Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2015
Course Outline

Instructor: Don Bailey, ME 4438, bailey@sce.carleton.ca. Office hours are posted on cuLearn.

Undergraduate Calendar Course Description
SYSC 1005 [0.5 Credit]
Introduction to Software Development
Software development as an engineering discipline, using a modern programming language. Language syntax. Algorithm design. Tracing and visualizing program execution. Testing and debugging. Program style, documentation, reliability. Lab projects are drawn from a variety of application domains; for example, digital image manipulation, computer games and robotics. Precludes additional credit for ECOR 1606, SYSC 1100, COMP 1005 and COMP 1405. Lectures two hours a week, tutorial one hour a week, laboratory three hours a week.

Course Aims
By the end of this course, students should:

- know the fundamental concepts of procedural programming, using Python as the programming technology.
- have gained practical experience using lightweight, modern software engineering practices to design and implement small-scale programs.
- have developed a "mental model" of computation; in other words, learned how to reason about the execution of code by drawing diagrams that depict program state.
- understand the use of software experiments as an aid to learning.

Learning Outcomes
By the end of this course, students should:

- 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.
- know the syntax and semantics of the fundamental control constructs provided by Python. These constructs include: sequential execution as determined by the ordering of executable statements; selection (`if`, `if-else` and `if-elif-else` statements); repetition (`while` and `for` statements); function calls.
- be able to trace the execution of short programs without using a computer and
    - describe what the program does;
    - draw diagrams that illustrate how code execution changes the program's state, as represented by the variables in the program's global frame and function activation

frames (function arguments and local variables).

- be able to design and implement functions that satisfy detailed specifications. Students should be able to use the Python shell to interactively test their functions. Students should be able to apply simple debugging techniques (e.g., inserting `print` statements to instrument code) to locate faults in their code.

- be able 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.

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

Prerequisite
None, but registration is normally restricted to students enrolled in the Software Engineering and Computer Systems Engineering programs.

Primary Textbook

- *Practical Programming (Second Edition): An Introduction to Computer Science Using Python 3*, Paul Gries, Jennifer Campbell, Jason Montojo, Pragmatic Bookshelf, 2013, ISBN-13: 978-1-93778-545-1.

  An eBook (PDF, epub and mobi formats) or paperback copy of this book can be purchased from the publisher's Web site: http://pragprog.com. If you only want a printed copy, it's probably less expensive to order it through the Web site of a large book retailer (e.g., amazon.ca).

Secondary Textbook

- *Think Python: How to Think Like a Computer Scientist*, Allen Downey, Green Tea Press, 2015 (Version 2.0.16).

  This book is available under a license that allows readers to freely copy and distribute the text. A PDF copy of the most recent version can be downloaded from the Green Tea Press Web site: http://greenteapress.com/thinkpython.

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

For help and support, go to http://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.

Intellectual Property
Classroom teaching and learning activities, including lectures, labs, discussions, presentations, etc., by both instructors and students, are copy protected and remain the intellectual property of their respective author(s). All course materials, including course outlines, lecture and lab materials, tests and exams, and other materials, are also protected by copyright and remain the intellectual property of their respective author(s).

Students registered in this course may take notes and make copies of course materials for their own educational use only. Students are not permitted to reproduce or distribute lecture notes and other course materials publicly for commercial or non-commercial purposes without express written consent from the copyright holder(s).

Web-based Resources

- *CodingBat* online programming exercises (visit http://codingbat.com.)

  This site provides numerous small-scale programming problems to help students develop fundamental programming skills (e.g., writing code that uses Boolean logic, loops, lists and strings). The problems are "live": you type your Python code in a box displayed in a Web browser window, then test your solution by clicking a button. You receive immediate feedback indicating which tests passed and which tests failed.

  This site is free, and you can use it without creating an account. (I recommend creating a CodingBat account - it's free, and the code you write while logged in will be saved between sessions.)

Software
All the software used in this course is free. Our lab computers run Windows versions of this software. If you want to install this software on your own computer, here's what you need to know:

- Python 3.4.3 can be downloaded from:

  python.org/downloads/release/python-343/

  For Windows platforms, a 32-bit Windows x86 MSI installer (file python-3.4.3.msi) and a 64-bit Windows x86-64 MSI installer (file python-3.4.3.amd64.msi) are available. For this course, it doesn't matter whether you install the 32-bit or 64-bit version.

- Pillow 2.9.0 can be downloaded from:

  http://pypi.python.org/pypi/Pillow/2.9.0

  Scroll down towards the bottom of the list of files to find the installers for Windows.

  Important: there are several different installers, each one corresponding to a different Python release (i.e., Python 2.6, 2.7, 3.2, 3.3 and 3.4). Please ensure that you download

the Pillow installer for Python 3.4. If you'll be installing the 32-bit version of Python, download Pillow-2.9.0.win32-py3.4.exe. If you'll be installing the 64-bit version of Python, download Pillow-2.9.0.win-amd64-py3.4.exe.

- Wing IDE 101 v. 5.1.7-1 can be downloaded from:

  http://wingware.com

  Please ensure that you download and install Wing IDE 101, not Wing IDE Professional or Wing IDE Personal. The latter two have a free 30-day trial license, after which a license must be purchased. Wing IDE 101 is free, and does not require you to purchase a license.

  On Windows platforms, one installer handles both 32-bit and 64-bit installations. The file name is: wingide-101-5.1.7-1.exe.

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

We are unable to provide support for students who prefer to use OS X or Linux. Please note there are known issues with OS X, Python's Tkinter module and the third-party Tcl/Tk GUI toolkit on which it depends (these issues are documented at python.org).

Attendance
Students are expected to attend all lectures and lab periods. The University requires students to have a conflict-free timetable. For more information, see the current Undergraduate Calendar, *Academic Regulations of the University*, Section 1.2, *Course Selection and Registration* and Section 1.5, *Deregistration*.

Requests to accommodate a missed midterm test, lab periods, due dates, etc., because of conflicts with other courses, 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: http://sce.carleton.ca/courses/health-and-safety.pdf.

Evaluation and Grading Scheme
Students will be evaluated primarily by means of a midterm test and a final exam. In addition, the marks assigned for lab work and online quizzes will contribute towards the final grade.

To pass the course, students must pass the final examination (50% or better). For students who pass the final exam, a numeric mark out of 100 will be calculated by weighting the course components as follows:

Lab work and online quizzes	15%
Midterm test	25%
Final exam	60%

This mark will be converted to your final letter grade, using the table of percentage equivalents shown in Section 2.3 of the *Academic Regulations of the University*.

Lab Periods
Attendance at the scheduled laboratory periods is mandatory, and attendance will be taken. During the labs you will work on short programming exercises that are intended to help you understand particular concepts that have been introduced in the lectures. You will normally be required to demonstrate your lab work and/or submit it to cuLearn by the end of the lab period (or other specified deadline), as indicated in that week's lab "handout".

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

- *Satisfactory* means that you were present at the lab and made reasonable progress towards completing the exercises. Note that you do not have to finish all the exercises to receive a *satisfactory* grade.

- *Marginal* means that you made some progress towards completing the exercises, but your solutions were not sufficiently complete to warrant a *satisfactory* grade. This grade indicates that you may be falling behind and should take steps to remedy this situation.

- *Unsatisfactory* means that you were absent from the lab period, or you attended but 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. You will also receive *unsatisfactory* if you do not demonstrate or submit your work before the deadline **or if it is apparent to the TA that you did not do enough of the lab work on your own; that is, you relied on your colleagues to explain the exercises and provide solutions (approach, algorithms or code)**.

For each s*atisfactory* or *marginal* grade, you will receive 2/2 towards the lab component of the course. For those labs that require you to submit your lab work via cuLearn, 0.5 marks will be deducted if you do not submit your work by the deadline or if the names of the files you submit are incorrect. Each *unsatisfactory* grade will receive 0/2.

If you are unable to attend a lab because of illness and would like to receive a mark for that lab, you must provide a medical certificate to your instructor before the start of your next lab period. In these cases, it is up to you to do the missed lab work on your own time, and you must make arrangements with your instructor to demonstrate your completed work in a timely manner.

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 can use the Systems and Computer Engineering undergraduate computer labs whenever the Mackenzie Building, Minto CASE and the Canal Building are open, except for those times when labs are reserved for specific courses.

Online Quizzes
Throughout the term, there will be short quizzes on cuLearn. You will receive 1/1 for each quiz you attempt, even if your solutions are entirely wrong. The only way you will receive 0/1 for a quiz is if you do not attempt it. In other words, the marks you receive for the quizzes are based on effort, not the correctness of your solutions. As such, there is nothing to be gained by copying solutions from a colleague instead of attempting the questions on your own: it makes absolutely no difference to your quiz score and completely defeats the purpose of the quizzes.

Exams
There will be one closed-book midterm test, which will be held approximately one-half of the way through the term. The date of the test will be announced in class and posted on cuLearn.

Computers will **not** be used during the midterm test.

Students who are unable to write the test because of illness or other circumstances beyond their control must provide in cases of illness a medical certificate dated no later than one working day after the test, or appropriate documents in other cases. Medical documents must specify the date of the onset of the illness, the (expected) date of recovery and the extent to which the student was/is incapacitated during the time of the test. If this information is provided to the instructor no later than five working days after the test, the weight of the final exam will be increased to cover the missed test; otherwise, the mark for the missed test will be 0.

Requests for accommodation because of poor performance on the midterm test will not be considered. There will be no “make-up” test. So, if you are ill on the day of the midterm test, don't write the test and later claim that your performance was impaired because you were unwell. You are better off to miss the test and request that the weight of your final exam be increased, by following the procedure outlined earlier.

A closed-book final exam will be held during the University's December examination period. The *Academic Regulations of the University* permit instructors to specify requirements that must be satisfied for students to be eligible to write the final examination or, where circumstances warrant, the deferred final examination.

- All students are eligible to write the final examination, regardless of the marks they received during the term.
- Students who miss the final exam, but earned at least 60% on the lab and quiz component and wrote the midterm test (or provided acceptable documentation to explain their absence from the test), will receive the grade ABS. These students will be deemed to have

performed satisfactorily during the term when their applications for a deferral of the final examination are considered. For more information, see the *Academic Regulations of the University*, Section 2.2, *The Course Outline*; Section 2.3, *Standing in Courses/Grading System*; and Section 2.5, *Deferred Final Examinations*.

- Students who miss the final exam but have not satisfied the conditions for receiving ABS, as listed above, will receive the grade FND. These students are ineligible to write the deferred final exam.

Computers will **not** be used during the final exam.

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 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.

Appeal of Grade
The processes for dealing with questions or concerns regarding grades assigned during the term and final grades is described in the *Academic Regulations of the University*, Section 2.7, *Informal Appeal of Grade* and Section 2.8, *Formal Appeal of Grade*.

Early Feedback
See Section 2.2.1 of the *Academic Regulations of the University*.

The weekly lab exercises will normally be graded during the lab period. Outside of the scheduled labs, you can obtain feedback during office hours or by making an appointment to see your instructor.

Academic Integrity
Students should be aware of their obligations with regards to academic integrity. Please review the information about academic integrity provided at the Office of Student Affairs Web site:

http://carleton.ca/studentaffairs/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 accommodation, the processes are as follows:

Pregnancy
Email me with any requests for academic accommodation during the first two weeks of term, or as soon as possible after the need for accommodation is known to exist. For more details, read the *Student guide to academic accommodation*, which is available from the Equity Services website:

http://carleton.ca/equity

Religious Obligations
Email me with any requests for academic accommodation during the first two weeks of term, or as soon as possible after the need for accommodation is known to exist, but in no case later than the second-last week of classes. For more details, read the *Student guide to academic accommodation*, which is available from the Equity Services website: http://carleton.ca/equity

Academic Accommodations for Students with Disabilities
The Paul Menton Centre for Students with Disabilities (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 me 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 me to ensure accommodation arrangements are made. Please consult the PMC website for the deadline to request accommodations for the formally-scheduled exam: http://carleton.ca/pmc/students/dates-and-deadlines

For more information, refer to the *Students* section of the Paul Menton Centre Web site:

http://carleton.ca/pmc/students

List of Topics

1. Introduction to Python: using the Python shell as a calculator. Fundamental types (`int`, `float` and `str`); operations supported by these types. Construction and evaluation of expressions. The assignment operator and binding values to variables.
2. Calling built-in functions from the shell. Importing functions from modules. Introduction to Python's `tuple` type. Image processing: importing and calling functions to load and display images.
3. Defining functions and modules. Passing arguments to functions, returning values from functions. Using the shell to interactively test functions.
4. Image processing: defining functions that manipulate images. Repetition (`for` and `while` statements), iterating over a sequence. Making decisions (`if` statements).
5. Interactive programs (scripts): keyboard input, printing to the console.
6. Lists: Python's `list` type; creating `list` objects; `list` operators, built-in functions and methods; developing functions that work with lists. Comparison of tuples and lists.
7. Reading strings and numbers from text files.
8. Sets: Python's `set` type; creating `set` objects; `set` operators, built-in functions and methods; developing functions that combine lists, tuples and sets.
9. Dictionaries: Python's `dict` type; creating `dict` objects; `dict` operators, built-in functions and methods.
10. If time permits, we will introduce the functional programming paradigm and some of the Python constructs that support functional programming.

Edited: August 31, 2015.