**Carleton University**
**Department of Systems and Computer Engineering**
**Course Outline**

# ECOR 1051 — Fundamentals of Engineering I — Fall 2019

## Instructors

Sections A/G and C/I: Don Bailey; Office: ME 4438; Email: donald.bailey@carleton.ca

Sections B/H: Cheryl Schramm; Office: ME 4444; Email: cheryl.schramm@carleton.ca

## Office Hours

Office hours are posted on cuLearn.

## Undergraduate Calendar Course Description

**ECOR 1051 [0.5 credit]**
**Fundamentals of Engineering I**
Software development as an engineering discipline, using a modern programming language. Tracing and visualization of program execution. Testing and debugging. Data management: digital representation of numbers; numerical algorithms; storing data in files; container data types: sequences, sets, maps.
Includes: Experiential Learning Activity
Precludes additional credit for ECOR 1606, SYSC 1005.
Prerequisite(s): This course may not be taken concurrently with ESLA 1300 or ESLA 1500.
Lectures three hours per week, laboratories three hours per week.

## Prerequisites

No prior experience in computer programming is required. The only background we assume is:

- *language*: reasonable proficiency in reading and writing English;
- *math*: understanding of integers and operations on integers; understanding of functions as mappings from one set (the domain) to another set (the codomain);
- *logic*: familiarity with logical *and*, *or* and *not*;
- *computer literacy*: ability to use email, browse the World Wide Web, and edit text files.

## Course Objectives

By the end of this course, students should:

- know the fundamental concepts of procedural programming, using Python as the programming language.
- have gained practical experience using lightweight, modern software engineering practices in a team environment to design and implement small-scale programs.
- have developed a "mental model" of computation; in other words, learned how to reason

about and visualize the execution of program code.

- understand the use of software experiments as an aid to learning.

**Learning Outcomes**

By the end of this course, students should:

1. be able to evaluate expressions consisting of literal values, variables and operators, using the same evaluation rules as Python. In other words, students should be able to predict the values that Python calculates when it evaluates expressions.

2. know the syntax and semantics of Python's fundamental constructs for controlling program execution. These constructs include: sequential execution as determined by the ordering of executable statements; selection (`if`, `if-else` and `if-elif-else` statements); repetition (`for` and `while` statements); function calls.

3. be able to trace short Python programs and

    - explain what happens, step-by-step, as the computer executes each statement;

    - visualize code execution; in other words, draw diagrams that depict the variables in the program's global frame and function activation frames (function arguments and local variables), and the objects that are bound to the variables.

4. be able to design algorithms for functions that satisfy detailed specifications, convert these algorithms to code, and use an integrated development environment to edit and execute the functions.

5. be able to test their functions interactively, using the Python shell, and by using a simple unit-testing framework. Students should be able to apply simple debugging techniques (e.g., inserting `print` statements to instrument code or by tracing the code using a program visualization tool) to locate faults in their code.

6. understand how signed and unsigned decimal integers can be represented as binary numbers. Students should understand some of the limitations of real-number computation using floating-point arithmetic. Students should know some simple numerical algorithms.

7. know the "client-side" view of four Python containers for organizing data, namely: lists, tuples, sets and dictionaries. Students should be able to select and use appropriate container(s) in their programs.

8. be able to work in a small team to iteratively and incrementally design, implement and test a small-scale, interactive program that is partitioned into multiple modules, given a detailed specification of the functional requirements.

**Modules**

This course consists of two modules. *Introduction to Computation and Programming* runs from the start of classes until approximately half-way through the term. It introduces students to concepts that are central to understanding computation and teaches students how to design, code,

test and debug small-scale programs written in the Python programming language. The second module, *Data Management*, runs from Break Week until the end of term. It focuses on developing Python programs that can process potentially large quantities of data. This module also includes an introduction to numerical methods (algorithms for engineering and scientific applications) and a significant team software development project.

**Graduate Attributes (GAs)**

The Canadian Engineering Accreditation Board requires graduates of undergraduate engineering programs to possess 12 attributes.[1] Courses in all four years of our programs evaluate students' progress towards acquiring these attributes. Aggregate data (typically, the data collected in all sections of a course during an academic year) is used for accreditation purposes and to guide improvements to our programs. Some of the assessments used to measure GAs may also contribute to final grades; however, the GA measurements for individual students are not used to determine the student's year-to-year progression through the program or eligibility to graduate.

This table lists the GAs that will be measured in this course, along with the learning outcomes that are intended to develop abilities related to these attributes.

| **Graduate Attribute/ Indicator** | **Instructional Level[2]** | **Learning Outcomes** |
|---|---|---|
| 1.3 Knowledge base for engineering: Fundamental engineering concepts | I | all |
| 5.3 Use of engineering tools: Tools for design, experimentation, simulation, visualization and analysis | I | 1 - 5, 8 |
| 6 Individual and team work<br>● 6.1 Personal and group time management<br>● 6.2 Group culture, group dynamics<br>● 6.3 Leadership: initiative and mentoring, areas of expertise, and interdisciplinary teams | I | 8 |
| 7.1 Communications skills: Giving and following instructions | I | 4, 5 (following)<br>8 (giving) |

Final grades will be converted to measurements for GA 1.

[1] Criterion 3.1, *2018 Accreditation Criteria and Procedures*, Canadian Engineering Accreditation Board, November 2018.

[2] The instructional level of course content related to graduate attributes is classified by the content-level codes I (Introduced), D (Developed) and A (Applied). These codes are defined in *A Guide to Outcomes-Based Criteria*, Version 1.25, Canadian Engineering Accreditation Board, 1 March 2015.

A short GA measurement exercise, which will be held in one of the labs approximately half-way through the course, will be used to measure GAs 5 and 7 (following instructions).

Some of the team project deliverables will be used to measure GAs 6 and 7 (giving instructions).

In addition, this course will introduce students to design in the context of software development. This will prepare students to undertake learning activities in subsequent courses that develop competence in GA 4 (Design)[3]. GA 4 will not be evaluated in this course.

**Course Web Site**

This course uses cuLearn, Carleton's learning management system. To access your courses on cuLearn, go to carleton.ca/culearn.

For help and support, go to carleton.ca/culearnsupport/students. Any unresolved questions can be directed to Computing and Communication Services (CCS) by phone at 613-520-3700 or via email: ccs_service_desk@carleton.ca.

**Textbook**

*Practical Programming (Third Edition): An Introduction to Computer Science Using Python 3.6*, Paul Gries, Jennifer Campbell, Jason Montojo, Pragmatic Bookshelf, 2017, ISBN-13: 978-1-68050-268-8.

Despite the phrase "Python 3.6" in the title, the book is equally applicable to the Python release we'll be using.

An eBook (PDF, epub and mobi formats) or paperback copy of this book can be purchased directly from the publisher's website: pragprog.com. *Make sure you purchase the third edition, not the second edition.* The eBook costs $26.95 USD, and you can download the eBook in all formats at no extra charge. If you prefer a paper book, it's priced at $41.95 USD. If you order the eBook and paper book at the same time, the cost is $52.95 USD; but if you prefer to read paper books, it's probably less expensive to purchase the eBook and print the chapters we cover in the course.

A paperback copy can be purchased from amazon.ca. (At the time this document was prepared, the price was about $55 CAD and it was eligible for free shipping. Third-party sellers with storefronts on amazon.ca sell new and used copies, but prices, shipping costs and delivery times vary.)

[3] "An ability to design solutions for *complex*, *open-ended* engineering problems and to design systems, components or processes that meet specified needs with appropriate attention to health and safety risks, applicable standards, and economic, environmental, cultural and societal considerations." Criterion 3.1, *2018 Accreditation Criteria and Procedures*, Canadian Engineering Accreditation Board, November 2018.

**Software**

All the software used in this course is free, and is available for the Windows, macOS and Linux operating systems.

- Python 3.7.4 can be downloaded from: www.python.org/downloads/release/python-374/

  Scroll down the page to find the list of installers. For Windows platforms, download the Windows x86-64 executable installer: python-3.7.4-amd64.exe. If you're using macOS 10.9 or a newer release, python.org recommends that you download the 64-bit installer: python-3.7.4-macosx10.9.pkg.

- Pillow 6.1.0 can be downloaded from: pypi.org/project/Pillow

  Click Download files to display the long list of installers. **Important:** there are several different 32-bit and 64-bit Pillow installers, each one corresponding to a different Python release (i.e., Python 2.7, 3.5, 3.6 and 3.7). Ensure that you download the 64-bit installer for Python 3.7. For Windows platforms, the installer file is named: Pillow-6.1.0.win-amd64-py3.7.exe. For macOS, the installer is a Wheel file with a very long name: Pillow-6.1.0-cp37-cp37m-macosx_..._64.whl.

- The integrated development environment (IDE) we'll use is Wing 101 v. 7.1.0. It can be downloaded from: wingware.com.

  For Windows platforms, the installer for Wing 101 is wing-101-7.1.0.2.exe. This installer supports both 32-bit and 64-bit installations. For macOS (referred to as OS X on the Wingware website), the 64-bit installer is wing-101-7.1.0.2.dmg.

  Two other IDEs are available from this website: Wing Personal and Wing Pro. Don't install Wing Pro. This product has a free 30-day trial license, after which a license must be purchased.

Installing the software is easy: just run the three installers. Install Python first, followed by Pillow, and then install Wing 101.

*Python Tutor* (visit pythontutor.com).

Learning how to trace and explain the execution of short programs is an important learning outcome in this course. Throughout the course, we'll use Python Tutor, a free Web-based tool that helps us visualize what happens as the computer executes each line of a program's source code, step-by-step.

**Evaluation and Grading Scheme**

Second-year status is a prerequisite for the second-year engineering, science and mathematics courses that are part of your program. One of the requirements for achieving second-year status is a minimum grade of C- in each of ECOR 1051, ECOR 1052, ECOR 1053 and ECOR 1054.

In ECOR 1051, each module is graded and is 50% of the final course grade. To receive a final course grade of C- or higher, a grade of C- or higher is required in each module.

The final course grade will be calculated as follows:

1. The grade for each module will be calculated as described below and added to create the *total calculated grade* in the course.
2. If a grade of at least C- (60%) is received for both modules, the final course grade will be the *total calculated grade*.
3. If a grade below C- (60%) is received for either module, the final course grade will be the *total calculated grade* or D+, whichever is lower.

Students who receives a final course grade below C- must repeat ECOR 1051. If the course is repeated within the next academic year, the student does not have to repeat a module in which a grade of C- or higher was earned. Instead, the student may elect to have that module grade "carried forward" to the next attempt at the course.

**Grading Scheme for *Introduction to Computation and Programming***

Students will be evaluated primarily by means of a midterm exam and a final exam. In addition, the marks assigned for the online reading assignment quizzes, tutorial labs, GA measurement exercise and completion of Learning Support Workshops (Incentive Program) will contribute towards the module grade.

Students who fail the final exam will receive a module grade of F, regardless of their marks in the other components. For students who pass the final exam, a numeric mark out of 100 will be calculated by weighting the course components as shown in this table:

| **Component** | **Weight** |
|---|---|
| Reading assignment quizzes | 5% |
| Learning support workshops | 5% |
| GA measurement exercise | 5% |
| Tutorial labs | 10% |
| Midterm exam | 25% |
| Final exam | 50% |

This mark will be converted to a letter grade, using the table of percentage equivalents shown in the *Undergraduate Calendar*, *Academic Regulations of the University*, Section 5.4, *Grading System*.

**Grading Scheme for *Data Management***

Students will be evaluated primarily by means of a team project and a final exam. In addition, the marks assigned for the online reading assignment quizzes, tutorial labs and project labs will contribute towards the module grade.

Students who fail the final exam will receive a module grade of F, regardless of their marks in the other components. For students who pass the final exam, a numeric mark out of 100 will be calculated by weighting the course components as shown in this table:

| Component | Weight |
|---|---|
| Reading assignment quizzes | 5% |
| Tutorial labs and project labs | 5% |
| Team project | 40% |
| Final exam | 50% |

This mark will be converted to a letter grade, using the table of percentage equivalents shown in the *Undergraduate Calendar*, *Academic Regulations of the University*, Section 5.4, *Grading System*.

**Early Feedback**

See Section 5.3 of the *Academic Regulations of the University*.

Your solutions to the tutorial lab exercises will normally be graded no later than one week after they are submitted. Midterm exam marks will be available shortly after Break Week.

**Overview of Graded Term Work**

Information about the midterm exam is provided in the section, *Exams*.

**Online Reading Assignment Quizzes**

Reading assignments from the textbook will be posted on cuLearn, to introduce the terminology and concepts that will be applied in upcoming lectures and labs. Short multiple-choice quizzes related to the reading assignments will be posted on cuLearn. You will receive full marks for each quiz you attempt; in other words, your quiz marks are essentially participation marks, and are not dependant on the number of questions you answer correctly.

**Learning Support Workshops (Incentive Program)**

Students will complete three Learning Support Workshops as part of the Incentive Program administered by the Centre for Student Academic Support. Information about participation in this program is provided on cuLearn.

**Team Project**

The team project will begin immediately after Break Week, and will run until the end of term. This project will provide you with practical experience using some lightweight, modern software engineering practices to develop a small-scale, interactive program that is partitioned into multiple modules. Information about the project will be provided on cuLearn.

**Labs**

This course has two 80-minute labs every week. During the *Introduction to Computation and*

*Programming* module, all of the labs will be tutorial labs. During the *Data Management* module, most of the labs will be project labs, but there will be a small number of tutorial labs.

**Tutorial Labs**

During tutorial labs you will work on short programming exercises that are intended to help you understand concepts that have been introduced in the lectures.

On Fridays, we will post the "handouts" for the following week's labs. To receive credit for a lab, you must submit your lab work to cuLearn no later than 11:55 pm on the Sunday after the lab. A calendar of lab deadlines will be posted on cuLearn.

If you complete all the exercises before your scheduled lab, you do not have to attend the lab, but it is your responsibility to submit your lab work on time. If you "forget" to submit your solutions to the exercises by the deadline, you will receive 0 for that lab, regardless of how far in advance you completed the work.

Students who are keeping up with the reading assignments and lectures should be able to complete most of the exercises within the 80-minute labs; however, we recommend that you attempt at least a few of the exercises before your labs, so that you know ahead of time that you require help.

Students with little or no programming experience are strongly encouraged to attend each tutorial lab, even if they finish the exercises ahead of time. You can then use the lab session to work on extra exercises, ask the teaching assistants questions about some of the more challenging concepts, prepare for exams, etc.

Your lab work will be graded *satisfactory*, *marginal*, or *unsatisfactory*.

- *Satisfactory* means that you made reasonable progress towards completing the exercises. Note that you do not have to finish all the exercises to receive *satisfactory*. For each s*atisfactory*, you will receive 2 marks towards the module's tutorial lab component..
- *Marginal* means that you made some progress towards completing the exercises, but your solutions were not sufficiently complete to warrant *satisfactory*. This grade indicates that you may be falling behind and should take steps to remedy this situation. For each *marginal*, you will receive 1 mark towards the module's tutorial lab component..
- *Unsatisfactory* means that you made little or no progress towards completing the lab exercises. This indicates that you are likely having difficulty understanding important concepts and should seek help from your instructor as soon as possible. Each *unsatisfactory* grade will receive 0 marks.

Students cannot "make up" a missed lab by attending a lab section other than the one in which they are enrolled. The lab handouts will be posted ahead of time, so if you know that you have to miss a lab because you have a medical appointment, job interview, travel plans, etc., it is your responsibility to complete the lab work on your own time and submit it by the deadline.

If you are unable to submit your lab work on time because of illness, injury or extraordinary circumstances beyond your control, contact your instructor no later than 3 days after the submission was due. **You must provide your instructor with appropriate documentation**

**that confirms the reason you could not complete the lab work on time**, either in person or by emailing a scanned copy. For these cases, alternate arrangements will be made for you to submit your lab work for grading. For more information, see the *Academic Regulations of the University*, Section 4.4, *Deferred Term Work*.

If you are unable to submit your lab work on time for reasons related to disabilities, pregnancy, religious obligations or participation in certain student activities, please contact your instructor to arrange appropriate accommodations. (See *Academic Accommodations*, towards the end of this document).

Serious long-term illness will be dealt with on an individual basis; in these circumstances, please contact your instructor to discuss appropriate arrangements.

Portions of the designs and code from any lab may be reused and refined in subsequent labs, and doing the labs is the best way to learn the course material and prepare for the exams, so students are encouraged not to "write off" any particular lab just because of its relatively low weight in the overall grading scheme.

Students are responsible for backing up their lab work; for example, we recommend that you copy your files to a USB flash drive or to a cloud-based file hosting service; e.g., Google Drive, Dropbox, OneDrive, etc. **Requests for extra time to work on a lab, because you don't have the required files from a previous lab, will not be approved.**

**Project Labs**

Some lab sessions will be devoted to working on the team project. Team members may be asked by a TA to demonstrate and provide detailed explanations of your software; for example, discuss your design decisions, explain how you would modify your code to reflect different requirements, etc.

Attendance at these labs is mandatory: your attendance record and your interactions with the TAs will count towards your lab mark for the *Data Management* module.

If you have a reasonable reason for missing a project lab, contact your instructor no later than 3 days after the lab. **You may be asked to provide documentation that confirms the reason for your absence (e.g., a medical certificate, appointment card or email from a doctor's office, or other appropriate supporting documentation).**

**GA Measurement Exercise**

A short exercise to measure your progress towards achieving Graduate Attributes 5.3 and 7.1 will be held in one of the labs. Computers will be used during this exercise. It is not intended to evaluate your programming skills, so you will be provided with all the Python code your require. The exercise will evaluate your ability to use the programming tools you've learned during the term, and your ability to follow instructions.

### Important Considerations When Submitting Files to cuLearn

It is your responsibility to ensure that all submissions (e.g., tutorial lab work, project deliverables) are completed on time. As explained earlier, there are a very limited number of exceptions to this policy, and all such requests must have appropriate supporting documentation.

Technical problems do not exempt you from the requirement to meet the submission deadlines. If you wait until the last minute and there are issues with your Internet connection, you will not be given an extension. You are advised to:

- periodically submit your "work in progress" as a draft; for example, submit the file(s) containing the work you've completed at least once a day. **Important:** in ECOR 1051, when you submit a file to cuLearn its submission status will be *Draft (not submitted)*. This allows you to keep a draft version of your submission on the system and, by clicking the Edit submission button, delete or replace it. Don't click the Submit assignment button until you are ready to submit the final version of your files for grading. After you do this, the submission status will change to *Submitted for grading*. You will not be able to make any more changes to your submission, and that's the one that will be graded!

- submit your final submission for grading at least one hour in advance of the deadline.

Make sure that each file you submit has the specified filename and format. For example, if you submit a Python source code file with an incorrect filename, you will receive 0 for that submission, because the test harness provided to the TAs won't run. Similarly, if you are asked to submit a PDF file and you instead submit a Word file, you will receive 0 for that submission.

Make sure that every Python source code file you submit can be opened and run in the Wing 101 IDE. If it doesn't, you will receive 0 for that submission. We recommend that, after submitting source code files, you download the files from cuLearn and verify that:

- the downloaded files have the correct filenames (see the previous paragraph);

- the source code in those files can be viewed in the Wing 101 editor;

- no syntax errors occur when the code is run.

## Exams

### Midterm Exam

There will be one closed-book midterm exam, which will be held in class shortly before Break Week. Computers will **not** be used during the midterm exam.

Students who are unable to write the midterm exam because of illness or other circumstances beyond their control (e.g., family or religious obligations) should contact their instructor to request accommodation for the missed exam. These requests must be made no later than 3 working days after the exam date, and must fully supported by appropriate documentation (in cases of illness, a medical certificate is required). For more information, see the *Academic Regulations of the University*, Section 4.4, *Deferred Term Work*.

Requests for accommodation because of poor performance on the midterm exam will not be

considered. So, if you are ill on the day of the midterm exam, don't write the exam and later claim that your performance was impaired because you were unwell. You are better off to miss the exam, obtain a medical certificate and request accommodation.

### Final Exam

A closed-book, three-hour final exam will be held during the University's December examination period. All students are eligible to write the final exam, regardless of the marks they received during the term. Computers will **not** be used during the final exam.

The final exam will have two parts: Part A is the final exam for the *Introduction to Computation and Programming* module and Part B is the final exam for the *Data Management* module. Both parts will be handed out at the beginning of the exam. Good time management is crucial to ensure that you complete both parts: we recommend that you allocate at most 90 minutes for each part, and after attempting each part, use any remaining time to attempt any unanswered questions.

Students who miss the final exam because of illness or other circumstances beyond their control may apply to write a deferred examination. For more information, see the *Academic Regulations of the University*, Sections 4.3.1, *Deferred Final Examinations*, 4.3.2, *Missed Deferred Examinations*, and 4.3.3, *Early Departure from Final Examinations*

The final examination is for evaluation purposes only and will not be returned to students. You will be able to make arrangements with your instructor or with the department office to see your marked final examination after the final grades have been made available. Your exam will not be remarked during this meeting and solutions to the exam questions will not be provided.

## General Regulations

### Copyright on Course Materials

The materials created for this course (including the course outline and any slides, notes, program source code, labs, projects, assignments, quizzes, exams and solutions) are intended for personal use and may not be reproduced or redistributed or posted on any website without prior written permission from the author(s).

### Attendance

The University requires students to have a conflict-free timetable. For more information, see the current *Undergraduate Calendar*, *Academic Regulations of the University*, Section 2.1.3, *Course Selection and Registration* and Section 2.1.7, *Deregistration*. Requests to accommodate absences from labs and midterm exams because of conflicts with jobs or vacation plans will not be considered.

### Health and Safety

Every student should have a copy of our Health and Safety Manual. A PDF copy of this manual is available online: sce.carleton.ca/courses/health-and-safety.pdf.

**Deferred Term Work**

Students who claim illness, injury or other extraordinary circumstances beyond their control as a reason for missed term work are held responsible for immediately informing the instructor concerned and for making alternate arrangements with the instructor and in all cases this must occur no later than three (3.0) working days after the term work was due. The alternate arrangement must be made before the last day of classes in the term as published in the academic schedule.

For more information, see the *Academic Regulations of the University*, Section 4.4, *Deferred Term Work*.

**Appeal of Grades**

The processes for dealing with questions or concerns regarding grades assigned during the term and final grades are described in the *Academic Regulations of the University*, Section 3.3.4, *Informal Appeal of Grade* and Section 3.3.5, *Formal Appeal of Grade*.

**Academic Integrity**

Students should be aware of their obligations with regards to academic integrity. Please review the information about academic integrity at: carleton.ca/registrar/academic-integrity/

This site also contains a link to the complete Academic Integrity Policy that was approved by the University's Senate.

**Academic Accommodations**

You may need special arrangements to meet your academic obligations during the term. To request academic accommodations, the processes are as follows:

**Pregnancy or Religious Obligations**

Email your instructor with any requests for academic accommodations during the first two weeks of term, or as soon as possible after the need for accommodation is known to exist. For more details, see the *Student Guide to Academic Accommodation* and the *Accommodation* pages at the Equity Services website: carleton.ca/equity/accommodation

**Academic Accommodations for Students with Disabilities**

The Paul Menton Centre for Students with Disabilities (PMC) (website: carleton.ca/pmc) provides services to students with Learning Disabilities (LD), psychiatric/mental health disabilities, Attention Deficit Hyperactivity Disorder (ADHD), Autism Spectrum Disorders (ASD), chronic medical conditions, and impairments in mobility, hearing, and vision. If you have a disability requiring academic accommodations in this course, please contact PMC at 613-520-6608 or pmc@carleton.ca for a formal evaluation. If you are already registered with the PMC, contact your PMC coordinator to send your instructor your Letter of Accommodation at the beginning of the term, and no later than two weeks before the first in-class scheduled test or exam requiring accommodation. After requesting accommodation from PMC, meet with your instructor to ensure accommodation arrangements are made. Please consult the PMC website (carleton.ca/pmc/students/dates-and-deadlines) for the deadline to request accommodations

for the formally-scheduled final exam.

**Survivors of Sexual Violence**

As a community, Carleton University is committed to maintaining a positive learning, working and living environment where sexual violence will not be tolerated, and its survivors are supported through academic accommodations as per Carleton's Sexual Violence Policy. For more information about the services available at the university and to obtain information about sexual violence and/or support, visit carleton.ca/sexual-violence-support.

**Accommodation for Student Activities**

Carleton University recognizes the substantial benefits, both to the individual student and for the university, that result from a student participating in activities beyond the classroom experience. Reasonable accommodation will be provided to students who compete or perform at the national or international level. Please contact your instructor with any requests for academic accommodations during the first two weeks of class, or as soon as possible after the need for accommodations is known to exist. For more information, read the Senate Policy on Accommodation for Student Activities.

**List of Topics**

Module *Introduction to Computation and Programming*

1. Introduction to Python: using the Python shell as a calculator. Numeric types (`int` and `float`); operations supported by these types. Construction and evaluation of expressions. The assignment operator: binding values to variables. Using variables in expressions. Using Python Tutor to visualize assignment operations.

2. Calling built-in functions from the shell. Importing functions from modules. Evaluating call expressions.

3. Defining functions. Passing arguments to and returning values from functions. Using the shell to interactively test functions. Using Python Tutor to trace and visualize the execution of function definitions (creation of function objects), function calls and function execution. Using local variables in function definitions. Introduction to Python's `tuple` type.

4. Simple programs (scripts). The `print` function. Functions that don't have `return` statements.

5. A step-by-step recipe for designing functions.

6. Using a simple unit-testing framework to automate testing.

7. Character strings (type `str`). Interactive programs (scripts): the `input` function.

8. The binary number system. Representation of integers in binary. Issues with representing real numbers in binary.

9. Boolean values (type `bool`). Boolean operators. Relational operators. Evaluating expressions that have numbers and boolean and relational operators. Comparing strings. Making decisions: (`if`, `if-else`, `if-elif-else` and `if-elif` statements).

10. Lists: Python's `list` type; creating `list` objects; `list` operators, built-in functions and methods. Using a `for` loop to iterate over a list. Using the `range` function to generate a sequence of list indices. Comparison of tuples and lists.

11. Repeating actions until a condition is satisfied (`while` statements).

Module *Data Management*

12. The binary number system. Representation of signed and unsigned integers in binary.

13. Floating point representation of real numbers. Limitations of real-number computation using floating-point arithmetic.

14. Reading and writing text files.

15. More container data types. Sets: Python's `set` type; creating `set` objects; `set` operators, built-in functions and methods. Dictionaries (maps): Python's `dict` type; creating `dict` objects; `dict` operators, built-in functions and methods.

16. More about testing, in the context of the term project.

17. Introduction to numerical methods. Algorithms for calculating square roots: exhaustive enumeration, bisection search, Newton-Raphson. As time permits, we will explore some other numerical algorithms; for example, simple linear regression (method of least squares).

August 29, 2019