Introduction To CPSC 217

- Procedural programming
- The python language
- Problem solving

Contact Information (James Tam)

- How to reach me
  - Email: tam@ucalgary.ca
  - Make sure you **specify the course name and number** in the subject line of the email 'CPSC 217' (otherwise I might miss it).
- Office time
  - Tuesday and Thursday from (3:50 – 4:30 PM) https://ucalgary.zoom.us/u/abyM8lbYG1 (passcode = 'hope')
  - Other help resources are available ('Help tutorial/CT' to be described later).

CPSC 217: A Student Question

- This course is targeted towards non-majors.
- Why am here?
- Why do I need 217 for my degree (this specific course or 217 as a choice among many).
- Degree requirements are agreed upon the faculty of individual departments (the “Profs”).
  - It’s based on their knowledge of their specific area.
  - The courses should include things they feel will prepare students for success.
  - Outsiders aren’t privy to this process (e.g. Tam has no idea what courses would be useful for students in actuarial sciences).
- **If you want to know why CPSC 217 is needed to for your program (e.g. how it will help you in your future) then contact an advisor in your department.**

James Tam

Course Resources

- Required resources:
  - D2L site (You **must** get the notes from here before lecture)
    - Course website can be reached via D2L (`Content->Course Information`) or https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/
      - The website includes all resources (not all of which may be individually linked to in D2L(and may display better than viewing them from within D2L.
      - Recommended but not required textbook:

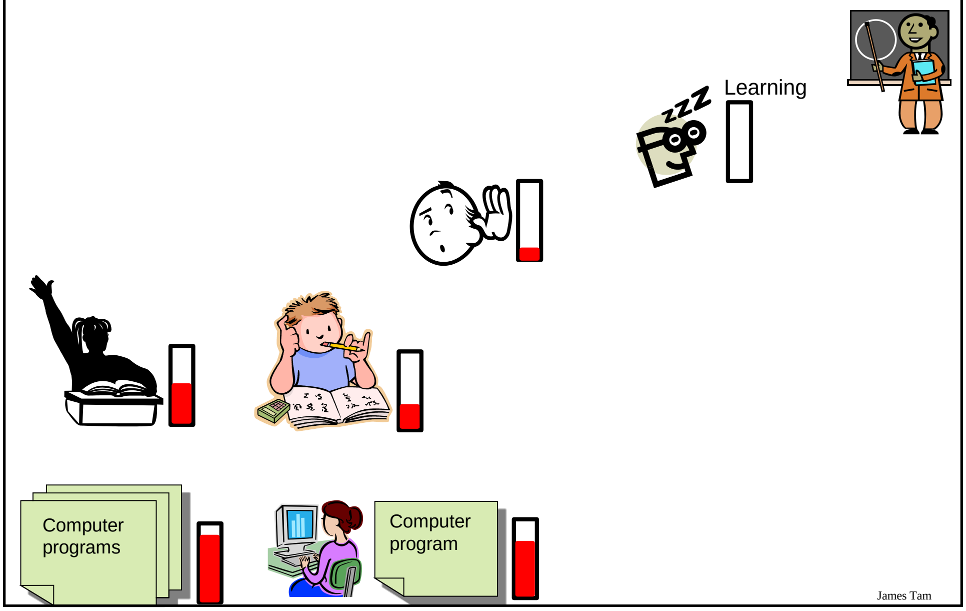
        "*Starting Out with Python*"(Gaddis T.) Addison-Wesley.

        5th edition directly via the publisher (Pearson).

        4th or 3rd edition can be purchased in the “previously enjoyed” version
  - Alternatively you can access any book licensed by the university (‘for free”) on the library web site:
    - (One of many books available) “*Visual QuickStart guide*” http://proquest.safaribooksonline.com.ezproxy.lib.ucalgary.ca/

James Tam

Your Engagement Level $\xrightarrow{+}$ Your Learning



James Tam

Lecture: How To Use The Course Resources

- They are provided to support and supplement this class.
  - The notes outline the topics to be covered
  - *At a minimum* look through the notes to see the important topics.
  - However the notes are just an outline and just looking at them without coming to class isn't sufficient to do well
  - You will get additional details (e.g., explanations) during lecture time
    - Take notes!
    - If you miss a lecture then get a copy of the in-class notes from another student (who takes detailed notes)

James Tam

Lecture Content

- You need to attend class (even if it’s virtual)
  - Make sure you review the video if you miss class
- New programming concepts
  - E.g. repetition, functional decomposition…
- Filtering concepts (i.e. you are told what to focus on): programming languages are extensive, even professional software developers aren’t experts in all parts of a language.
  - You will be informed as to what concepts are important for this course
  - What parts of the language are important.
  - How to do things with language e.g. how to use a random number function as computer simulation.
- While “Googling it” can provide links to example programs Google won’t show you all the answers e.g. what’s important for this course.
- Beyond this lecture will provide some opportunities to develop your skills e.g. we’ll go over practice problems

Tutorials

- There’s two types: teaching and help tutorials.
- **Both of them will be in person**.
- Teaching tutorials (you registered in a specific section when you signed up for the course).
  - Similar to lecture, again you should be attending class and catching up if you miss a class.
    - Reinforce concepts but the audience size is smaller (easier to ask questions about content) and hearing an alternative explanation for complex concepts can be beneficial.
    - The teaching assistants will also go over assignments.
    - Some additional hands on work (exercises) will allow you to develop your skills.

Tutorials (2)

- Help tutorials (CT/Continuous tutorial)
  - Attendance is optional.
  - It’s your opportunity to ask questions.
  - Located in the computer lab (near MS151 at a desk labelled as the "217 CT" or something similar).
  - Staffed by the teaching assistants (who teach the tutorials):
    - Aminbeidokhti, Amir
    - Farajian, Samin
    - Al Marouf, Ahmed

James Tam

Tutorial: Days/Times & Instructor Information

Tutorial information from the course website:

Tutorial	Location	Instructor name	Contact information
T01: MW 09:00-10:50	MS 176	Aminbeidokhti, Amir (a MAC user)	amir.aminbeidokhti@ucalgary.ca
T02: MW 17:00-18:50	MS 160	Farajian, Samin (a MAC user)	samin.farajian@ucalgary.ca
T03: MW 15:00-16:50	MS 160	Farajian, Samin (a MAC user)	samin.farajian@ucalgary.ca
T04: TR 16:00-17:50	MS 160	Al Marouf, Ahmed (a MAC user)	ahmedal.marouf@ucalgary.ca
T05: MW 11:00-12:50	MS 176	Aminbeidokhti, Amir (a Windows user)	amir.aminbeidokhti@ucalgary.ca

James Tam

Evaluation Components

- Assignments:
  - 4 full assignments: 82% total
  - 6 mini assignments (3% each) = 18% total
- Information about all assignments will be made available here:
  - https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/#Main_grid:_course_schedule_for_the_lecture,_lecture_notes,_assignment_information
  - Links to individual assignments can be found in D2L under: `Content->Assignments`

Student Questions?

- Q: How do I know how I did on an assignment? Can we get the grading for previous assignments before the next assignment is due?

- A: The two questions are actually inter-related.
  - With all assignments you will be graded according to the number of program functions that are correctly implemented ("program functionality").
    - The required program functions along with grading weights will be specified beforehand.
    - If you test your program thoroughly before submitting the final version then you should get a pretty clear idea of "how you did".
  - The larger assignments will include style and documentation requirements.
    - Again these requirements will be listed in the assignment description.
    - Details will be provided throughout the semester but these requirements are first introduced in the "Introduction to computer programming" components.
  - So **even though the marking for an earlier assignment may not be available prior to the due date of the next assignment** you should have the ability to know "how you did".

Implementing Assignments

- You will create a working and executable computer program.
- Use a text editor (similar to a word processor minus the fancy formatting capabilities) to create it and you will electronically submit the text file (to D2L) for marking.
- The official editor that we will support for this class was developed by the same people who developed the python language: IDLE (with the Windows installation for python automatically comes with download and install).
- Although you may be given some time in tutorial to work on your assignments (during the "open tutorial") mostly you will complete your work on your own time.
  - Don't underestimate the time/effort required.
  - Creating a good working program is harder than it may first appear.
- But on the flip side previous students have "gotten through the course" with no prior programming experience if they approached the course properly.

The Programming Language For Assignments

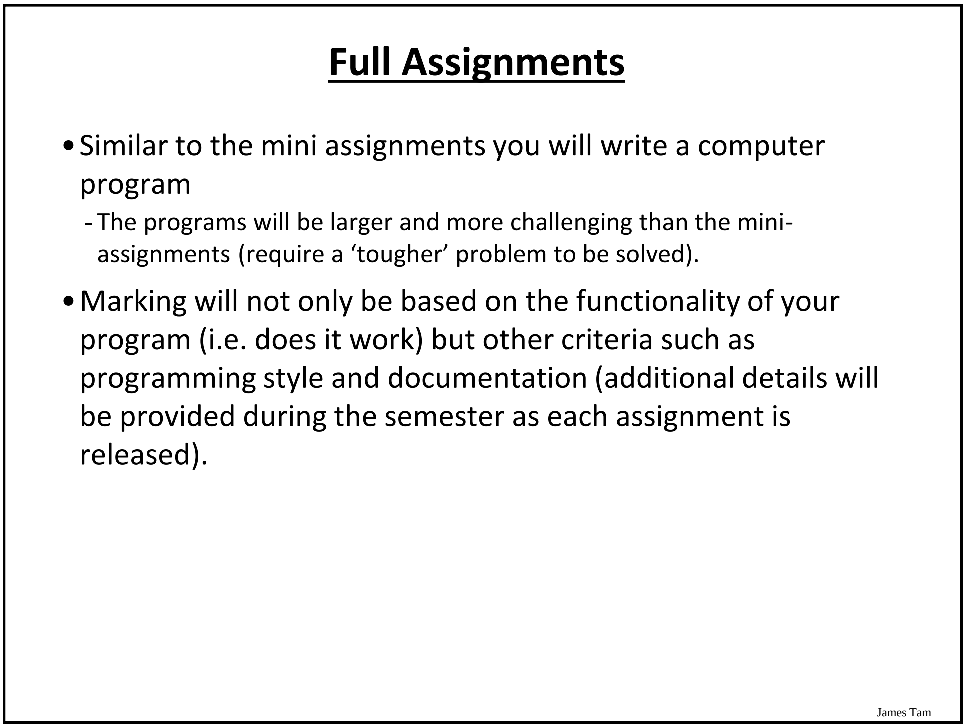
- All evaluation components must be completed by writing a program using Python version 3.x (not version 2.x) runnable in the CPSC labs. (MS 160 or 176 where your tutorials are scheduled).
  - **If your program doesn't work under these conditions then it will not be marked.**
  - The labs are accessible 24/7 (with card access).
  - If you have access problems then contact campus security:
    - https://www.ucalgary.ca/pubs/calendar/current/student-campus-services-campus-security.html
- Installing python isn't mandatory but here's some information
  - https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/notes/pdf/installing_accessing_python.pdf
  - Note: A4 will include a short writing component (D2L quiz) related to the content of the lectures (more details provided later).

Collaboration

- Each student must work on his/her own assignment (no group work is allowed for this class)
- Each student must individually submit an assignment
- Students must not see each other’s assignment code
  - Also students should not see the code from other students if that code will be graded for this course (e.g. code for examinations).
- Additional details will be provided later during the semester

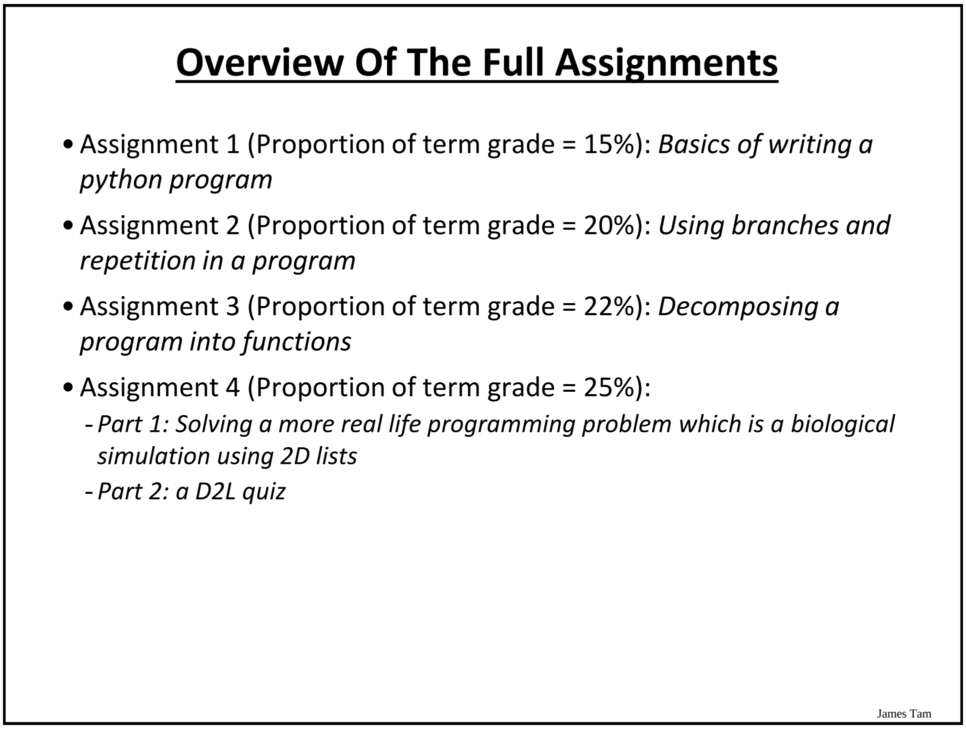
Mini Assignments

- As the name implies mini-assignments are a smaller version of the (full) assignments.
- The purpose is to allow you to apply the technical concepts (e.g., branches, functions, loops etc.) without the real-world complexities of a full program.
- Marking will focus on ‘functionality’: getting the program to work.
  - However some mini-assignments may have specific design requirements so make sure you are familiar with each assignment’s requirements.
- Although you shouldn’t ignore other things such as style and documentation these things won’t be graded for the mini-assignments



Full Assignments

- Similar to the mini assignments you will write a computer program
  - The programs will be larger and more challenging than the mini-assignments (require a ‘tougher’ problem to be solved).
- Marking will not only be based on the functionality of your program (i.e. does it work) but other criteria such as programming style and documentation (additional details will be provided during the semester as each assignment is released).



Overview Of The Full Assignments

- Assignment 1 (Proportion of term grade = 15%): *Basics of writing a python program*
- Assignment 2 (Proportion of term grade = 20%): *Using branches and repetition in a program*
- Assignment 3 (Proportion of term grade = 22%): *Decomposing a program into functions*
- Assignment 4 (Proportion of term grade = 25%):
  - *Part 1: Solving a more real life programming problem which is a biological simulation using 2D lists*
  - *Part 2: a D2L quiz*

Assignment Grading

- Assignments will be marked by the tutorial instructor who teaches your tutorial section.
- **After the deadline**: If have questions or concerns about grading **the tutorial instructor** (in the tutorial that you are registered) **is the first person to contact**.
  - For your convenience the link to tutorial information has been provided again: https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/#Tutorial_information_(teaching_and_help_tutorials)
  - Your tutorial section was the one you were required to select when you registered.
  - If you don't know how to find that information then you can find it the "Student Center" in PeopleSoft:
    - https://cas.ucalgary.ca/cas/login?service=https://portal.my.ucalgary.ca/psp/paprd/?cmd=start&ca.ucalgary.authent.ucid=true
- **Before the deadline**: extensions require the approval of the course instructor (more on this later)

Submitting Assignments

- Submit your work to the appropriate D2L Dropbox link.
  - http://d2l.ucalgary.ca/
  - Find the appropriate course name/number for the semester.
  - Within the D2L Dropbox setup for the course there's a separate link for each assignment.

Submitting Assignments (2)

- **Bottom line: it is each student's responsibility to make sure that the correct version of the program was submitted on time.**
  - Alternate submission mechanisms e.g., email, uploads to cloud-based systems such as Google drive, time-stamps, TA memories cannot be used as alternatives if you have not properly submitted into D2L
  - **Only files submitted into D2L by the due date is what will be marked**
- Late assignments:
  - Full assignments: they may be submitted for a progressive per day penalty (see the assignment description).
  - Mini-assignments: a most mini-assignments can be submitted one day late (with a penalty).
  - If you are ill then a sworn declaration is required.
  - Contact your **course instructor** and not your tutorial instructor to get permission for a late submission

JT's Hint: Electronically Submitting Work

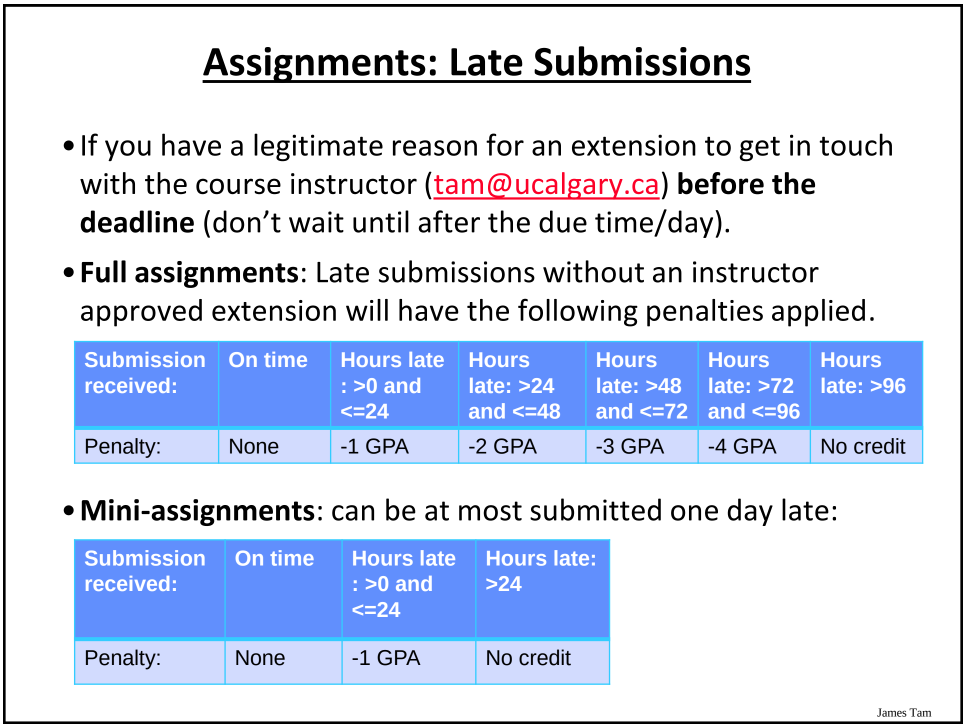
- Bad things sometimes happen!
  - A virus, hardware failure, you screwed up the submission.
- Rules of thumb for assignment submissions:
  - Do it early! (Get familiar with the system)
  - Do it often! (If somehow real disaster strikes and you lose everything at least you will have a partially completed version that your TA can mark).
  - *Check your work*.
    - Don't assume that everything was submitted OK.
    - Don't just check file names but at least take a look at the actual file contents (not only to check that the file wasn't corrupted but also that you submitted the correct version).
    - Assignment 0 'A0':
      - An exercise in tutorial where you practice submitting and checking your work
      - Not directly graded but still important to complete

How To Verify Submissions In DropBox

- There is a help link provided with each assignment description.
- When Teaching Assistants go over Assignment zero you will have the opportunity to work through the assignment.
- Assignment zero:
  - Not graded but important practice
  - Learning objectives:
    - How to submit assignments using D2L (most know this).
    - **How to properly check if a submission "went through"** (main learning objective as most students do not do this).
- Resource file for checking submissions:
  - http://pages.cpsc.ucalgary.ca/~tamj/resources/Verifying_D2L_Submissions.pdf

Backing Up And Submitting Your Work

- Bottom line: **it is up to you** to make sure things are done correctly and on time.
- If you have questions beforehand then do ask (make sure you ask your questions early enough so you can receive an answer before the due time).
- But don't wait until after the due date (it's too late).
  - If your work isn't in D2L before the final due date then **you will be awarded no credit**.



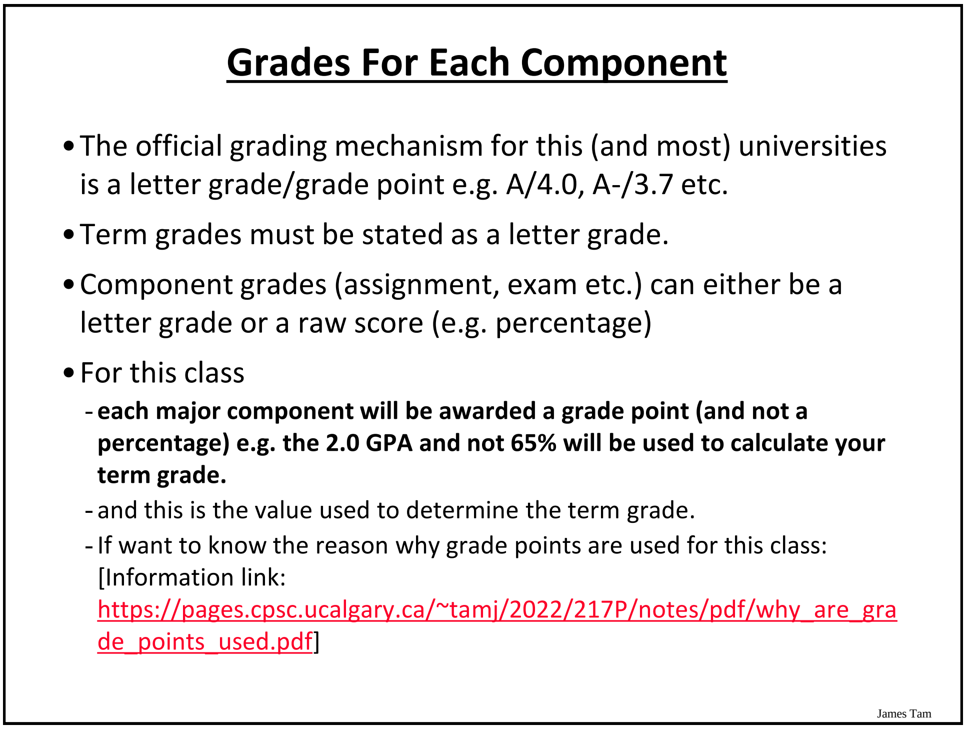
Assignments: Late Submissions

- If you have a legitimate reason for an extension to get in touch with the course instructor (tam@ucalgary.ca) **before the deadline** (don't wait until after the due time/day).
- **Full assignments**: Late submissions without an instructor approved extension will have the following penalties applied.

Submission received:	On time	Hours late : >0 and <=24	Hours late: >24 and <=48	Hours late: >48 and <=72	Hours late: >72 and <=96	Hours late: >96
Penalty:	None	-1 GPA	-2 GPA	-3 GPA	-4 GPA	No credit

- **Mini-assignments**: can be at most submitted one day late:

Submission received:	On time	Hours late : >0 and <=24	Hours late: >24
Penalty:	None	-1 GPA	No credit

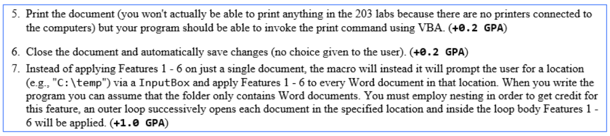


Grades For Each Component

- The official grading mechanism for this (and most) universities is a letter grade/grade point e.g. A/4.0, A-/3.7 etc.
- Term grades must be stated as a letter grade.
- Component grades (assignment, exam etc.) can either be a letter grade or a raw score (e.g. percentage)
- For this class
  - **each major component will be awarded a grade point (and not a percentage) e.g. the 2.0 GPA and not 65% will be used to calculate your term grade.**
  - and this is the value used to determine the term grade.
  - If want to know the reason why grade points are used for this class: [Information link: https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/notes/pdf/why_are_grade_points_used.pdf]

Grading: Course Components

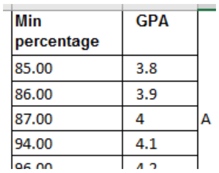
- Each course component will have a grading key
- Sample from a past assignment (different course)



5. Print the document (you won't actually be able to print anything in the 203 labs because there are no printers connected to the computers) but your program should be able to invoke the print command using VBA. (**+0.2 GPA**)
6. Close the document and automatically save changes (no choice given to the user). (**+0.2 GPA**)
7. Instead of applying Features 1 - 6 on just a single document, the macro will instead it will prompt the user for a location (e.g., "`C:\temp`") via a `InputBox` and apply Features 1 - 6 to every Word document in that location. When you write the program you can assume that the folder only contains Word documents. You must employ nesting in order to get credit for this feature, an outer loop successively opens each document in the specified location and inside the loop body Features 1 - 6 will be applied. (**+1.0 GPA**)

  - A student who completes only Features #5 – 7 will be awarded an assignment grade point of 0.2 + 0.2 + 1.0 = 1.4 (just over a D+ grade)

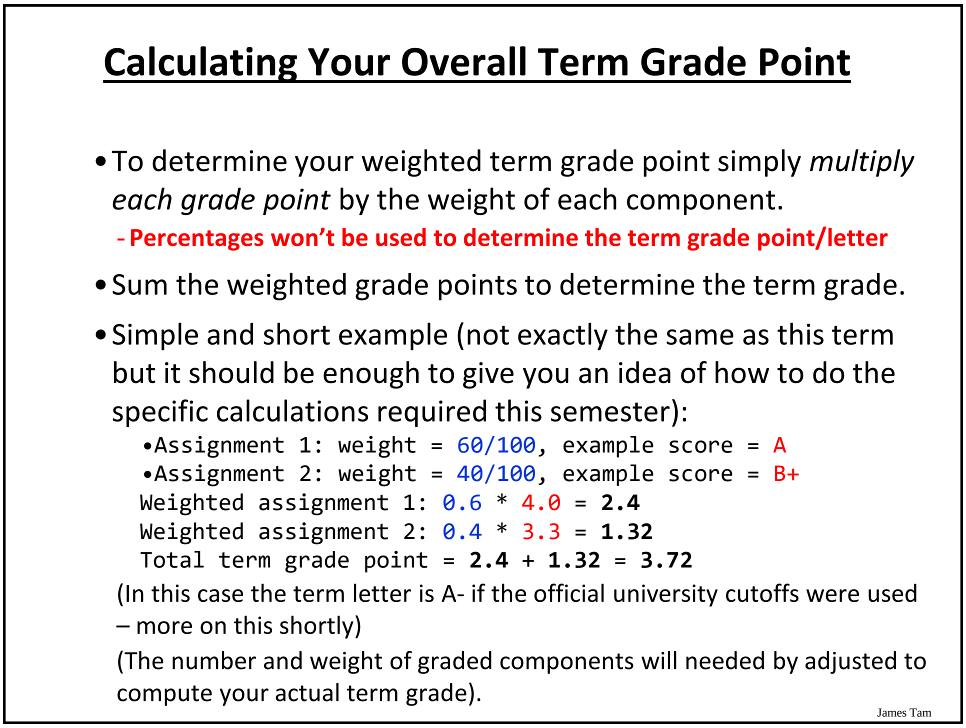
- Examinations will include a lookup table (raw percentage score to grade point). Sample from another course.



Min percentage	**GPA**	
85.00	3.8	
86.00	3.9	
87.00	4	A
94.00	4.1	
96.00	4.2	

Grade Points Are Letter Grades Not Percentages

- For examinations the mapping between a raw score and a grade point occurs one way (raw score mapped to grade point)
  - Example (purely for illustration purposes) 65 – 69% = C/2.0, 70 – 74% = C+/2.3
  - But **grade points don't correlate back to percentages**
    - e.g. I was awarded a 66% on midterm and then I see this is a 2.0 GPA (out of 4.0)
    - Does this mean that my percentage 'went' from a 66% to a 50%!!!???
    - No.
      - A 'C'/2.0 does not mean that 50% was awarded as a course grade.
      - To put this in perspective a passing grade point in this university is a 1.0/D in a course. If a grade point mapped back to a percentage this would mean that anyone getting a 25% or higher would pass any course here.
  - The mapping of the midterm to grade point will be posted sometime after the midterm grades have been released.
  - The mapping of the final exam to grade point will be posted sometime after the final exam grades have been released.



Calculating Your Overall Term Grade Point

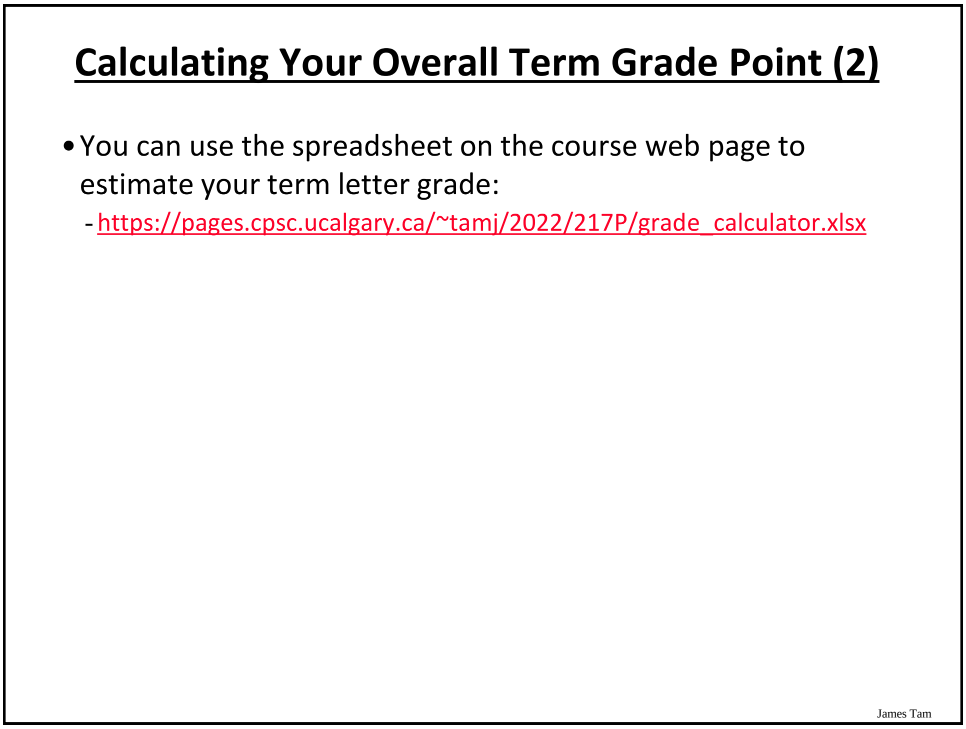
- To determine your weighted term grade point simply multiply *each grade point* by the weight of each component.
  - **Percentages won't be used to determine the term grade point/letter**
- Sum the weighted grade points to determine the term grade.
- Simple and short example (not exactly the same as this term but it should be enough to give you an idea of how to do the specific calculations required this semester):

```
•Assignment 1: weight = 60/100, example score = A
•Assignment 2: weight = 40/100, example score = B+
Weighted assignment 1: 0.6 * 4.0 = 2.4
Weighted assignment 2: 0.4 * 3.3 = 1.32
Total term grade point = 2.4 + 1.32 = 3.72
```

(In this case the term letter is A- if the official university cutoffs were used – more on this shortly)

(The number and weight of graded components will needed by adjusted to compute your actual term grade).

James Tam



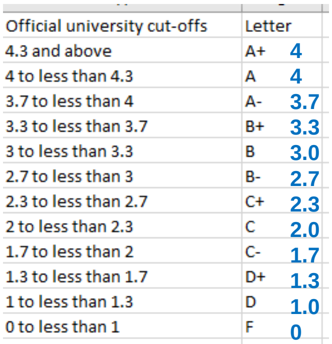
Calculating Your Overall Term Grade Point (2)

- You can use the spreadsheet on the course web page to estimate your term letter grade:
  - https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/grade_calculator.xlsx

James Tam

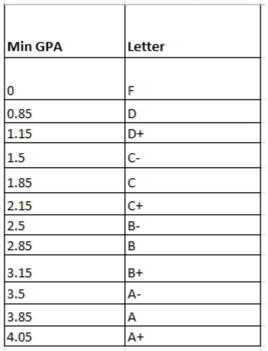
Contrast The Cut-Offs

Official UC cutoffs



Official university cut-offs	Letter	
4.3 and above	A+	4
4 to less than 4.3	A	4
3.7 to less than 4	A-	3.7
3.3 to less than 3.7	B+	3.3
3 to less than 3.3	B	3.0
2.7 to less than 3	B-	2.7
2.3 to less than 2.7	C+	2.3
2 to less than 2.3	C	2.0
1.7 to less than 2	C-	1.7
1.3 to less than 1.7	D+	1.3
1 to less than 1.3	D	1.0
0 to less than 1	F	0

The Tam cutoffs (this course)



Min GPA	Letter
0	F
0.85	D
1.15	D+
1.5	C-
1.85	C
2.15	C+
2.5	B-
2.85	B
3.15	B+
3.5	A-
3.85	A
4.05	A+

- The cutoffs in the spreadsheet are **significantly more lenient** (almost everyone "gets a big break" e.g. instead of 3.7 for an A- it's 3.5 (midpoint between A-/3.7 and B+/3.3 is the higher letter grade)
- Do not expect a further "rounding up" at the end of the term e.g. 3.4999999999999999999999999999999999999999999999 is soooooooooooooooooooooooooooooooooooooooooooooo close to 3.5 can't you get an "A-"
- No or using an Internet emphasis
- NOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO!!!

Common Computer Skills Assumed

- You know what a computer is!
- You've used a computer in some form (e.g., turn on, turn off, open a file, played a game, gone online etc.)
- You have experience *using common applications* (specifically email, web browsers, text editing using a word processor).
- That is, you already have basic computer literacy skills coming into this course.

What This Course Is About

- Writing/creating computer programs.
- But it is *not assumed* that you have prior knowledge of Computer Science (or even experience writing programs)
- It can be a lot of work.

Late night 'coding'

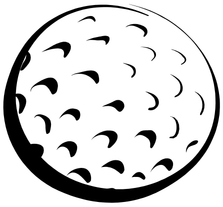
Satisfaction coming from
solving that tough algorithm!

- The course can be completed by students with a normal course load (many already have gotten through it!)
- But be cautious if you already have many other commitments

Wav file from "Tam"

Actual Practice: Common Interview Questions

- Besides looking at degrees granted and grades received, some tech companies (e.g., Google) may ask you questions that appear non-technical:
  - You're asked to solve puzzles during the interview.



- There is a relationship between skill at solving puzzles ("problem solving") and success in a (technically oriented) industry.
  - You will develop these skills writing programs for this class.

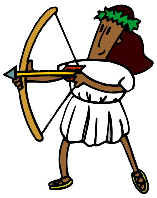
Example list of questions
http://www.businessinsider.com/15-google-interview-questions-that-will-make-you-feel-stupid-2009-11

Course Goals

- Know the basic structure of a computer program (rules for laying out a program and how the basic constructs such as repetition and branching work)
  - If you don't know and understand these concepts then your program won't work at all and you can't proceed to the next goals.
- Develop basic problem solving and analysis skills.
  - As mentioned this is a skill that you will need to develop for "the real world"
- Learn good programming principles.
  - It's not good enough if you know how to write a program that 'works' but the instructions are unclear and logic is not self evident so it is difficult to understand and modify.
  - A rough analogy: Writing a short story that is grammatically correct but is poorly written, hard to follow and in short not entertaining.

How To Succeed

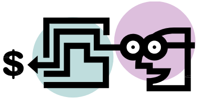
- Successful people



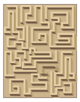
How To Succeed In This Course

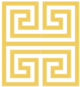
1. Practice things yourself (not by getting the answer from someone/someplace else).

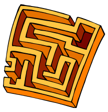
Providing solutions to assignments may be popular among students but useless for learning



What's needed is for me to teach you the skills to solve any reasonable size problem



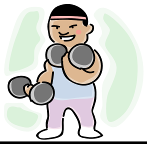




- How Computer Science works: You get better by doing things for yourself (this is a 'hands-on' field of study and work).

Similar to getting fit: you can't just watch

You have to do it yourself



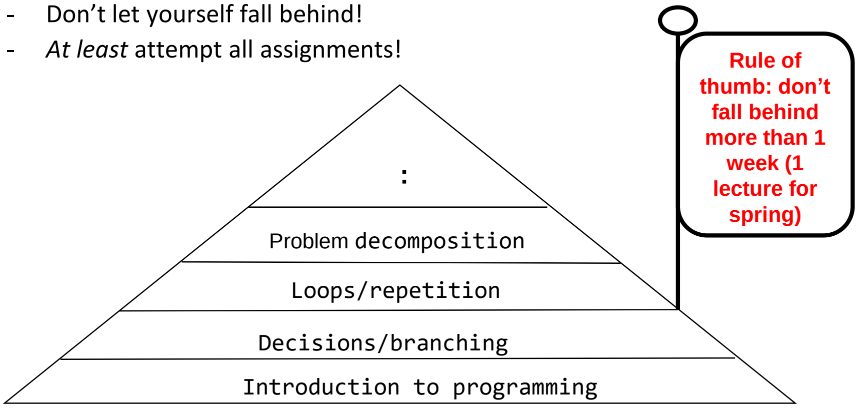
How To Succeed In This Course (2)

- Write lots programs.
  - At the *very least* attempt every assignment.
  - Try to do some additional practice work (some examples will be given in class, some practice assignments will be available on the course web page).
  - Write lots of little 'test' programs to help you understand and apply the concepts being taught.

- Trace lots of code (computer programs)
  - Involves reading through programs that other people have written, and executing it 'by hand' in order to understand how and why it works
  - This is an essential skill.
  - Relying on just running the program and observing the results won't always work (errors?)
- Extra practice problems for the course:
  - https://pages.cpsc.ucalgary.ca/~tamj/2022/217P/assignments/practice/index.html

How To Succeed In This Course (3)

2. Make sure that you keep up with the material
   - Many of the concepts taught later depend upon your knowledge of earlier concepts.
   - Don’t let yourself fall behind!
   - *At least* attempt all assignments!


:
Problem decomposition
Loops/repetition
Decisions/branching
Introduction to programming
Rule of thumb: don’t fall behind more than 1 week (1 lecture for spring)

How To Succeed In This Course (4)

- If you find concepts unclear trying to research the answer on your own can be beneficial (because this is a ‘hands on’ field).
  - Read alternate explanations of the concepts covered in class in the text book (or other textbooks: remember that electronic books accessible through the library-Safari are ‘free’).
  - Looking at online resources:
    - Keep in mind that some programming resources online just like other online information may not always be a good source.
    - Start with more reputable sources
      - http://proquest.safaribooksonline.com.ezproxy.lib.ucalgary.ca/
      - www.python.org
- Addendum to the previous point #2 and a point raised earlier “ask questions”.
  - If you are still unclear on concepts then make sure that you ask for help.
  - Don’t wait too long (more than a few days) to do this because latter concepts may strongly depend on your understanding of earlier concepts.

How To Succeed In This Course (5)

3. Look at the material before coming to lecture so you have a rough idea of what I will be talking about that day:
   a) Read the slides
   b) Look through the textbook(s)

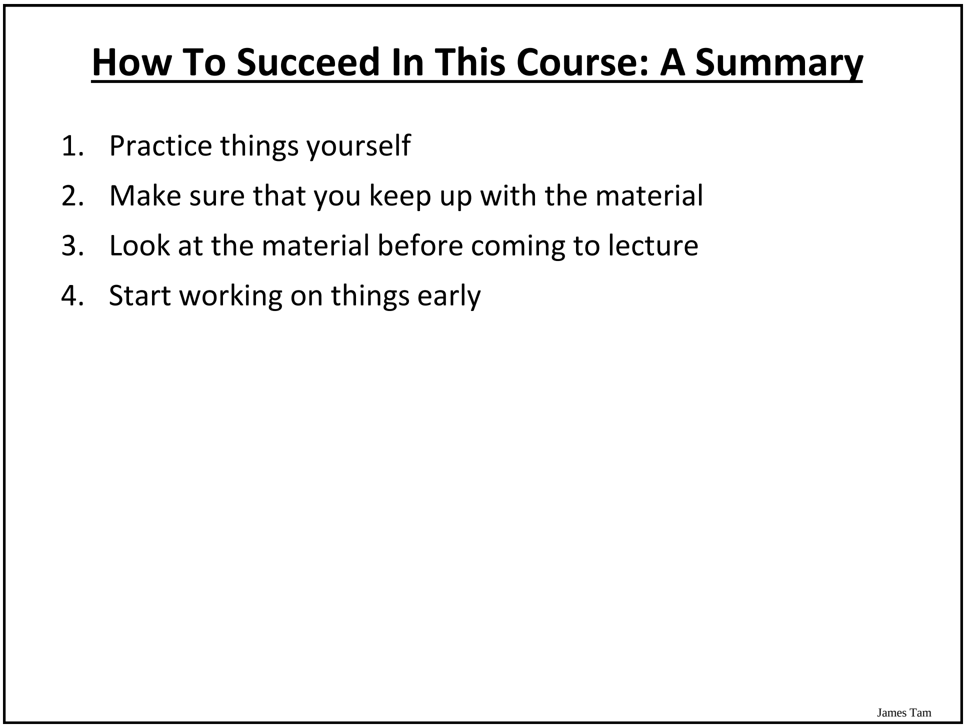
When we get to more complicated programs that appear to ‘jump around’ in how they execute (“section: problem decomposition/functions”) just having an idea of the scope and components of the program beforehand can be useful when I cover it in class.

James Tam

How To Succeed In This Course (6)

4. Start working on things as early as possible:
   - Don't cram the material just before the exam, instead you should be studying the concepts as you learn them throughout the term.
   - It’s important to work through and understand concepts *before* you start assignments.
   - If you try to learn a new concept *and* work out a solution for the assignment at the same time then you may become overwhelmed.
   - Some assignments may require the application of multiple concepts, not all the concepts have to be completely covered before you start working on an assignment.
     - Start working based on what’s currently been covered (this will teach you how to decompose a program and work on it a part at a time).

James Tam



How To Succeed In This Course: A Summary

1. Practice things yourself
2. Make sure that you keep up with the material
3. Look at the material before coming to lecture
4. Start working on things early

James Tam