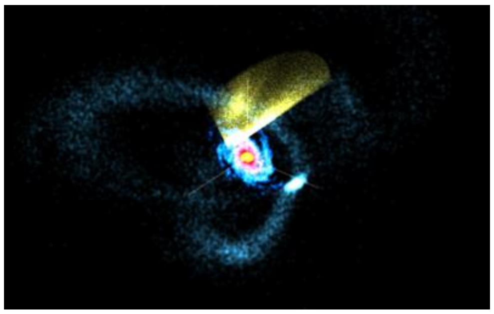

Temple University [1]
Department of Physics
Scientific Computing I: 1.5 Credits
PHYS 2511 – CRN 36041
2022 Spring (7A)

Room: SERC 456
Time: TR 2:00-3:20 pm
Instructor: Prof. Matthew Newby
Email: matthew.newby@temple.edu
Office: SERC 476
Office Hours: TBD; or by appointment.

[1] Course Learning Outcomes

Students who complete this course will learn the following:

- How to write and run programs.
- How to think about problems from a computational perspective.
- How to solve applied problems using simple numerical methods.
- How computers work (very basic computer science).
- Proper coding practices (organization, debugging).
- How to express results in a clear graphical format.
- To understand the errors implicit to numerical solutions.
- The use of function libraries.
- The difference between mark-up, interpreted, and compiled languages

[2] Course Description

An introduction to computing as a tool for solving scientific problems. No previous programming experience is assumed. Students completing this course will be able to write their own computer programs to solve introductory science problems, and will be prepared to understand more advanced techniques. Topics include the basic computer science needed to understand hardware and software processes, program organization and debugging, spreadsheet programs (Excel), interpreted programming languages (Python), compiled programming languages (C++), basic numerical methods

[1] Image created by Shane Reilly and his LModL library for the MilkyWay@home computing project (https://github.com/Milkyway-at-home/milkywayathome_client/tree/master/LModL).

for solving physics problems, basic error analysis, and information visualization techniques.

Prerequisites/Co-requisites:
Prerequisite: A grade of C- or higher in Physics 1021, 1061, 2021, or 2921.
Prerequisite: A grade of C- or higher in Math 1021 and 1022, or test-out equivalent.

[3] Text

Optional Text: *Elementary Mechanics Using Python*
Author: Anders Malthe-Sørenssen
Publisher: Springer
ISBN 978-3-319-19595-7
ISBN 978-3-319-19596-4 (eBook)
This text is available as a PDF through Temple University Library.

[4] Computer Access and Software

As this is a computational course, you will need access to a computer and necessary software (below). While the classroom will have all necessary software installed, a personal computer with software installation i**s required** to be able to work from home (and in the event of a move to online teaching).

Required:
- A text editor, such as gedit, Atom or Notepad++. Microsoft Notepad is not sufficient)
- Git code versioning software, and a Github account
- Python (preferably version 3.x) – https://www.python.org
  - Numpy (version 1.10 or higher) – https://numpy.org
  - Scipy (version 1.0 or higher) – https://scipy.org
  - Matplotlib (version 2.1 or higher) – https://matplotlib.org

Optional:
- Excel (or another spreadsheet program, such as OpenOffice)
- Gitkraken (for easy Git/Github integration, see Student Developer Pack)
- Atom (a useful, hackable, multiplatform text editor, also in Student Developer Pack)

Online Resources:
- Student Developer Pack: https://education.github.com/pack
- Bash Shell for Windows: https://www.howtogeek.com/249966/how-to-install-and-use-thelinux-bash-shell-on-windows-10/
- Stackoverflow (programming forums and advice) https://stackoverflow.com/
- A Guide to Git: http://guides.beanstalkapp.com/version-control/common-git-commands.html

[5] Advice

1. This is a **fast-paced short** course. We have a few short weeks to work together. Make sure to take time each day to work on the course so that you do not get lost.
2. Engage in the class. Think ahead to the next step and learn what questions to ask yourself when you first confront a challenge. Each student is responsible for all materials covered in all lecture sessions.
3. Engage in lab activities. At worst it does not work the first N times you try, but be willing to try things. Watching and reading about coding is not effective.

4. Practice Makes Perfect — The central part of your learning is the set of lab assignments.
    a. Lab assignments can be one of the best tools for learning Python — or the most abused portion of the course. The serious student will work the problem conscientiously and in a consistent manner, while the student who is not careful will rely on others and rush to do it at the end. These assignments can help you harness the power of computing in science or hinder you, depending on how you approach them.
    b. Although this is a 1.5 credit course, during the first half of the semester it will require the time of a 3 credit course. Make sure to put in the time so you do not fall behind. There is little time to catch up.
5. Teamwork — Work together; discuss and argue about issues. That does NOT mean copying final code, but it can mean getting a piece of the code as long as you are clear with the history of it (something that Github will help you do). You are not competing against each other for grades. Pull each other up and push each other on.
6. Help: Get help from a variety of sources, **especially the professors' office hours.** *BUT*, after getting help, go practice a similar technique on your own to solidify your understanding.

[6] Instructor Information

- Professor Matthew Newby (matthew.newby@temple.edu) Office: SERC 476

[7] Canvas/Communication

- The course has an on-line component. All course related materials, announcement and assignments, including current grades, are available on Canvas.
- All communication will be through the course Canvas page or Github.

[8] Computing Help

- For help outside of class you should visit the instructor during office hours. If the office hours do not fit into your schedule, appointments may be made.
- The Department of Computer and Information Science has tutoring at a wide variety of times listed at https://cis.temple.edu/labs#LabSchedules
- The Student Success Center offers free **Computer Programming tutoring** for Python at any time they have a tutor for **CIS 1051: Intro to Python**. During STEM tutoring sessions, a tutor will work with you one-on-one to review and understand the concepts in this course. More information about STEM tutoring, as well as other services offered by the SSC, is available at https://studentsuccess.temple.edu/, by calling 215-204-0702, or by visiting the SSC's STEM Learning Lab in Charles Library, room 340.
- Google can lead you to useful examples of code or techniques as well as alert you to common issues.
- Stackoverflow can lead you to useful examples of code or techniques as well as alert you to common issues.

[9] Tips for Good Coding

- If it's not backed up, assume it will be lost.
- Document your code – for a future you if nobody else.
- Set aside long blocks of time to write code. This is a subject which requires a bit of flow rather than piecemeal work.
- Test everything. Try to break it.

- Your goal is not to be a computer scientist, but to learn to be comfortable with coding as a tool that helps you in science.

[10] Lecture Attendance/Participation

The scheduled lecture period will consist of short lectures, quizzes, and hands-on programming labs. Attendance will not be taken, but as in-class work is important to your final grade it is in your best interest not to miss classes.

[11] Programming Labs

The bulk of your grade in this course will be from programming labs. Programming labs will begin during class meetings and will be due 1 week after they are assigned. One or two programming lab assignments will be due each week (see Schedule below).

All lab work will be due via Github. This will require you to create a Github account. Github accounts are free for public, open source code, and private accounts are available for free. More details will be provided during the first day of class.

You will need to be able to write and run Python language programs in order to pass this course. Multiple approaches will be discussed during the first class meeting. Ultimately, your instructor will need to be able to run your program (which means you must be careful using additional libraries or modules – this will be discussed in class).

Programming lab submission grades will depend on the following: timely submission; proper submission through Github; original contribution; ability to solve the given problem; and ability to meet all of the specific objectives put forward on the assignment. Each assignment will be graded out of 100 points, although scores of higher than 100 may be possible for assignments with bonus objectives. A grading breakdown will be provided alongside each assignment.

Each programming lab will be weighted equally. No lab grades will be dropped. Details and individual programming labs will be posted on Github and discussed in class.

As professional coding is often collaborative, you are welcome to work together on all programming labs, but you will be held responsible for your individual assignments, and your part of a cooperative assignment. Each student's submission must also be unique; a completely or slightly modified program copied from online sources or from other students will not be accepted, resulting in a "0" grade for that assignment (and may constitute plagiarism, depending on the context).

Important: When collaborating, it is vital that you do so through Github (such as pushing a change to someone else's code or forking someone else's work.) This will keep a record of your effort, which will be useful beyond this course.

[12] Program Resubmission

The final programming assignment in this course is to take any or all of assignments #1 through #5 and improve upon them. For each assignment improved, up to 50% of the missing potential grade may be earned (for example, an assignment with a 50% grade can earn up to 25% more). If you are happy with your grades on all assignments, you may choose to skip this assignment.

This assignment provides you with the opportunity to improve programs written with weaker skills or hurried assignments, and will give you the opportunity to revisit old problems and work with your previous code.

Your instructor will post a survey alongside this assignment, in which you will clearly indicate which assignments you want to be considered as resubmissions.

You may choose to improve any of assignments #1 through #5, and this resubmission is due on **Friday, February 25**. Anything submitted after February 11 will not be counted towards your grade.

[13] Conceptual Reading

Once or twice a week you will receive concepts that you will be expected to research and briefly summarize. Your summary will need to answer the question(s) asked in the assignment, and feature links or references to the material(s) used. Wikipedia (and the like) is an acceptable source, but you may find it to be too technical at times; a second opinion is always prudent (but not required). Conceptual reading assignments will be due 1 week after they are assigned.

There are 12 planned conceptual reading assignments, but this number may vary if new interesting articles or news items become available during the run of this course.

[14] Quizzes

Each week (except for the first) will end with a short online quiz on programming concepts and applications. Quizzes will follow course material and conceptual readings closely, but may also include written-out Python code snippets and ask you for the output or to find an error in the code.

Quizzes will be completed on Canvas outside of class time.

[15] Overall Grade Assessment

Assignment Type	Number of Assignments	Total Portion Of Final Grade
Programming Labs	6	60%
Conceptual Reading	~12	10%
Quizzes	6	30%

Grades in this class will not be curved at the end of the semester. Rounding of grades will be limited to maximum of one-tenth of a percent.

A	**A-**	**B+**	**B**	**B-**
92%...100%	89% ...<92%	85% ...<89%	81%...<85%	77%...<81%
C+	**C**	**C-**	**D**	**F**
73%...<77%	69%...<73%	65%...<69%	55%...<65%	< 55%

[16] Quizzes/Exams

- All quizzes and exams are “closed book”. Your instructor will provide you with an equation sheet (if needed) for all quizzes and exams.
- Note that you will **not be** allowed to use your mobile phone during any quiz.

[17] Academic Honesty Policy:

Students are encouraged to work together on labs and homework but are expected to submit their own answers. Evidence of dishonest work on any assignment will result – at minimum – in a “zero” grade for that assignment. Dishonest work on a quiz, or repeated abuses, will result in an automatic failing grade in the course. Any instance of dishonest work may warrant a formal charge of academic dishonesty.

All students who are enrolled in Temple University have agreed to Temple University’s honesty policy. Excerpt of the honesty policy:

> *Temple University believes strongly in academic honesty and integrity. Plagiarism and academic cheating are, therefore, prohibited… Plagiarism is the unacknowledged use of another person's labor, another person's ideas, another person's words, another person's assistance. ... the penalty for academic dishonesty can vary from receiving a reprimand and a failing grade for a particular assignment, to a failing grade in the course, to suspension or expulsion from the university.*
>
> • Full policy may be found at:
> http://bulletin.temple.edu/undergraduate/about-temple-university/student-responsibilities/#academichonesty

[18] Disabilities and Accommodations

Temple University is committed to the inclusion of students with disabilities and provides accessible instruction, including accessible technology and instructional materials. *Any student who has a need for accommodations based on the impact of a documented disability or medical condition should contact Disability Resources and Services (DRS) in 100 Ritter Annex (drs@temple.edu; 215-204-1280) to request accommodations and learn more about the resources available to you. If you have a DRS accommodation letter to share with me, or you would like to discuss your accommodations, please contact me as soon as practical. I will work with you and with DRS to coordinate reasonable accommodations for all students with documented disabilities. All discussions related to your accommodations will be confidential.*

[19] Academic Freedom

Freedom to teach and freedom to learn are inseparable facets of academic freedom. The University has adopted a policy on Student and Faculty Academic Rights and Responsibilities (Policy # 03.70.02) which can be accessed through the following link: http://policies.temple.edu/getdoc.asp?policy_no=03.70.02.

[20] Key Dates and Schedules:

- Programming labs are due 1 week after they are assigned.
- Programming lab resubmission is due on Friday, February 25.
- Quizzes are due the Friday of the week they are assigned.
- Conceptual Readings will be due (on Canvas) 1 week after they are assigned.

- First day of 7A course**:** Monday January 10
- 7A course add/drop date: Tuesday January 18
- Last day to withdraw from 7A course: Thurs February 24
- Spring break: MTWRF February 28 - March 4
- Last day of 7A courses: Monday March 7

Detailed Schedule:

Week	**Topics Covered**	**Programming Labs**	**Quiz (Due Fri)**
#1: Jan 10 – 14	I. The Great Insights of Computer Science: Binary (Leibniz, et al.), Turing Machine (Turing), Logical Operations (Bohm & Jacopini) II. Computers as machines III. How software works IV. Summaries of major areas/problem types V. Binary math, binary representations of data (ASCII) VI. Intro to Programming VII. Intro to Python	0. Github Access 1. Magic 8-Ball	None
#2: Jan 17 – 21	VIII. Debugging IX. Programming structure: Functions, Recursion, Libraries. X. Numerical Python (Numpy), Scientific Python (Scipy)	2. Virtual Board Game	#1
#3: Jan 24 – Jan 28	XI. Arrays XII. Matplotlib Library: Visualization and plotting	3. Reading a Classic	#2
#4: Jan 31 – Feb 4	XIII. Randomness, intro to Stochastic Methods	4. Blindfolded Archer	#3
#5: Feb 7 – 11	XIV. Area under a curve: Numerical integration XV. Intro to N-Body	5. Numerical integration	#4
#6: Feb 14 – 18	XVI. Orders calculations XVII. Classes	6. Data Exploration	#5
#7: Feb 21 – Feb 25	XVIII. Compiled Programs	(+). Program (1-5) Resubmission	#6
#8: Feb 28 Mar – 4	***Spring Break***		
#9: Mar 7	7A classes end Monday		