

CMPU 101 §02

Problem-Solving and Abstraction

Fall 2022



Hello, computer



Hello, computer

We use computers every day as electronic *black boxes* that do amazing things by

collecting,

storing,

retrieving, and

transforming *data*.

Computers only do very basic things.

Numerical calculations:

Add

Subtract

...

Symbolic manipulations

Compare two numbers

Substitute one string of letters and numbers for another

...

But when trillions of these simple operations are arranged in the right order, amazing computations can be carried out:

forecasting tomorrow's weather

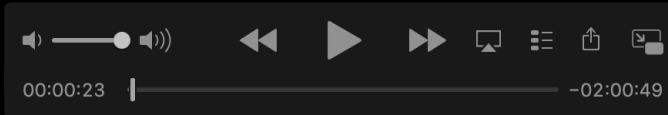
deciding where to drill for oil

finding which physical places are most likely to be visited by a person

figuring out which two people would make a great couple 🥰💕

...

A long time ago in a galaxy far,
far away.....







WIKIPEDIA
The Free Encyclopedia

Not logged in | [Talk](#) | [Contributions](#) | [Create account](#) | [Log in](#)

Article
[Talk](#)
Read
[View source](#)
[View history](#)

Computer

From Wikipedia, the free encyclopedia

For other uses, see [Computer \(disambiguation\)](#).

A **computer** is a [digital electronic machine](#) that can be programmed to *carry out sequences of arithmetic or logical operations (computation)* automatically. Modern computers can perform generic sets of operations known as [programs](#). These programs enable computers to perform a wide range of tasks. A **computer system** is a "complete" computer that includes the [hardware](#), [operating system](#) (main [software](#)), and [peripheral](#) equipment needed and used for "full" operation. This term may also refer to a group of computers that are linked and function together, such as a [computer network](#) or [computer cluster](#).

A broad range of [industrial](#) and [consumer products](#) use computers as [control systems](#). Simple special-purpose devices like [microwave ovens](#) and [remote controls](#) are included, as are factory devices like [industrial robots](#) and [computer-aided design](#), as well as general-purpose devices like [personal computers](#) and [mobile devices](#) like [smartphones](#). Computers power the [Internet](#), which links billions of other computers and users.

Early computers were meant to be used only for calculations. Simple manual instruments like the [abacus](#) have aided people in doing calculations since ancient times. Early in the [Industrial Revolution](#), some mechanical devices were built to automate long tedious tasks, such as guiding patterns for [looms](#). More sophisticated electrical [machines](#) did specialized [analog](#) calculations in the early 20th century. The first [digital](#) electronic calculating machines were developed during [World War II](#). The first [semiconductor transistors](#) in the late 1940s were followed by the [silicon](#)-based [MOSFET](#) (MOS transistor) and [monolithic integrated circuit](#) (IC) chip technologies in the late 1950s, leading to the [microprocessor](#) and the [microcomputer revolution](#) in the 1970s. The speed, power, and versatility of computers have been increasing



Computers and computing devices from different eras – clockwise from top left:
 Early vacuum tube computer ([ENIAC](#))
 Mainframe computer ([IBM System 360](#))
 Desktop computer ([IBM ThinkCentre S50](#) with monitor)
 Supercomputer ([IBM Summit](#))

Tools

- [What links here](#)
- [Related changes](#)
- [Special pages](#)
- [Permanent link](#)
- [Page information](#)
- [Cite this page](#)
- [Wikidata item](#)

Print/export

- [Download as PDF](#)
- [Printable version](#)

In other projects



The magic of a computer is its ability to become almost anything you can imagine...

...as long as you can explain *exactly* what that is.

When we program a computer to do something, everything needs to be described precisely.

Say you tell an accounting program to bill your clients the amount each owes.

Should the computer send a weekly bill for \$0 to clients who owe nothing?

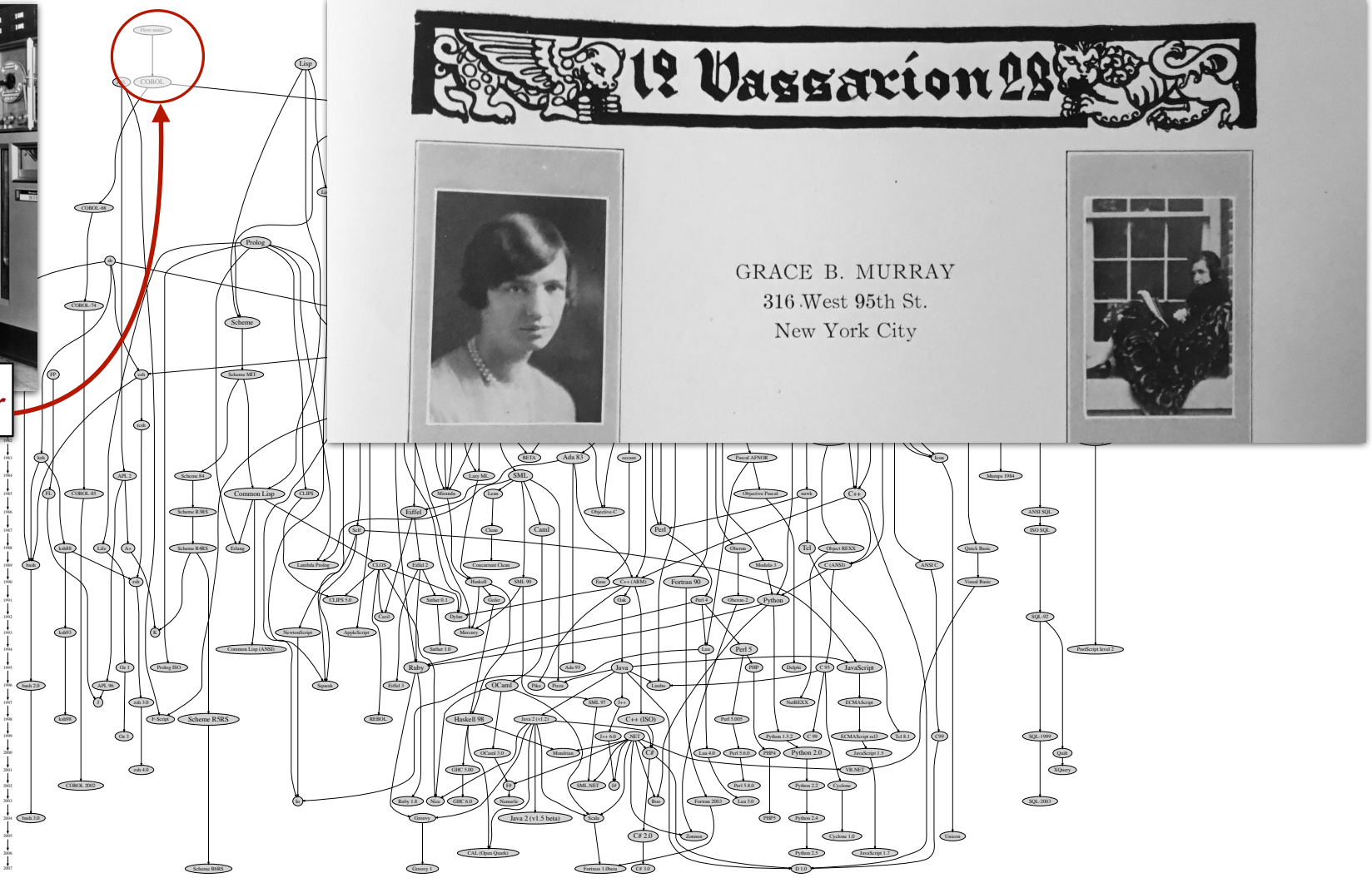
If you tell the computer to send a threatening letter to clients who haven't paid, then clients who owe nothing will receive threatening letters until they send you a payment of \$0!

When computers behave intelligently, it's because a person used *their* intelligence to design an intelligent program.

There are many programming languages we can use to write these programs.

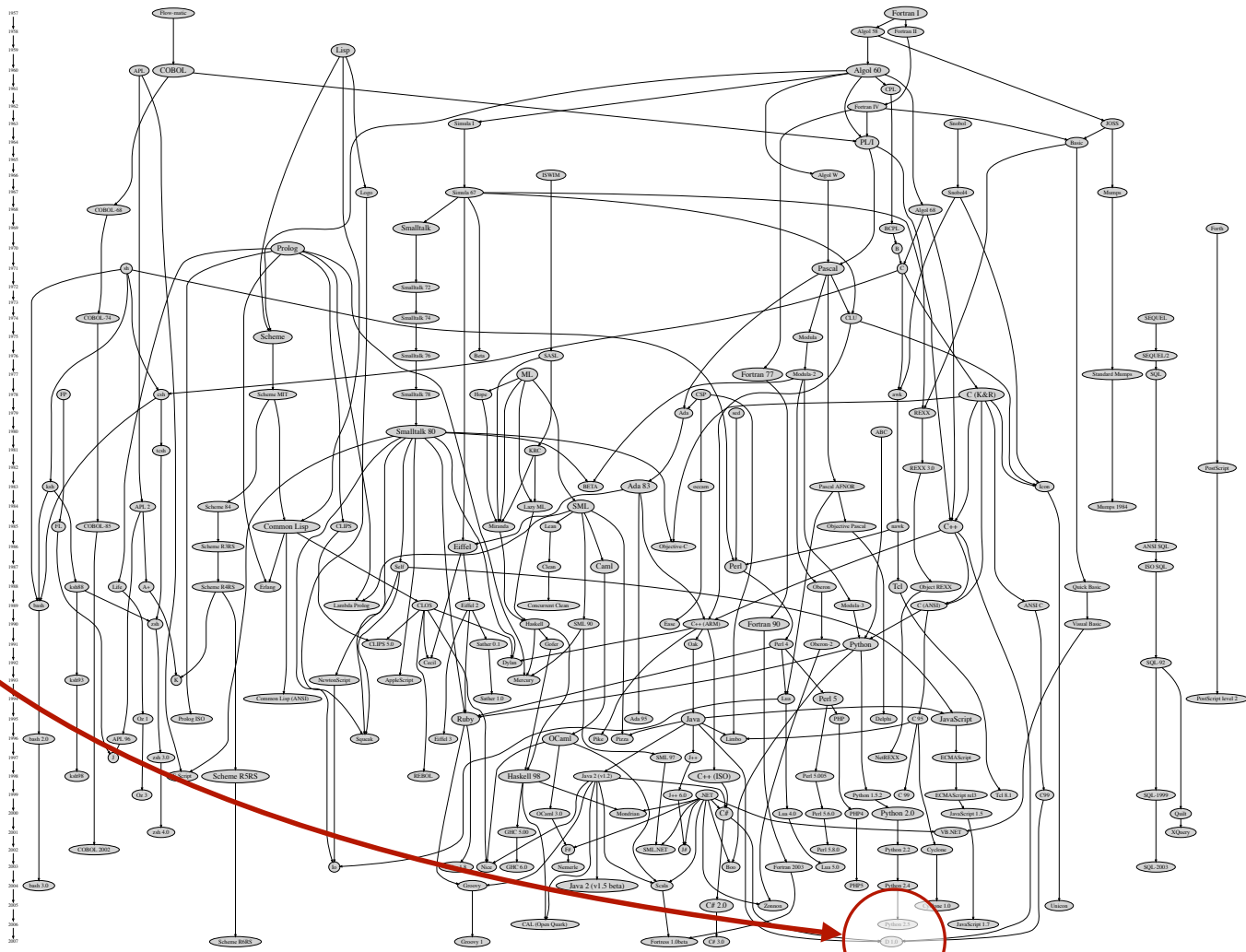


Grace Hopper



github.com/stereobooster/programming-languages-genealogical-tree

Us



github.com/stereobooster/programming-languages-genealogical-tree

There are many programming languages due to

intended use

history

habit

taste

Ancient history (my childhood)



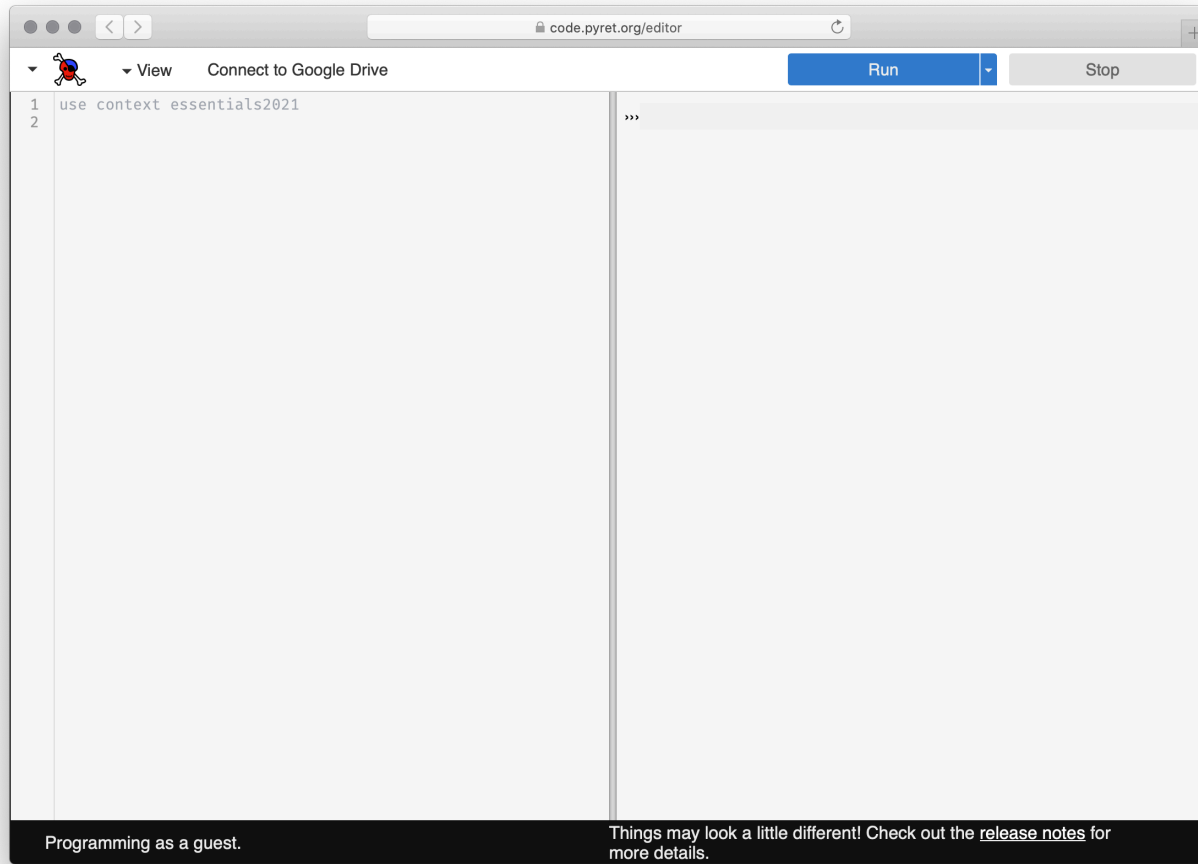
In this course, we'll be working
in two programming languages:



Why join the navy
if you can be a pirate?

famous Steve Jobs quote

—with apologies to Grace Hopper, who was actually a rear admiral in the Navy!




code.pyret.org/editor

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
```

```
>>> include image
>>> circle(30, "solid", "red")
>>> |
```



Programming as a guest.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
```

```
>>> include image
>>> circle(30, "solid", "red")
>>> circle(30, "solid", "yellow")
>>> |
```

Programming as a guest.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
```

```
>>> include image
>>> circle(30, "solid", "red")
>>> circle(30, "solid", "yellow")
>>> above(circle(30, "solid", "red"), circle(30, "solid", "yellow"))
>>>
```

Programming as a guest.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
```

```
>>> circle(30, "solid", "yellow")
>>> above(circle(30, "solid", "red"), circle(30, "solid", "yellow")
>>> above(above(circle(30, "solid", "red"), circle(30, "solid", "
```

Programming as a guest.

Drawing pictures this way is fun – but also a lot of typing.

Here's where things gets interesting:


We can define new words.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
3 g = circle(30, "solid", "green")
4 y = circle(30, "solid", "yellow")
5 r = circle(30, "solid", "red")
```

» above(above(g, y), r)



»|


Programming as a guest.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
3 fun traffic-light():
4   g = circle(30, "solid", "green")
5   y = circle(30, "solid", "yellow")
6   r = circle(30, "solid", "red")
7   above(above(g, y), r)
8 end
```

»» traffic-light()



»»

Programming as a guest.

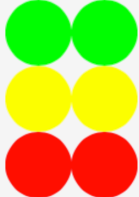
In extending the language, the programmer uses the power of functional abstraction to *create new building blocks*.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
3 fun traffic-light():
4   g = circle(30, "solid", "green")
5   y = circle(30, "solid", "yellow")
6   r = circle(30, "solid", "red")
7   above(above(g, y), r)
8 end
9
10 fun intersection():
11   beside(traffic-light(), traffic-light())
12 end
```

»» intersection()



»»|


Programming as a guest.

code.pyret.org/editor


View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
3 fun traffic-light(size):
4   g = circle(size, "solid", "green")
5   y = circle(size, "solid", "yellow")
6   r = circle(size, "solid", "red")
7   above(above(g, y), r)
8 end
9
10 fun intersection():
11   beside(traffic-light(), traffic-light())
12 end
```

» traffic-light(10)



» traffic-light(60)



»|

Programming as a guest.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
3 fun traffic-light(size):
4   g = circle(size, "solid", "green")
5   y = circle(size, "solid", "yellow")
6   r = circle(size, "solid", "red")
7   above(above(g, y), r)
8 end
9
10 fun intersection():
11   beside(traffic-light(), traffic-light())
12 end
```

... intersection()

This application expression errored:

definitions://:10:9-10:24

```
11 beside(traffic-light(), traffic-light())
```

9 arguments were passed to the operator.

The operator evaluated to a function defined to accept 1 parameter:

definitions://:2:0-7:3

```
3 fun traffic-light(size):
4   g = circle(size, "solid", "green")
5   y = circle(size, "solid", "yellow")
6   r = circle(size, "solid", "red")
7   above(above(g, y), r)
8 end
```

An application expression expects the number of parameters and arguments to be the same.

(Show program evaluation trace...)


...>>>

Programming as a guest.

code.pyret.org/editor

View Connect to Google Drive Run Stop

```
1 use context essentials2021
2
3 fun traffic-light(size):
4   g = circle(size, "solid", "green")
5   y = circle(size, "solid", "yellow")
6   r = circle(size, "solid", "red")
7   above(above(g, y), r)
8 end
9
10 fun intersection(size):
11   beside(traffic-light(size), traffic-light(size))
12 end
```

```
>>> intersection(10)

>>> |
```

Programming as a guest.

What will we do in this course?

- 1 Identify and organize the data needed to solve a problem *Data design*
- 2 Break a problem down into subproblems that can be solved with computations *Programming*
- 3 Express computations over the data *Programming/CS*
- 4 Test those computations to make sure they're doing what they're supposed to *Testing*

- 0 Think about whether it's a good idea to solve the problem, and how your solution might affect the world around you.

This course teaches skills that will help you both in computer science and beyond.

Three major units:

Tabular / data science data

Other core CS data structures and how to program with them

Additional data structures and programming techniques

| *Pyret*

| *Python*

Goals

Apply fundamental data-organizations (called data structures) to capture the information in a computing problem.

Break down a computing question into manageable smaller problems.

Write programs to compute answers to questions over fundamental data structures.

Check whether your programs behave as intended/required.

Course information

Class:

Monday & Wednesday, 9:00–10:15 a.m.

Sanders Classroom 006

Lab:

Friday, 9:00–11:00 a.m.

Sanders Classroom 006

The screenshot shows a web browser window with the URL <https://www.cs.vassar.edu/courses/cs101-2022-02/top>. The page is titled "COMPUTER SCIENCE | VASSAR COLLEGE" and features a search bar. The main content area is titled "CS1: Problem-Solving and Abstraction" and includes the following information:

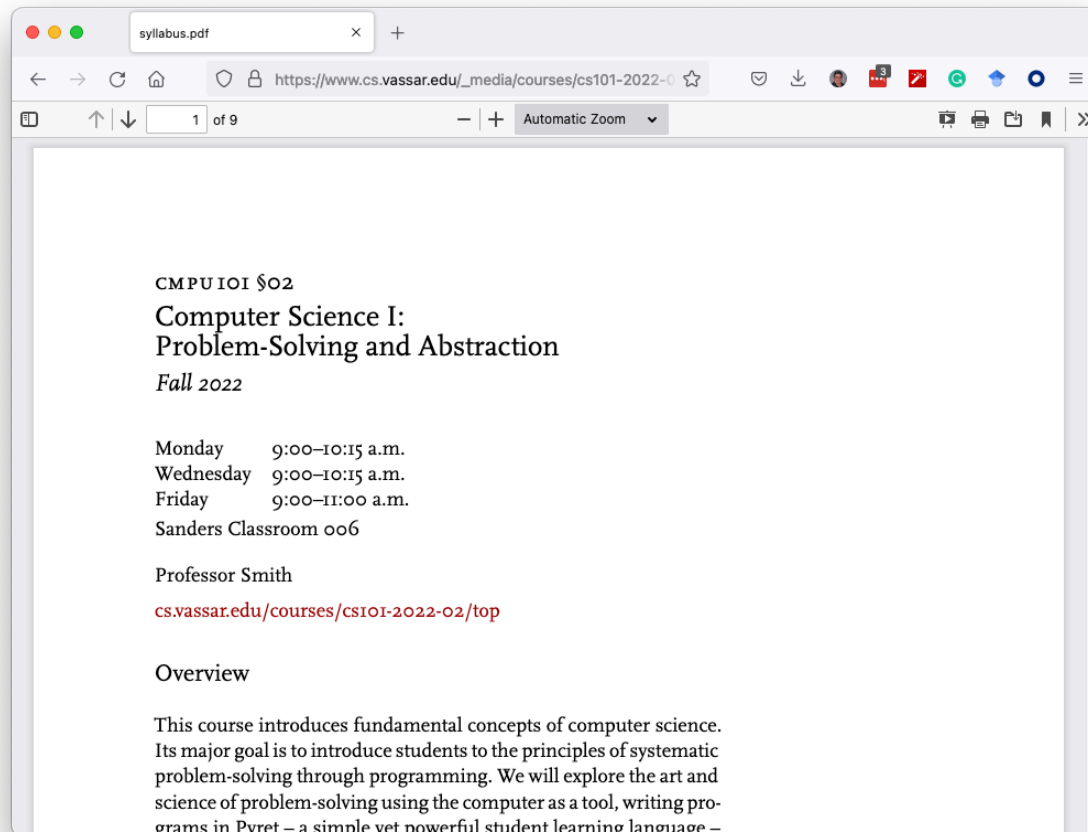
- CMPU-101 Section 02**
Vassar College, Fall 2022
Syllabus / Course Wiki
- Welcome to our course wiki and syllabus. It will be updated throughout the semester with important course information, so check here regularly.
- Contact Information**

Professor:	Marc Smith
Office:	SP 104.5
Virtual:	Zoom
Phone:	845 437 7497
Email:	mlsmith (best way to contact me!)
- Course Coordinates**

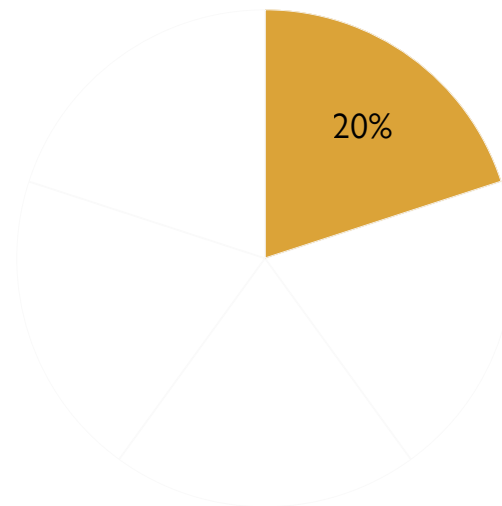
Lectures:	Mon/Wed 9:00-10:15am in SC 006
Labs:	Fri 9am-11am in SC 006
Office Hours:	tbd; and by appointment
Wiki:	https://www.cs.vassar.edu/courses/cs101-2022-02/top
Gradescope:	https://www.gradescope.com/

The left sidebar contains navigation links for "Top", "Syllabus", "Schedule", "Coaching Hours", "Remote Access", and "Resources". The "Resources" section lists "DCIC (Text)", "Table Functions", "CPO (Pyret)", and "Pyret Style Guide". The footer of the page displays the "Computer Science | Vassar College" logo.

<https://www.cs.vassar.edu/courses/cs101-2022-02/top>

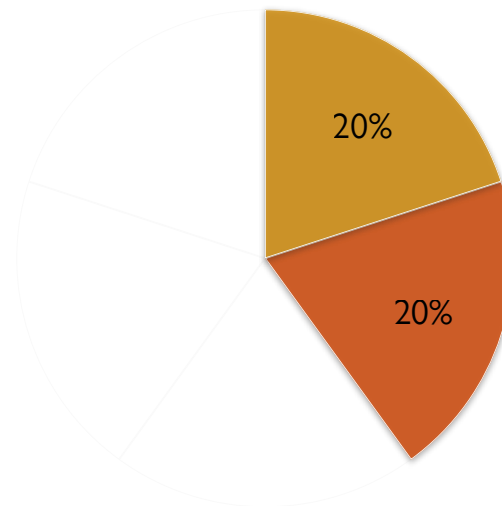


Grading



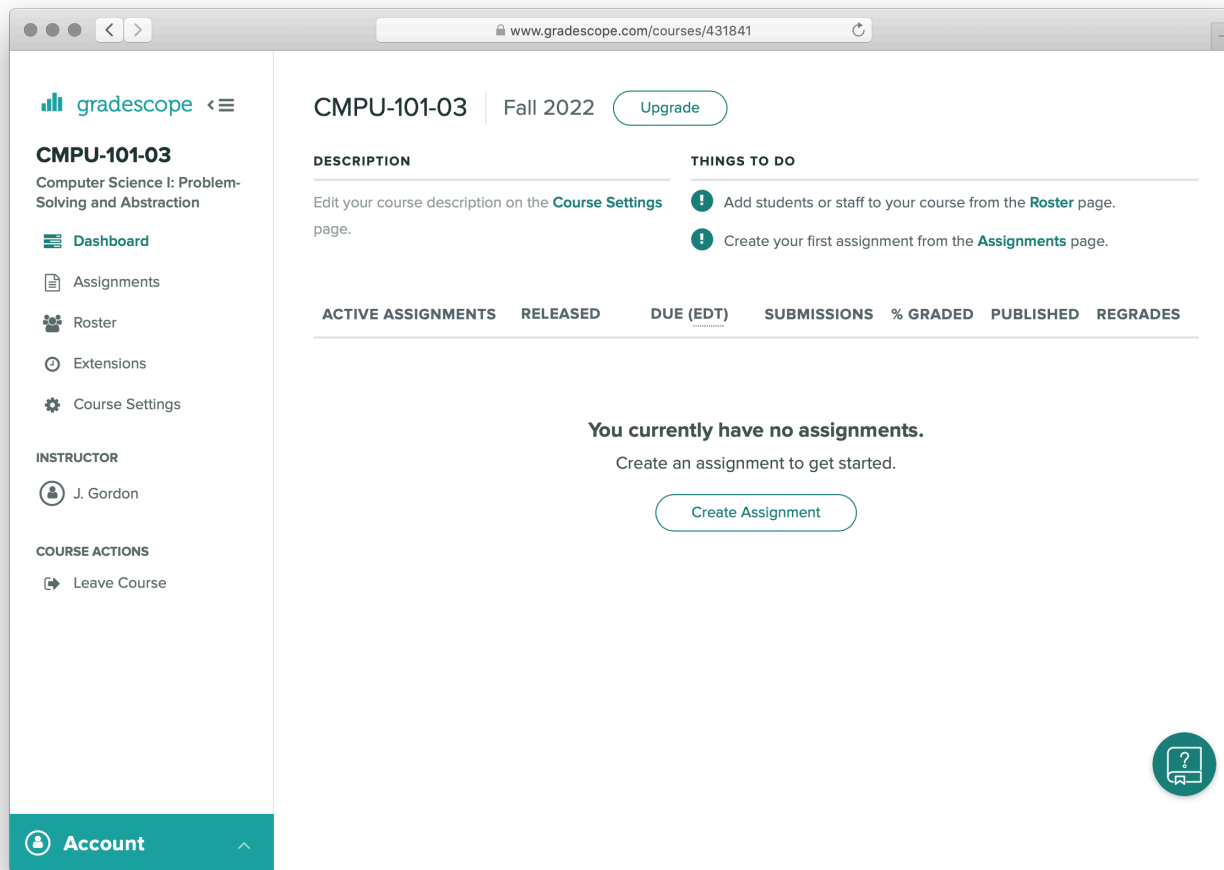
Weekly lab exercises

Grading



- Labs
- Assignments
-
-
-

Weekly programming assignments



gradescope.com

The image shows a browser window displaying the Vassar CS Student Integrity Guide. The browser's address bar shows the URL www.cs.vassar.edu/integrity. The page header includes the Vassar College logo and the text "COMPUTER SCIENCE | VASSAR COLLEGE". A search bar and navigation icons are located in the top right. On the left side, there is a sidebar menu with links: CS Wiki Home, Courses, Add policy, Dep Graph, CS Integrity Guide, Course Galleries, Events, History, People, Students, Resources, FAQs, Interactive Tree Idx, and Dropboxes, etc.. The main content area features the title "Vassar CS Student Integrity Guide" and an introductory paragraph explaining the guide's purpose. Below this, there are two paragraphs discussing the goal of computer science courses and the importance of understanding over competition. The page is structured with a "1. Policy" section, which includes a sub-section "1.1. Guidelines for individual work". A "Table of Contents" sidebar on the right lists the page's sections. Edit buttons are visible at the end of several sections.

COMPUTER SCIENCE | VASSAR COLLEGE

Search

CS Wiki Home
Courses
Add policy
Dep Graph
CS Integrity Guide
Course Galleries
Events
History
People
Students
Resources
FAQs
Interactive Tree Idx

Dropboxes, etc..

Vassar CS Student Integrity Guide

This guide is designed to clarify [Vassar College's academic integrity policy](#) as it applies to the Computer Science Department. Furthermore, it provides advice on how to best navigate integrity issues in the context of the field, where source code authorship is a central issue.

The goal of our computer science courses is to promote understanding of the field, not competition among students. As such, students are encouraged to discuss class material, ideas, sample exercises, etc., with other students.

However, when it comes to graded work (e.g., programming assignments, programming labs, take-home exams), it is important to know when to collaborate and when to work individually. Taking shortcuts, while seemingly beneficial in the short term, will inevitably backfire later on. Conversely, the challenges of working through a problem will pay off greatly in future courses and postgraduate life, as they will enable students to be more independent in their work.

1. Policy

1.1. Guidelines for individual work

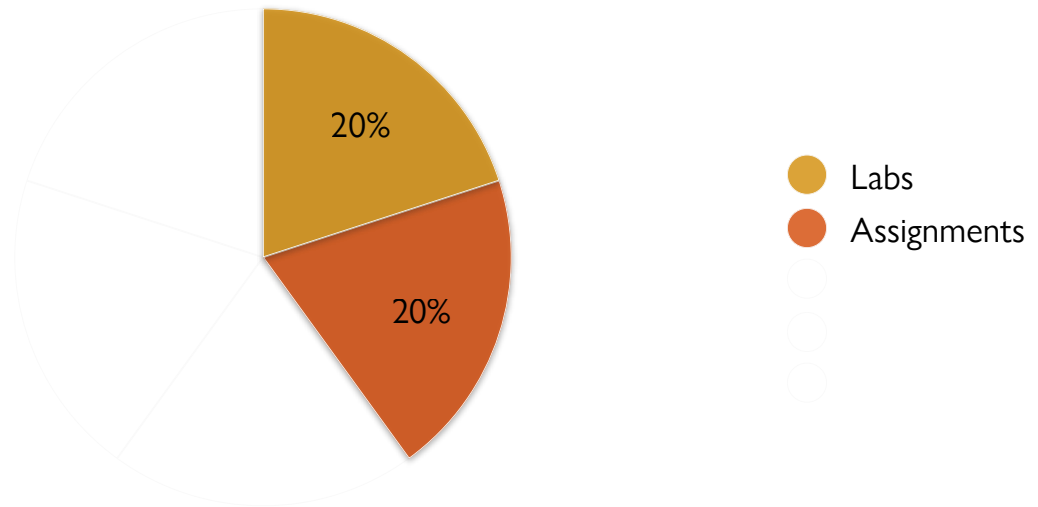
The goal of individual work is to assess the learning of each person in isolation. The guidelines are the following:

Table of Contents

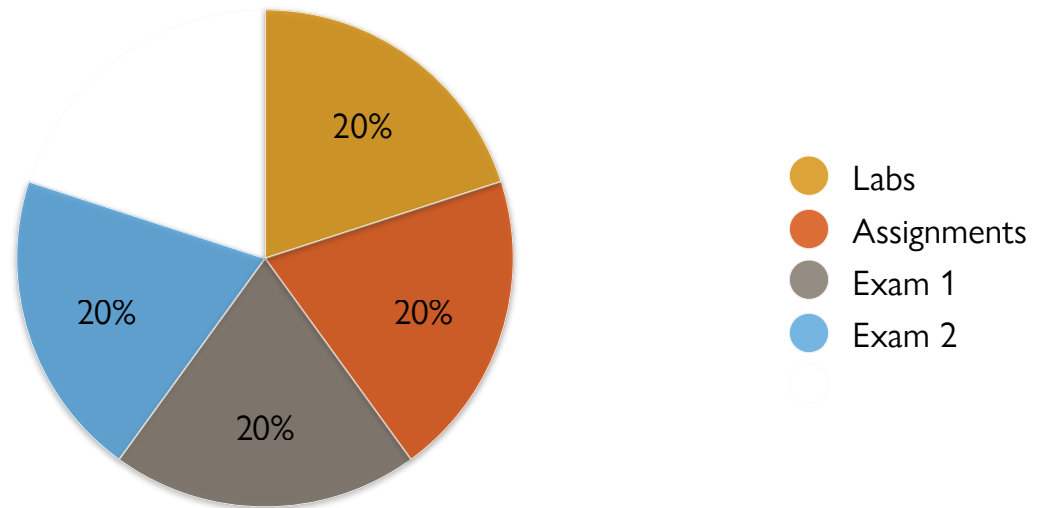
- Vassar CS Student Integrity Guide
- 1. Policy
 - 1.1. Guidelines for individual work
 - 1.2. Guidelines for partnered/group work
- 2. Frequently Asked Questions (FAQ)
- 3. Cautionary tales
 - 3.1. Past examples of academic integrity violations
 - 3.2. Statistics
- 4. Other helpful resources

www.cs.vassar.edu/integrity

Grading

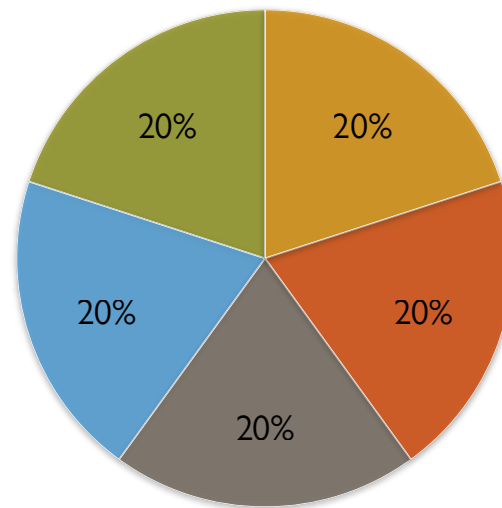


Grading



Two midterm exams

Grading



- Labs
- Assignments
- Exam 1
- Exam 2
- Exam 3

Regularly scheduled final exam

“All through our education, we are being taught a kind of reverse mindfulness. A kind of Future Studies where – via the guise of mathematics, or literature, or history, or computer programming, or French – we are being taught to think of a time different to the time we are in. Exam time. Job time. When-we-are-grown-up time.

To see the act of learning as something not for its own sake but because of what it will get you reduces the wonder of humanity. We are thinking, feeling, art-making, knowledge-hungry, marvelous animals, who understand ourselves and our world through the act of learning. It is an end in itself. It has far more to offer than the things it lets us write on application forms. It is a way to love living right now.”

Matt Haig, *Notes on a Nervous Planet*

So...

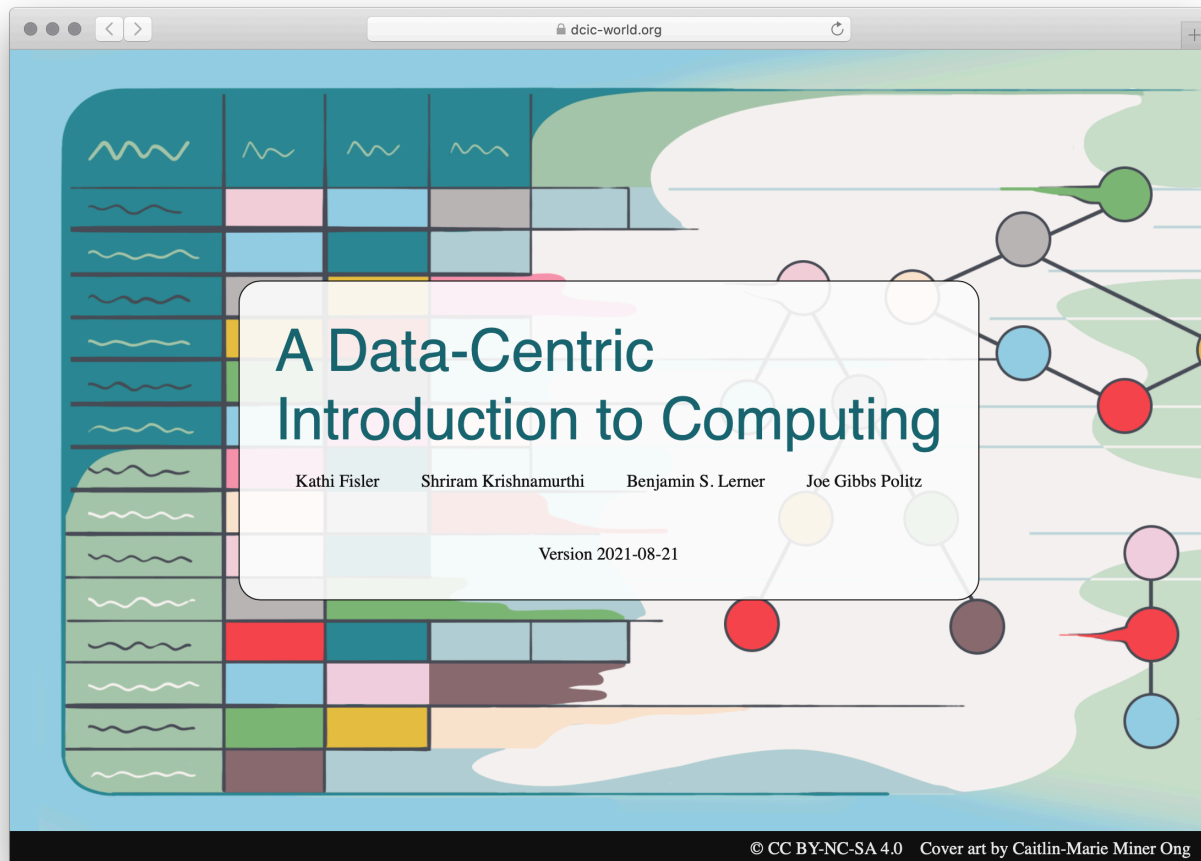
Don't just focus on grades.

Don't just focus on what happens if you get an A in the course.

Be here now.

If you worry about whether you understand what we're doing in class, in lab, and on the assignments, your grades will take care of themselves.

Trust me.



dcic-world.org



We've got a big journey ahead of us. I hope you're excited!

Acknowledgments

This class incorporates material from:

Peter J. Denning and Matti Tedre, *Computational Thinking*

Kathi Fisler and Doug Woos, Brown University

W. Daniel Hillis, *The Pattern on the Stone*