CSCI-400: Principles of Programming Languages

Spring 2018: Course Syllabus

Instructor: Jack Rosenthal (jrosenth@mines.edu)

Lecture: Tuesdays and Thursdays, 9:30 AM to 10:45 AM in BE 243

Office Hours: See http://inside.mines.edu/~jrosenth for a current schedule, or Email to schedule appointment.

TA: Ben Tarman (btarman@mines.edu)

Course Website: https://lambda.mines.edu

Prerequisites: Either CSCI-262 (Data Structures) or CSCI-306 (Software Engineering)

Important

This syllabus is not a legal contract, nor is it set in stone. Anything is subject to change.

```
#include <std/disclaimer.h>
from disclaimers import standard
```

Course Description & Learning Objectives

CSCI-400 explores the elements of structure and interpretation in computer programming languages. As a class, we will explore the Haskell programming language. Then, students will be given an opportunity to explore a language of their choice (Language Explore Project). In the second half of the course, we will explore the design philosophies of the Python programming language. Finally, students will design a language of their own and justify their design choices using the concepts presented in the course (Language Design Project).

By the end of this course, students should be able to:

1. **Classify** programming languages as either **procedural**, **functional**, **declarative**, or **concatenative** (or as none of the previously listed).
2. **Explain** language concepts such as **structured programming**, **lexical scoping**, **dynamic scoping**, **static typing**, **dynamic typing**, **weak typing**, **strong typing**, **duck typing**, and *demonstrate* understanding of the concepts by explaining them in a language of their own design.
3. **Create** a simple parser in Haskell, Python, or a language of the student's choosing.

Textbooks

- **Required**: LYH: Learn You a Haskell for Great Good (Available free online: http://learnyouahaskell.com/chapters)
- **Optional (supplementary)**: RWH: Real World Haskell (Available free online: http://book.realworldhaskell.org/read)
- **Optional (supplementary)**: TCA: The Coder's Apprentice: Learning Programming with Python 3 (Available free online: http://www.spronck.net/pythonbook)

Learning Groups

This class makes use of **Learning Groups** to facititate education. You will be randomly assigned a new learning group every few weeks, and your instructor will let you know your new group via Email. You are expected to:

1. Sit with your learning group during lecture.
2. Complete your part of learning group assignments, dividing problems amongst group members if requested by the assignment.
3. Share results from learning group assignment at the beginning of lecture with the rest of your group.
4. Arrange study sessions outside of class for quizzes and exams with your learning group.

Group Quiz Grade

Part of your grade will be derived from the average quiz scores from your learning group (missing, zero, or significantly outlying quiz scores will not be included in this average unless they are your own). You should make sure to study with your learning group members and help them understand *even if you think you know the material.* Teaching is the (hidden) highest level of Bloom's Taxonomy.

If you have issues with any of your group members, you should let your instructor know so that groups can be reassigned.

Autograded Assigments

Most programming assignments are automatically graded. The autograder scripts are designed to give all fully-correct programs 100%, and do the best job possible assigning partial credit where applicable. But under some cases, you may not receive as many points as you deserve. If you believe this to be the case, contact the course instructor so they can either update the autograder to handle your case better, or manually grade your assignment.

You are only allowed 4 submissions by default on autograded programming assignments. This is because your grade is received quickly after submitting, and it may be tempting to use the autograder as your only means of testing. Since students are expected to test their own code before submitting, this is to help encourage that. If you run out of submissions and need more, then Email your instructor. Your instructor may assign you more if you have a legitimate reason.

Warning

Autograded assigments are reviewed for plagarism a few weeks after the deadline, not when graded. While copying an assigment from another student may give you a good grade in the short term, the long term consequences could be severe. Plagarism is taken very seriously on these assignments.

Mailing List

A student mailing list has been created for this course. If you are registered for this course, your `@Mines.EDU` email should have already been subscribed. Use this mailing list for questions related to the course or its assignments: either the instructor or another student may be able to answer your question. **Do not use this mailing list for matters which need to remain private between you and the course instructor**, messages will be delivered to everyone enrolled in the course.

To send mail to the list, deliver your message to `csci-400-s18@lists.640k.net`.

List Guidelines

When using the list, please respect these guidelines:

1. Be polite, and respect others. This includes taking "back and forth" conversations to a separate Email thread rather than spamming everyone's INBOX with content relevant to a select few (in such a case, please make sure to CC the course instructor).
2. Do not send large snippets of code for a course assignment to the list, even if asking or answering a question. Note that some of the languages we are using in the course may allow you to express the answer very succinctly, so in some cases, even a single line may be considered a "large snippet of code". When in doubt if a snippet of code is too large, post pseudocode instead, or answer the question using a similar example (rather than for the assignment itself).
3. Use complete sentences, correct spelling, punctuation, and grammar, to the best extent you can.
4. Quote only relevant parts of the message, and place your reply **after** the quotation. Note that the Gmail web client does this incorrectly by default, and you will need to format your message properly.

   ### Example

   For example, **do not** send a reply like this to the list:

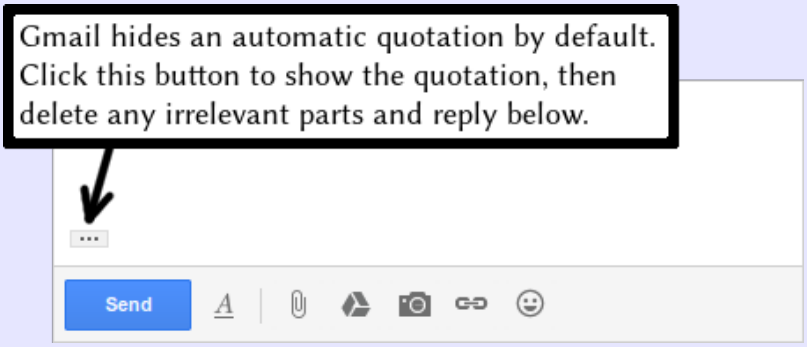
   ```
   You will need to download and install a newer version of ...

   On 2017-12-24 at 08:30, Blaster the Burro wrote:
   > Hi Friends,
   >
   > My telnet client is broken! I am using ...
   >
   > Thanks,
   >
   > Blaster
   ```

   What was wrong with that message? Two things: the reply came before the quotation, and more than the relevant part of the message was quoted (`Hi Friends`, `Thanks`, etc.).
   This message might be correctly formatted as:

   ```
   On 2017-12-24 at 08:30, Blaster the Burro wrote:
   > My telnet client is broken! I am using ...

   You will need to download and install a newer version of ...
   ```

   In the Gmail web application, you will need to click the button shown above to edit the quote and reply properly.
5. Do not send HTML-only mail to the list. Your message must at least include a plain text version.

Grading Policy

Your course grade will be divided into 8 major parts:

Programming Assignments	16%
Language Explore Project	11%
Exam 1	17%
Exam 2	17%
Language Design Project	24%
Quizzes (Individual Grade)	3%
Quizzes (Learning Group Average)	7%
Learning Group Participation	5%

Your grade letter will be derived using the standard plus-minus grading scale. In other words, if your grade in percents is X, then your grade letter will be:

Grade	**If X is...**
A+[1]	$X \geqslant 96.5$
A	$93.5 \leqslant X < 96.5$
A-	$90.0 \leqslant X < 93.5$
B+	$86.5 \leqslant X < 90.0$
B	$83.5 \leqslant X < 86.5$
B-	$80.0 \leqslant X < 83.5$
C+	$76.5 \leqslant X < 80.0$
C	$73.5 \leqslant X < 76.5$
C-	$70.0 \leqslant X < 73.5$
D+	$66.5 \leqslant X < 70.0$
D	$63.5 \leqslant X < 66.5$
D-	$60.0 \leqslant X < 63.5$
F	$X < 60$

Warning

If your **individual quiz grade average** is less than 60%, your **final grade letter will be reduced by one whole letter**. For example, if you earned an average of 40% on the quizzes, and you would normally earn a C+ in the course, you will actually get a D+.

Late Work Policy

This course uses a **slip day policy**. You will be given 4 slip days at the beginning of the semester, but there may be opportunities to earn more slip days later.

For each 24 hours you turn in an assignment late, it will cost you 1 slip day to turn in an assignment. You will not get any points off, you just need enough slip days to turn it in. For example, if you have 3 slip days left, and you turn in an assignment 4 hours late, you will have 2 slip days left after turning in the assignment.

Slip days can only be used for programming assignments, the language explore project, and the language design project. You cannot use slip days for exams, quizzes, or learning group activities.

[1] For some reason, I cannot put this in Trailhead. But you will still get the bragging rights.

Regardless of how many slip days you have, the following rules apply:

1. No more than 5 slip days may be used on a single assignment without the instructor's permission.
2. You must turn in all work by 8 AM on Dead Day, even if you still have slip days left.

You can check in on your current slip day balance on the course website. There is no benefit to keeping slip days at the end of the semester; they will not translate to extra credit.

Note

If you need an extension for good reasons (other than for just being busy, slip days are intended to cover that) then Email your instructor. You can still get traditional extensions.

Regrade Policy

If you feel you deserve more points on an assignment or a problem than you got, then send an Email to your instructor. There may be an opportunity to get points back.

Collaboration Policy for Programming Projects in CS Courses

The following policy exists for all courses in the CS department. This policy is a minimum standard; your instructor may decide to augment this policy.

1. If the project is an individual effort project, you are not allowed to give code you have developed to another student or use code provided by another student. If the project is a group project, you are only allowed to share code with your group members.
2. You are encouraged to discuss programming projects with other students in the class, as long as the following rules are followed:

    a) You view another student's code only for the purpose of offering or receiving debugging assistance. Students can only give advice on what problems to look for; they cannot debug your code for you. **All changes to your code must be made by you**.
    b) Your discussion is subject to the **empty hands policy**, which means you leave the discussion without any record (electronic, mechanical, or otherwise) of the discussion.

3. Any material from any outside source such as books, projects, and in particular, from the Web, should be properly referenced and should only be used if specifically allowed for the assignment.
4. To prevent unintended sharing, any code stored in a hosted repository (e.g., on GitHub) must be private. For group projects, your team members may, of course, be collaborators.
5. If you are aware of students violating this policy, you are encouraged to inform the professor of the course. Violating this policy will be treated as an academic misconduct for all students involved. See the Student Handbook for details on academic dishonesty.

Disability Support

The Colorado School of Mines is committed to ensuring the full participation of all students in its programs, including students with disabilities. If you are registered with Disability Support Services (DSS) and your instructor has received your letter of accommodations, please contact your instructor at your earliest convenience so you can discuss your needs in this course. For questions or other inquiries regarding disabilities, we encourage you to visit Disability Support Services (DSS) for more information.