in a given lecture. Note that participating in a clicker question requires simply providing an answer to a question, regardless of correctness. If you do not meet the threshold for full credit, your clicker grade will be the percent of lectures you participated in (N/YY * 8%), minus a 2% penalty. You may opt out of participating in clicker questions. To do so, e-mail the course email by DATE [Please wait for final version of syllabus to details on this]. If you choose to do this, this portion of your grade will be added to your project.

Coding Labs - 12%

Sections will be used for hands-on work time and 'coding labs', including specific tutorials and activities that are focused on preparing you for the assignments and projects. Across the 10 week quarter, sections will include 8 coding labs, that are graded as Pass/Fail. To get full credit for this portion of the class, you must complete at least 6 of these assignments. Coding labs can only be completed within the section time (this means that you do need to be able to attend your section time).

Assignments - 40%

There will be 5 assignments, each worth 8% of your final grade. Assignments will be hands-on coding assignments, that will typically be due 1 week after release, and are to be completed individually. Late assignments will be accepted at 75% credit for the first week after the due date. After one week, the answers for the assignment will be released, and assignments can no longer be submitted for credit.

Midterm - 15%

There will be a written midterm exam, in class, during week 6.

Individual Project - 25%

To complete this class, you will be expected to pursue an independent coding project, worth 25% of your final grade. A full project description will be available elsewhere, briefly: you will be asked to choose one of several options in which you will take one of the assignments, and expand what you started into a bigger project, adding original elements.

Contact Information & Policies

Please review and follow the following guidelines for how and when to contact course staff. Note that for all content and programming related questions, course staff may ask you to come and see them in person in either section and/or office hours, if it would be more effective and/or efficient to do so.

I have a question about the organizational structure of the class:

First, please review the information provided in the course syllabus (this document), as well as other informational documents covering the details of the course schedule, coding labs, assignments, and project, as appropriate, to double check if the question is available. If you still cannot find the answer, ask the question on Piazza.

I have a question about something about Python:

If possible, it is much preferred that you come to sections and office hours to ask about content questions - it is much more efficient to be able to talk through ideas, and this allows course staff to ask follow up questions, and look through more code. Otherwise, all code related questions should be asked on Piazza.

I'm stuck on something - but I don't even really know what the question is:

Programming can be frustrating, and it may not be obvious what is going wrong when something isn't working. That's fine - if you are stuck, you can and should reach out for help, even if you do not have a specific questions. If you are not sure when to reach out, consider as a rule of thumb the **2-hour rule**: if you are working on a particular problem, and you are stuck, work for at most an hour on that thing in particular. Then, take a 30 break, and do something else. After that, come back to the problem for another half hour or so. If you are not making progress at that point, stop, and contact us (come to section or office hours, or post on piazza). You don't need to bring or post a specific question - just include the information you have (copy the code as you have it, and say what's happening).

I'm having a technical problem submitting assignments:

If you are having a technical problems submitting assignments, please e-mail the course e-mail (cogs18python@gmail.com). If there is a deadline, you can e-mail submissions to that e-mail if (and only if) the normal method of submission is not working.

I have a questions about grades:

All grading questions must be sent to the course email: cogs18python@gmail.com. Make sure to include your Student ID number with any communications to this email.

I have a question about something specific from section

If you have a question for a particular TA based on something from section, you can e-mail them directly.

It's something else

If you some other question, you should e-mail the instructor directly.

Required Course Materials

Textbook

This class has no required textbook.

Clickers

We will be using clickers for Peer Instruction (PI) during the lecture sessions, meaning you will need a clicker, that you register on TritonEd, that you can bring to lecture sessions.

Software

You will need some installed software for this course, notably Python 3.6 with anaconda distribution, including Jupyter Notebooks. Detailed instructions for these installations will be listed on the course website, and hands-on help offered across week 1. All of the required software is freely available for download. If you do not have a computer available, you will be able to complete the course requirements on UCSD computers.

Class Conduct

Through class lectures and coding labs, you will be encouraged and at times required to interact and work together with your class peers, as well as the course staff. Throughout these interactions, you are expected to be respectful of everyone you interact with. This includes following the UC San Diego principles of community. There will be zero tolerance for bullying, harassment discrimination, or disrespect based on race, ethnicity, gender identity, sexual orientation, disability, national origin, age, appearance, accent or any other personal attribute. Any evidence for harassment or bullying and/or suspicion of verbal, physical or emotional abusive behaviour will be reported to the appropriate university offices. If you see or experience, such behaviours, you are encouraged to reach out to the course instructor, and/or directly to relevant university offices.

Please review the principles of community here: https://ucsd.edu/about/principles.html

Academic Integrity

Don't cheat. Although you are encouraged to work together, and help each other, you are ultimately responsible to submit your own work. Copying and pasting large segments of code will be considered plagiarism, and may be used as ground to fail an assignment and/or the project. For assignments, you are required to write all submitted code yourself. Projects may include ideas and code from other sources - but these other sources must be documented with clear attribution.

Please review academic integrity policies here: http://academicintegrity.ucsd.edu