CSCA08 Fall 2017

Week 1 - Welcome

Brian Harrington & Marzieh Ahmadzadeh

University of Toronto Scarborough

September 4-8, 2017

UNIVERSITY OF TORONTO SCARBOROUGH

Welcome to CSCA08

- This course is an introduction to computer science using Python programming
  - NOT the other way around
- Designed for people with no programming experience
- Designed as an introduction for people intending on taking further CS courses

Who are we?

- Brian Harrington
  - brian.harrington@utsc.utoronto.ca
  - BSc. University of Toronto Scarborough
  - MSc & D.Phil University of Oxford
  - Teaching at UTSC since 2012
  - Office: IC342
  - Office Hours: Tues & Thurs 2-3pm
- Marzieh Ahmadzadeh
  - marzieh.ahmadzadeh@utoronto.ca
  - Bsc is in Software Engineering from Isfahan University
  - MSc and PhD in Computer Science from Nottingham University, UK
  - Worked as a software engineer for 4 years in both UK and Iran
  - 11 years of teaching experience in UK, Iran and Canada
  - Office: IC468
  - Office Hours: Tuesday 4-5 & Friday 2-3

CSCA08 vs CSCA20

- CSCA08
  - Prelude to CSCA48
  - Practical + Theoretical work
  - Introduction to CS concepts
  - Some CMS programs and one Management program require CSCA08
- CSCA20
  - Prelude to CSCB20 (or taken as a terminal course)
  - Intended for those not planning on doing A48 and beyond
  - Focused on practical implementation
  - Most degrees allow CSCA20 or CSCA08 (check latests version of reqs)

Course Website

- http://www.utsc.utoronto.ca/~bharrington/csca08/
- Most of the info from this lecture can be found on the course info sheet
- Assignments/Exercises will be posted here
- You should also check regularly for announcements
- Links to discussion forums, MarkUs, necessary software, etc.

Discussion Forums

- Your first point of contact
- Running a Piazza forum
- Link on the website, need utoronto e-mail address
- Questions will be answered here much faster than via e-mail
  - E-mails that should be forum posts will be deferred to the forum
  - Repeat questions will be removed
  - There's only 2 of us, and 800+ of you. Crowdsource your learning
  - Be sensible/respectful

Lectures

- 2 hour lecture (traditional), 1 hour lecture (inverted)
- Lecture Slides & Code will be posted online
  - So you don't have to copy frantically
- 3 REASONS YOU MUST STILL COME TO LECTURE
  1. Important announcements will be made
  2. Lecture slides do not contain all the info we cover in class
  3.

Tutorials

- Held every week (starting next week)
- TAs will be presenting new material and reviewing material from lecture
- No attendance taken, but material presented is fair game for exam
- Will also have quizzes
- Meant to be a more interactive version of lectures

Practical Sessions

- Held in BV computer labs (see website for exact times/rooms)
- Drop-in sessions (as little or as much as you want/feel you need)
- Practice Questions, Challenge Questions, Fun Questions
- Tracademic

Course Structure

Work	Weight	Comment
Term Work	15%	Weekly Exercies & Tutorial Work
Assignments	20%	3 Assignments
Term Tests	25%	2 tests, dates TBA
Final exam	40%	You must get 40% or above on the exam to pass the course

Exercises

- Due most weeks of the course
- Give you practice with basic concepts covered in lecture
- TAs will help you during tutorials/practical sessions
- Meant to make sure you're grasping the concepts

Quizzes

- Held in tutorial
- Marked on a scale of 0-1
  - 0 = didn't show/didn't try
  - 0.5 = made an effort/understood core concept(s)
  - 1 = got it

Assignments

- Larger pieces of work
- Must be completed entirely on your own
- TAs will give some help, but don't expect them to do any debugging
- Much more challenging than exercises
- Start early
- Must be completed ENTIRELY on your own

Exams

- 2 term tests, one final
- No I don't know the dates yet
  - I will post info online as soon as I know
- Closed book, written tests
- Get used to working without the Python interpreter as a safety net

Textbook(s)

- No course textbook
- We will be using online resources instead
- Why?
  - Save money
  - More flexibility
  - Better resources
- You will still have weekly readings, just not a physical textbook
- If you really want a textbook, let me know and I can recommend some

How to Succeed in CSCA08 (and University in General)

- Be a student
- Get involved
- Keep on top of your work
- Get help when you need it

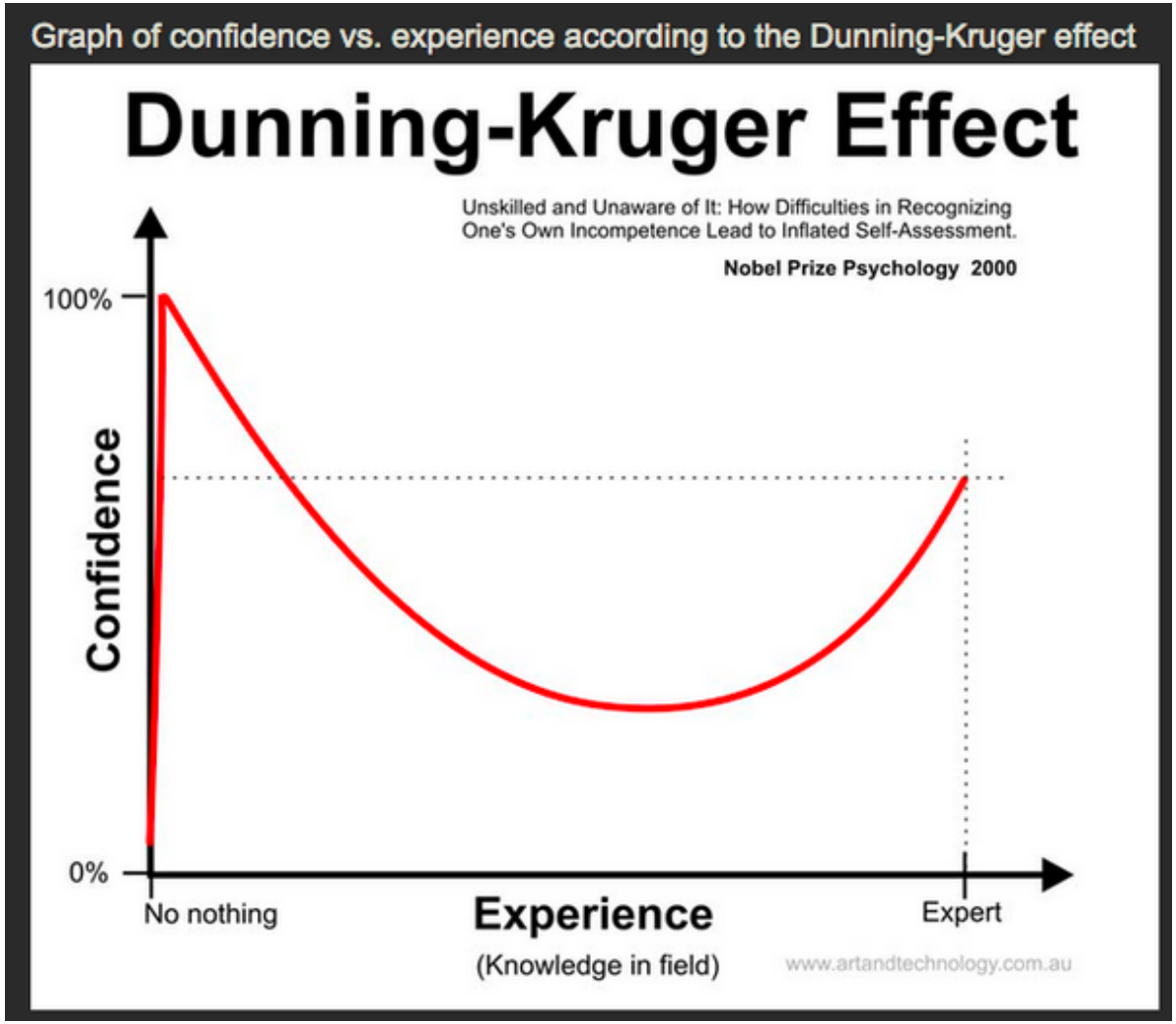
Getting Help

- Resources: in the order you should try them:
  - Website forums
  - Tutorials
  - Practicals
  - Office Hours

Remember

- If you're not enjoying it. You're doing it wrong!

Break


Graph of confidence vs. experience according to the Dunning-Kruger effect
Dunning-Kruger Effect
Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessment.
Nobel Prize Psychology 2000
100%
0%
Confidence
No nothing
Experience
(Knowledge in field)
Expert
www.artandtechnology.com.au

What is Computer Science?

- Easier to define what it is NOT
  - Solitary
  - Nerd Culture
  - ‘hacking’
  - Programming (not mainly anyway)

What is(n't) Computer Science?

- Not a Science
- Not (necessarily) about Computers
- All about **Algorithms**

What is an Algorithm?

- A set of instructions
- Broken up into simple to follow steps
- Followed step-by-step

Algorithms

- Cooking recipies
- Driving directions
- Assembly instructions

Computers are...

- STUPID!
- They can follow very simple directions very fast
- But they can't do anything you don't explicitly tell them to

Granularity

- "Walk to my office."
- "Exit this building. Cross Ellesmere. Enter the Instructional Centre..."
- "Walk to the door. Open the door. Walk through the door..."
- "Move your right foot 10cm forwards..."
- ...

Granularity

- Low granularity (e.g., English)
  - Easy to write
  - Lots of ambiguity
- High granularity (e.g., Punch cards)
  - Easy for the computer
  - More difficult to write
- Computer Science can be thought of as the process of turning low granularity ideas into high granularity algorithms
- One of the mantras of this course: *"Break it down"*. Break a complex problem down into 2 or more smaller problems, which are easier to solve.
  - If you break it down far enough, even a dumb computer can solve it!

Algorithms

- Computers can follow **simple** instructions very quickly
- Programming language = A set of instructions that the computer can understand
- Code = An algorithm written as a set of instructions in a programming language