CS234

MODULE 1 - INTRODUCTION

• Course Staff
• Course Contents
• Course Structure
• Misc. Knowledge
Welcome to CS234!

Four Questions you (might) have:

• Who’s this guy?
• Who are we?
• What is this course?
  • What do we learn?
  • How do we learn it?
• Why, though?
Who’s this guy?

• Dr. Daniel James Holtby
  • Dan for short
  • email: djholtby@uwaterloo.ca
  • Office: DC2130
  • Office Hours: 12:30P – 2:00P, Tues. and Thur.

• Dan: Tell them about yourself
Course Staff

• IA
  • Xiang Fang
  • Deals with Marks, Remarks, Piazza, and holds Office Hours
  • Email: x2fang@uwaterloo.ca

• ISC
  • Travis Bartlett
  • Email: travis.bartlett@uwaterloo.ca
  • Deals with accommodations (e.g. illness), cheating cases, and other course administration details
Who are we?

Don’t you know?

Here’s what I think you are:

• Interested in Computer Science
  • (Maybe even in minoring in it?)

• Not a CS Major

• Someone who has passed CS116 or CS136
What is this course all about?

**Algorithms and Data Structures!**

You are going to

- Learn about the basic data structures
  - This is (mostly) language independent
- Gain experience writing programs that are **efficient**
- Analyze algorithms to work out their efficiency
- Design all new algorithms and data structures
Awww man, efficiency?

In previous courses your definition of “a good program” might have been “gives the right answer (mostly)".

It’s true, a program should be **correct**

It should also be:

- Maintainable (easy to update and modify)
- Simple (easy to write and/or easy to understand)
- Reusable (easy to reuse modules in other programs)
- **Efficient** (makes good use of resources, e.g. time and/or space)
Why are we learning this?

“The difference between a bad programmer and a good one is whether [the programmer] considers code or data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships.”

-Linus Torvalds
How will we learn all that?

• Lectures
  • Slides (PDF files will be online a few days ahead of time)
  • Sometimes I’ll entertain you by coding and/or running pre-written code

• Assignments

• There are no labs in this course!
  • There will be a Python review session though
Assignments

• There are 4 assignments, each due on Friday 4pm
  • No late assignments will be accepted
  • For verified excuses weight will be shifted
• A mix of written and programming questions
  • Programming is in Python 3 (make sure you’re using 3)
  • Suggested IDE: WingIDE 101
• Must be written individually (no collaboration is permitted)
Assignment Tips

• Start Early!
• Try to solve a simpler version of the problem, then think about how to extend it
• Try to break the problem into sub-problems
• I’m here to help
  • But not to give you the solution
• Assignments are there to help you learn
Grades

- **Assignments – 30%**
  - That works out to 7.5% each
- **Midterm Exam – 25%**
  - Tuesday, June 19th, 4:30P-6:20P
- **Final Exam – 45%**
  - Date TBD

Note that you must pass the weighted exam average to pass the course
Appeals

- Request a remark within 2 (two) weeks of the marks being released
- Look at the solutions first!
  - The problem might be with your understanding, not with the marking

Procedure:
1. Email the IA (not me) and explain why you think the mark is incorrect
2. Be prepared for your mark to go up or down (we will remark the entire assignment)
3. If you do not think the remark is fair, then (and only then) loop me in on the issue.
Textbook

Data Structures and Algorithms using Python

- Optional
Piazza

• Do not discuss solutions (even partial)
• Search before you ask
• Good title:
  • A1Q1 What does <terminology> mean?
• Bad titles:
  • A1Q1
  • What does this mean?
Cheating

The University values: honesty, trust, fairness, respect, and responsibility

How to avoid an academic integrity violation:

• Do your own work!
• Do not try to find solutions online (unless permission has specifically been granted)
• It’s OK to search online for what an error message means
• It’s not OK to get online help fixing your solution
Pseudo-Code

Many examples will be given in pseudocode rather than in Python (or any other language). You will also be asked to write your own.

```plaintext
function sum(list)
    count ← 0
    for every v in list
        count ← count + v
    return count
```
Isn’t that Python?

Pseudo-code is whatever you want it to be. It’s not for a computer, it’s for a person. As long as the reader understands, it’s all good!

```python
function sum(list)
    count ← sum of items in list
    return count
```

```latex
\text{function } \text{sum}(\text{list}) \\
\hspace{1cm} \text{return } \sum_{v \in \text{list}} v
```