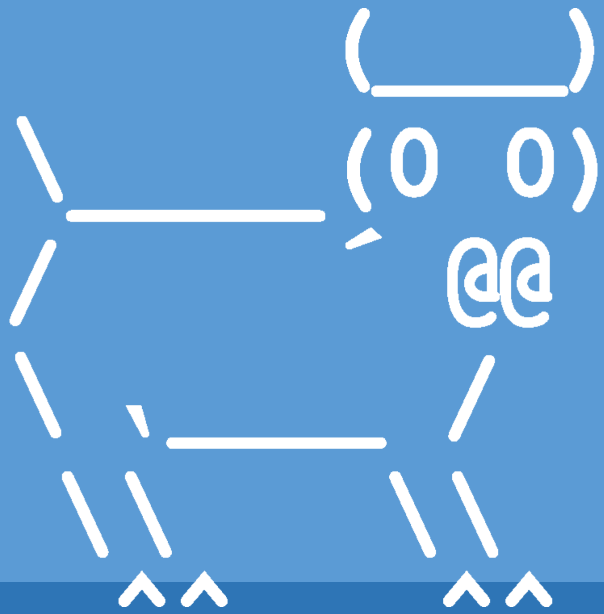
COMMON MISCONCEPTIONS

REVIEW



Introduction to Computer Science

Iris Howley

LAST LECTURES

Almost there!!

REMAINING DELIVERABLES

- Quiz 3 is this **Friday, May 15**!
  - (*topics include up to, but not including recursion*)
- Quiz 4 is Friday, May 22
  - (*up to and including recursive data structures*)
- **SCS Forms**
- Remote Learning Feedback Google Form
  - *(link emailed later)*

THE PURPOSE OF QUIZZES/EXAMS

Showing you know the knowledge without having to consult an external source.

If you need to consult a book, check with python, etc. for every programming problem, do you *know* the concepts?

Quizzes are intended to be doable without notes, without python!

Quiz Strategies

- Quiz 3 will be 1 hour (~6 questions)
  - + 30 minutes for technical difficulties
- These quizzes are meant to be doable with *pencil and paper*
  - Do not let double-checking your work in Python take too much time!
- The points allotment tells you which questions you should spend more time on!
- There's partial credit for incorrect responses
- *Studying*: Practice homeworks & labs without python

Python is best used to check your work, if you have leftover time!

LAST LECTURES

- Friday
  - Common Misconceptions
  - Computer Science opportunities

TODAY'S LESSON

Reviewing some common misconceptions.

(Look out for these! Entire practice worksheet available on course website.)

Practice worksheet, question 1

1. Convert the following `if/else` into a one-line return statement:

```
if mylist is not None and len(mylist) > 24:
    return True
else:
    return False
```

PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 1

Possible Answer

```
return mylist is not None and len(mylist) > 24
```

even better:

```
return mylist and len(mylist) > 24
```

If we don't check that `mylist` exists, we'll get an error with `len(mylist)`

Practice worksheet, question 2

2. Convert the following loop into a one-line list comprehension:

```
newList = []
for v in mylist:
    newList.append(v + 5)
```

PAUSE WHILE ANSWERING THE QUESTION

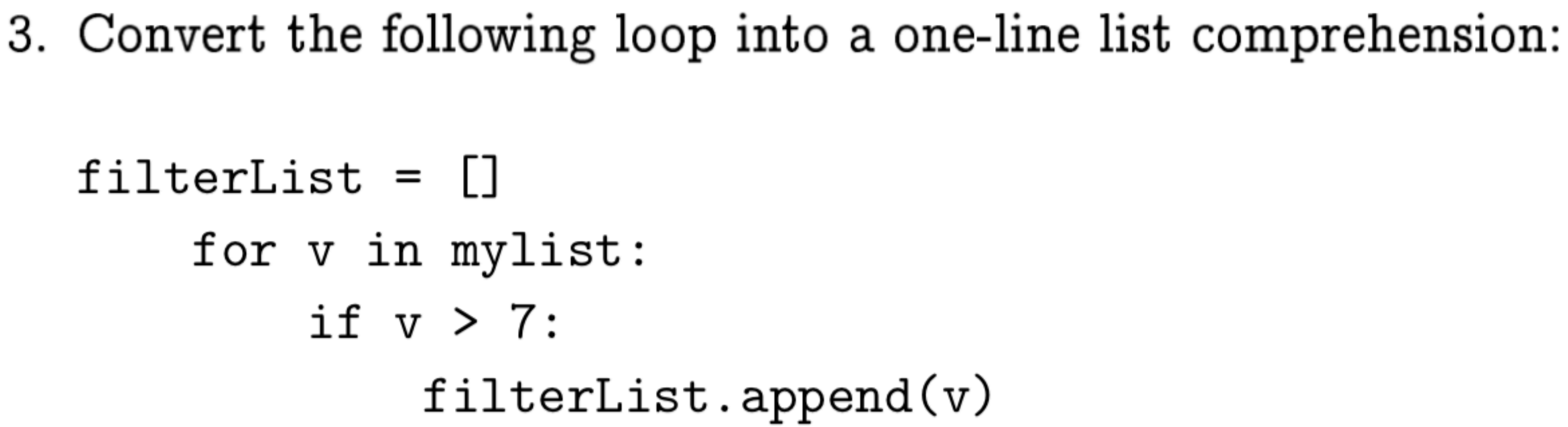
Practice worksheet, question 2

Possible Answer

```
newList = [ v+5 for v in mylist ]
```

Mapping new values from mylist to newList

Practice worksheet, question 3



3. Convert the following loop into a one-line list comprehension:

```
filterList = []
    for v in mylist:
        if v > 7:
            filterList.append(v)
```

PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 3

Possible Answer

```
filterList = [v for v in mylist if v > 7]
```

Filtering values from mylist to filterList based on a criteria (> 7)

Practice worksheet, question 4

4. Fix the following generator, which should generate a stream of lowercase letters in a given String, `mystr`:

```
def findLowers(mystr):
    currentIndex = 0
    while True:
        if mystr[currentIndex].islower():
            return mystr[currentIndex]
            currentIndex += 1
```

Could fix with `while currentIndex < len(mystr)`, but there's better options

Looks like it will run forever, strings are finite!

Generators `yield`, not return!

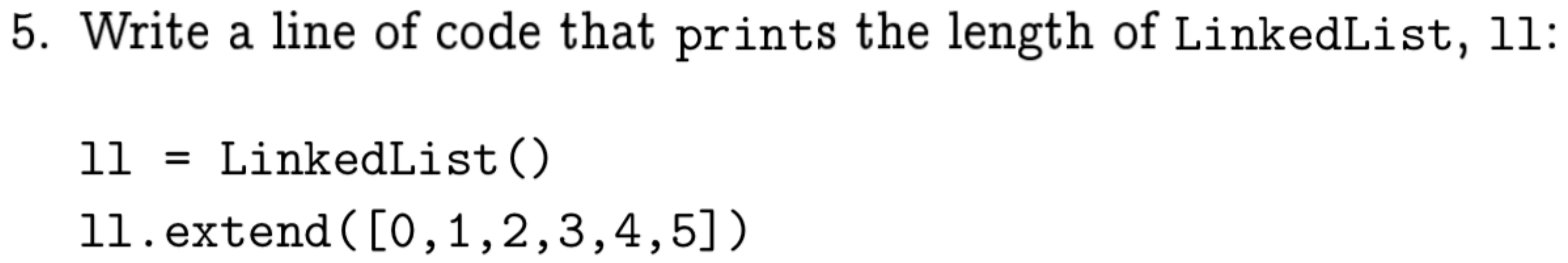
PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 4

Possible Answer

```
def findLowers(mystr):
  for ch in mystr:
    if ch.islower():
      yield ch
```

Practice worksheet, question 5



5. Write a line of code that `prints` the length of `LinkedList`, `ll`:

```
ll = LinkedList()
ll.extend([0,1,2,3,4,5])
```

PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 5

Possible Answer

```
ll = LinkedList()
ll.extend([1,2,3,4])
print(len(ll))
```

Practice worksheet, question 6

6. `sum(self)` is a method within the `LinkedList` class. Write a line of code that uses this method to print the LinkedList, `ll`'s, sum:

PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 6

Possible Answer

```
ll = LinkedList()
ll.extend([1,2,3,4])
print(ll.sum())
```

Why is `ll.sum()` called differently from `len(ll)` ?

Practice worksheet, question 7

7. `__contains__(self, val)` is a special method in Python. Write a line of code that implicitly calls this method, to see if our LinkedList, `ll`, contains the value 24:

PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 7

Possible Answer

```
ll = LinkedList()
ll.extend([1,2,3,4])
print(24 in ll)
```

Special methods!!

Practice worksheet, question 8

8. Recall our `Tree` class. Write a method for Tree objects that counts the number of leaf nodes in the tree:

PAUSE WHILE ANSWERING THE QUESTION

Practice worksheet, question 8

Possible Answer

```
def countLeaves(self):
  if self.isLeaf:
    return 1
  if self.left:
    lCount = self.left.countLeaves()
  if self.right:
    rCount = self.right.countLeaves()
  return lCount + rCount
```

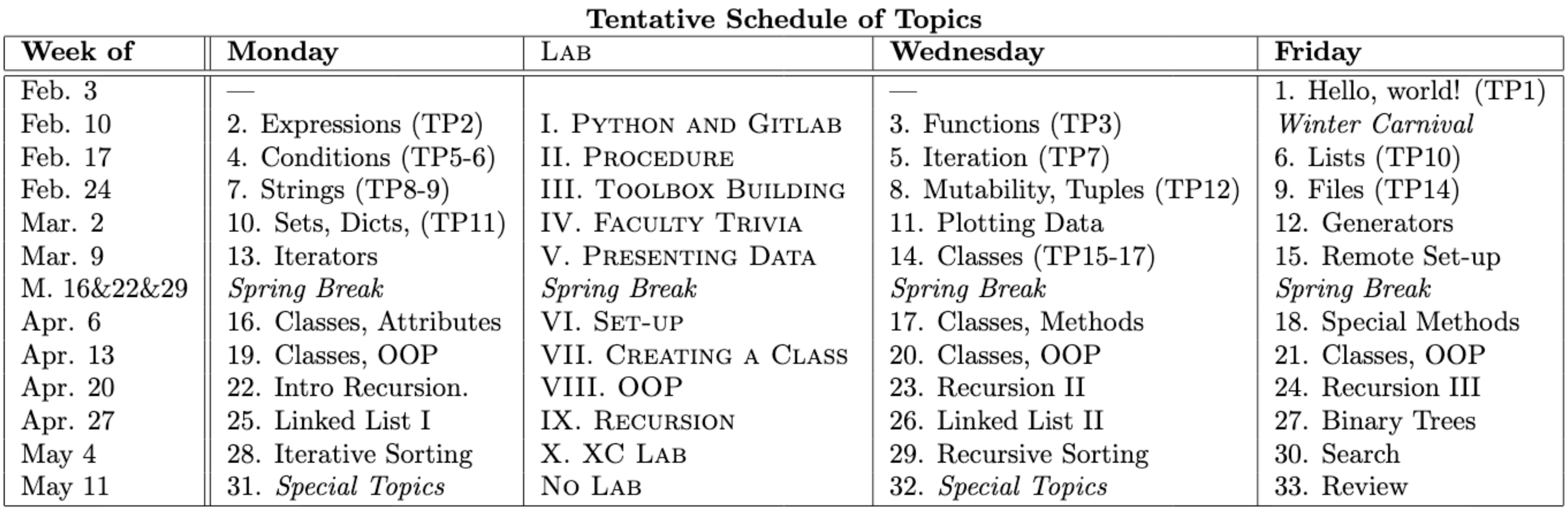
Will exit the method if we're a leaf

Check the left tree exists

Note: `if`, not `elif`!

Need to check the right sub-trees, even if left exists!

WHAT WE'VE LEARNED THIS SEMESTER



Tentative Schedule of Topics

Week of	**Monday**	Lab	**Wednesday**	**Friday**
Feb. 3	—		—	1. Hello, world! (TP1)
Feb. 10	2. Expressions (TP2)	I. Python and Gitlab	3. Functions (TP3)	*Winter Carnival*
Feb. 17	4. Conditions (TP5-6)	II. Procedure	5. Iteration (TP7)	6. Lists (TP10)
Feb. 24	7. Strings (TP8-9)	III. Toolbox Building	8. Mutability, Tuples (TP12)	9. Files (TP14)
Mar. 2	10. Sets, Dicts, (TP11)	IV. Faculty Trivia	11. Plotting Data	12. Generators
Mar. 9	13. Iterators	V. Presenting Data	14. Classes (TP15-17)	15. Remote Set-up
M. 16&22&29	*Spring Break*	*Spring Break*	*Spring Break*	*Spring Break*
Apr. 6	16. Classes, Attributes	VI. Set-up	17. Classes, Methods	18. Special Methods
Apr. 13	19. Classes, OOP	VII. Creating a Class	20. Classes, OOP	21. Classes, OOP
Apr. 20	22. Intro Recursion.	VIII. OOP	23. Recursion II	24. Recursion III
Apr. 27	25. Linked List I	IX. Recursion	26. Linked List II	27. Binary Trees
May 4	28. Iterative Sorting	X. XC Lab	29. Recursive Sorting	30. Search
May 11	31. *Special Topics*	No Lab	32. *Special Topics*	33. Review

Online SCS Evaluations

Available via Glow:

‘Course Evaluations’ on your dashboard

On your Glow dashboard you’ll see a course called “Course Evaluations.” Click on this and then follow the instructions you see on the screen. If you have trouble finding the evaluation, you can reach out to ir@williams.edu

Pro Tip: Be *specific* so your instructors can fix it in the future!

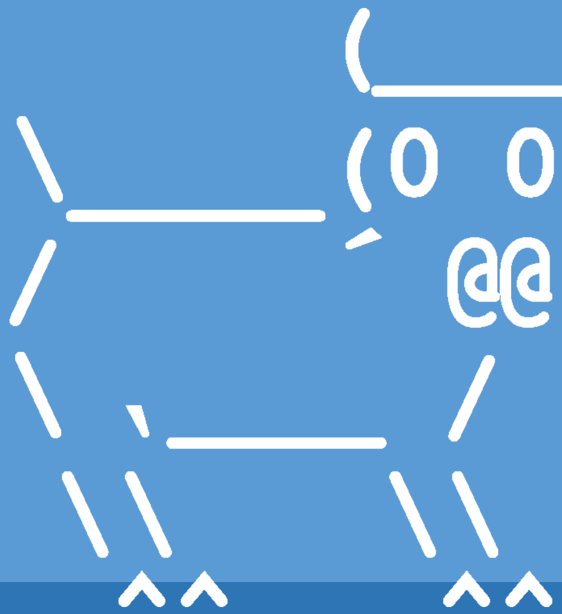
Script for the Student Course Survey

- Every term, Williams asks students to participate in end-of-semester course evaluations. Your feedback will help improve this course for other students taking it in the future, and help shape the [department/program name] curriculum.
- You may skip questions that you don't wish to answer, and there is no penalty for choosing not to participate. All of your answers are confidential and I will only receive a report on your responses after I have submitted all grades for this course. While evaluations are open, I will receive information on how many students have filled out the evaluations, but I won't know which of you have and haven't completed them. I won't know which responses are associated with which student unless you identify yourself in the comments.
- To access the online evaluations, log into Glow (glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "Course Evaluations." Click on this and then follow the instructions on the screen. If you have trouble finding the evaluation, you can ask a classmate or reach out to Institutional Research at ir@williams.edu. You can complete the online evaluation through the end of reading period. If you haven't filled it out by the beginning of reading period, you will start receiving email reminders from Institutional Research.

QUESTIONS?

Please contact me!

Continuing in Computer Science



Introduction to Computer Science

Iris Howley

TODAY'S LESSON

Need something to learn this summer?
Or interested in continuing in the major?

(Some resources for moving onward and upward!)

Things to Learn Over the Summer

- Using computers effectively
- Data Structures/Java to make the first few weeks of CSCI136 smoother
- Intermediate python skills for building cool projects
- A personal project built on GitHub as a means to begin building a software development portfolio.
- Preparing for the Discrete Math proficiency exam

Things to Learn Over the Summer

Using computers effectively

- Using computers effectively
  - MIT's "*The Missing Semester of Your CS Education*": https://missing.csail.mit.edu/
  - Command-line, scripting, debugging, etc.
  - Concepts that will make you a more efficient computer scientist

Things to Learn Over the Summer

Learning Java

- Enrolling in low stakes Data Structures/Java courses that might make the first couple weeks of CS136 smoother.
  - https://www.edx.org/professional-certificate/pennx-computer-science-essentials-for-software-development

- The Java version of the CS134 textbook is also available

MOOC: Massive Open Online Course.

Free online courses with large enrollments. Not a lot of instructor interaction, more reliant on interaction with peers in the discussion forums (or via peer evaluation of projects). Sometimes interactive programming environments with immediate feedback.

Things to Learn Over the Summer

Intermediate python for cool projects

- Enrolling in a low stakes intermediate python course using web data, or databases, or other useful concepts
  - https://www.coursera.org/specializations/python
  - "Using python to access web data"
  - "Using databases with python"

- And then build some cool projects!

Things to Learn Over the Summer

Building a software development portfolio

- GitHub
  - Like GitLab, but with a public-facing option as well
  - Can work on public projects, or share your own personal projects
  - Useful for internship interviews, too. Shows interest in programming!

DO NOT post Williams coursework publicly!

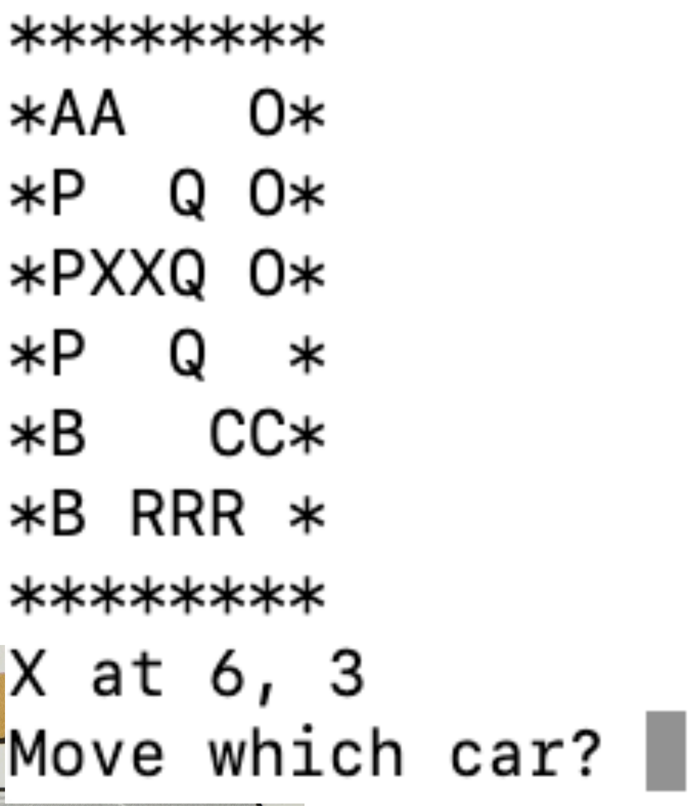
(Honor Code Violation)

(But would be *great* for version control + publicizing your <u>personal</u> projects)

Things to Learn Over the Summer

Building a software development portfolio

- Making cool stuff with programs! Personal projects!
  - A script to automatically generate new workout routines
  - Games
    - A sudoku game generator
    - MineSweeper game generator & solver
    - Rush Hour sliding block puzzle game



```
********
*AA   O*
*P  Q O*
*PXXQ O*
*P  Q  *
*B   CC*
*B RRR *
********
X at 6, 3
Move which car?
```

Things to Learn Over the Summer

Building a software development portfolio

- Making cool stuff with programs! Personal projects!
  - A script to automatically generate new workout routines
  - Games
    - A sudoku game generator
    - MineSweeper game generator & solver
    - Rush Hour sliding block puzzle game
  - A quilt generator, and then make the quilt…

```
********
*AA   O*
*P  Q O*
*PXXQ O*
*P  Q  *
*B   CC*
*B RRR *
********
X a
Mov
```

Interested in Being a CS major?

- See "*Admission to the Major*":
  - https://csci.williams.edu/major-requirements/
  - *Short version*: Complete CS134 & CS136 & Discrete Math requirement prior to end of sophomore year to declare CS as a major

- Discrete Math: MATH 200 or Proficiency Exam:
  - https://csci.williams.edu/required-proficiency-in-discrete-mathematics/
  - 'Could take a Discrete Math class this summer to prep for the exam

- Discrete Math is a prerequisite for core course CSCI256, which is in turn a prerequisite for many other courses in the major

QUESTIONS?

Please contact me!

Leftover Slides

DO THE PRACTICE WORKSHEET

Available on the course website

Use a pencil and paper! Only use the computer to *check* your work.

PAUSE WHILE COMPLETING THE PRACTICE WORKSHEET

Giving Constructive Feedback

```
        (___)
\_______(0 0)
/      ´  @@
\  `_____/
 \ \    \ \
  ^ ^    ^ ^
```

Introduction to Computer Science

Iris Howley

TODAY'S LESSON

Giving useful feedback

(A reminder for the online SCS forms for the course!)

```
GRADE SHEET FOR CS134 LAB 8 ("Cipher Classes"):

Requirements of this lab:
   1. Requirements of the lab:
      * Passes doctests (see output below)
      * rotateLetter uses ord-->chr, not dictionary
      * Properties/Setters used appropriately
      * super() only used in initializers, others inherited w self.
      * Message Class:  __slots__ = ['_text']
        * wordCount: .split() not .split(' '). See: ' hello,  world '
      o Plaintext:
        * __slots__ = ['_shift']
   2. Style
      * Good choice of Python commands.
      * Each .py file has a top-level docstring describing the module
      * Includes docstrings for *each* method & Class: """...."""
      * Informative one-line comments
      * Meaningful names used in declarations (vars, functions,..)
      * Good and consistent formatting
```

Tips for Critics

Borrowed from CSCI376: Human-Computer Interaction

- In Design Thinking/Research there critiquing strategies:
  - Hamburger method
  - I like, I wish, what if
  - Rose, Bud, Thorn
  - Socratic method, …
- These provide ways to give critique that can help the conversation go smoothly
  - Can give you a question to ask when you do not have one, provide a way to ask that is productive and less likely to create defensive reaction

Tips for Critics: Hamburger Method

Borrowed from CSCI376: Human-Computer Interaction

- Bun:
  - Something fluffy and nice
- Meat:
  - Criticism on how to improve
- Bun:
  - Something fluffy and nice

How to Survive a Critique by Cheng (2013)
https://www.aiga.org/how-to-survive-a-critique/

Tips for Critics: I Like, I Wish, What If

Borrowed from CSCI376: Human-Computer Interaction

- I Like:
  - Lead with something nice
- I Wish:
  - Some criticism, often leading from what you like
- What If:
  - An idea to spark further conversation, better than:“I think you should have…” or “Why didn’t you …”
  - Gives the presenter benefit of the doubt if they did already think of your idea, can present rationale

Stanford Design School

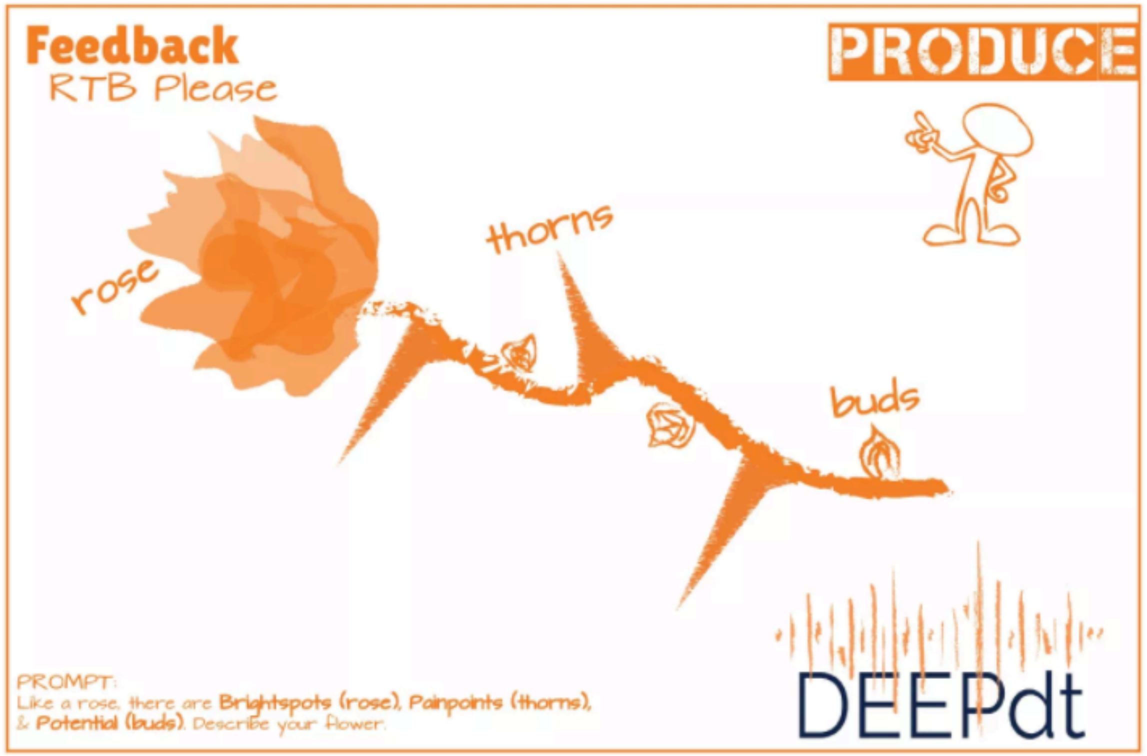
http://dschool-old.stanford.edu/wp-content/themes/dschool/method-cards/i-like-i-wish-what-if.pdf

Tips for Critics: Rose, Bud, Thorn

Borrowed from CSCI376: Human-Computer Interaction

- Roses – bright spots
- Thorns – pain points
- Buds – Potential

- Can be used as a self-evaluation, too



PROMPT:
Like a rose, there are Brightspots (rose), Painpoints (thorns), & Potential (buds). Describe your flower.

Rose, Bud, Thorn Technique by Cantwell (2014)
https://www.deepdesignthinking.com/deep-design-thinking/rose-thorn-bud-technique-im-a-little-behind-the-curve

Student Evaluations

1. What are the strongest features of your professor’s teaching? In other words, what contributes most to your learning?

2. What specific suggestions do you have for improvements that your professor can make?

Online SCS Evaluations

Available via Glow:

‘Course Evaluations’ on your dashboard

On your Glow dashboard you’ll see a course called “Course Evaluations.” Click on this and then follow the instructions you see on the screen. If you have trouble finding the evaluation, you can reach out to ir@williams.edu

QUESTIONS?

Please contact me!