Course 11-605: Python for Data Science II

Course Description & Syllabus

Carnegie Mellon University
Spring 2024

1. Overview

Title: Python for Data Science II

Units: 12 units
Pre-requisites: none

Carnegie Mellon's LMS access: https://canvas.cmu.edu

Zoom links for each course meeting and TA office hour sessions will be posted to the zoom area on Canvas for your convenience.

Instructor team:

Carolyn Rose
cprose@cs.cmu.edu
GHC 5415

Vicente Malave
vlm@cs.cmu.edu
GHC 5414

2. Course Description

This is the second mini of a two-course sequence on the Python programming language, with a focus on learning concepts, techniques, skills, and tools needed for developing programs in Python. In comparison with the first mini, this course covers more advanced ways to load, format, and store data, including APIs, databases and pandas. This course also provides an introduction to tools that are useful when developing large software projects, including coding standards, version control, unit testing and debugging. Finally, the course covers more abstract building blocks including implementing more complex data structures and using numpy, both of which are the foundation of training and applying modern models and methods.

Course projects include working with real datasets from a variety of sources.

This course is taught by an instructor team working together. Prof. Carolyn Rose will oversee the course and participate in the course meetings, but the activities and discussion will primarily be led by Vicente Malave, who is the course developer. If you have concerns about the course materials or the way the course is being run, please reach out to Prof. Rose. The TAs will be the primary people responsible for assisting both with in class activities and out of class activities. They will be the primary people answering questions on Piazza and running weekly office hours sessions.

3. Course Goals

In this course, students gain hands-on experience solving real-world problems by completing programming projects in Python focused on programming language fundamentals, software development practices, data manipulation and analysis. Specifically, students are exposed to realistic software development projects, real-world data, and scenarios in order to learn how to:

1. Explain and use fundamental constituents of the Python programming language.
2. Use the Python language, Jupyter, and git to write code to solve a variety of computational problems.
3. Retrieve, process, analyze and visualize real-world data using Python data science libraries, APIs, RDMBS databases, SQL, and pandas.

Through this process, we aspire for our students to become independent and resilient problem solvers who are able to enact their problem solving in the Python language, and in so doing, are able to overcome challenges and learn.

4. Learning Goals

In this section we break down the learning goals for each unit. These describe what you will be able to do by the end of each week.

- **Week 1 Data Cleaning and Table Manipulation**
    - Clean up a dataset with missing and noisy data.
    - Use Pandas to create, modify and output formatted data, including indexing, joining and subsetting data frames.
- **Week 2 APIs (Application Programming Interfaces)**
    - Use the requests library to connect to an API and retrieve information.
    - Use an API Key appropriately and keep your identifying information secure.
    - Evaluate the results of your query to handle errors, exceptions, and combine queries from several API endpoints.
- **Week 3 Debugging and Instrumentation**
    - Recognize different types of bugs including syntax errors, logic errors, and errors with dependencies.
    - Use the python debugger (pdb) to inspect functions.
    - Design and implement unit tests.
    - Use %timeit in jupyter and other utilities to profile your code.
- **Week 4 Databases**
    - Compare Relational Database Management Systems to pandas data frames.

- Design queries using Structured Query Language (SQL).
- Query a database instance using SQL and Python libraries.

- **Week 5 Software Development Skills**
  - Recognize good python style, apply PEP 8 standards, and use automated tools to detect and fix formatting issues.
  - Understand version control and branching.
  - Refactor code with unit tests to verify that it conforms to the initial requirements
  - Use git to clone a repository, create a branch, and push changes on that branch.
- **Week 6 Tries and Advanced Data Structures**
  - Explain the problems being solved, and contrast a trie with a list of lists.
  - Explain why a Trie uses fewer comparisons to search.
  - Implement a python class to store data in a trie.
- **Week 7 Numpy**
  - Compare the speed of using numpy vs writing naive code yourself
  - Recognize issues with floating point, including comparing for equality.
  - Process data using numpy expressions to speed up calculation.

○ Course organization and weekly activities

This course is designed to have several steps of engagement with the material, so there are multiple sources of instruction and tasks for you to do every week. There are some tasks you will need to complete before class, so that we can make the class meeting more interactive.

1. The readings in Canvas are in Jupyter Notebooks, and are interactive: this means you can run all the code examples in the same place as the readings that explain how it works. You can pause, modify, and execute the code to get more hands-on understanding of what is going on. You should work through the notebooks before the class meeting.
2. At the end of each notebook, there are short programming exercises designed to test your engagement with material, and allow you some practice with the concepts. You will also complete these before the class meeting.
3. There is also a short quiz on the readings to complete before class. Quizzes may be taken multiple times to improve your score and must be completed before the class session.
4. Course meetings are mandatory. Our course meetings are designed to be interactive sessions to respond to your learning needs, instead of a traditional lecture. We will discuss questions you post about the material as part of the discussion in the course meeting, and respond to specific topics you may ask for clarification or more detail on. Also, in the course meeting, students will work together in small groups to answer some small programming exercises. This is a chance for

additional practice, and also to discuss code with other students, building skills you will make use of in future courses.

5. After the session, you will have a homework assignment to work on, which is normally due by the Friday of the week in which it is assigned. These will involve more substantial individual work on the same topics. Assistance (office hours, piazza) will be available during the week, and we encourage you to start early and ask questions!

6. After the homework is turned in and you have completed the readings in preparation for the subsequent course meeting, we will ask you to post a reflection on what you have learned from the assignment, and what you have struggled with in that assignment and in the readings. This should be posted to the course discussion forum in Canvas and will be used by the instructor team to enable us to better tailor the course activities to your needs. Additionally, we ask that you follow up the discussion after everyone has contributed. This allows you to share things you might have learned with each other, to reflect and build connections in your own learning These posts are due by 8pm ET the night before each class meeting.

5. Getting help

Students are encouraged to ask questions about content and projects through thePiazza Q & A forum, hosted by Piazza, which is available from Canvas by clicking the Piazza link. This is separate from the course discussion forum, which is available for assigned reflections.

6. Policies

Working Alone on Projects

Projects that are assigned to single students should be performed individually. Making your assignment code visible to another student before it is due is considered cheating. Using code from another student or any other source in an assignment is also cheating. Furthermore, using ChatGPT or any other generative AI coding support is considered cheating. You may look up examples on StackOverflow, but you are not allowed to copy any of the code and turn it in as your own. Cheating is an extremely important issue, so these issues are revisited again below.

Handing in Assignments

All assessments are due at 11:59 PM ET (one minute before midnight) on the due dates specified on Canvas. All hand-ins are electronic.

Appealing Grades

After each assignment is graded, you have seven calendar days to appeal your grade. All your appeals should be provided by email to the instructor.

Grace Days

A limited number of grace days are provided to students as an extension without penalty in order to complete work. Each grace day extends a project deadline for 24 hours beyond the original deadline. Students can contact the course staff to request a grace day after the original deadline and within 24 hours.

Students can use at most two (2) grace days throughout the course. Students can only use up to one (1) grace day for each project.

AI Assistance

Seeking assistance from an artificial intelligence system (e.g., a large language model such as ChatGPT) is strictly prohibited.

"Found Code"

Students are encouraged to explore various online and offline resources, such as books and instructional materials, to deepen their understanding of the learning materials presented in class. These resources may include an example or pseudo code, which can help clarify an algorithm or implementation detail. However, when crafting a solution for an assignment, students must set all materials aside and develop their code entirely independently, starting "from scratch." It is important not to use any code found or encountered in the process. If students encounter code that implements any portion of their assignment, they must disclose this in their collaboration statement. By adhering to this policy, students can ensure a rewarding and authentic learning experience.

7. Assessment

We make a distinction between formative feedback and grading. Assignments are graded and constitute a substantial portion of your grade for the course. Quizzes and other activities are primarily provided for learning rather than assessment. We want you to feel comfortable making errors and then learning from them in these activities. Thus, for quizzes we offer multiple attempts so that you can achieve full points with enough effort. And for the other activities, the errors you make do not count against your grade. We simply use these as opportunities to offer feedback and instruction.

The assignments at the end of each week are designed to test and assess your mastery of the material, and will be automatically graded after you hand them in. You will receive a personalized feedback report on how well you have done, and in the assignments are a number of test cases to pass to make sure your code is correct. Passing the tests we provide is necessary to get full points on the assignment, but assignments are graded on the correctness of the code, not simply on passing the tests we provide in the assignment. We will use additional tests on your submitted code and manual grading to verify correctness of your solutions.

This course also includes a final exam that will be completed at the end of the course. The final exam is similar to the individual assignments but it is not auto-graded. Instead, the instructors and teaching staff will provide students with feedback and grading.

Type	Number	Weight
Learn by Doing and Practice Programming Questions (pre-Class)	7	5%
Quizzes	6	10%

Assignments	6	40%
Discussion	7	5%
In Class Challenge Problems	7	10%
Final Exam	1	20%
Extra Credit	1	Up to 5%
Total Grade		100%

8. Cheating

We urge each student to carefully abide by the course policy on academic integrity, which outlines the policy on cheating, plagiarism, or unauthorized assistance. It is the responsibility of each student to produce her/his own original academic work. Collaboration or assistance on academic work to be graded (i.e., the formal assignments and the final exam) is not permitted unless explicitly authorized by the course instructor. Each unit checkpoint quiz or assignment submitted must be the sole work of the student turning it in. Student work on the cloud is logged, submitted work will be closely monitored by automatic cheat checkers, and students may be asked to explain any suspicious similarities with any piece of code available. The following are guidelines on what collaboration is authorized and what is not:

What is cheating?

1. Sharing a solution, code, or other electronic files by either copying, retyping, looking at, or supplying a copy of any file. Copying any solution, code from the internet (stackoverflow.com or GitHub or others). Do not use other students' solutions or code to "test" the auto-grader. Anything you submit to the auto-grader must be your work.
2. Copying answers to any checkpoint quiz from another individual, published or unpublished written sources, and electronic sources.
3. Collaborating with another student or another individual on checkpoint quizzes or project modules.
4. Sharing written work, looking at, copying, or supplying work from another individual, published or unpublished written sources, and electronic sources.

What is **not** cheating?

1. Clarifying ambiguities or vague points in-class handouts.
2. Helping others use computer systems, networks, compilers, debuggers, profilers, or system facilities.

3. Helping others with high-level design issues.

4. Guiding others through code debugging but not debugging for them.

Cheating on assignments will also be strictly monitored and penalized. Be aware of what constitutes cheating (and what does not) while interacting with students. You cannot share or use solutions, written code, and other electronic files from students. If you are unsure, ask the teaching staff.

Be sure to store your work in protected directories. The penalty for cheating is severe, and might jeopardize your career – cheating is simply not worth the trouble. By cheating in the course, you are cheating yourself; the worst outcome of cheating is missing an opportunity to learn. In addition, you will be removed from the course with a failing grade. We also place a record of the incident in the student's permanent record.

9. Schedule

Monday	Topic	Due Dates for HW
March 11	Data Cleaning and Pandas	March 18
March 18	APIs (Application Programming Interfaces)	March 25
March 25	Debugging	April 1
April 1	Databases and SQL	April 8
April 8	Software Development Skills	April 15
April 15	Tries and Advanced Data Structures	April 22
April 22	numpy	April 26 (Friday)
	Final For Mini 1	April 29-30

- Before the first course meeting (on March 11th), you should do the readings for Module 1.

10. Accommodations for Students with Disabilities

If you have a disability and have an accommodations letter from the Disability Resources office, I encourage you to discuss your accommodations and needs with the instructor as early in the semester as possible. I will work with you to ensure that accommodations are provided as appropriate. If you suspect that you may have a disability and would benefit from accommodations but are not yet registered with the Office of Disability Resources, I encourage you to contact them at access@andrew.cmu.edu.

11. Take care of yourself

Do your best to maintain a healthy lifestyle this semester by eating well, exercising, avoiding drugs and alcohol, getting enough sleep, and taking some time to relax. This will help you achieve your goals and cope with stress.

All of us benefit from support during times of struggle. You are not alone. There are many helpful resources available on campus and an important part of the college experience is learning how to ask for help. Asking for support sooner rather than later is often helpful.