CMPU 101 § 53

Problem-Solving and Abstraction

Spring 2024

Hello, computer


Hello, computer

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

collecting,

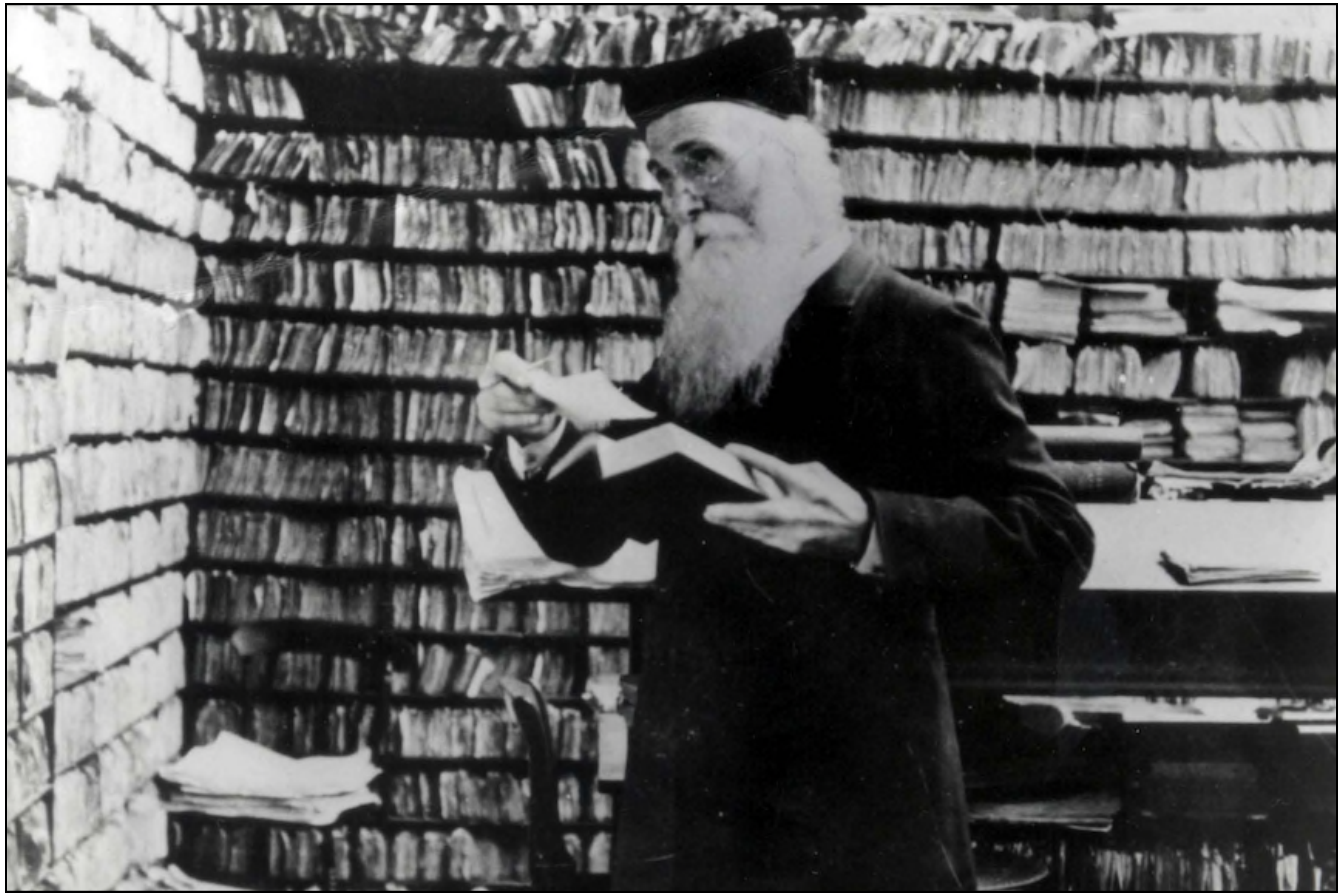
storing,

retrieving, and

transforming *data*.

“Many people think of data as numbers alone, but data can also consist of words or stories, colors or sounds, or any type of information that is systematically collected, organized, and analyzed…”

D’Ignazio & Klein, *Data Feminism*, 2020



James Murray compiling the *Oxford English Dictionary*, c. 1928.

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 places a person’s most likely to visit 🚗
- figuring out who would make a great couple 😘😍
- …


Star Wars
A long time ago in a galaxy far,
far away. . . .
00:00:23
-02:00:49


Compare
Copy
Paste
Undo
Redo
View:
109%
PRESETS
STORE
UNDO
REDO
KEYS
ANIMOOG
ARP
RIBBON
907A
OSCILLATOR DRIVER
OSCILLATOR
OSCILLATOR
904A
SEMITONE
OCTAVE
WIDTH OF RECTANGULAR WAVE
CONTROL INPUTS
FREQUENCY
WIDTH
RANGE
SYNCH. WEAK–STRONG
SINE
SYNCH. IN
TRIANGULAR
A.C. MODULATE
SAWTOOTH
D.C. MODULATE
RECTANGULAR
WHITE
PINK
FREQUENCY
RANGE
RECTANGULAR WIDTH
AUX. OUT. WAVEFORM
CLAMPING POINT
AUX. OUT. LEVEL
FREQUENCY CONTROL INPUTS
AUXILIARY OUTPUTS
VOLUME
GLIDE
PITCH
MOD
HOLD
ARP
OFF
ON
Model 15

Computer science

Contents hide

From Wikipedia, the free encyclopedia

For other uses, see Computer science (disambiguation).

Computer science is the study of computation, information, and automation.[1][2][3] Computer science spans theoretical disciplines (such as algorithms, theory of computation, and information theory) to applied disciplines (including the design and implementation of hardware and software).[4][5][6] Though more often considered an academic discipline, computer science is closely related to computer programming.[7]

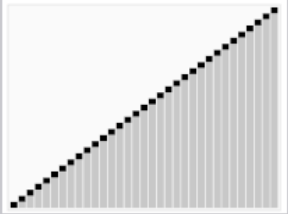
Fundamental areas of computer science

$0 := \lambda f.\lambda x.x$

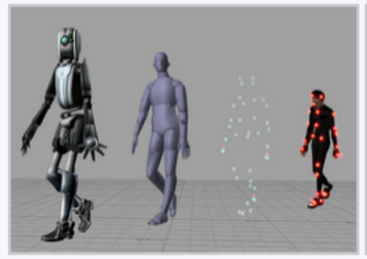
$1 := \lambda f.\lambda x.f\ x$

$2 := \lambda f.\lambda x.f\ (f\ x)$

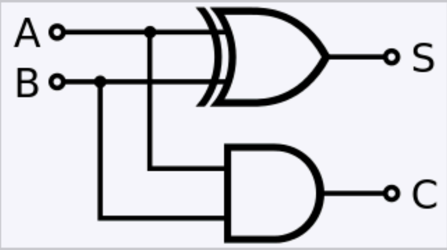
Programming language theory



Computational complexity theory



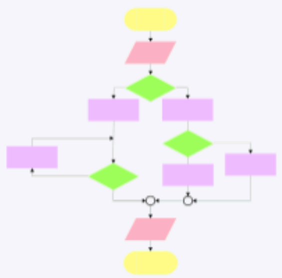
Artificial intelligence



Computer architecture

Algorithms and data structures are central to computer science.[8] The theory of computation concerns abstract models of computation and general classes of problems that can be solved using them. The fields of cryptography and computer security involve studying the means for secure communication and for preventing security vulnerabilities. Computer graphics and computational geometry address the generation of images. Programming language theory considers different ways to describe computational processes, and database theory concerns the management of repositories of data. Human–computer interaction investigates the interfaces through which humans and computers interact, and software engineering focuses on the design and principles behind developing software. Areas such as operating systems, networks and embedded systems investigate the principles and design behind complex systems. Computer architecture describes the construction of computer components and computer-operated equipment. Artificial intelligence and machine learning aim to synthesize goal-orientated processes such as problem-solving, decision-making, environmental adaptation, planning and learning found in humans and animals. Within artificial intelligence, computer vision aims to understand and process image and video data, while natural language processing aims to understand and process textual and linguistic data.

Computer science



History
Outline
Glossary
Category

V • T • E

The fundamental concern of computer science is determining what can and cannot be automated.[2][9][3][10][11] The Turing Award is generally recognized as the highest distinction in computer science.[12][13]

History

Main article: History of computer science

The earliest foundations of what would become computer science predate the invention of the modern


J. Gordon - Computer (Black to Move) — Edited
8
7
6
5
4
3
2
1
A B C D E F G H

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

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

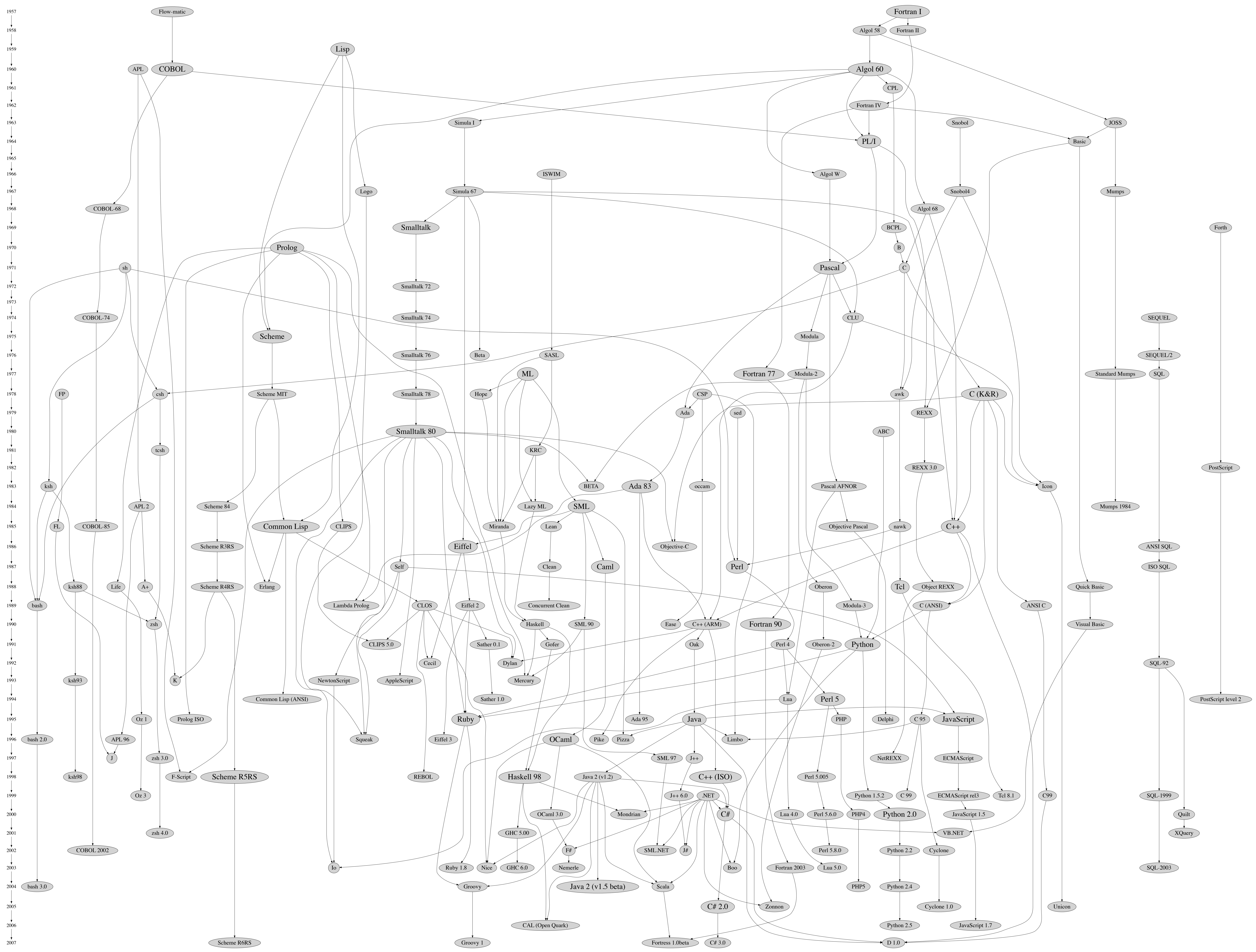
…as long as you can explain *exactly* what that is.

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

eblong.com/zarf/zero-bill.html

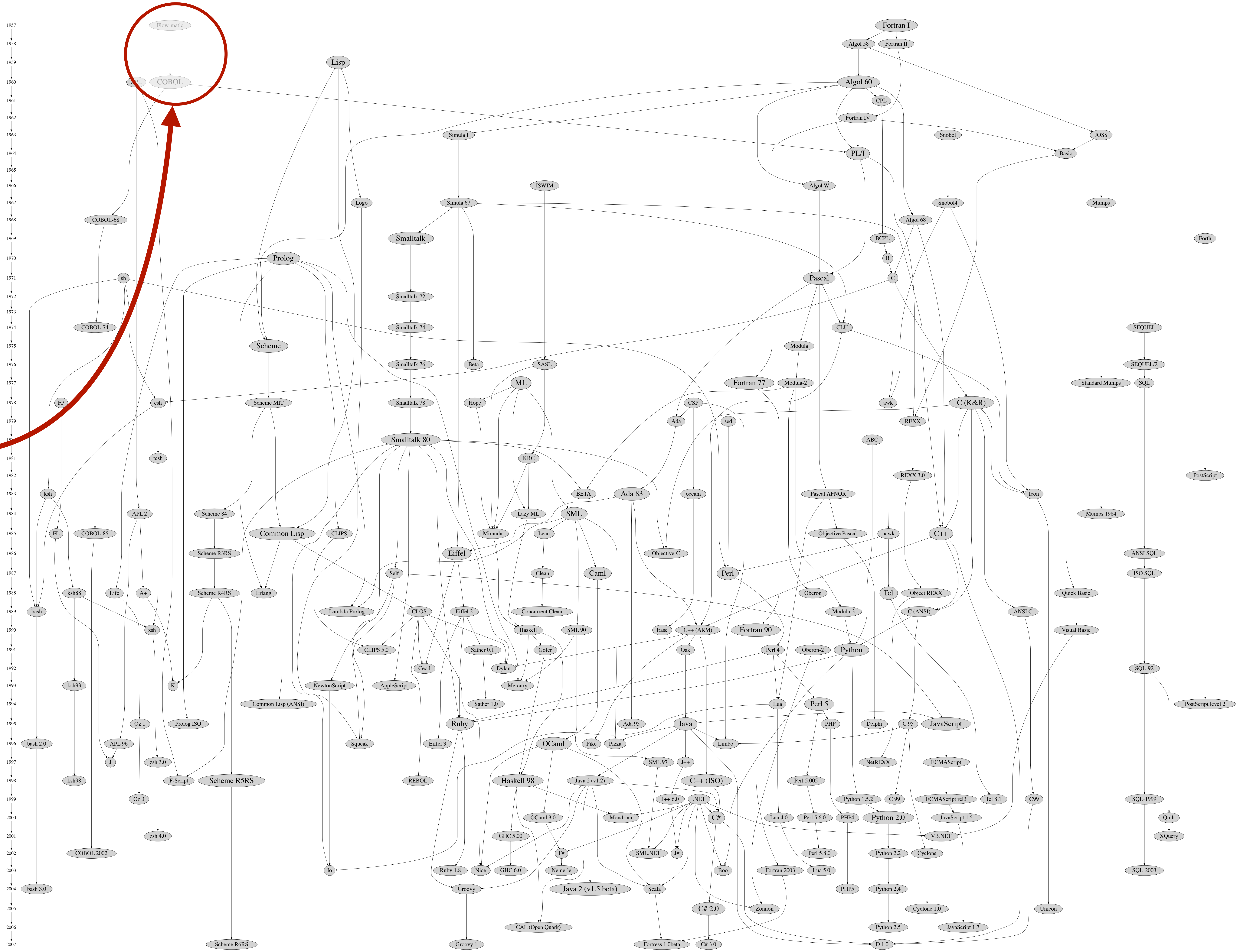
When computers behave intelligently, it's because a person used *their* intelligence to design an intelligent program.

To tell the computer exactly how to behave, we give it instructions using a ***programming language***.



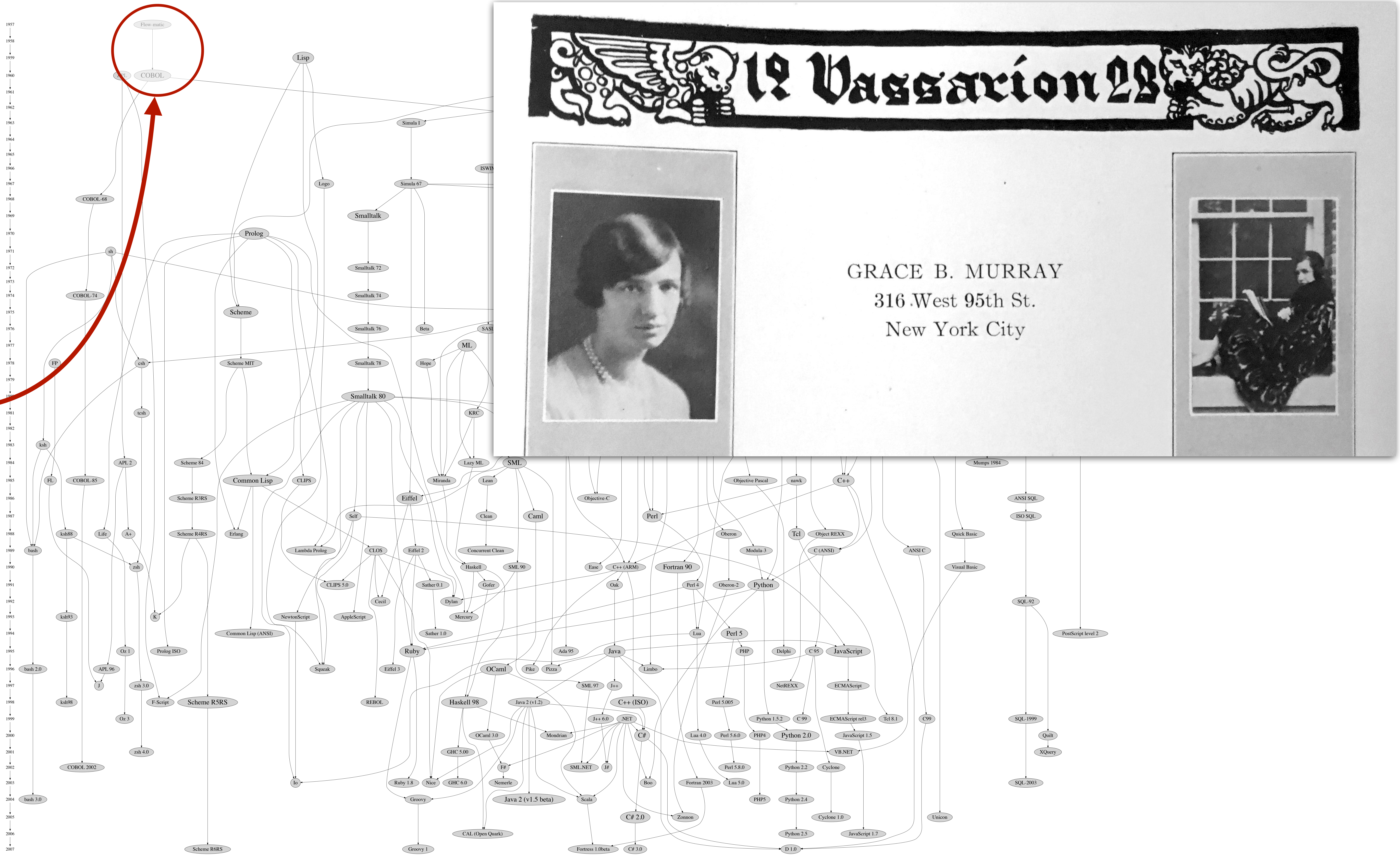
github.com/stereobooster/programming-languages-genealogical-tree

Grace Hopper

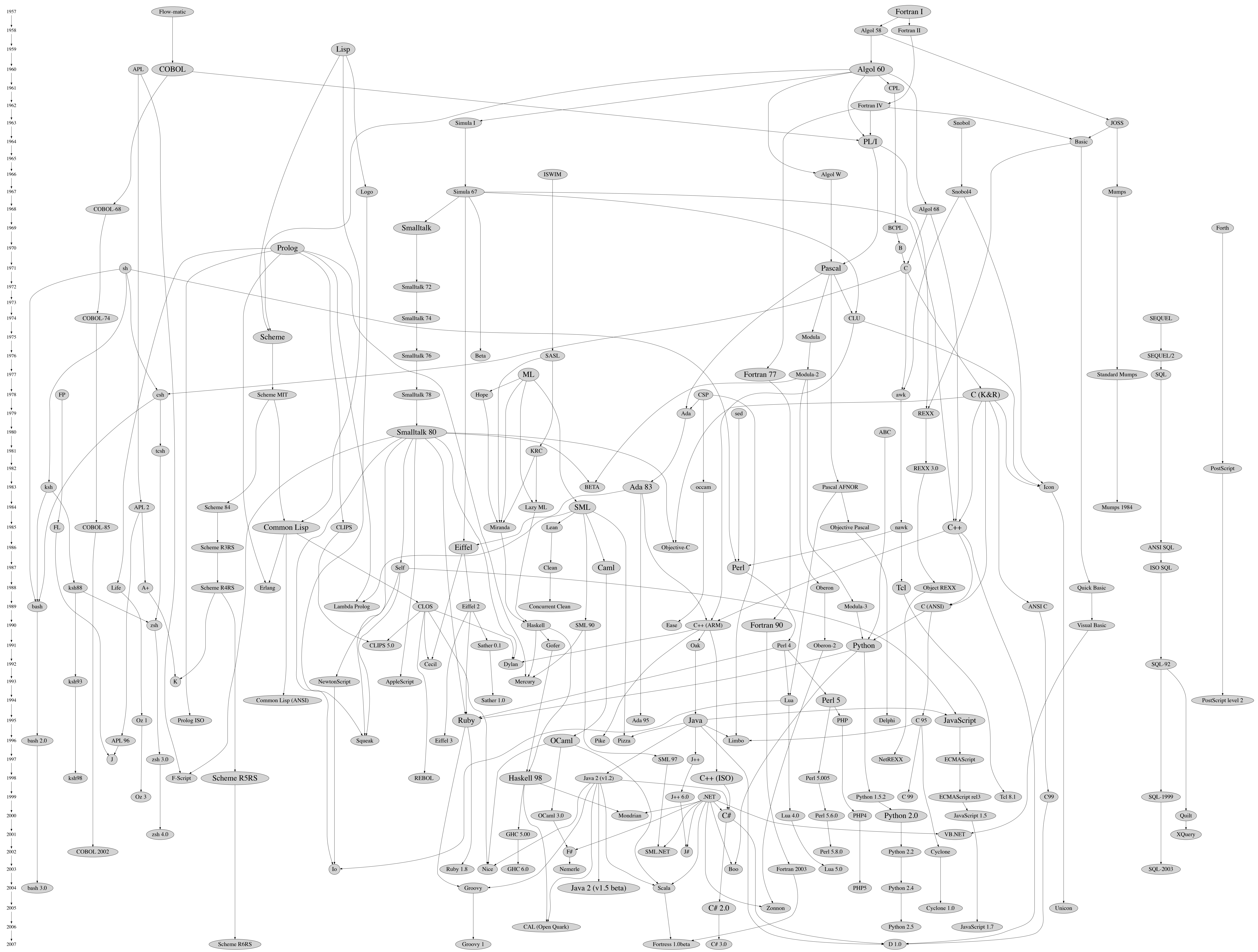


github.com/stereobooster/programming-languages-genealogical-tree

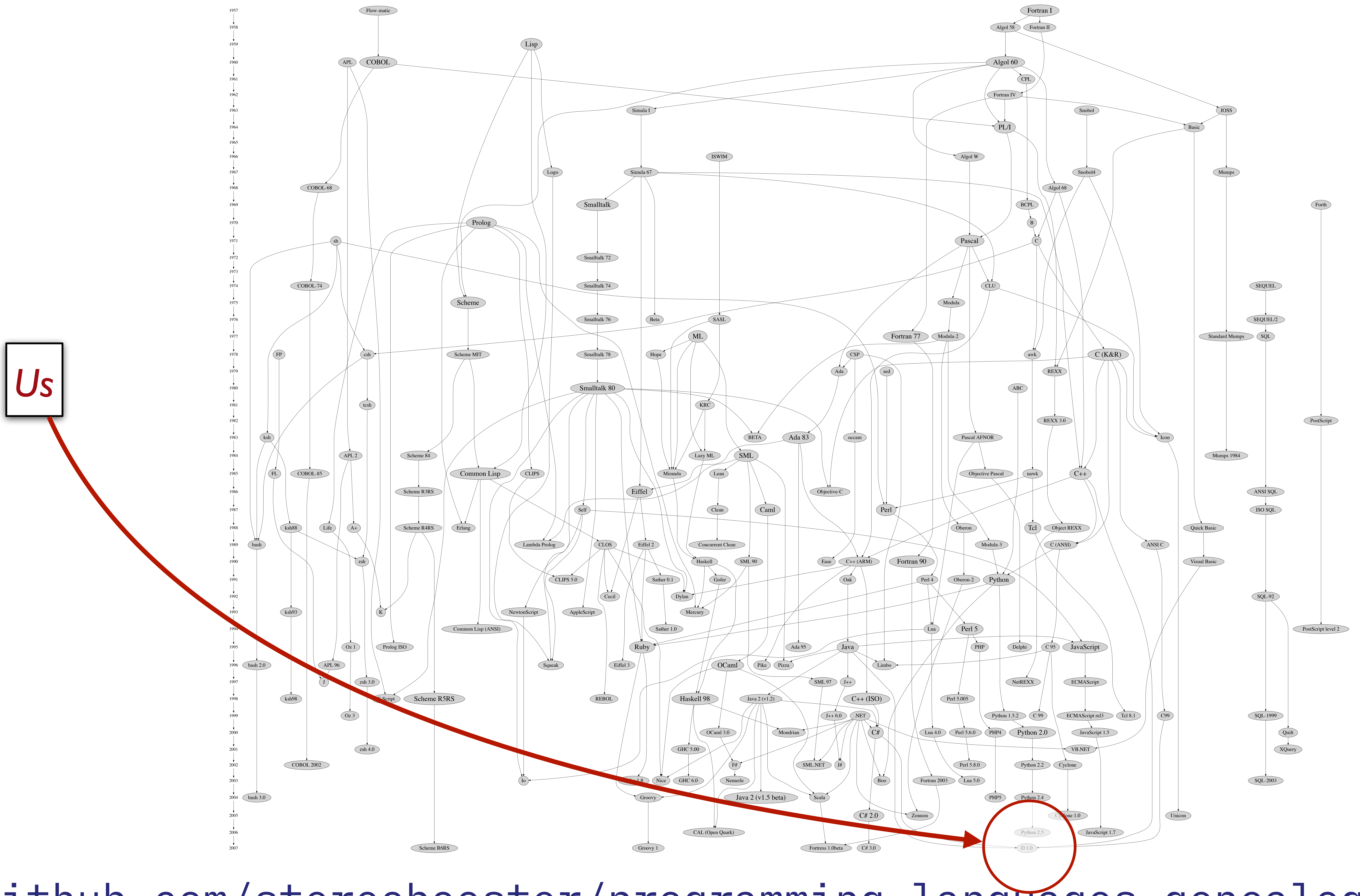
Grace Hopper



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



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



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

There are many programming languages due to

- intended use
- history
- habit
- taste

Ancient history (my childhood)

Ancient history (my childhood)

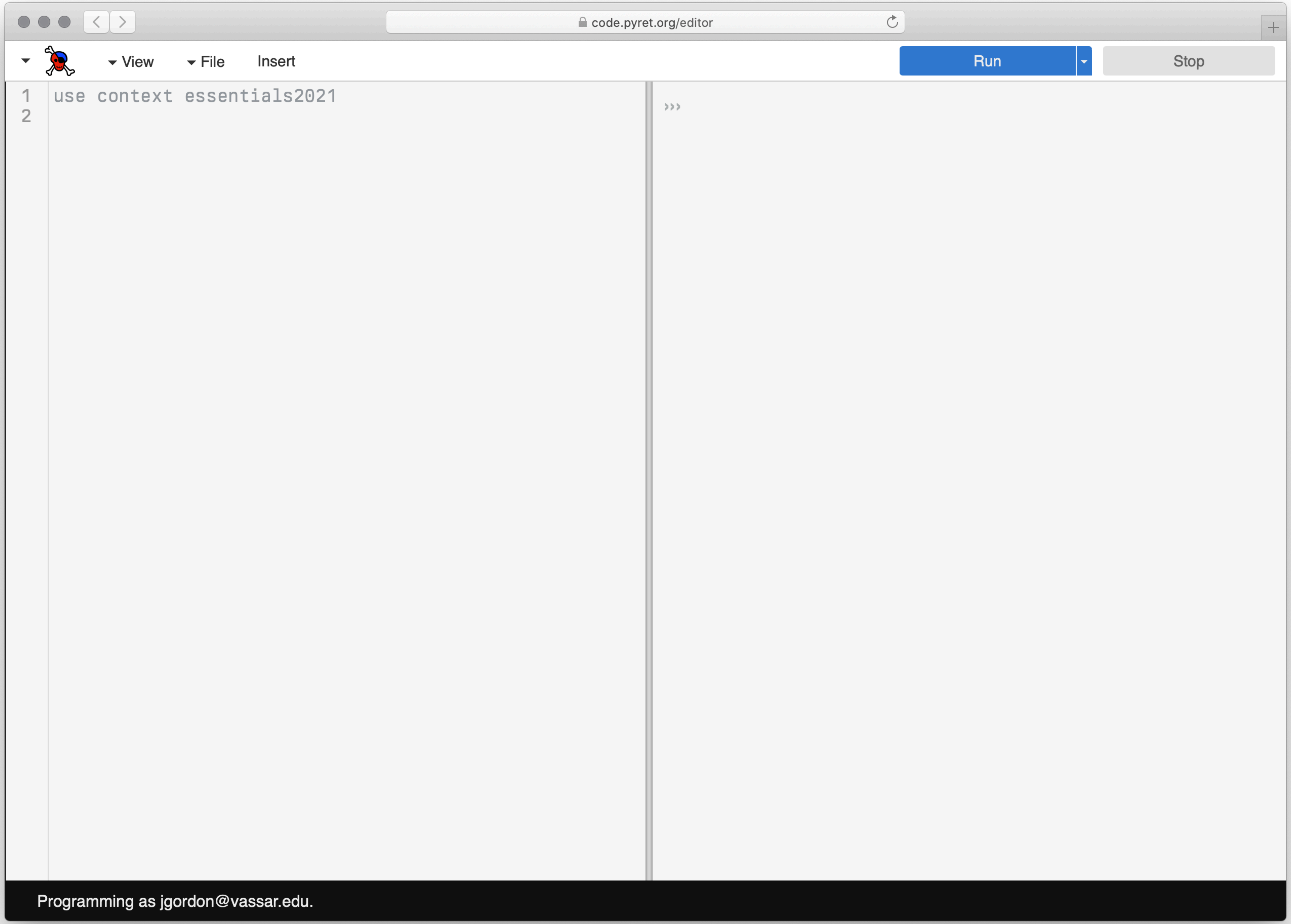
In this course, we’ll be working in two programming languages:

Pyret

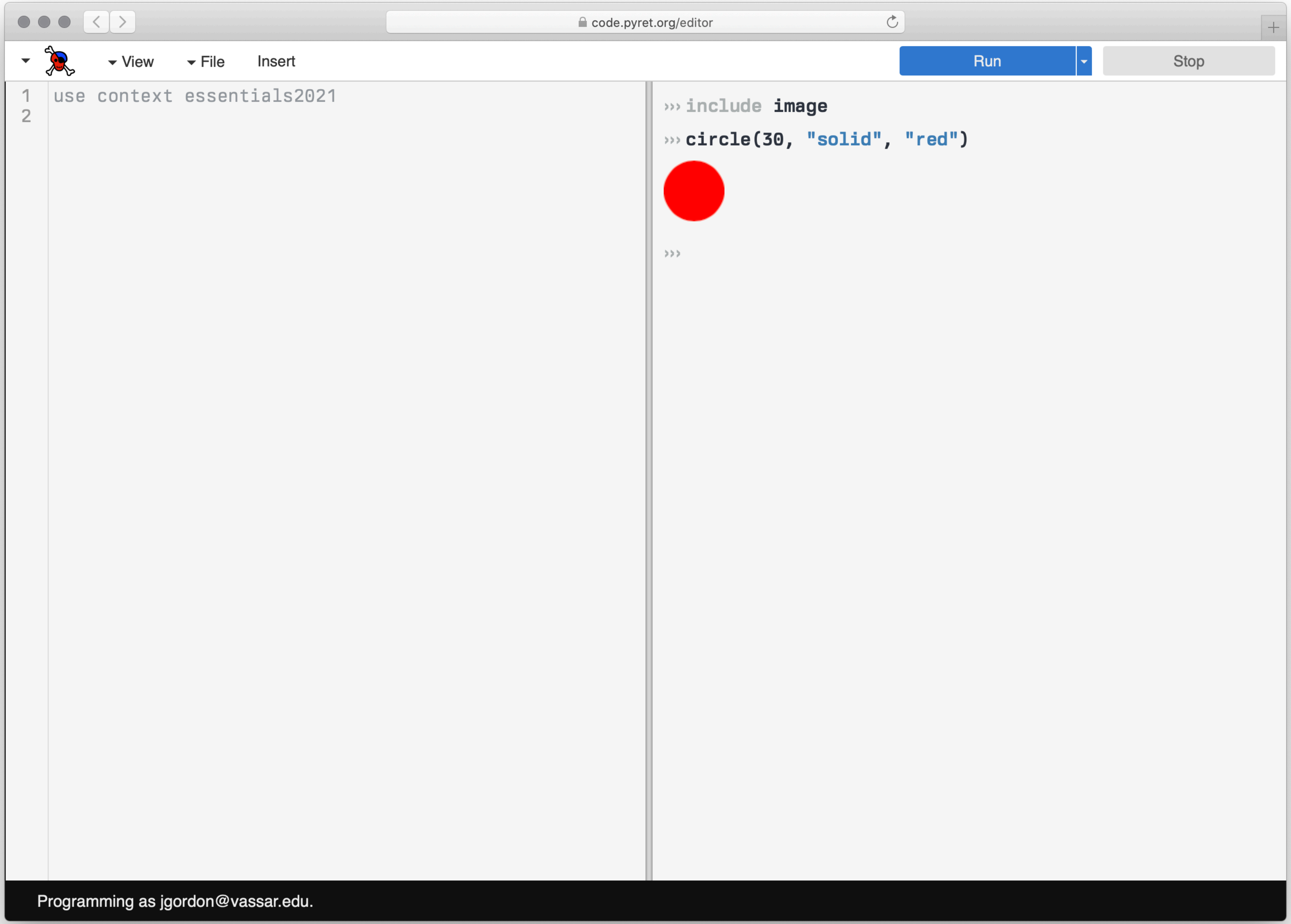
In this course, we'll be working in two programming languages:

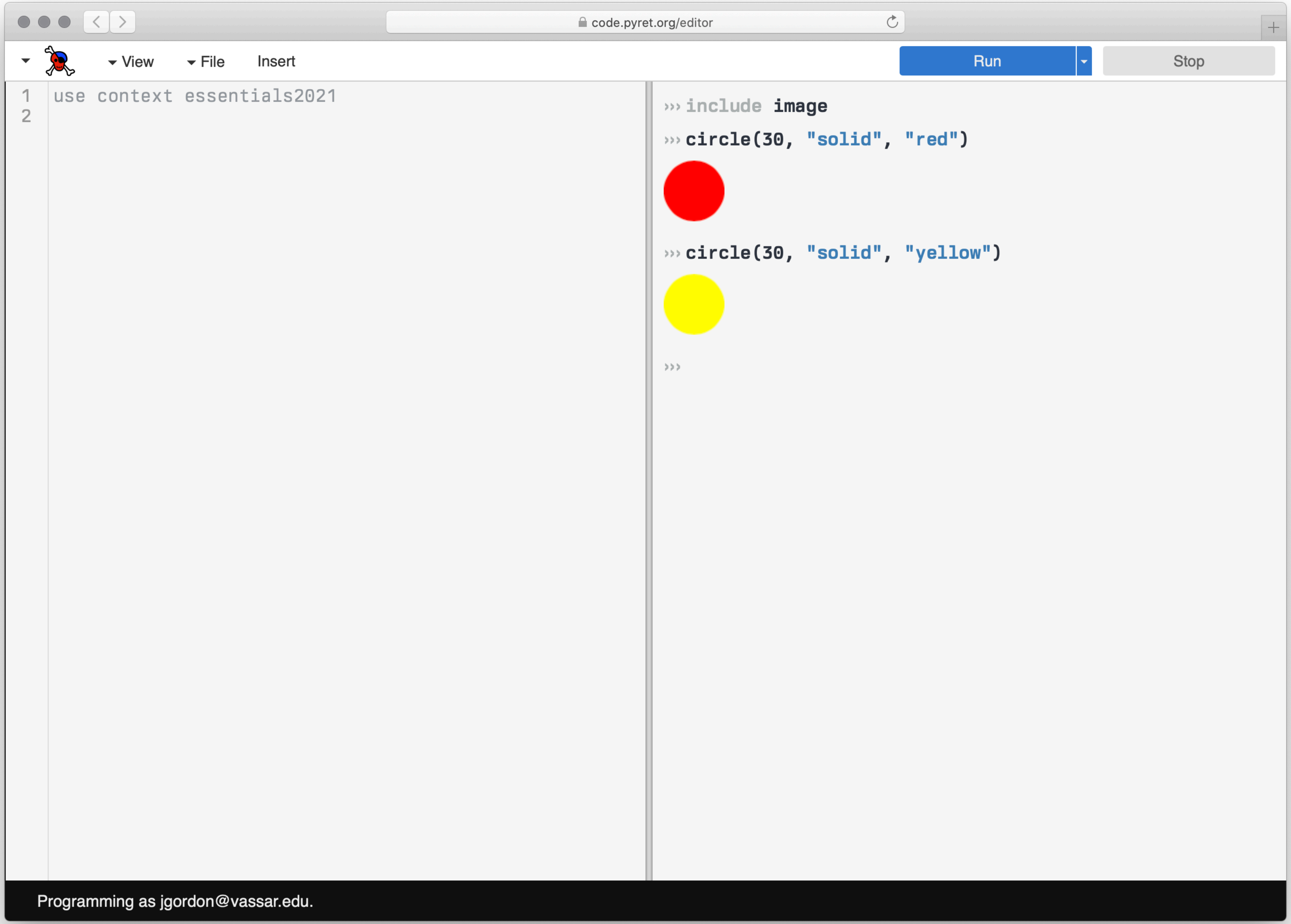
In this course, we'll be working in two programming languages:

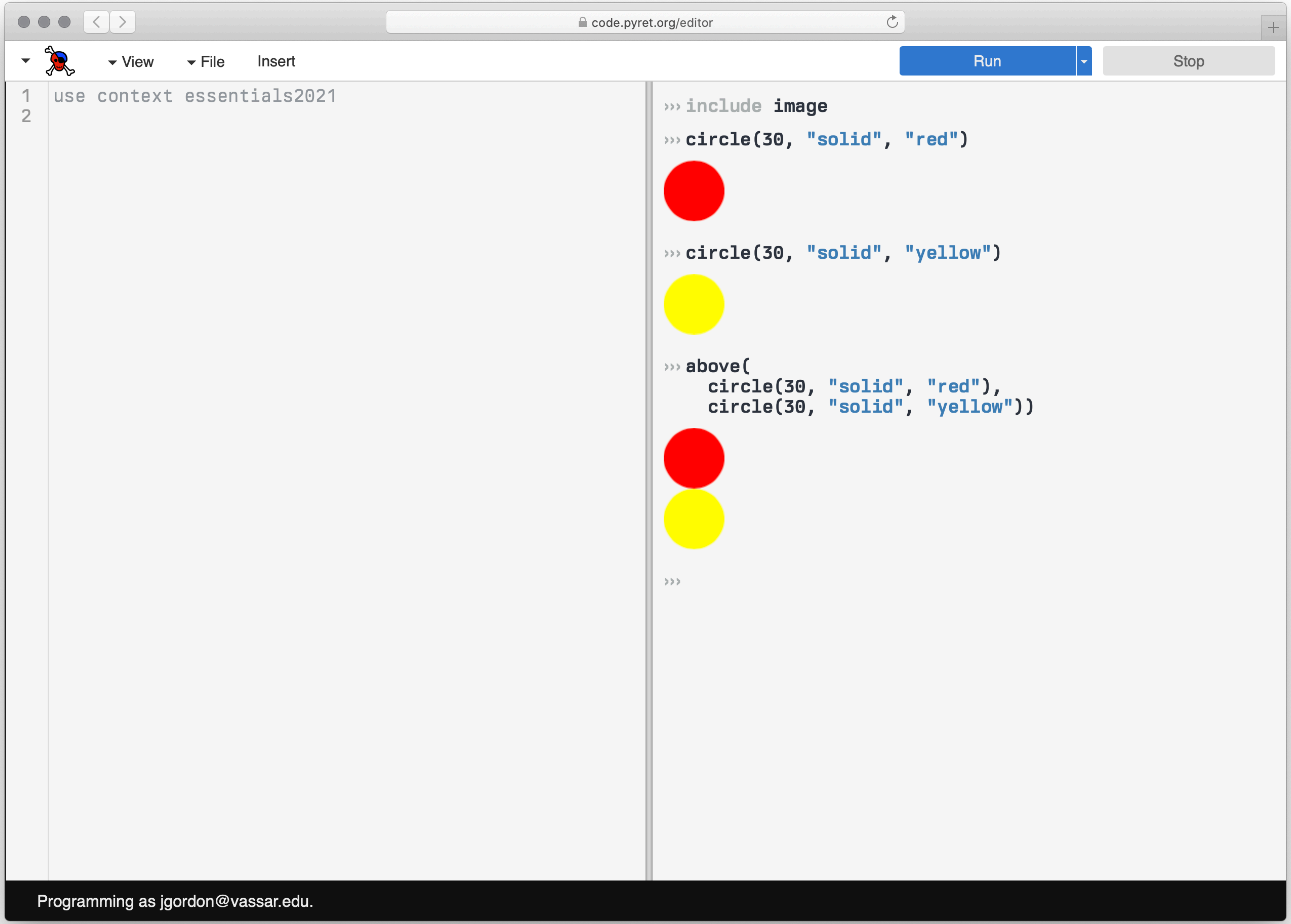
Why join the navy
if you can be a pirate?

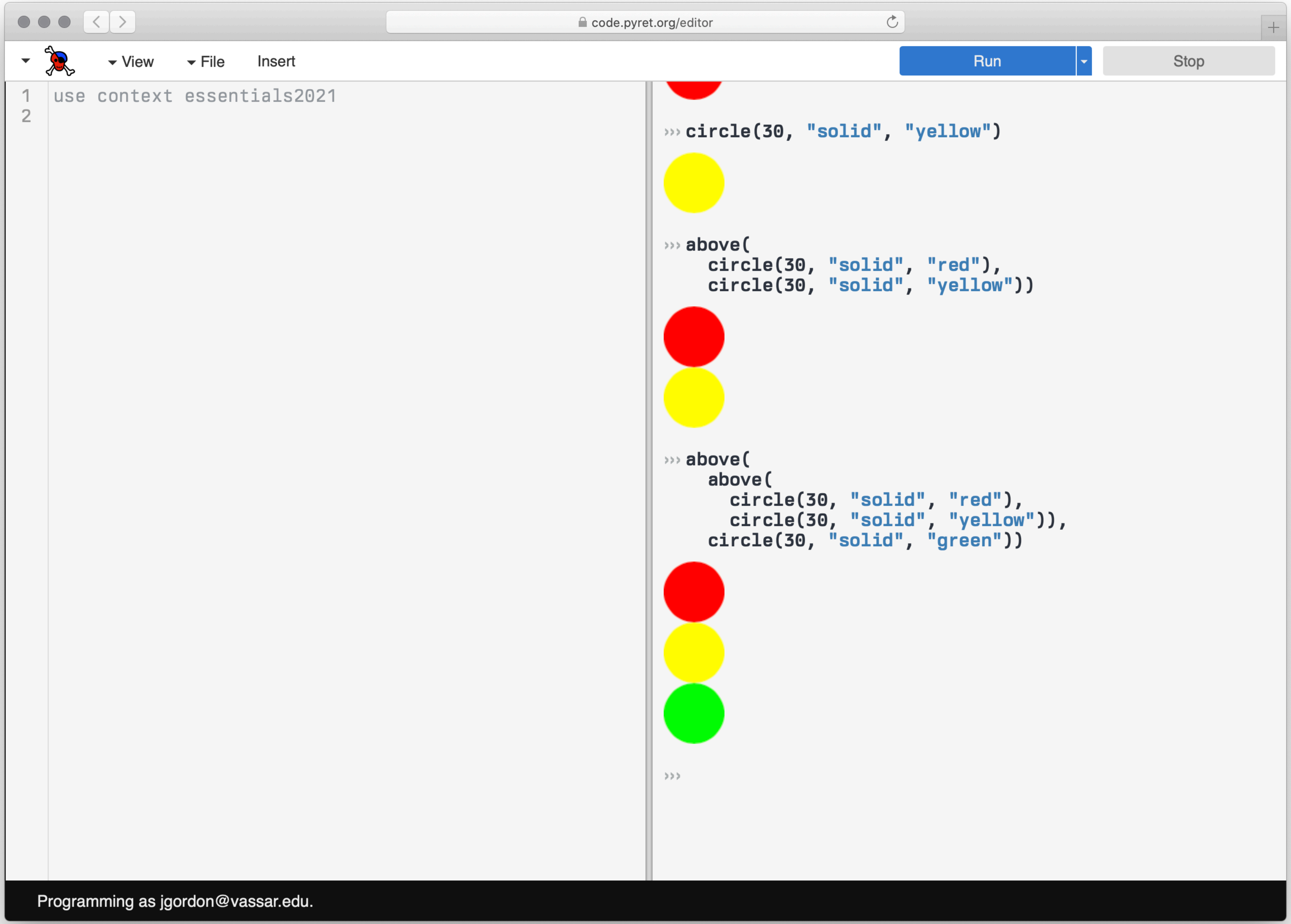


code.pyret.org/editor


code.pyret.org/editor
View
File
Insert
Run
Stop
use context essentials2021
››› include image
››› circle(30, "solid", "red")
›››
Programming as jgordon@vassar.edu.


code.pyret.org/editor
View
File
Insert
Run
Stop
1 use context essentials2021
2
››› include image
››› circle(30, "solid", "red")
››› circle(30, "solid", "yellow")
›››
Programming as jgordon@vassar.edu.

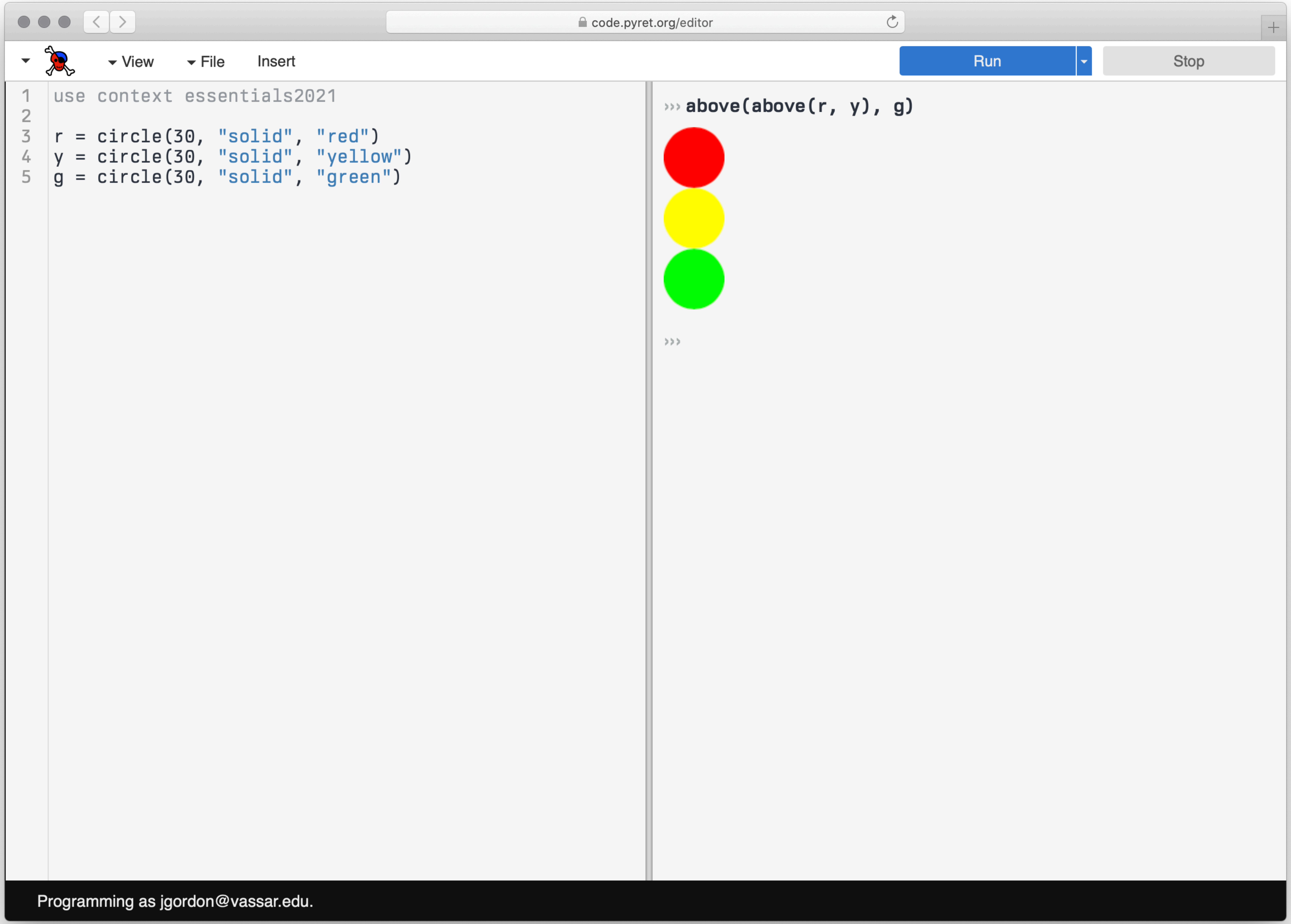

code.pyret.org/editor
View
File
Insert
Run
Stop
use context essentials2021
››› include image
››› circle(30, "solid", "red")
››› circle(30, "solid", "yellow")
››› above(
circle(30, "solid", "red"),
circle(30, "solid", "yellow"))
›››
Programming as jgordon@vassar.edu.

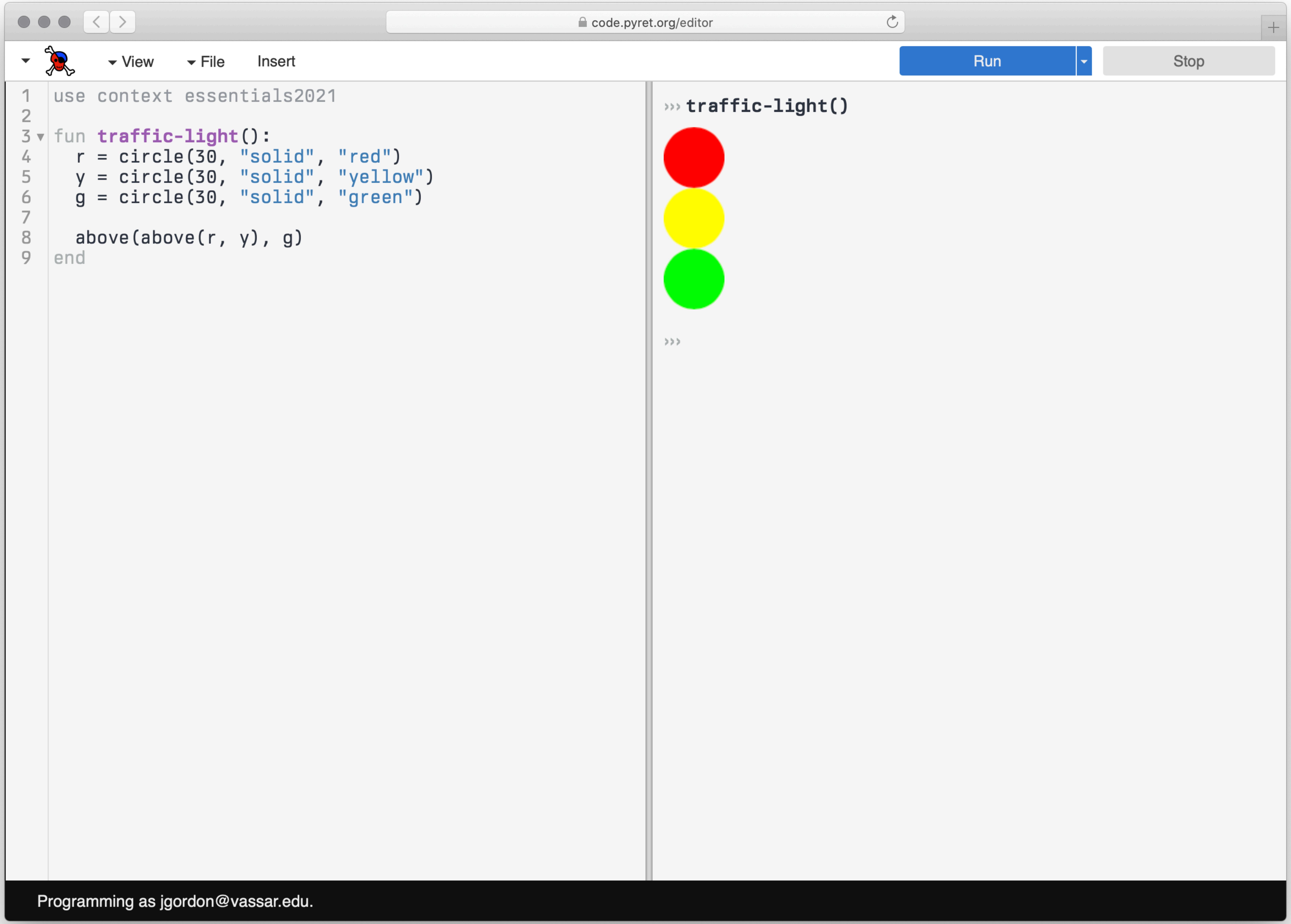

code.pyret.org/editor
View
File
Insert
Run
Stop
1 use context essentials2021
2
››› circle(30, "solid", "yellow")
››› above(
circle(30, "solid", "red"),
circle(30, "solid", "yellow"))
››› above(
above(
circle(30, "solid", "red"),
circle(30, "solid", "yellow")),
circle(30, "solid", "green"))
›››
Programming as jgordon@vassar.edu.

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
File
Insert
Run
Stop
use context essentials2021
r = circle(30, "solid", "red")
y = circle(30, "solid", "yellow")
g = circle(30, "solid", "green")
››› above(above(r, y), g)
›››
Programming as jgordon@vassar.edu.


code.pyret.org/editor
View
File
Insert
Run
Stop
use context essentials2021
fun traffic-light():
r = circle(30, "solid", "red")
y = circle(30, "solid", "yellow")
g = circle(30, "solid", "green")
above(above(r, y), g)
end
››› traffic-light()
›››
Programming as jgordon@vassar.edu.

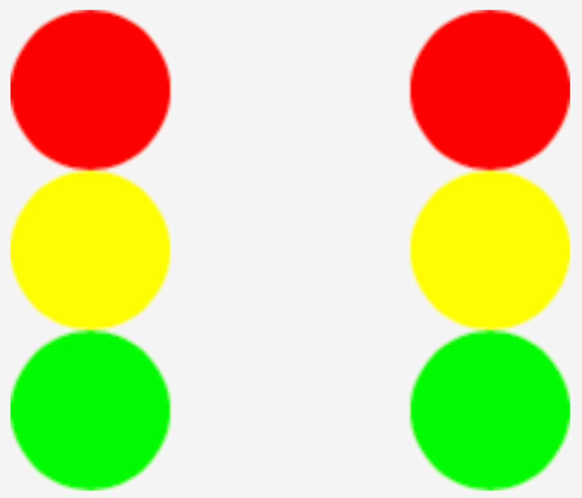
In extending the language, the programmer uses the power of functional abstraction to ***create new building blocks***.

code.pyret.org/editor

View File Insert Run Stop

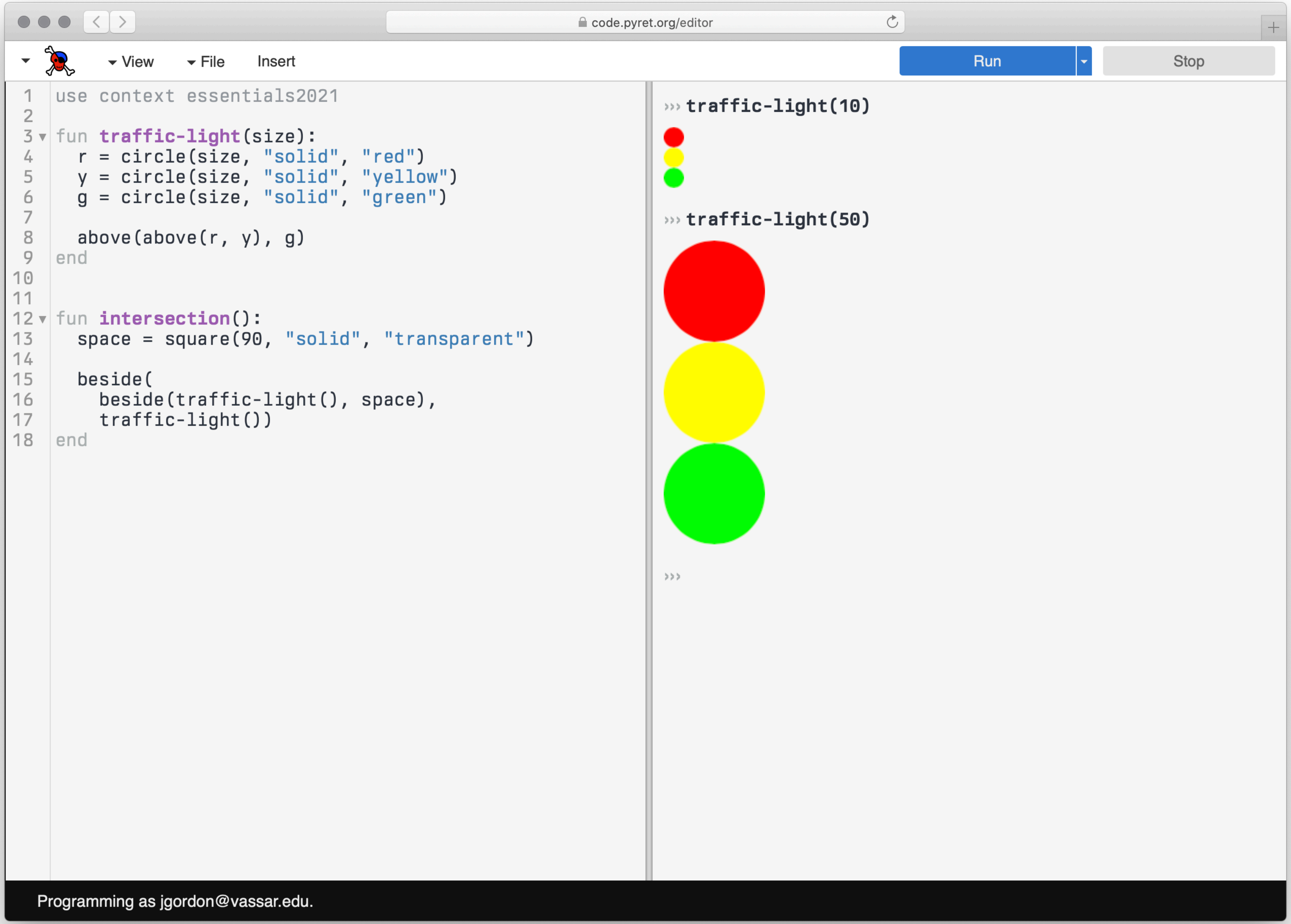
```
use context essentials2021

fun traffic-light():
  r = circle(30, "solid", "red")
  y = circle(30, "solid", "yellow")
  g = circle(30, "solid", "green")

  above(above(r, y), g)
end


fun intersection():
  space = square(90, "solid", "transparent")

  beside(
    beside(traffic-light(), space),
    traffic-light())
end
```

```
››› intersection()
```



```
›››
```

Programming as jgordon@vassar.edu.

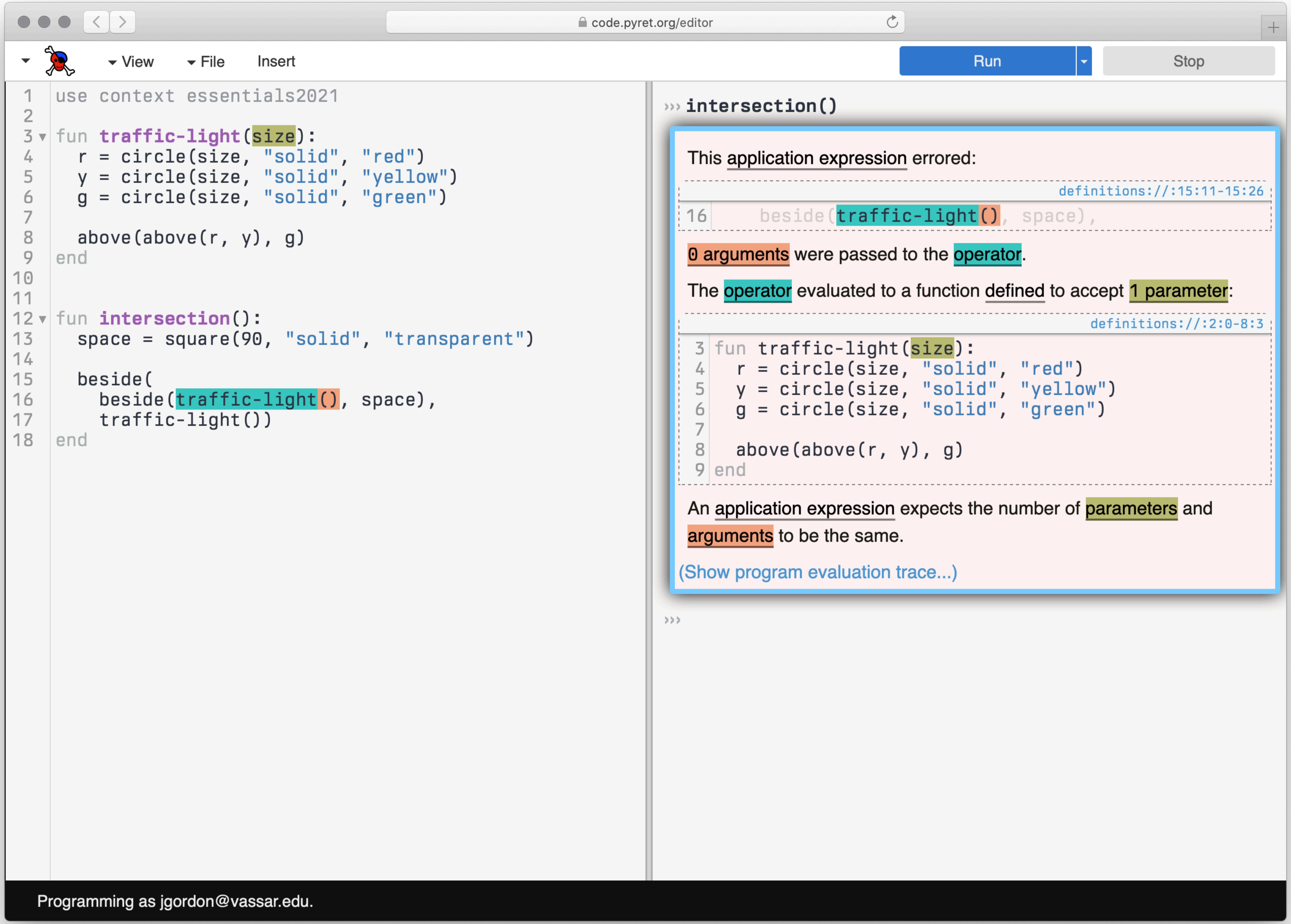

code.pyret.org/editor
View
File
Insert
Run
Stop
use context essentials2021

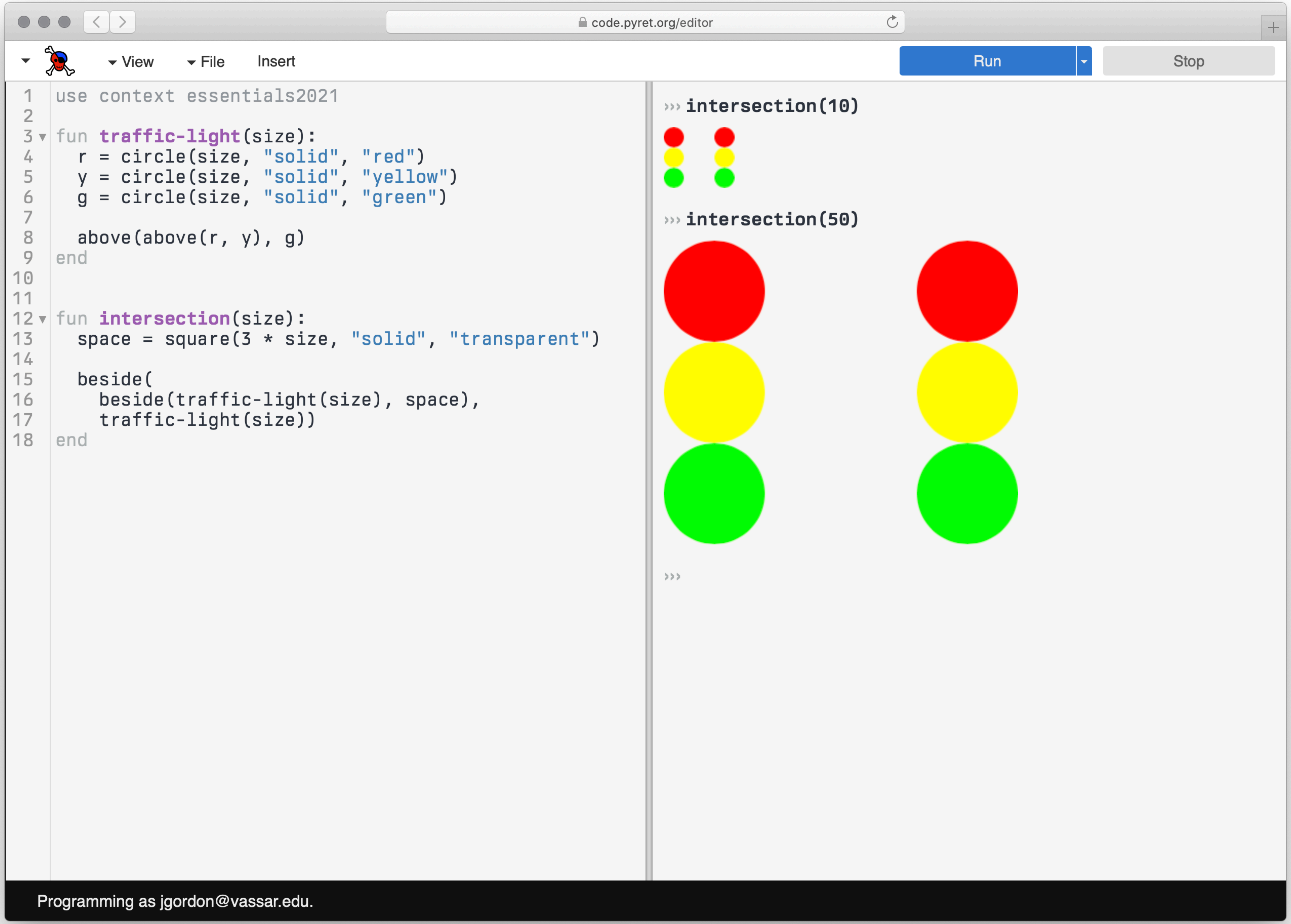
fun traffic-light(size):
  r = circle(size, "solid", "red")
  y = circle(size, "solid", "yellow")
  g = circle(size, "solid", "green")

  above(above(r, y), g)
end

fun intersection():
  space = square(90, "solid", "transparent")

  beside(
    beside(traffic-light(), space),
    traffic-light())
end
››› traffic-light(10)
››› traffic-light(50)
›››
Programming as jgordon@vassar.edu.


code.pyret.org/editor
View
File
Insert
Run
Stop
use context essentials2021
fun traffic-light(size):
r = circle(size, "solid", "red")
y = circle(size, "solid", "yellow")
g = circle(size, "solid", "green")
above(above(r, y), g)
end
fun intersection():
space = square(90, "solid", "transparent")
beside(
beside(traffic-light(), space),
traffic-light())
end
››› intersection()
This application expression errored:
definitions://:15:11-15:26
16 beside(traffic-light(), space),
0 arguments were passed to the operator.
The operator evaluated to a function defined to accept 1 parameter:
definitions://:2:0-8:3
3 fun traffic-light(size):
4 r = circle(size, "solid", "red")
5 y = circle(size, "solid", "yellow")
6 g = circle(size, "solid", "green")
8 above(above(r, y), g)
9 end
An application expression expects the number of parameters and arguments to be the same.
(Show program evaluation trace...)
Programming as jgordon@vassar.edu.



code.pyret.org/editor
View
File
Insert
Run
Stop
use context essentials2021
fun traffic-light(size):
  r = circle(size, "solid", "red")
  y = circle(size, "solid", "yellow")
  g = circle(size, "solid", "green")
  above(above(r, y), g)
end
fun intersection(size):
  space = square(3 * size, "solid", "transparent")
  beside(
    beside(traffic-light(size), space),
    traffic-light(size))
end
››› intersection(10)
››› intersection(50)
›››
Programming as jgordon@vassar.edu.

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.

First half:

Functional programming

Atomic, tabular, and recursive data

Pyret

Second half:

Imperative programming

Messy real-world data

Python

Goals

Apply fundamental data structures to capture the information in a computing problem.

Break down a computing question into smaller, manageable problems.

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

Check whether your programs behave as intended/required.

Course information

docs.google.com/forms/d/e/1FAIpQLSetFWP6-2HNUoU3lW18aBY9HYzalfkHt9Q8tLv

CMPU-101-53 Student Information

Please fill out this short form to help me better prepare for the start of the semester.

jgordon@vassar.edu Switch account

* Indicates required question

Email *

☐ Record **jgordon@vassar.edu** as the email to be included with my response

What name would you like me to call you? *

Your answer

What are your preferred pronouns? (optional)

Your answer

forms.gle/HfHMcA5PjyCuZwgX9

Class:

Tuesday & Thursday, 3:10–4:25 p.m.

Sanders Classroom 006

Lab:

Friday, 3:10–5:10 p.m.

Sanders Classroom 006

www.cs.vassar.edu/~cs101/53/

CMPU 101

⌂ LABS ASSIGNMENTS RESOURCES GRADESCOPE

Computer Science I: Problem-Solving and Abstraction

Spring 2024 · §53

Tuesday 3:10–4:25 p.m.
Thursday 3:10–4:25 p.m.
Friday 3:10–5:10 p.m.
Sanders Classroom 006

Professor Gordon

↓ Current

	Tuesday	Thursday	Friday
Introduction		*Jan. 18*	*Jan. 19*
– Read Syllabus – Read How to succeed – Read 3.1 Getting started		Problem-solving and abstraction	Lab 1: Getting started

cs.vassar.edu/~cs101/53

CMPU 101 §53

Computer Science I: Problem-Solving and Abstraction

Spring 2024

Tuesday	3:10–4:25 p.m.
Thursday	3:10–4:25 p.m.
Friday	3:10–5:10 p.m.

Sanders Classroom 006

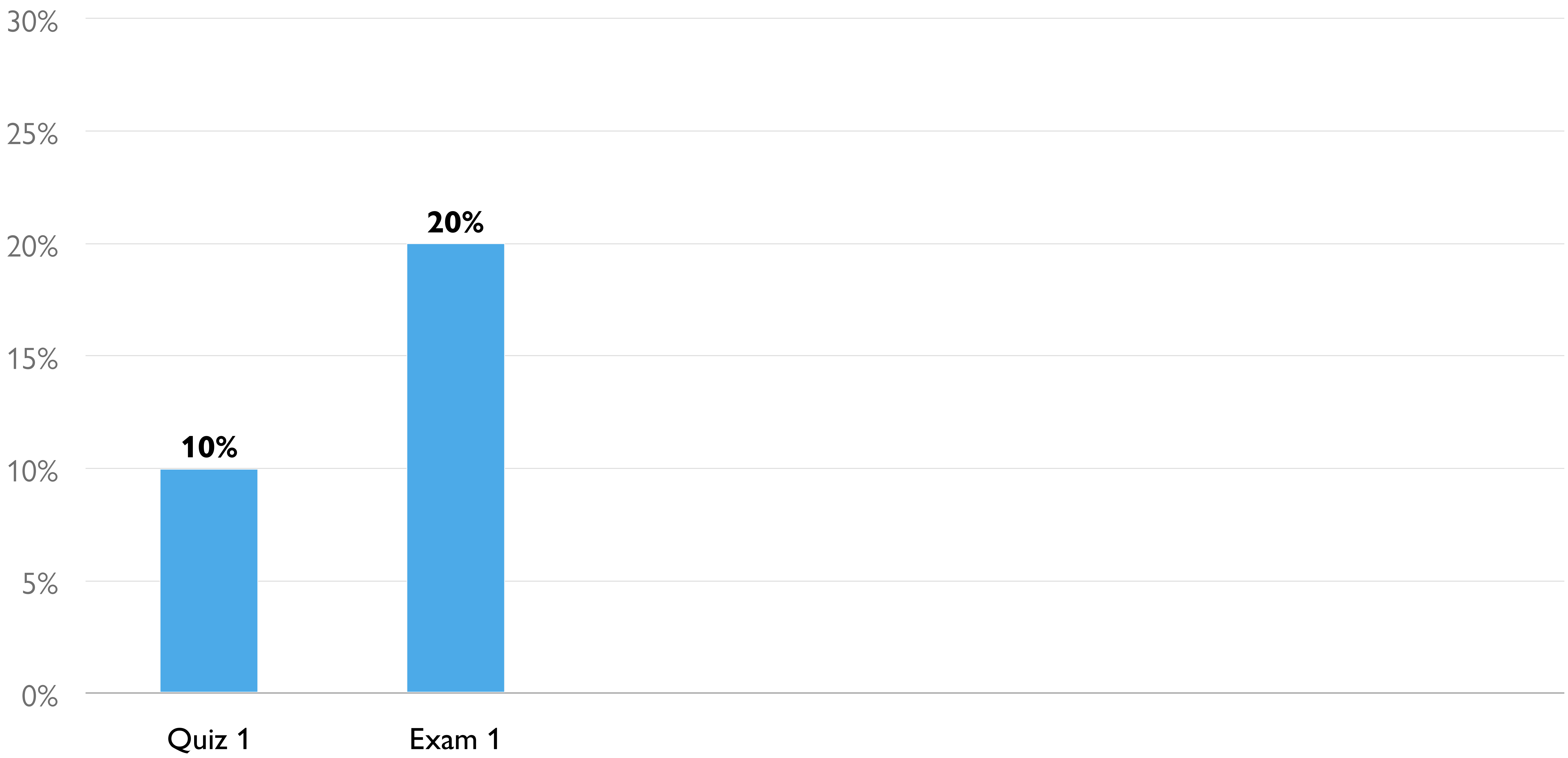
Professor Gordon

cs.vassar.edu/~cs101/3

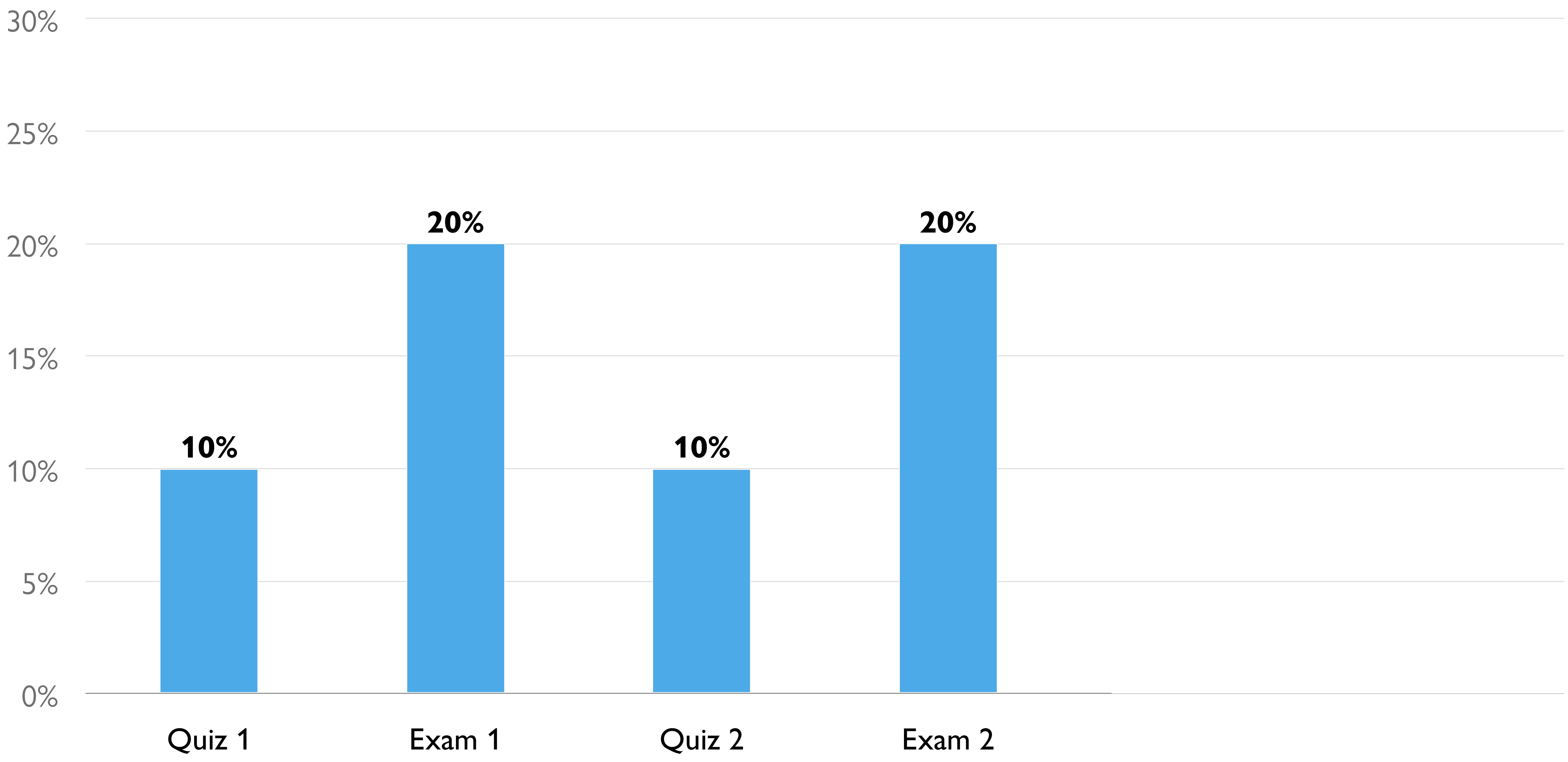
Overview

This course introduces fundamental concepts of computer science

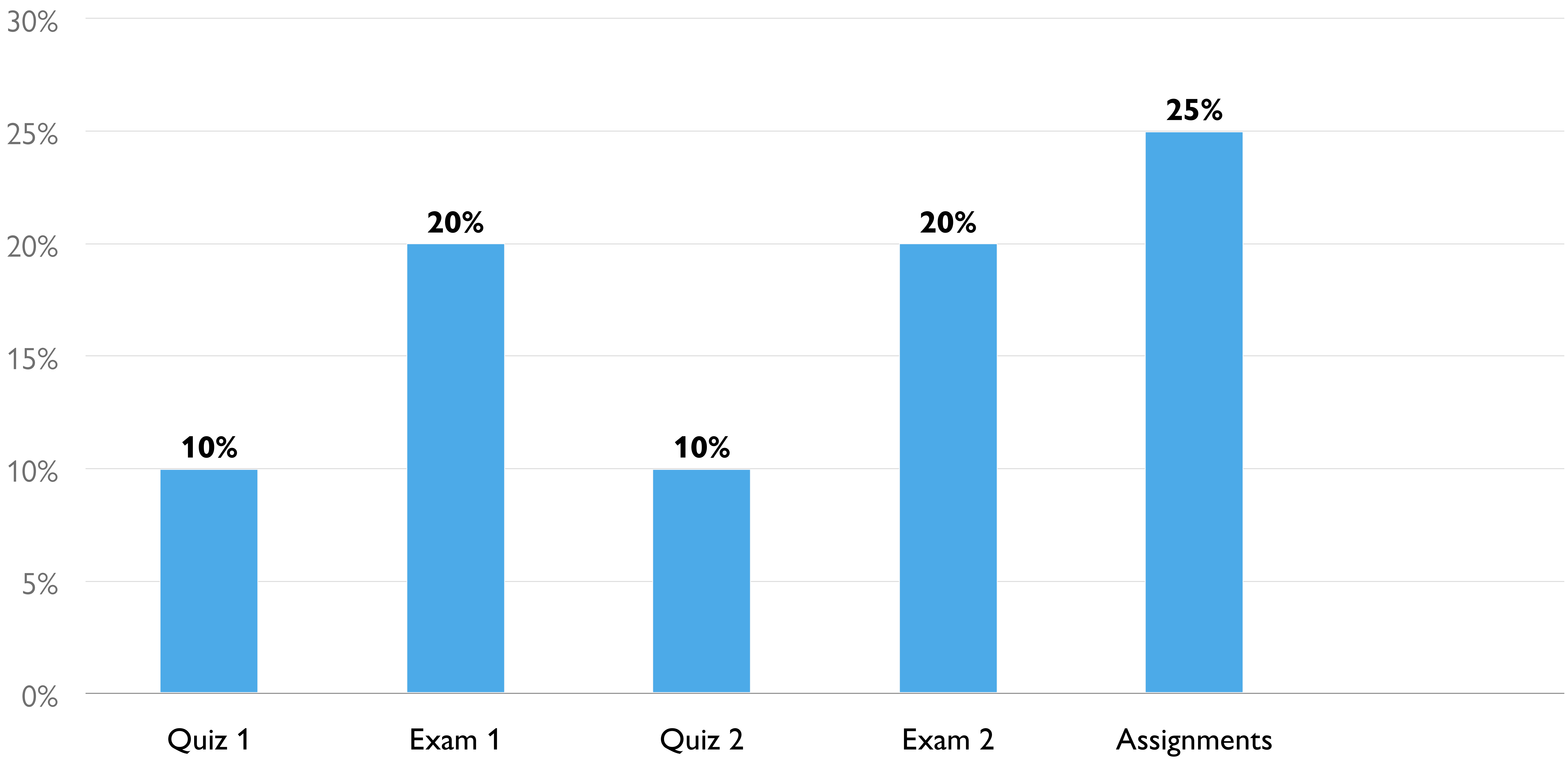
Grading

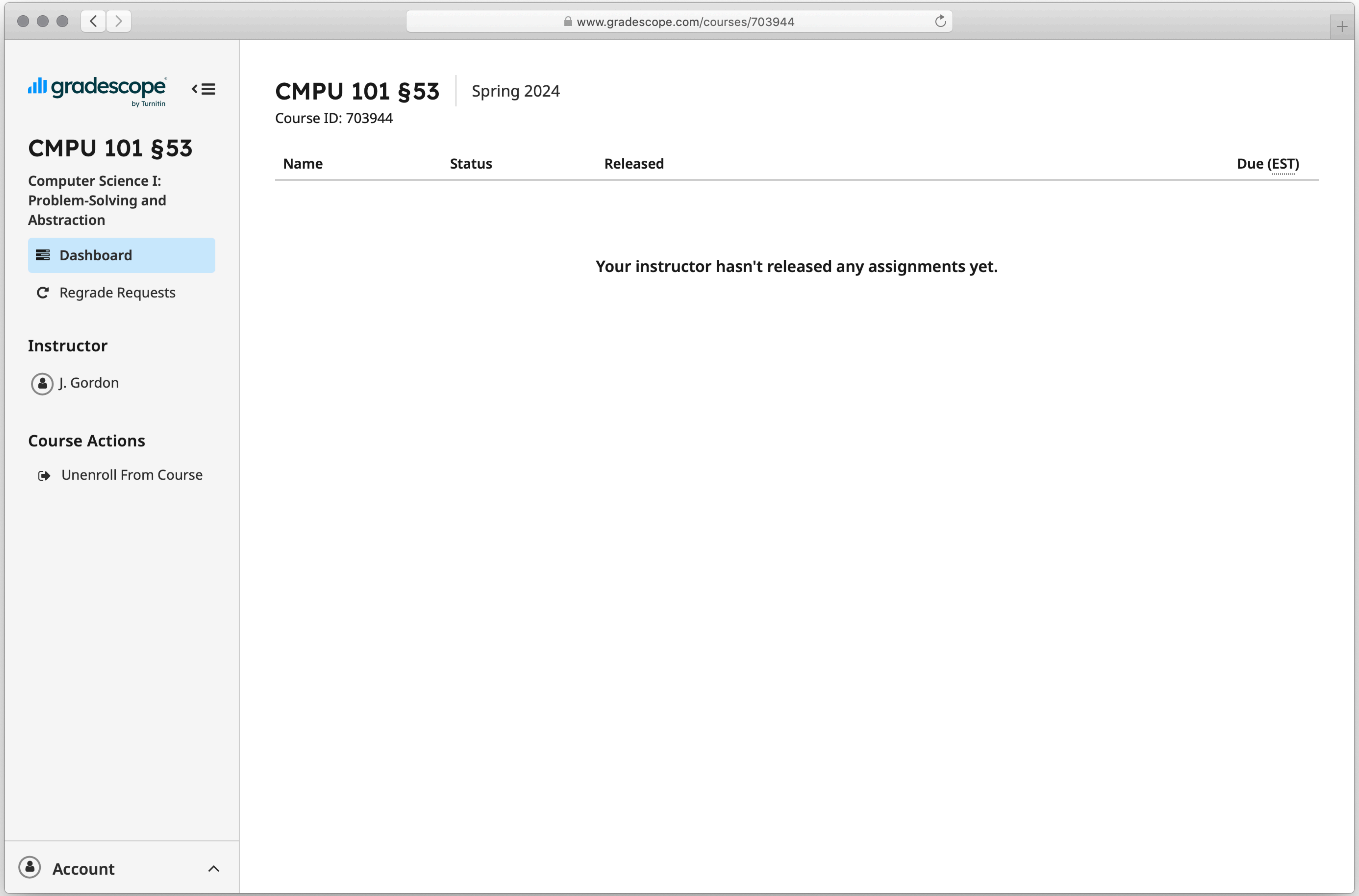


Grading



Grading


30%
25%
20%
15%
10%
5%
0%
10%
20%
10%
20%
25%
Quiz 1
Exam 1
Quiz 2
Exam 2
Assignments



gradescope.com

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

Vassar CS Student Integrity Guide

Table of Contents

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:

1. The work submitted should be solely authored by the person submitting it.

cs.vassar.edu/integrity

www.cs.vassar.edu/~cs101/53/resources/syllabus.pdf

Generative AI

Generative AI – such as ChatGPT, Bing Chat, or Anthropic Claude – can be powerful tools to help you in writing and debugging programs. At times during the semester, we may encourage you to try these tools in class and in lab. For this class, you may also use generative AI on your own when you are studying or working on homework assignments.

If you use generative AI while working on a homework assignment, your submission must include a comment acknowledging the use of these tools, and you may be asked to include a transcript showing your interactions. (This will help us to understand both the difficulties students are having in the class and the ways that generative AI can help them!)

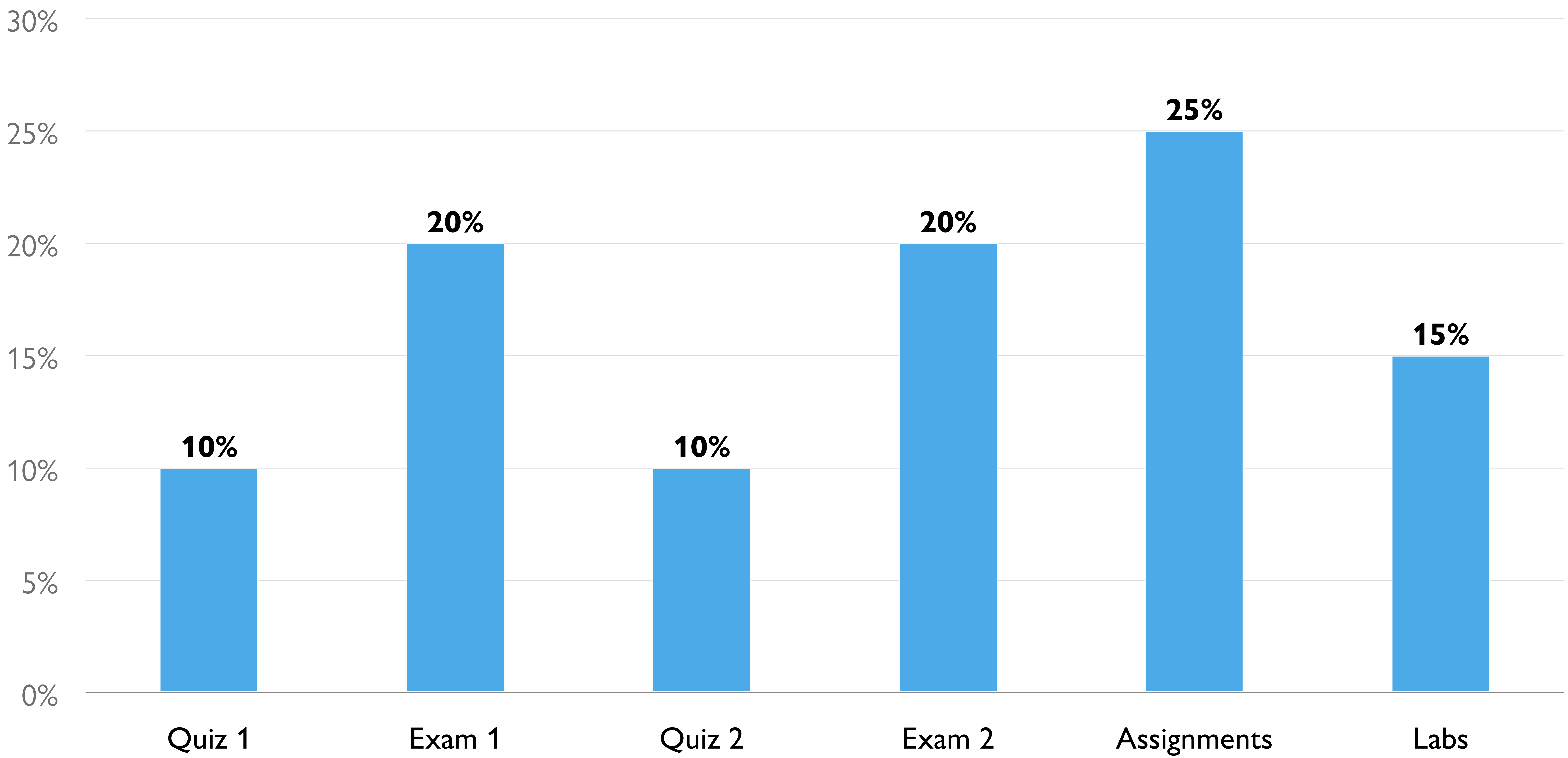
Quizzes and exams will be taken in person, on paper, without the use of AI, so be careful that you are using generative AI to *help* you learn rather than as a way to *avoid* learning!

Academic integrity

Please read the CS department's guide to academic integrity. In particular, note that:

cs.vassar.edu/integrity

Grading



30%
25%
20%
15%
10%
5%
0%
10%
20%
10%
20%
25%
15%
Quiz 1
Exam 1
Quiz 2
Exam 2
Assignments
Labs

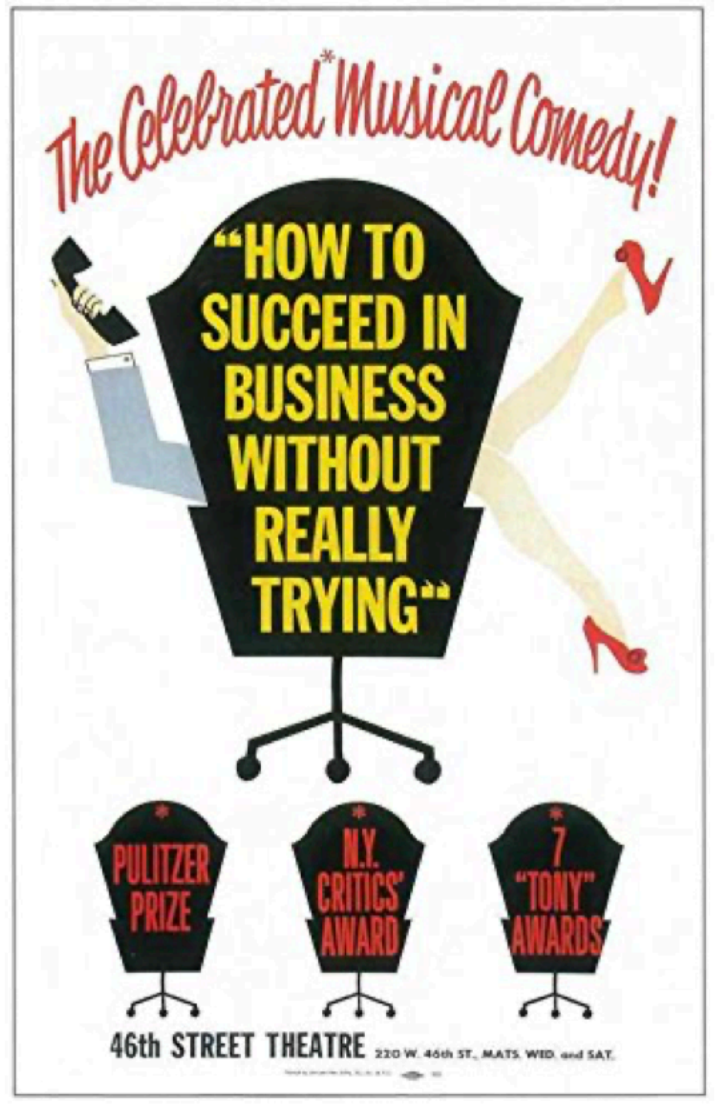
www.cs.vassar.edu/~cs101/53/resources/succeed.pdf

CMPU 101 §53 · Spring 2024

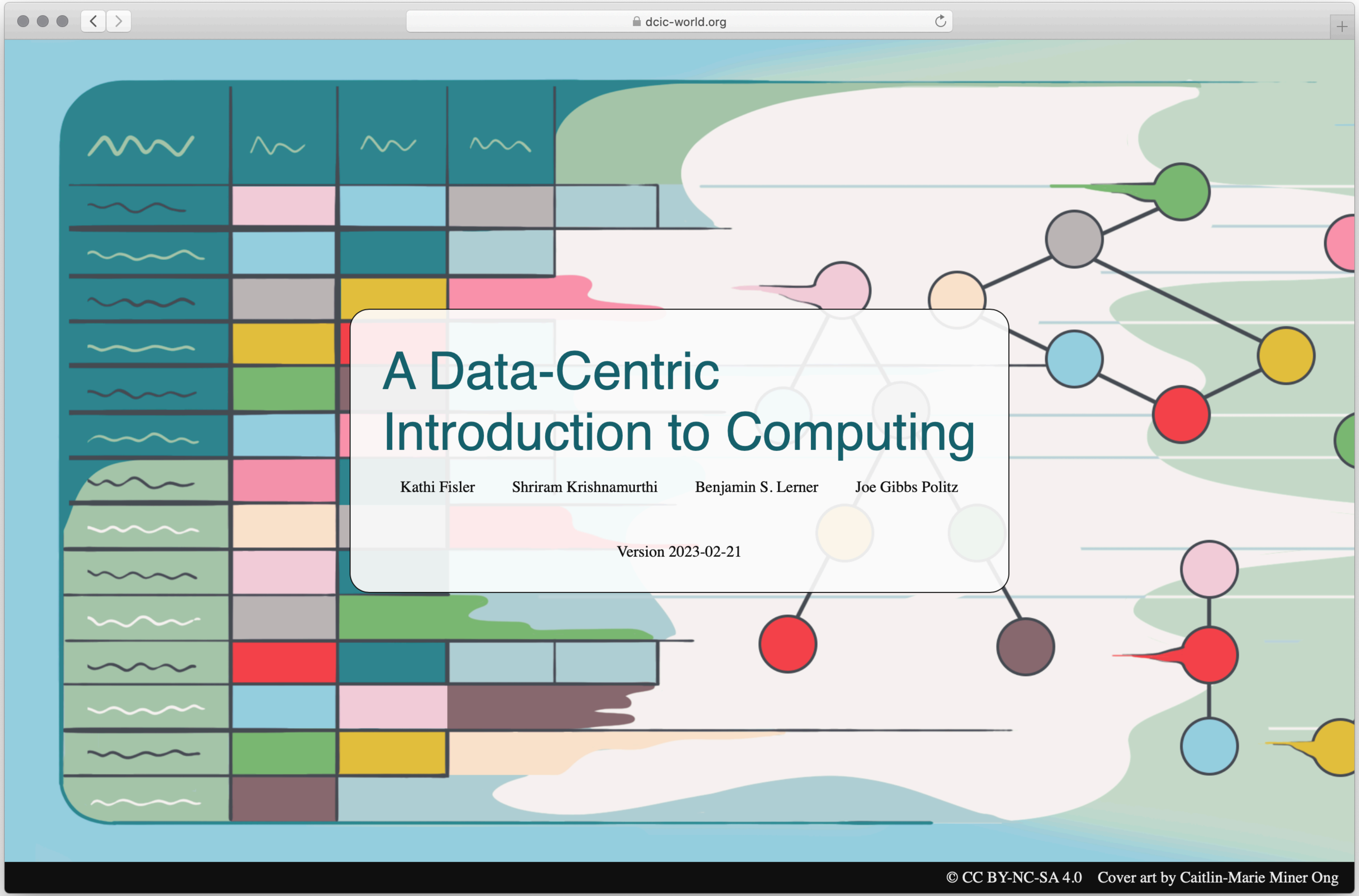
How to Succeed in CMPU 101 by Really Trying

I want you to be successful in CMPU 101 this semester. "Success" in the course is more than just good grades. It means that you are being challenged to grow as a learner, that you are engaging actively with tasks that feed your growth, and that you are creating excellent work in computer science by completing tasks with an appropriate level of support. It also means that you are building your lifelong learning skills so that once the course is over, you are better and stronger as a learner, both so you can succeed in courses that have CMPU 101 as a prerequisite – and so you can continue to learn new things independently.

This is designated as a quantitative course, satisfying one of Vassar's requirements for graduation. Like any quantitative course, it will challenge you to think abstractly, analytically, and logically. It's my sincere hope that you enjoy this class, but the unavoidable truth is that learning computer science takes time, effort, and practice. So,



Close – but try harder!



dcic-world.org


How do you draw so well?
Practice.
It must be an innate gift...
A gift from God...
It's practice.
I'll never understand how some people are so talented...
A mystery...
Practice.

"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*

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*