**Instructor:** Danielle Presgraves, Ph.D.

**Text:** Currently there is no textbook for this course. While our particular focus will be on solving biological problems, if you need or want more practice learning Python fundamentals, there are many **free** online options. I have had recent students recommend the following (if you have other suggestions please let me know so I can tell your peers):

* codeacademy is useful for cementing basic Python (material covered in Module 2 and some Module 3): <https://www.codecademy.com/learn/learn-python-3>
* A Byte of Python: <https://python.swaroopch.com>
* Google Python class: <https://developers.google.com/edu/python/>
  + The above suggestions have *free* versions; I do not suggest that you pay a fee for the more advanced version of these websites (unless you want to for personal enrichment purposes)*.*
  + There are also many free resources available on youtube; if you find a particularly excellent one, please let me know so I can share with the rest of the class!

**Content:** The majority of the class will focus on applying the open-source programming language Python since it is the current choice for a large proportion of bioinformatics, genomics and computational biology. More generally, we will also gain experience using the NumPy, Pandas, BioPython and SQLite Python packages that are important in data science. Besides its ubiquity, Python has a straightforward computer syntax that makes it a good choice for an introductory language. Once you have experience with one language, it will be easier for you to learn other computer programming languages in the future – which you will almost certainly need to do since the popularity of a language changes over time and by field.

We will cover a small amount of **SQLite** (using a Python library which emulates the logic and syntax of SQLite) since understanding the fundamental logic of database manipulation is important when dealing with the large datasets typical in modern biology.

**Learning Outcomes:** By the end of this course, students will be able to use the Python programming language and do the following:

* Clearly deconstruct a problem into manageable smaller problems
* Apply appropriate (and efficient) programming tools to solve a problem
* Understand and interpret other peoples' code
* Identify problems/deficiencies in existing code (ie. 'troubleshoot' code, your own code as well as peers) and reformulate the code with incorporated improvements
* Evaluate the quality of coding solutions through guiding principles such as encapsulation, cohesion, optimization
* Utilize appropriate libraries
* Be able to explain your solution to an audience of peers

**Acquiring Python:** We will use the Anaconda manager (<https://www.anaconda.com>) to download Python (**version 3.XX**)and the **Jupyter** notebook.Use the default most updated versions. Importantly: we will not use version 2.7X.

**Lectures:** As an online course, you are free to complete aspects of this course of asynchronously — that is, you are free to watch the content videos, work through the accompanying Jupyter notebook, and complete the problem sets when you see fit to do so as long as you hand the problem sets in before the completion dates.

**Problem Sets:** The problem sets are different lengths and have differing levels of challenge. Accordingly, they have differing amounts of points. The general rubric for how your solutions on problem sets will be graded is as follows:

* Approx. 55-70% of your points will come from having a correct solution (ie. does your program run, and does it solve the assigned problem?)
* Approx. 30-45% of your points will derive from style **(**ie. have you submitted an efficient solution or is your code five times longer than it needs to be? Is your code commented adequately?).
* If you fail to attribute code chunks from elsewhere, you will automatically earn 0 on a problem set (and potentially have academic honesty involvement).

An elegant, well-commented solution that uses the appropriate tools will earn you100%. You will usually be given instructions to incorporate the theme of the week into your solution, for instance if/else loops, which means that even though there may be other ways to solve the problem you will only get full points for applying what you learned during the affiliated lectures/notebooks.

Additionally, each problem set asks that you create a novel question (using the tools emphasized in the accompanying problem set) that you will post to a discussion thread in the module. There are a few parts to this question:

1. Create a novel question using tools discussed in module (the question specifies the tools you are expected to use)
2. You will need to answer one unanswered question posted by a peer. No question should be answered twice (so every student will answer one unanswered question. This should mean that every question has an answer).
3. After someone answers your question, to provide an explanation (if they have answered it incorrectly; if they have answered it correctly, you can simply confirm or explain other aspects of the question).

**Due dates:**See module schedule, syllabus, calendar schedule and/or checklist – there may be some modifications to due dates as we work through the course material. I will give you as much notice as possible for any changes and I will post changes on blackboard and email them to the course participants. It is essential to your success in this course that you begin the problem sets early and that you stay on top of the material of the course. However, I recognize that sometimes crises arise that are out of your control - especially over the last couple of years! - and may compromise your ability to hand in a problem set on time. **If you have exceptional circumstances that are likely to result in a late submission, reach out to your TAs and/or me as soon as you can before the deadline.** I may be able to delay posting the answer key or otherwise work with you to get your problem sets submitted. Reaching out to me after the deadline will not be useful since I will post the answer key to Blackboard immediately after the due date. **Once the answer key is posted, no late submissions will be accepted for credit.**

**How/Where to hand-in Problem Sets:** Blackboard. Blackboard will give you a 'receipt' or a notice that you have successfully submitted your work.

**Final Project:** We will discuss the final projects during class in the mid-October. The projects will be more challenging and a bit deeper than the regular weekly problem sets but are manageable! I have a number of general suggestions that I will post in Module 5 but you should feel free to suggest projects, especially if you are working in a lab and want to analyze/present/program with your data/question.

As a general note: in all aspects of this project, you should consider the heuristic of the **S.T.A.R.** method that is often used in interviews for technology companies (**S**ituation, **T**ask**, A**ction**, R**esult). You can google this method, but it forces you to organize your project around a digestible story for your audience: the narrative of a prompting **s**ituation (or problem) during which you give your audience enough background to understand why your problem is important, a description of the **t**ask (ie. your responsibilities in the solution), there is an **a**ction (in the case of your project, a programming strategy), and a **r**esult (does your solution solve the original interesting problem? What are the additional constraints? Can you break your solution to test its robustness?).

The final project consists of a one- or two- page proposal of the general problem that your group is proposing and any relevant details about how you will solve it. In this proposal, you will have a two-paragraph summary of your problem, you will assign group members tasks and outline general "deliverables" and due dates for each group member. You will also answer – as a group – some additional prompts that will be provided about the goals of collaborative learning. Your entire group can hand in just one proposal, i.e. you don't all need to hand in the same proposal each. After it is handed in, I will either approve or request more information about your project. You will then complete your programming project and give a ten-minute presentation to your peers about your solution (including relevant information such as trouble shooting, why it is an interesting problem and trying to ‘break your solution’ to see how robust it is). Your presence is required for the presentations of your fellow classmates! Finally, your group will hand in the code itself to ensure that it runs and it is well commented and reasonably efficient. You will individually hand in a brief write up (~2 pages) of your problem, how you approached solving it and detail how you overcame challenges that arose. Your final write up should be an individual (not a group) effort and everyone will hand in their own individual write up.

**Academic Honesty:** A specific note about programming academic honesty: On assignments, I encourage collaboration. It is acceptable (and often useful!) to work in groups but it is not acceptable to copy another’s work or to allow your work to be copied. I have two additional thoughts about working in groups:

1. Sir Tony Hoare stated an expression that is repeated A LOT in the programming world thanks to Donald Knuth: "Premature optimization is the root of all evil". In the case of problem sets, I take this expression to mean that to truly learn how to program, or any new skill, you need to work through feeling uncomfortable, incompetent and, sometimes, a bit frustrated in order to receive the eventual dopamine rewards of solving a new challenge. You will improve with practice. Learning to persevere and resolve a difficult problem is an enormous benefit to learning to program and basically every other type of learning you will need to do in life. If you simply find the solution online – or through a group of peers – you aren't going to be successful in building up the habits and resiliency that help you become an effective programmer or learner. This is especially true if you rely on your peers/google at the beginning of this course when the problem sets are relatively simple (I promise that you won't feel they are simple until you look back on them at the end of the semester and are amazed at how sophisticated your programming has become).
2. If you work in groups on assignments, you MUST include the names of members in your group on the assignment, and assignments MUST be individually completed. You should thoroughly comment the code that you are using to demonstrate that you understand the logic of it. Be aware that copying of homework assignments is not only a reportable violation of our academic honesty policy, it also deprives the instructor of an important cue that you are having trouble with course material. It's also simply true that you are cheating yourself – you enrolled in this class because you either need to learn to code or you appreciate that coding is an important skill. So take the opportunity to actually learn to do it!

Despite my warning to hand in your own work, I strongly encourage you to effectively – and liberally - utilize google (or your favourite search engine). One of the goals of this course is to empower you to eventually be autonomous when using Python. One of the best ways to achieve competence in any programming language is to learn to google your error terms in order to refine your ability to trouble-shoot your own code. I guarantee that other individuals have also had a similar problem to whatever one you are encountering and it is wonderful to have access to their knowledge about what might be wrong with your own code! This is one of the most glorious benefits of 'open access' languages like Python and R.

If you use snippets of code found in places like stackoverflow in your work **include *attribution.* Attribution means that you should clearly state that you used or heavily borrowed (with modification) code snippets, include a web link to where you found the link AND thoroughly explain what the code snippet is actually doing.** Not only will I appreciate this acknowledgment, but you will fulfill the spirit of 'academic honesty' AND it will help 'future you' when you return to try to figure out a similar problem by re-reading your own code. This happens more than you might think.

Please remember that sharing of course material without permission is a copyright violation, and is prohibited under our academic honesty policy.

**Note: A "due date checklist" follows this schedule as well as a calendar-format with due dates listed.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week/**  **Concepts** | **Learning Objectives** | **Means of Assessment**  **(order of activity in module)** | | **Instructional Activities and Instructional Tools & Resources** |
| **Module 1: Week 1**  Welcome to the course!  Course expectations, Syllabus, Resources available  **Aug 21-Aug 29** | 1. Students will demonstrate awareness of course schedule (due dates) and expectations of participation 2. Students will create a video to introduce themselves 3. Students use and manipulate Jupyter notebooks | 1. Introduction and academic honesty quiz | | Activities:   1. Download Anaconda 2. Work through "**Lecture\_1**" Jupyter notebook (and accompanying video)   **Online resources (supplementary)::**   * <https://medium.com/edureka/jupyter-notebook-cheat-sheet-88f60d1aca7> * <https://www.datacamp.com/community/blog/jupyter-notebook-cheat-sheet> |
| **Module 2: Weeks 2-4**  Introduction to basic Python  **Aug 30- Sept 20** | 1. Recognize:    1. Syntax of Python including the use of whitespace and appropriate commenting of code    2. Basic elements of Python: variables, data types, including lists, strings, floats, integers and methods    3. Breaking down a problem using control statements (if/elif/else and AND/OR/NOT) 2. Implement:    1. Appropriate methods of data types including slicing    2. Loops    3. Functions    4. modules 3. Debug:    1. Assertions    2. Using other resources (ie. google and exception statements) 4. Design:    1. Efficient solutions that use control statements 5. Analyze and interpret:    1. Complex code written by others    2. Scope of variables | 1. **Problem set 1 (8 pts):** Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question LO covered:   1a, 1b, 1c, 2a, 3b   1. **Problem set 2 (8 pts):**  Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question   LO covered:  1(a,b,c), 2(a,b,c,d,e), 3(a,b), 4(a), 5(a,b) | | Activities:   1. Watch lecture videos:  * 2Ai,2Aii, 2B, 2Ci,2Cii, 2D, 2E  1. Work through accompanying Jupyter Notebooks (for each video):  * Lecture\_2Ai * Lecture\_2Aii * Lecture\_2B * Lecture\_2Ci * Lecture\_2Cii * Lecture\_2D * Lecture\_2E   **Online resources (supplementary):**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **This is an incredibly useful resource that allows you to run your code and check it!**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***    Python Tutor: <http://pythontutor.com> |
| **Module 3: Weeks 5-9**  More Advanced Python  **Sept 21-Oct 17** | 1. Recognize: 2. Complex data types  * Tuples, dictionaries etc  1. Regular expressions 2. Iterators and generators 3. List comprehensions 4. Identify problems that are best solved using recursion 5. Functional programming tools 6. Identify problems that best use OOP 7. Manifestations of Abstract Data Types – stacks, queues, Deques, lists and how to implement ADT using classes 8. Apply: 9. Rules of regular expressions 10. Recursion 11. Functional programming 12. Iterators and generators 13. OOP 14. ADT 15. Debug:   a. Identify problems in code  b. Correct flaws in program  4. Refactor:   1. Redesign previous problems using more efficient tools   5. Analyze and interpret:   * 1. Scope of variables | **1. Problem set 3 (7 pts** Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question  LO covered:  1a, 1b, 2a, 3a  **2. Problem set 4 (6 pts):** Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question  LO covered:  1(c,d), 2b, 3(a,b), 4a, 5a  **3. Problem set 5 (6 pts):**  Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question  LO covered:  1e, 2(c,d), 3(a,b), 4a,5a  **4. Problem set 6 (6 pts):**  Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question  LO covered:  1f, 2e, 3(a,b), 4a, 5a | | Activities:   1. Watch lecture videos:  * 3A, 3B, 3C, 3D, 3E, 3F  1. Work through accompanying Jupyter Notebooks (for each video):  * Lecture\_3A * Lecture\_3B * Lecture\_3C * Lecture\_3D * Lecture\_3E * Lecture\_3F * Lecture\_3G (Note: not created yet)   **Online resources (supplementary):**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **Yes, I recommend the same website for module 3 as well as module 2. It's just THAT awesome.**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***    Python Tutor: <http://pythontutor.com> |
| **Week 10**  **Wednesday, Oct 27th - Exam 1: Modules 1-3 (inclusive)** | | | | |
| **Module 4:**  **Weeks 9-12**  Libraries  **Oct 18- Nov 14** | 1. Recognize: Important libraries and their features   a. NumPy   * + Arrays   + Subsetting   + application of conditions   + import/export   + major methods  1. Pandas    * Import/export    * Basic tools    * Loc versus iloc    * Applying criteria    * Groupby 2. BioPython 3. Data Visualization libraries/packages 4. SQLite package and how it compares to 'traditional' SQL    * SELECT, APPEND, SORT, FILTER, JOIN, AGGREGATE 5. How to access libraries/packages using Anaconda 6. Apply:    1. Rules of NumPy    2. Rules for Pandas    3. Rules/methods of Biopython    4. SQLITE package    5. When to use/optimize libraries 7. Debug:   a. Identify problems in code  b. Correct flaws in program  4. Refactor:   1. Redesign previous problems using more efficient tools from libraries/packages   5. Analyze and interpret:   1. When should you use libraries 2. Why is NumPy used widely? 3. Why is BioPython used? 4. Types of problems appropriate to use SQLite   6. Design:  a. Build a pipeline between libraries (ie. manipulate the data through Numpy and Pandas and then visualize the results with another library) | **1. Problem set 7 (7 pts):** Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question by the given date LO covered: 1(a,b,f), 2a, 3(a,b), 4a, 5(a,b), 6a  **2. Problem set 8 (6 pts):**  Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question by the given date LO covered:  1(c,d,f), 2b, 3(a,b), 4a, 5(a,c), 6a  **3. Problem set 9 (6 pts):**  Upload Problem set to Blackboard by the due date and upload your created question to the appropriate discussion thread and answer peers posted question by the given date LO covered:  1(e,f), 2(d,e), 3(a,b), 4a,5d,6a | Activities:   1. Watch Lecture Videos    * 4A, 4B, 4C, 4D, 4E 2. Work through accompanying Jupyter Notebooks (for each video):  * Lecture\_4A * Lecture\_4B * Lecture\_4C * Lecture\_4D * Lecture\_4E | |
| **Module 5:**  **Weeks 13-15**  Projects  **Nov 15-Dec 7** | 1. Identify    1. Define a challenging but tractable problem 2. Effectively manage group work    1. Assign tasks    2. Complete tasks on time 3. Design    1. Decompose larger problem into smaller problems using guiding principles of abstraction and modularity    2. Connect smaller solutions together into a pipeline or schema    3. Demonstrate creativity 4. Debug and Refactor    1. Try to 'break' your code    2. Determine limitations of your code    3. Attempt measurement of efficiency and apply iterative optimization/improvement 5. Translate and Present Solution    1. use clear language to explain your approach    2. deliver a concise, and relevant oral presentation that summarizes why your problem is an interesting one and includes analysis of your solution strategy 6. Communicate Professionally    1. Write up a 2 page (approx. 500 words) summary of both your project and code (this could include, depending on your particular project, challenges, tests of robustness of code, optimization, future suggestions for the project) | 1. Group: Short Proposal and group 'contract' with deliverables assigned and questions answered (1-2 pages, less than 500 words)  2. Group program code   1. Group presentation (10 minutes) 2. Individual write-up (500 words, 2 pages) | Online resources:   * *You are expected to propose a project that is commensurate with the number of individuals in your group, ie. the larger the group, the most challenging a problem you will need to pick.* * ***You are strongly encouraged to develop your own project but, in case you are not able to do so, here is a website that is a starting place:***   http://rosalind.info/problems/tree-view/ | |
| **Week 16**  **Wednesday, Dec 8th - Exam 2: Cumulative (Modules 1-4)** | | | | |

**Schedule (and Checklist) Fall, 2021:**

|  |  |
| --- | --- |
| Week | Important Activities in each module |
| 1 | * Fri, Aug 21: **Module 1 opens** * Wed, Aug 25: Classes begin at UR  1. Watch Panopto videos 2. Install Anaconda 3. Open and manipulate Jupyter notebook |
| 2 | * Mon, Aug 30 by noon: **Module 2 opens**  1. Watch video 2A 2. Work through accompanying Jupyter notebook  * Wed, Sept 1:   1. Watch video 2B   2. Work through accompanying Jupyter notebook |
| 3 | **Module 2 continues**   * Mon, Sept 6th: No office hours (Labour Day) * Wed, Sept 8:  1. Watch video 2C 2. Work through accompanying Jupyter notebook |
| 4 | **Module 2 continues**   * Mon, Sept 13:   1. Watch video 2D   2. Work through accompanying Jupyter notebook * Wed, Sept 15:  1. Watch video 2E 2. Work through accompanying Jupyter notebook  * Fri, Sept 17 (by 11:59PM):  1. **Submit PS 1** 2. **Provide made up question to the appropriate discussion board in module 2**  * Sat, Sept 18 (by 11:59PM):  1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered). 2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 5 | * Mon, Sept 20 by noon:  **Module 3 opens**  1. Watch video 3A 2. Work through accompanying Jupyter notebook  * Wed, Sept 22:  1. Watch video 3B 2. Work through accompanying Jupyter notebook  * Fri, Sept 24 (by 11:59PM):  1. **Submit PS 2** 2. **Post made up question to appropriate discussion board in module 2**  * Sat, Sept 25 (by 11:59PM):   1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered).   2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 6 | **Module 3 continues**   * Mon, Sept 27:  1. Watch video 3C 2. Work through accompanying Jupyter notebook  * Wed, Sept 29:  1. Watch video 3D 2. Work through accompanying Jupyter notebook  * Fri, Oct 1(by 11:59PM):  1. **Submit PS 3** 2. **Post made up question to appropriate discussion board in module 3**  * Sat, Oct 2 (by 11:59PM):  1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered) 2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 7 | **Module 3 continues**   * Mon, Oct 4:  1. Watch video 3E 2. Work through accompanying Jupyter notebook  * Wed, Oct 6:  1. Watch video 3F 2. Work through accompanying Jupyter notebook  * Fri, Oct 8 (by 11:59PM):  1. **Submit PS 4** 2. **Post made up question to appropriate discussion board in module 3**  * Sat, Oct 9 (by 11:59PM):   1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered)   2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 8 | **Module 3 continues**   * Mon, Oct 11: Fall break, no Office Hours * Wed, Oct 13:  1. **Project group designation and sign up** 2. Watch video 3G and work through accompanying Jupyter notebook  * Fri, Oct 15 (by 11:59PM): * **Submit PS 5** * **Post made up question to appropriate discussion board in module 3** * Sat, Oct 16 (by 11:59PM): * **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered) * Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 9 | * Mon, Oct 18 by noon: **Module 4 opens**  1. Watch video 4A 2. Work through accompanying Jupyter notebook  * Wed, Oct 20:  1. Watch video 4B 2. Work through accompanying Jupyter notebook  * Fri, Oct 22 (by 11:59PM):  1. **Submit PS 6** 2. **Post made up question to appropriate discussion board in module 3**  * Sat, Oct 23th (by 11:59PM)   1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered)   2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 10 | * Mon, Oct 25 by noon: **Review** * **Wednesday, Oct 27th - EXAM 1: Modules 1,2 and 3** * Fri, Oct 29 (by 11:59PM): * **Submit (as a group): two paragraph proposal of project and group member contract and questions with 'deliverables' assigned** |
| 11 | **Module 4 continues**   * Mon, Nov 1:  1. Watch video 4C 2. Work through accompanying Jupyter notebook  * Wed, Nov 3:  1. Watch video 4D 2. Work through accompanying Jupyter notebook  * Fri, Nov 5 (by 11:59PM):  1. **Submit PS 7** 2. **Post made up question in appropriate discussion thread in module 4**  * Sat, Nov 6 (By 11:59PM):  1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered) 2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 12 | **Module 4 continues**   * Mon, Nov 8:   1. Watch video 4E  2. Work through accompanying Jupyter notebook   * Wed, Nov 9:   1. Watch remaining videos for 4E  2. Work through accompanying Jupyter notebook   * Fri, Nov 12 (by 11:59PM):  1. **Submit PS 8** 2. **Post made up question in appropriate discussion thread in module 4**  * Sat, Nov 13(by 11:59PM)   1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered)   2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 13 | * Mon, Nov 15: **Work on group project** * Wed, Nov 17: **Work on group project** * Fri, Nov 19 (by 11:59PM):  1. **Submit PS 9** 2. **Post made up question in appropriate discussion thread in module 4**  * Sat, Nov 20 (by 11:59PM)  1. **Provide answer** (and explanation of your answer) to unanswered peers posted question (so that everyone answers one question, and all questions are answered) 2. Mark answer to your question as correct or provide additional explanation/clarification if the answer was incorrect |
| 14 | * Mon, Nov 22: **Work on group project**   THANKSGIVING BREAK |
| 15 | * Mon, Nov 30**: Presentations** * Wed, Dec 2**: Presentations** |
| 16 | * Mon, Dec 7: **Presentations** * **Wed, Dec 8th:**   1. **EXAM 2: CUMULATIVE (Modules 1, 2, 3, and 4)**   2. **By 11:59PM – Hand in your individual project write-up and submit code and presentation from entire group.** |

**Calendar format follows on next page….**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sun** | **Mon** | **Tues** | **Wed** | **Thurs** | **Fri** | **Sat** |
| **August**  22 | 23 | 24 | **Lecture 1**  25 | 26 | 27 | 28 |
| 29 | **Lecture 2A**  30 | 31 | **September**  **Lecture 2B**  1 | 2 | 3 | 4 |
| 5 | Labour Day  Holiday  6 | 7 | **Lecture 2C**  8 | 9 | 10 | 11 |
| 12 | **Lecture 2D**  13 | 14 | **Lecture 2E**  15 | 16 | **PS 1 DUE**  17 | **Answer peer question**  18 |
| 19 | **Lecture 3A**  20 | 21 | **Lecture 3B**  22 | 23 | **PS 2 DUE**  24 | **Answer peer question**  25 |
| 26 | **Lecture 3C**  27 | 28 | **Lecture 3D**  29 | 30 | **October**  **PS 3 DUE**  1 | **Answer peer question**  2 |
| 3 | **Lecture 3E**  4 | 5 | **Lecture 3F**  6 | 7 | **PS 4 DUE**  8 | **Answer peer question**  9 |
| 10 | Fall Break – no Office hours  11 | 12 | **Lecture 3G**  **Sign up for Project**  13 | 14 | **PS 5 DUE**  15 | **Answer peer question**  16 |
| 17 | **Lecture 4A**  18 | 19 | **Lecture 4B**  20 | 21 | **PS 6 DUE**  22 | **Answer peer question**  23 |
| 24 | **REVIEW**  25 | 26 | **EXAM 1**  27 | 28 | **Proposal DUE**  29 | 30 |
| 31 | **November**  **Lecture 4C**  1 | 2 | **Lecture 4D**  3 | 4 | **PS 7 DUE**  5 | **Answer peer question**  6 |
| 7 | **Lecture 4E**  8 | 9 | **Catch-up**  **Lecture**  10 | 11 | **PS 8 DUE**  12 | **Answer peer question**  13 |
| 14 | **Dedicated**  **Project time**  15 | 16 | **Dedicated**  **Project time**  17 | 18 | **PS 9 DUE**  19 | **Answer peer question**  20 |
| 21 | **Dedicated**  **Project time**  22 | 23 | **Break**  24 | **Break**  25 | **Break**  26 | 27 |
| 28 | **Presentations**  29 | 20 | **December**  **Presentations**  1 | 2 | 3 | 4 |
| 5 | **Presentations**  **PROJECT**  6 **DUE** | 7 | **EXAM 2**  8 |  |  |  |

* Modules will be opened on the dates given on Bb.
* There is a quite a bit of flexibility in this course and it can be easy to forget what is due and when. I have provided this calendar to remind you about problem set due dates (firm) and suggested dates by which you SHOULD have worked through the lecture material indicated in order to complete the problem sets on time.
* "**Office hours" will be held during scheduled lecture time and the focus material of those office hours are listed on this calendar.** This is mostly my attempt to ensure that questions are at a similar level for all participants. If an individual is asking a question about module 3 when we are covering the material in module 2, that could be disruptive and anxiety-inducing, at worst, and not helpful to other participants, at best (note: if you are the only individual who shows up for class office hours, you can ask any question you like!).
* I am quite happy for you to 'work ahead' but questions about a different module are better directed at office hours with low attendance or with your Teaching Assistants since other students won't necessarily benefit from those questions in a group session.