What have we been doing this semester?
use context essentials2021

"Hello, computer!"

>>>
"Hello, computer!"

'Hello, computer!'
We’re not especially interested in Pyret – or Python!

If you’re programming 20 years from now, it will be in a different language, using different tools.
What have we been *doing* in these languages?
use context essentials2021

# Assignment 1: Fun with Flags
# Example Solutions
# CMPU 101, Spring 2024

# You can scale the flags up or down by changing this width.
# Note that the flags have different width-to-height ratios, so they don't all use the 'height' defined below.
width = 300
height = width / 1.5

# The country abbreviations used below:

# Switzerland
#
ch-bg = square(width, "solid", "red")
ch-bar = rectangle(0.6 * width, 0.2 * width, "solid", "white")
ch-cross = overlay(ch-bar, rotate(90, ch-bar))
ch = frame(overlay(ch-cross, ch-bg))
ch

# The Republic of the Congo
#
Programming as a guest.
Prediction accuracy

Since we have the actual child heights, we can see how accurate our prediction is:
get_art_info(456524)

{'isHighlight': False,
 'acquisitionNumber': '49.41',
 'acquisitionYear': '1949',
 'isPublicDomain': True,
 'additionalImages': [],
 'constituents': [{'constituentID': 161947,
 'role': 'Artist',
 'name': 'Vincent van Gogh',
 'constituentULAN_URL': 'http://vocab.getty.edu/ulan/500155588',
 'constituentWikidata_URL': 'https://www.wikidata.org/wiki/Q5382',
 'gender': ''},
 'department': 'European Paintings',
 'objectName': 'Painting',
 'title': 'Sunflowers',
 'culture': '',
 'period': '',
 'dynasty': '',
 'reign': '',
 'portfolio': '',
 'artistRole': 'Artist',
 'artistPrefix': '',
 'artistDisplayName': 'Vincent van Gogh',
 'artistDisplayBio': 'Dutch, Zundert 1853-1890 Auvers-sur-Oise',
 'artistSuffix': '',
 'artistAlphaSort': 'Gogh, Vincent van',
 'artistNationality': 'Dutch',
 'artistBeginDate': '1853',
 'artistEndDate': '1890',
 'artistGender': '',
 'artistWikidata_URL': 'https://www.wikidata.org/wiki/Q5382',
 'artistULAN_URL': 'http://vocab.getty.edu/ulan/500155588',
 'objectDate': '1887',
 'objectBeginDate': '1887',
 'objectEndDate': '1887'}
We’ve been practicing *computational thinking*.
“Modern computer science is the last 1 percent of the historical timeline of computational thinking. Computer scientists inherited and then perfected computational thinking from a long line of mathematicians, natural philosophers, scientists, and engineers all interested in performing large calculations and complex inferences without error.”

Peter J. Denning & Matti Tedre, *Computational Thinking*
Origins of computational thinking
I’m a computer!
Teams of human computers engaged in computational thinking long before the invention of electronic computers.
Early computational thinking can be seen going back to the records of the Babylonians, who wrote down general procedures for solving mathematical problems around starting around 1800 BCE.
Long before this class, you learned these kind of computational methods.
27182818284590
+31415926535897
\[\underline{27182818284590 + 31415926535897 = 58608744820487}\]
2718281828459\theta \\
+31415926535897 \\
----------------- \\
\quad 7
\[
\begin{align*}
271828182845 \pm 314159265358 & \quad \text{97} \\
+314159265358 & \quad 90 \\
\hline
\end{align*}
\]

\[87\]
\[
\begin{align*}
27182818284590 & \quad + 31415926535897 \\
\hline
11 & 487
\end{align*}
\]
27182818284590
+31415926535897
\[= 0487\]
\[
\begin{align*}
\text{1111} & \quad \text{284590} \\
+31415926 & \quad 535897 \\
\hline
\text{820487}
\end{align*}
\]
18284590
+6535897
\[ \underline{4820487} \]
27182818284590
+314159226535897
\[\underline{44820487} \]
\[
1111111

27182818284590
+31415926535897
\]

\[
744820487
\]
\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2718 & 2 & 8182 & 8459 & 0 \\
+3141 & 5 & 9265 & 3589 & 7 \\
\hline
8744820487 \\
\end{array}
\]
27182818284590
+31415926535897
98744820487
\[
\begin{array}{ccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
27182818284590 \\
+31415926535897 \\
\hline
585987444820487
\end{array}
\]
Euclid’s algorithm

Around 300 BCE, the Greek mathematician Euclid gave a method to find the greatest common divisor (GCD) of two numbers, which is the largest integer that divides both numbers.
Euclid’s algorithm

Euclid noticed that the GCD of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.
Euclid’s algorithm

Euclid noticed that the GCD of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.

$\text{gcd}(48, 18)$
Euclid’s algorithm

Euclid noticed that the GCD of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.

\[
gcd(48, 18) \rightarrow gcd(30, 18)
\]
Euclid’s algorithm

Euclid noticed that the GCD of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.

\[
gcd(48, 18) \\
\rightarrow gcd(30, 18) \\
\rightarrow gcd(12, 18)
\]
Euclid’s algorithm

Euclid noticed that the GCD of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.

\[
gcd(48, 18) \\
\rightarrow gcd(30, 18) \\
\rightarrow gcd(12, 18) \\
\rightarrow gcd(12, 6)
\]
Euclid’s algorithm

Euclid noticed that the GCD of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.

\[
gcd(48, 18) 
\rightarrow gcd(30, 18) 
\rightarrow gcd(12, 18) 
\rightarrow gcd(12, 6) 
\rightarrow gcd(6, 6)
\]
Euclid’s algorithm

Euclid noticed that the \text{GCD} of two numbers divides their difference.

So, he repeatedly replaced the larger number of the two numbers with their difference until both were the same.

\[
gcd(48, 18) 
\rightarrow gcd(30, 18) 
\rightarrow gcd(12, 18) 
\rightarrow gcd(12, 6) 
\rightarrow gcd(6, 6) 
\rightarrow 6
\]
Sieve of Eratosthenes

This is another famous method dating back to the ancient Greeks, used to find all the prime numbers up to some limit.
Sieve of Eratosthenes

We begin with a list of all the integers, from 2 to some chosen limit.
Sieve of Eratosthenes

We cross out all the multiples of 2.
We cross out all the multiples of 2.

Then all the multiples of 3.
We cross out all the multiples of 2.

Then all the multiples of 3. And 5.
Sieve of Eratosthenes

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

We cross out all the multiples of 2.

Then all the multiples of 3. And 5.

And so on, leaving you with only the primes between 2 and the limit you chose.
Sieve of Eratosthenes

After each round of elimination, a new prime will be revealed, and the next round crosses out all its multiples.
These are computational procedures, carried out by hand!
Programmable computers
No matter how simple and unambiguous the steps are made, human computers make mistakes – and lots of them!

So, inventors through the ages have sought to make computing *machines* to allow people to perform longer computations with fewer errors.
This was a slow process, taking us from…
Slide rule

c. 1620
Blaise Pascal’s mechanical calculator

1642
Precursors to the idea of a *programmable* computer originated well before the electronic computing age. In the early 1700s, French textile weavers experimented with machines that could weave complex patterns using an automatic loom.
One of the more well known is the Jacquard loom, which was controlled by long chains of punched cards.
Plan for Babbage’s Analytical Engine

1840
Babbage collaborated with a gifted mathematician, Ada Lovelace, who designed algorithms for the Analytical Engine, even though there was no machine to run them on.
Lovelace saw the Analytical Engine not as a mere calculator but as a processor of *any* information that could be encoded in symbols.

This insight, that computing programs can calculate not only over numbers but over symbols that can stand for anything in the world, anticipated by a hundred years a key tenet of the modern computer age.

Lovelace saw the computer as an *information machine*. 
While Babbage’s designs for a programmable computer weren’t realized at the time, the age of electronics opened new possibilities.
ENIAC

c. 1945
Early computers were very difficult to program, working in languages that were closely tied to the hardware.
Grace Hopper ’28 popularized the idea of a compiler for machine-independent programming languages and defined FLOW-MATIC, the first English-like data processing language in the early 1950s.

Those ideas were later folded into the popular COBOL language (1959).
Since the 1950s, many programming languages have been defined, experienced popularity, and then been supplanted by new designs.
Today, Python is the programming language most often used for work in data science and artificial intelligence.
Programming no longer involves plugging in wires or punching cards, but it’s still hard!
“The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds castles in the air, from air, creating by exertion of the imagination…

Frederick Brooks,
The Mythical Man-Month, 1975
“Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures. Yet the program construct, unlike the poet’s words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself…

Frederick Brooks,
The Mythical Man-Month, 1975
“One types the correct incantation on a keyboard, and a display screen *comes to life*, showing things that never were nor could be… It prints results, draws pictures, produces sounds, moves arms. The magic of myth and legend has come true in our time…

Frederick Brooks, *The Mythical Man-Month*, 1975
“The computer resembles the magic of legend in this respect, too. If one character, one pause, of the incantation is not strictly in proper form, the magic doesn’t work. Human beings are not accustomed to being perfect, and few areas of human activity demand it. *Adjusting to the requirement for perfection is, I think, the most difficult part of learning to program.*”

Frederick Brooks, *The Mythical Man-Month*, 1975
Computing with data
We’ve seen some cool datasets during this semester – and you got to explore data that was of interest to you for your mini-projects – but there are many, many more datasets you can explore.
Data Is Plural

... is a weekly newsletter (and seasonal podcast) of useful/curious datasets, published by Jeremy Singer-Vine. There have been 371 editions, dating from October 21, 2015 to April 24, 2024. To receive future editions, sign up here:

Enter your email address
Subscribe

Some Nice Things People Have Said

- “a treasure trove of interesting datasets” — Julia Silge
- “delivers exactly what it promises, it’s delightful” — Simon Willison
- “required reading for anyone interested in data journalism” — Julia Angwin
- “consistently fascinating” — Robin Sloan
- “the only newsletter I open immediately” — Paul Ford
- “the newsletter I forward most frequently to colleagues who like datasets” — Melody Joy Kramer
- “definitely my favorite newsletter about data and possibly my favorite newsletter period” — Melanie Welsh
- “the best data newsletter ever” — François Briatte

Recent Editions

2024.04.24 • Public procurement, chain locations, wholesale electricity markets, food-system indicators, and 3D natural history.

2024.04.17 • Unregulated water contaminants, border crossings, earthquakes felt, automated decision-making in government, and aerial obstacles.

2024.04.10 • Opioid settlement payouts, groundwater wells, work-injury laws, colonial empire timelines, and objects launched into outer space.
Try Public APIs for free

Explore popular APIs and see them work in Postman.

APILayer

APILayer is the fastest way to integrate APIs into any product. They created this repository to support the community in easily finding public APIs. Explore their collections on the Postman API Network.

APILayer APIs

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
<th>Call this API</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Stack</td>
<td>Locate and identify Website Visitors by IP Address</td>
<td>Run in Postman</td>
</tr>
<tr>
<td>Marketstack</td>
<td>Free, easy-to-use REST API Interface delivering worldwide stock market data in JSON format</td>
<td>Run in Postman</td>
</tr>
<tr>
<td>Weatherstack</td>
<td>Retrieve instant, accurate weather information for any location in the world in lightweight JSON format</td>
<td>Run in Postman</td>
</tr>
<tr>
<td>Numverify</td>
<td>Global Phone Number Validation &amp; Lookup JSON API</td>
<td>Run in Postman</td>
</tr>
<tr>
<td>Fixer</td>
<td>Fixer is a simple and lightweight API for current and historical foreign exchange (forex) rates.</td>
<td>Run in Postman</td>
</tr>
</tbody>
</table>

Popular APIs

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
<th>Auth</th>
<th>Call this API</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Cat</td>
<td>Cat for every HTTP Status</td>
<td>No</td>
<td>Run in Postman</td>
</tr>
</tbody>
</table>
Building data models
Data is situated in the environment where it was gathered.

Consider Galton’s child-height data.

He gathered the data in England c. 1886.

What would happen if you tried to use it to predict heights in Poughkeepsie today? In Guatemala? In China?
When we collect data, it’s like making a map:

We’re constructing a model, where we choose what to represent, and how to represent it.
“...most of the data and data models we have inherited deal with structures of power, like gender and race, with a crudeness that would never pass muster in a peer-reviewed humanities publication.”

Let's see how things look when we look at the parents' heights combined into "mid-parent height" with this factor:

```r
galton.hist("mid-parent height", unit="inch")
```
Let's see how things look when we look at the parents' heights combined into "mid-parent height" with this factor:

galton.hist("mid-parent height", unit="inch")
About four-in-ten U.S. adults say forms should offer more than two gender options

Nikki Graf • December 18, 2019

States that offer a third gender option on driver’s licenses

Note: Hawaii, New Hampshire and Pennsylvania will make a third gender option available on driver’s licenses in 2020, as will Illinois beginning in 2024. Rhode Island has plans to begin offering the option within a year.

In 2019, at least seven states have started offering a third gender option on driver’s licenses for people who don’t identify as male or female, and at least four more plan to do so in 2020. A number of states have also added a third gender...
“I want us to be more ambitious, to hold ourselves to much higher standards when we are claiming to develop data-based work that depicts people’s lives.”

Data and privacy
### Dataset Card for "sentiment140"

<table>
<thead>
<tr>
<th>Text</th>
<th>Lengths</th>
<th>Date</th>
<th>String</th>
<th>Lengths</th>
<th>User</th>
<th>String</th>
<th>Lengths</th>
<th>Sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>@witchfoot <a href="http://twitpic.com/2zylx">http://twitpic.com/2zylx</a> - Awww, that's a bummer. You should get David Cazz of Third Day. I'm so sad.</td>
<td>28</td>
<td>Mon Apr 06 22:39:45 PTL 2009</td>
<td><em>TheSpecialOne</em></td>
<td>28</td>
<td>3</td>
<td>1</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>@manichaie I cried many times for the ball. Managed to save SRB. The rest go out of bounds.</td>
<td>20</td>
<td>Mon Apr 06 22:19:49 PTL 2009</td>
<td>mettycum</td>
<td>20</td>
<td>5</td>
<td>1</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>my whole body feels itchy and like its on fire</td>
<td>20</td>
<td>Mon Apr 06 22:19:13 PTL 2009</td>
<td>EllecFF</td>
<td>20</td>
<td>10</td>
<td>1</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Retionwideclass no, its not behaving at all. I'm mad. Why am i here? because I can't see you all.</td>
<td>20</td>
<td>Mon Apr 06 22:19:57 PTL 2009</td>
<td>Karoli</td>
<td>20</td>
<td>10</td>
<td>1</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>@KKunsfied not the whole crew</td>
<td>20</td>
<td>Mon Apr 06 22:19:57 PTL 2009</td>
<td>you wolf</td>
<td>20</td>
<td>10</td>
<td>1</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

Downloads last month: 599
- **Size of downloaded dataset files**: 81.4 MB
- **Size of the auto-converted Parquet files**: 124 MB
- **Number of rows**: 1,600,498

**Models trained or fine-tuned on sentiment140**
- AK776161/birdseye_roberta-base-tweet...
  - Text Classification
  - Updated May 11, 2023
- pig4431/Sentiment140_ELECTRA_S6
  - Text Classification
  - Updated Nov 7, 2021
- pig4431/Sentiment140_roBERTa_S6...
  - Text Classification
  - Updated Nov 6, 2021
- macildur/distilbert-base-uncasedv1-fi...
  - Text Classification
  - Updated Oct 5, 2022
<table>
<thead>
<tr>
<th>text</th>
<th>date</th>
<th>user</th>
<th>sentiment</th>
<th>query</th>
</tr>
</thead>
<tbody>
<tr>
<td>I'm itchy and miserable!</td>
<td>Sat May 16</td>
<td>roblaposky</td>
<td>0</td>
<td>itchy</td>
</tr>
<tr>
<td>@sokeemness no. I'm not</td>
<td>Sat May 16</td>
<td>EdwinLValencia</td>
<td>0</td>
<td>itchy</td>
</tr>
<tr>
<td>itchy for now. Maybe</td>
<td>22:42:44 UTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>later, lol.</td>
<td>2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT @jessverr I love the</td>
<td>Sat May 16</td>
<td>imusicmash</td>
<td>4</td>
<td>stanford</td>
</tr>
<tr>
<td>nerdy Stanford human</td>
<td>23:48:15 UTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>biology videos - makes</td>
<td>2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>me miss...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@spinuzzi: Has been a</td>
<td>Sat May 16</td>
<td>drewloewe</td>
<td>4</td>
<td>lyx</td>
</tr>
<tr>
<td>bit crazy, with steep</td>
<td>23:58:34 UTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>learning curve, but LyX</td>
<td>2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is reall...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I'm listening to &quot;P.Y.T&quot;</td>
<td>Sun May 17</td>
<td>VickyTigger</td>
<td>4</td>
<td>Danny Gokey</td>
</tr>
<tr>
<td>by Danny Gokey &lt;3 &lt;3</td>
<td>15:05:03 UTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;3 Aww, he's so</td>
<td>2009</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amazing...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>is going to sleep then</td>
<td>Sun May 17</td>
<td>babblyabbie</td>
<td>4</td>
<td>sleep</td>
</tr>
<tr>
<td>on a bike ride:]</td>
<td>17:27:45 UTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2009</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This account doesn’t exist
Try searching for another.
“The words and phrases we search for on Google, the times of day we are most active on Facebook, and the number of items we add to our Amazon carts are all tracked and stored as data – data that are then converted into corporate financial gain.”

D'Ignazio & Klein, *Data Feminism*, 2020
How Companies Learn Your Secrets

By Charles Duhigg
Feb. 16, 2012

Andrew Pole had just started working as a statistician for Target in 2002, when two colleagues from the marketing department stopped by his desk to ask an odd question: “If we wanted to figure out if a customer is pregnant, even if she didn't want us to know, can you do that?”

Pole has a master's degree in statistics and another in economics,
“As Pole’s computers crawled through the data, he was able to identify about 25 products that, when analyzed together, allowed him to assign each shopper a ‘pregnancy prediction’ score. More important, he could also estimate her due date to within a small window, so Target could send coupons timed to very specific stages of her pregnancy.

“One Target employee I spoke to provided a hypothetical example. Take a fictional Target shopper named Jenny Ward, who is 23, lives in Atlanta and in March bought cocoa-butter lotion, a purse large enough to double as a diaper bag, zinc and magnesium supplements and a bright blue rug. There’s, say, an 87 percent chance that she’s pregnant and that her delivery date is sometime in late August.”

Kashmir Hill, “How Target Figured Out a Teen Girl was Pregnant Before Her Father Did”, *Forbes*, 2012
Computing with data is complex, and it’s not just “technical” issues we need to concern ourselves with!
Computing with data gives us a lot of power!

We can do a lot of harm, but we can also do a lot of good!
Computer Science I
—or, where do you go from here?
<table>
<thead>
<tr>
<th>Data types</th>
<th>List and string slicing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naming values</td>
<td>List comprehensions</td>
</tr>
<tr>
<td>Evaluation</td>
<td>Numpy arrays</td>
</tr>
<tr>
<td>Conditionals (if and cases)</td>
<td>Sanitizing real-world data</td>
</tr>
<tr>
<td>Function signatures</td>
<td>Visualization</td>
</tr>
<tr>
<td>Testing functions</td>
<td>Dataclasses</td>
</tr>
<tr>
<td>Tabular data</td>
<td>Side effects and mutation</td>
</tr>
<tr>
<td>Higher-order functions</td>
<td>Functional vs imperative languages</td>
</tr>
<tr>
<td>Lambda expressions</td>
<td>Iteration (for)</td>
</tr>
<tr>
<td>Linked lists</td>
<td>Accumulators</td>
</tr>
<tr>
<td>Defining structured data</td>
<td>Debugging using <code>print</code> statements</td>
</tr>
<tr>
<td>Structurally recursive data and functions</td>
<td>Memory and aliasing</td>
</tr>
<tr>
<td>Trees (binary, n-ary)</td>
<td>Dictionaries (hash tables)</td>
</tr>
<tr>
<td>Reactive programs</td>
<td>Web APIs and JSON</td>
</tr>
</tbody>
</table>
Congratulations on making it this far!
CS courses at Vassar
Vassar Computer Science Course Map

Major-required courses
- CMPU 101 - Computer Science I: Problem-Solving and Abstraction
- CMPU 102 - Computer Science II: Data Structures and Algorithms
- CMPU 145 - Foundations of Computer Science
- CMPU 203 - Computer Science III: Software Design and Implementation
- CMPU 224 - Computer Organization
- CMPU 240 - Theory of Computation
- CMPU 341 - Analysis of Algorithms
- CMPU 331 - Compilers
- CMPU 334 - Operating Systems

300-level electives (at least one for major)
- CMPU 324 - Computer Architecture
- CMPU 353 - Bioinformatics
- CMPU 365 - Artificial Intelligence
- CMPU 368 - Computational Linguistics
- CMPU 375 - Computer Networks
- CMPU 377 - Parallel Programming
- CMPU 378 - Graphics
- CMPU 379 - Computer Animation: Art, Science and Criticism
- CMPU 395 - Advanced Special Topics

Correlate-required courses
- CMPU 101 - Computer Science I: Problem-Solving and Abstraction
- CMPU 102 - Computer Science II: Data Structures and Algorithms
- CMPU 145 - Foundations of Computer Science
- CMPU 240 or 241 - Theory of Computation or Analysis of Algorithms
- CMPU 2xx - Any other 200-level course
- CMPU 3xx - Any 300-level course

200-level electives (not required for major)
- CMPU 245 - Declarative Programming Models

Intensives (at least one for major)
- CMPU 310 - Topics in Virtualization
- CMPU 311 - Database Systems
- CMPU 312 - Applications of Artificial Intelligence
- CMPU 314 - Projects in Digital Media Production
- CMPU 315 - Computer Security

Extra-departmental
- MATH 221 - Linear Algebra

*At least two CMPU-200 level courses required for every CMPU-300 level course. †Prerequisites vary depending on topic.
Vassar Computer Science Course Map

**Major-required courses**
- CMPU 101 - Computer Science I: Problem-Solving and Abstraction
- CMPU 102 - Computer Science II: Data Structures and Algorithms
- CMPU 145 - Foundations of Computer Science
- CMPU 203 - Computer Science III: Software Design and Implementation
- CMPU 224 - Computer Organization
- CMPU 240 - Theory of Computation
- CMPU 241 - Analysis of Algorithms
- CMPU 331 - Compilers
- CMPU 334 - Operating Systems

**300-level electives (at least one for major)**
- CMPU 324 - Computer Architecture
- CMPU 353 - Bioinformatics
- CMPU 365 - Artificial Intelligence
- CMPU 368 - Computational Linguistics
- CMPU 375 - Computer Networks
- CMPU 377 - Parallel Computing
- CMPU 378 - Graphics
- CMPU 379 - Computer Animation: Art, Science and Criticism
- CMPU 395 - Advanced Special Topics

**Correlate-required courses**
- CMPU 101 - Computer Science I: Problem-Solving and Abstraction
- CMPU 102 - Computer Science II: Data Structures and Algorithms
- CMPU 145 - Foundations of Computer Science
- CMPU 240 or 241 - Theory of Computation or Analysis of Algorithms
- CMPU 3xx - Any other 200-level course
- CMPU 3xx - Any 300-level course

**200-level electives (not required for major)**
- CMPU 245 - Declarative Programming Models

**Intensives (at least one for major)**
- CMPU 310 - Topics in Virtualization
- CMPU 311 - Database Systems
- CMPU 312 - Applications of Artificial Intelligence
- CMPU 314 - Projects in Digital Media Production
- CMPU 315 - Computer Security

**Extra-departmental**
- MATH 221 - Linear Algebra

*At least two CMPU-200 level courses required for every CMPU-300 level course. †Prerequisites vary depending on topic.
<table>
<thead>
<tr>
<th>COURSE ID</th>
<th>TITLE</th>
<th>UNITS</th>
<th>SP</th>
<th>MAX</th>
<th>ENR</th>
<th>AVL</th>
<th>WL</th>
<th>GMD</th>
<th>YL</th>
<th>FR</th>
<th>PR</th>
<th>QA</th>
<th>PREREQ</th>
<th>FORMAT</th>
<th>DIV</th>
<th>DEPT</th>
<th>XLIST</th>
<th>DAYS</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPU-100-01</td>
<td>Programming with Data</td>
<td>1.0</td>
<td></td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MW</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-100-02</td>
<td>Programming with Data</td>
<td>1.0</td>
<td></td>
<td>6</td>
<td>14</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>FW</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-100-03</td>
<td>Programming with Data</td>
<td>1.0</td>
<td></td>
<td>7</td>
<td>13</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>1030AM-</td>
</tr>
<tr>
<td>CMPU-101-01</td>
<td>Problem-Solving &amp; Abstraction</td>
<td>1.0</td>
<td></td>
<td>2</td>
<td>19</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MW</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-101-02</td>
<td>Problem-Solving &amp; Abstraction</td>
<td>1.0</td>
<td></td>
<td>2</td>
<td>19</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>1100AM-</td>
</tr>
<tr>
<td>CMPU-101-03</td>
<td>Problem-Solving &amp; Abstraction</td>
<td>1.0</td>
<td></td>
<td>8</td>
<td>12</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MW</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-102-01</td>
<td>Data Structures and Algorithms</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>14</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MM</td>
<td>1200PM-</td>
</tr>
<tr>
<td>CMPU-102-02</td>
<td>Data Structures and Algorithms</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>7</td>
<td>17</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MM</td>
<td>1030AM-</td>
</tr>
<tr>
<td>CMPU-102-03</td>
<td>Data Structures and Algorithms</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>21</td>
<td>3</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MM</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-145-01</td>
<td>Foundations/Computer Science</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>20</td>
<td>4</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-145-02</td>
<td>Foundations/Computer Science</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>NR</td>
<td>QA</td>
<td>Y</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>0600AM-</td>
</tr>
<tr>
<td>CMPU-203-01</td>
<td>Software Design</td>
<td>1.0</td>
<td></td>
<td>20</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MW</td>
<td>1030AM-</td>
</tr>
<tr>
<td>CMPU-203-02</td>
<td>Software Design</td>
<td>1.0</td>
<td></td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>T</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-240-01</td>
<td>Theory of Computation</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-241-01</td>
<td>Analysis of Algorithms</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MM</td>
<td>0800AM-</td>
</tr>
<tr>
<td>CMPU-290-01</td>
<td>Community-Engaged Learning</td>
<td>0.5</td>
<td>SP</td>
<td>999</td>
<td>0</td>
<td>999</td>
<td>0</td>
<td>SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>OTH</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>SU</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-290-02</td>
<td>Community-Engaged Learning</td>
<td>1.0</td>
<td>SP</td>
<td>999</td>
<td>0</td>
<td>999</td>
<td>0</td>
<td>SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>OTH</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>SU</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-298-01</td>
<td>Independent Work</td>
<td>0.5</td>
<td>SP</td>
<td>999</td>
<td>0</td>
<td>999</td>
<td>0</td>
<td>SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>OTH</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>SU</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-298-02</td>
<td>Independent Work</td>
<td>1.0</td>
<td>SP</td>
<td>999</td>
<td>0</td>
<td>999</td>
<td>0</td>
<td>SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>OTH</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>SU</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-319-01</td>
<td>Modeling Minds, Brains, Behavior</td>
<td>1.0</td>
<td>SP</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>NR</td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>COGS</td>
<td>XCOGS</td>
<td>T</td>
<td>0310PM-</td>
</tr>
<tr>
<td>CMPU-311-01</td>
<td>Compilers</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>23</td>
<td>1</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MM</td>
<td>1200PM-</td>
</tr>
<tr>
<td>CMPU-365-01</td>
<td>Artificial Intelligence</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>13</td>
<td>11</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-377-01</td>
<td>Parallel Programming</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>20</td>
<td>4</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>TR</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-395-01</td>
<td>Robotics</td>
<td>1.0</td>
<td></td>
<td>24</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td></td>
<td></td>
<td>Y</td>
<td>CLS</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>MM</td>
<td>0130PM-</td>
</tr>
<tr>
<td>CMPU-399-01</td>
<td>Senior Independent Work</td>
<td>0.5</td>
<td>SP</td>
<td>999</td>
<td>0</td>
<td>999</td>
<td>0</td>
<td>SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>OTH</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>SU</td>
<td>0900AM-</td>
</tr>
<tr>
<td>CMPU-399-02</td>
<td>Senior Independent Work</td>
<td>1.0</td>
<td>SP</td>
<td>999</td>
<td>0</td>
<td>999</td>
<td>0</td>
<td>SU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
<td>OTH</td>
<td>NS</td>
<td>CMPU</td>
<td></td>
<td>SU</td>
<td>0900AM-</td>
</tr>
</tbody>
</table>

*For Prerequisite approval: See
Vassar Computer Science Course Map

Major-required courses
- CMPU 101 - Computer Science I: Problem-Solving and Abstraction
- CMPU 102 - Computer Science II: Data Structures and Algorithms
- CMPU 145 - Foundations of Computer Science
- CMPU 203 - Computer Science III: Software Design and Implementation
- CMPU 224 - Computer Organization
- CMPU 240 - Theory of Computation
- CMPU 341 - Analysis of Algorithms
- CMPU 331 - Compilers
- CMPU 334 - Operating Systems

300-level electives (at least one for major)
- CMPU 324 - Computer Architecture
- CMPU 353 - Bioinformatics
- CMPU 365 - Artificial Intelligence
- CMPU 368 - Computational Linguistics
- CMPU 375 - Computer Networks
- CMPU 377 - Parallel Programming
- CMPU 378 - Graphics
- CMPU 379 - Computer Animation: Art, Science and Criticism
- CMPU 395 - Advanced Special Topics

Correlate-required courses
- CMPU 101 - Computer Science I: Problem-Solving and Abstraction
- CMPU 102 - Computer Science II: Data Structures and Algorithms
- CMPU 145 - Foundations of Computer Science
- CMPU 240 or 241 - Theory of Computation or Analysis of Algorithms
- CMPU 2xx - Any other 200-level course
- CMPU 3xx - Any 300-level course

200-level electives (not required for major)
- CMPU 245 - Declarative Programming Models

Intensives (at least one for major)
- CMPU 310 - Topics in Virtualization
- CMPU 311 - Database Systems
- CMPU 312 - Applications of Artificial Intelligence
- CMPU 314 - Projects in Digital Media Production
- CMPU 315 - Computer Security

Extra-departmental
- MATH 221 - Linear Algebra
Try them out!

If you keep going with the CS major sequence, you work your way up to some really exciting courses, including…

* CMPU 240  Theory of Computation
* CMPU 366  Computational Linguistics

And, you know, probably some cool courses I don’t teach as well!
Further reading
"THIS IS THE BEST BOOK ON COMPUTERS I HAVE EVER READ."
—PETER THOMAS, NEW SCIENTIST

THE PATTERN ON THE STONE

THE SIMPLE IDEAS THAT MAKE COMPUTERS WORK

W. DANIEL HILLIS
That’s it!
We’ll meet some time during study week to review for Exam 2, working through practice problems and answering your questions.
Acknowledgments

This lecture incorporates material from:

Frederick P. Brooks, Jr, *The Mythical Man-Month*

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

Miram Posner, UCLA

Marc Smith, Vassar College