

CSCI134 : Intro to Computer Science Fall 2024

Syllabus

- [Logistics](#)
- [Course Description](#)
- [Organization](#)
- [Grading](#)
- [Programming Assignments and Pre-labs](#)
- [Homework](#)
- [Late Policy](#)
- [Exams](#)
- [Workload Expectations](#)
- [Community](#)
- [Honor Code](#)
- [Intellectual Property](#)
- [Accommodations](#)
- [Personal Health](#)
- [Public Health](#)
- [Lab Grading](#)
- [Getting Help](#)
- [Acknowledgements](#)

Logistics

Instructors	Bill Jannen (https://www.cs.williams.edu/~jannen/), Iris Howley (https://cs.williams.edu/~iris/)
Technical Support	Lida Doret (mailto:lida@cs.williams.edu)

Webpage & Email	https://www.cs.williams.edu/~cs134 (https://www.cs.williams.edu/~cs134), cs134staff@williams.edu (mailto:cs134staff@williams.edu)
Lectures	MWF 9:00-9:50am MWF 10:00-10:50am MWF 11:00-11:50am
Labs	Mon 1:00-2:30pm, Mon 2:30-4:00pm Tues 2:30-4:00pm
Classroom	Schow 030A (Lectures), TCL 216/217A (Labs)
Python Resources	Optional: Allen Downey's <i>Think Python, 2ed</i> , an online reference book (https://greenteapress.com/wp/think-python-2e/). Python documentation: https://www.python.org/doc/ (https://www.python.org/doc/).

Course Description

We are surrounded by information. This course introduces fundamental computational concepts for representing and manipulating data. Using the programming language Python, this course explores effective ways to organize and transform information in order to solve problems. Students will learn to design algorithms to search, sort, and manipulate data in application areas like text and image processing, scientific computing, and databases. Programming topics covered include procedural, object-oriented, and functional programming; control structures; structural self-reference; arrays; lists; streams; dictionaries; and data abstraction. This course is appropriate for all students who want to create software and learn computational techniques for manipulating and analyzing data.

Organization

We will meet three times each week for lecture and once a week for lab. During lecture hours, we will learn new concepts and problem solving strategies. During the 90-minute lab section, we will gain hands-on experience with the concepts through programming assignments.

Grading

Final grades will be determined according to the following:

- Programming assignments (Labs): 30%
 - Pre-lab assignment: 5% of Lab grade
- Homework: 10%
- Midterm exam: 30%
- Final exam: 30%

Each of these items are explained in detail in the following sections.

Programming Assignments and Pre-labs

Each week students will be assigned a lab project. Most lab assignments will be posted every Friday. Lab submissions are due:

- **Wednesday at 10pm for the Monday lab groups**
- **Thursday at 10pm for the Tuesday lab groups**

Pre-labs are short “paper and pencil” assignments that are meant to organize your thinking about important lab concepts before you arrive, and they are due at the start of the lab meeting itself. Pre-labs are graded pass/fail and together make up 5 out of the 30 percentage points allocated to lab assignments.

Labs will be graded on a letter scale (A–F). An assessment-sheet will be distributed with the lab each week that includes specific details about assignment expectations.

Homework

Each week students will be assigned homework. Homework assignments will be posted every Wednesday and are due **on Monday at 10pm**. Homework will be used to test comprehension on important course concepts and help students prepare for the exams. We will drop your lowest homework score from your final grade.

Late Policy

Students are expected to turn in all assignments by the due date to receive full credit. For up to two individual lab assignments, students may request a 12-hour extension. These extensions are meant to accomodate (potentially) unexpected happenings that come up throughout the semester, including atheltic competitions, illness, and other life events. A reason does *not* need to be given, but extensions must be requested before the assignment's original deadline.

There are other life events that fall outside of the scope of a 12-hour extension. The instructors will try to be reasonable and work with you to develop a plan that is appropriate for your unique circumstances. Please contact your instructors as soon as possible if you are facing challenges.

Exams

The midterm exam will be on the evening of **Thursday, October 17**. The final exam will be scheduled by the registrar's office to occur during the final exam period. The exams will be closed book, closed notes, and will test conceptual understanding of the material. Details regarding the specific format of the exams will be discussed in class.

Workload Expectations

Attendance is required in both lecture and lab. In general, beyond the 4 hours we spend together during our class and lab meetings, students should expect to spend (on average) approximately 10 hours per week on work related to class. Aside from completing the weekly lab and homework assignments, students are responsible for reading supporting material and investigating approved online resources (documentation, tutorials) as necessary.

Community

We embrace diversity. We welcome all students and we expect everyone to contribute and support a respectful and welcoming environment. If you have concerns, please share them with us or the college administration.

Honor Code

For programming assignments in computer science courses, the honor code is interpreted in very specific ways. Unless otherwise indicated, labs are expected to be the work of the individual student, designed and coded by them alone. Help interpreting error messages is allowed, but, if you are taking photos of someone else's screen, or if you are telling someone else what to type, it is likely the work is no longer the work of an individual student. **The following are all considered violations of the Honor Code: (a) giving your solution to other students, (b) submitting another person's solution as your own, or (c) using another person's solution as the starting point for your solution.**

One of the major goals of this course is to learn how to build programs. Therefore, you should never use artificial intelligence tools (like ChatGPT or GitHub Copilot) to assist you with labs or other assignments. **Any use of generative AI technology is considered a violation of the Honor Code.**

If you do not understand how the Honor Code applies to a particular assignment, consult your instructor. Students should be aware of the Computer Ethics outlined in the Student Handbook. Violations (including uninvited access to private information and malicious tampering with or theft of computer equipment or software) are subject to disciplinary action.

The College and Department also have computer usage policies that apply to courses that make use of computers. Read more about these policies [here \(http://csci.williams.edu/the-cs-honor-code-and-computer-usage-policy\)](http://csci.williams.edu/the-cs-honor-code-and-computer-usage-policy).

Intellectual Property

As per College policy, no part of this course may be reproduced and/or distributed. In particular, no videos recorded as part of this class may be shared with anyone external to the CSCI 134 course.

Accommodations

If formal accommodations need to be made to meet your specific learning or physical abilities, you should contact your instructors as soon as possible to discuss appropriate accommodations. You should also contact the [Office of Accessible Education \(https://www.williams.edu/accessible-education/contact-us/\)](https://www.williams.edu/accessible-education/contact-us/). We will work together to ensure this class is accessible and inclusive.

Personal Health

If you are experiencing mental or physical health challenges that are significantly affecting your academic work, you are encouraged to contact your instructor and/or reach out to the Dean's Office (x4171).

Public Health

To keep our classroom and lab environments as healthy as possible, use common-sense measures such as wearing a mask if you have even minor cold-like symptoms. If you have a fever or feel ill, **please do not come to class or lab** and inform the course staff of your absence. We will work with you to make sure you can make up any missed work and to develop a plan that allows you to continue making progress in the course as you recover from your illness.

Lab Grading

Programming labs will be graded on the following scale:

Grade	Description
A	A flawless submission that meets all the requirements and gets the job done in a particularly elegant way.
A-	A submission that satisfies almost all the requirements—a job well done.
B+	Submission meets the major requirements for the assignment with a few minor issues.
B	A submission that has problems serious enough to fall short of the requirements for the assignment.
C	A submission that has extremely serious problems, but nonetheless shows some effort & understanding.
D	A submission that shows little effort and does not represent passing work.

Getting Help

The course staff and TAs will hold weekly help hours for all students. To get the most up-to-date help hour schedule, add the [CSCI 134 Calendar \(https://calendar.google.com/calendar/embed?src=c_i1epr6fcdjvn2q1iobrqq023hoo%40group.calendar.google.com&ctz=America%2FNew_York\)](https://calendar.google.com/calendar/embed?src=c_i1epr6fcdjvn2q1iobrqq023hoo%40group.calendar.google.com&ctz=America%2FNew_York) to your Google Calendar.

1. **Staff Help Hours:** The course staff will hold weekly help hours for all students in the CS common room:
 - Wednesday 1-4 pm
 - Thursday 2-4pm
2. **TA Help Hours:** CSCI 134 TAs hold weekly help hours in TCL 217A/216. See [CSCI 134 Calendar \(https://calendar.google.com/calendar/embed?src=c_i1epr6fcdjvn2q1iobrqq023hoo%40group.calendar.google.com&ctz=America%2FNew_York\)](https://calendar.google.com/calendar/embed?src=c_i1epr6fcdjvn2q1iobrqq023hoo%40group.calendar.google.com&ctz=America%2FNew_York).
3. **Tutoring:** Free tutoring is available for any student enrolled in this course through the office of [Content Tutoring \(https://www.williams.edu/peer-academic-support/content-tutoring/\)](https://www.williams.edu/peer-academic-support/content-tutoring/). You are welcome to schedule an individual tutoring session or visit the Math and Science Resource Center (the MSRC) for help. The MSRC is for drop-in tutoring where you can ask tutors a quick question, collaborate with people in your class, or just get some work done among others. One note: tutors for this class will only be available in the MSRC on particular weeknights. To find the MSRC schedule or make an appointment with an individual tutor, please visit [Accudemia \(https://williams.accudemia.net/\)](https://williams.accudemia.net/).

Acknowledgements

Many individuals, including Jeannie Albrecht, Duane Bailey, Rohit Bhattacharya, Lida Doret, Molly Feldman, Stephen Freund, Mark Hopkins, Iris Howley, Kelly Shaw, Shikha Singh, Bill Jannen, and Brent Heeringa have contributed to the materials for this course.

CSCI 134

Introduction to Computer Science

[Home](#) | [Lectures](#) | [Assignments](#) | [Resources](#) | [Williams CS](#)

Course Schedule

The table below lists the topics we will discuss and any items associated with each class/date. Some of these resources will only be accessible from within the campus network. Information about the proxy server can be found [here](#).

The schedule is subject to change: we may wish to explore new topics in response to current events or student interest. You're more than welcome to work ahead, but please check with us first!

Mon	Tue	Wed	Thu	Fri
9/2	9/3	9/4	9/5	9/6 Welcome & Overview Course Syllabus Slides pogils 1: Intro to Python
9/9 Types & Expressions Slides videos 02-types-and-expressions pogils How To POGIL 2: Variables and Input 3: Arithmetic, Operators, Assignment	9/10	9/11 Functions Slides videos 0-functions pogils 4: None Functions 5: Value Functions	9/12	9/13 Booleans Slides videos Making Change 04-booleans pogils 6: Booleans 7: if elif else
9/16 Conditionals Slides	9/17	9/18 Strings & Iteration Slides	9/19	9/20 Lists & Iteration Slides

videos 05-conditionals 3 Ways to Run Python pogils 8: More elif 9: Nested if else		videos 06-strings-iteration pogils 10: Strings 11: for Each Loop		videos 07-lists-iteration pogils 12: Lists
9/23 Range & Nested Loops Slides flagVar.py videos 08-ranges-nestedLoops pogils 13: Range 14: Nested Loops	9/24	9/25 Nested Lists Slides videos 09-nestedLists pogils 15: Lists of lists	9/26	9/27 While Loops Slides guess.py sched.py videos 10-whileLoops-modules pogils 16: while Loops
9/30 Querying Sequences Slides schedule.py videos See 06 See 07 pogils 17: import Main 18: Sequence Algorithms	10/1	10/2 Mutability & Aliasing Slides videos 12-mutability-aliasing pogils 19: is Identity Value 20: List Aliasing	10/3	10/4 Scope Slides videos 13-scope pogils 21: Scope 22: Scope - Function Frame
10/7 Sets Slides sets.py videos 14-sets pogils 23: Sets	10/8	10/9 Files & plotting HTML code.zip videos 15-files-plotting pogils 24: Files 25: Plotting	10/10	10/11 Mountain Day?

10/14 Reading Period	10/15 Reading Period	10/16 Review Session Bring Questions!	10/17 Midterm Exam In TPL 203, from: • 6-7:30pm OR • 8-9:30pm Practice Exam Sample Solutions	10/18 Dictionaries Slides videos 16-dictionaries pogils 26: Dictionaries 27: Dicts of Dicts
10/21 Recursion Slides videos 17-recursion pogils 28: Recursion	10/22	10/23 More Recursion Slides videos 18-recursion-vs-iteration pogils 29: Recursion Function Frame 30: Recursion vs. Iteration	10/24	10/25 Graphical Recursion Slides videos 19-graphicalRecursion pogils 31: Turtle 32: Graphical Recursion
10/28 Methods, Classes, Objects Slides videos 20-methods-classes-objects pogils 34: Methods 35: Intro to Classes	10/29	10/30 Accessor/Mutator/Special Methods Slides videos 21-attributes-methods pogils 36: Accessor/Mutator Methods 37: Special Methods	10/31	11/1 Inheritance & sorted Slides videos 22-inheritance pogils 38: Inheritance 39: Key Sorting
11/4 Tic-Tac-Toe	11/5	11/6 Tic-Tac-Toe-Two	11/7	11/8 Tic-Tac-Toe-Three

Slides board.py graphics.py		Slides tttboard.py tttcube.py		Slides guessinggame.py tttgame.py
11/11 Tic-Tac-Toe-Four Slides pogils 40: Tuples	11/12	11/13 Special Methods, Linked Lists Slides linkedlist.py videos 27-specialmethods-linkedlists pogils 43: LinkedList 44: LinkedList-methods	11/14	11/15 Linked Lists, Efficiency Slides linkedlistIter.py videos 28-iterators pogils 45: LinkedList-Iterators
11/18 Efficiency Slides videos 29-efficiency pogils 46: Searching-Efficiency	11/19	11/20 Searching/Sorting Slides videos ← See videos 29 & 31 →	11/21	11/22 Sorting Slides videos 31-sorting pogils 47: Sorting
11/25 List Comprehensions Slides videos 32-listComprehensions-tuples-strMethods pogils 48: List Comprehensions	11/26	11/27 Thanksgiving Break	11/28	11/29 Thanksgiving Break
12/2 Java Slides	12/3	12/4 Wrapup Slides	12/5	12/6 Review (Jeopardy!)

videos 33-java-intro pogils 60: Java				Slides
12/9 Reading Period Review Session Time/Location: 10a-12p, TPL203 Bring Questions! Student Help Hours in CS Common Room, 1-3pm	12/10 Reading Period Student Help Hours in CS Common Room, 2:30-4:30p	12/11 Final Exam In Wachenheim B11, from: • 9:30-11:50a Practice Exam Sample Solutions	12/12	12/13
12/16	12/17	12/18	12/19	12/20

Name: _____

Partner: _____

Python Activity 38: Classes - Inheritance

When subclasses inherit features from parent classes, it reduces code redundancy!

Learning Objectives

Students will be able to:

Content:

- Explain how **parent** and **child classes** are relevant to **class inheritance**
- Predict how methods and attributes will be inherited by child/sub-classes

Process:

- Build sub-classes that inherit attributes and methods from the parent class.

Prior Knowledge

- Python concepts: user-defined classes, attributes, methods, special methods

Concept Model:

We want to build a data structure to represent the artworks at the local art museum. Before we begin programming, we should think about the different kinds of artwork, and how they are similar or different from one another:

```
class Artwork
  Attributes:
    artist, title, medium, owner, height, width...
  Methods:
    get_artist, set_owner, get_provenance, ...
```

```
class Photograph
  Attributes:
    same as Artwork plus: speed, aperture,
    camera, ...
  Methods:
    same as Artwork plus: get_speed,
    get_camera, ...
```

```
class Sculpture
  Attributes:
    same as Artwork plus: depth, ...
  Methods:
    same as Artwork plus: get_depth, ...
```

CM1. What are some of the common *attributes* shared by all three of these classes?

CM2. Does the Sculpture class have a speed attribute? _____

CM3. What are some of the common *methods* shared by all three of these [related] classes?

CM4. Does the Artwork class have a `get_depth()` method? _____

-  CM5. If we were to implement these three classes with our current concepts so far, how would we have to implement the shared attributes & methods in each of our classes?

Critical Thinking Questions:

1. Examine the following code below, which represents two of the classes from our *Concept Model*.

artwork.py	
<pre>0 class Artwork: 1 def __init__(self, title): 2 self._title = title 3 4 def get_title(self): 5 return self._title 6 7 def __str__(self): 8 return self._title</pre>	<pre>7 class Sculpture(Artwork): 8 def __init__(self, title, depth): 9 super().__init__(title) 10 self._depth = depth 11 12 def get_depth(self): 13 return self._depth 14 15 def __str__(self): 16 return self._title + str(self._depth)</pre>
<pre>24 if __name__ == "__main__": 25 tt = Sculpture("The Thinker", 140) 26 print(tt.get_depth()) 27 print(tt.get_title()) 28 print(tt)</pre>	

- a. Circle the class syntax that is new to us.
- b. Fill in the blank: `tt` is an *instance* of a _____ object.
- c. What do you think will be displayed by line 26?
- d. What do you think will be displayed by line 27?

 Line 27 displays The Thinker. How might Python know `tt`'s `_title` attribute?

-  e. What do you think `super().__init__(title)` on line 9 is doing?

What might happen if we remove the reference to `(Artwork)` on line 7?

- f. What do you think will be displayed by line 28? _____
Line 28 displays The Thinker140. What method must `print(tt)` be calling?

 How might python know which `__str__` method to execute?

2. Examine the following code below, that extends our previous code:

```
artwork.py (continued)
15 class Photograph(Artwork):
16     def __init__(self, title, cam, spd):
17         super().__init__(title)
18         self._camera = cam
19         self._speed = spd
20
21     def get_camera(self):
22         return self._camera
23
24     def get_speed(self):
25         return self._speed
```

- a. Write a line of code to create a new instance of a Photograph object:

- b. If we were to print this new Photograph object, what might the output be?

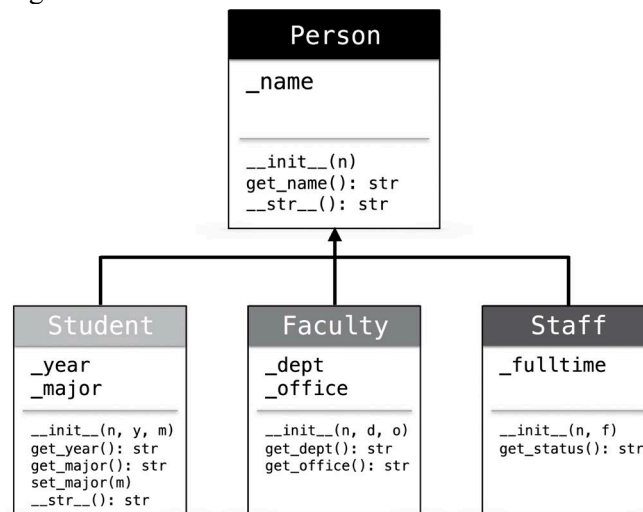
The output would be *just the Photograph object's title*. Which method is likely being called when we print this new Photograph object?

Compare your response to the response in Q1f. Why are they different?

-  c. Explain how Python chooses when to execute a method from a *child class* versus its *parent class*:

Application Questions: Use Python to check your work

1. Implement a new set of classes and sub-classes that inherit from one another, according to the following class diagram:



- a. Create a new instance object for each of the newly defined classes:
- b. Write some lines of code to use some of the shared and some of the specific-to-subclasses methods: