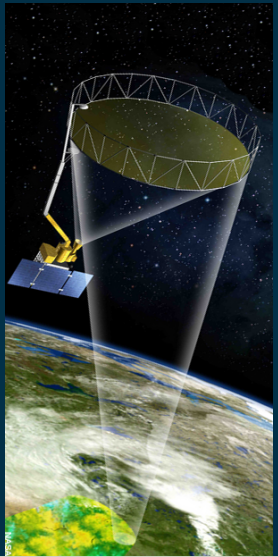
GPHY491/489: Programming for GIS

K. Arthur Endsley
Numerical Terradynamic Simulation Group
W.A. Franke College of Forestry and Conservation

January 18, 2024

UNIVERSITY OF MONTANA

Introductions



Hi, My Name is Arthur!

- Taught programming at Lawrence Berkeley National Labs, NASA Langley, the Federal Reserve Board...
- Technical lead for NASA Soil Moisture Active Passive (SMAP) satellite mission's Level 4 Carbon (L4C) Product: 1-km resolution, global coverage, over 500 million pixels!
- My research: How do the plants and soil respond to short-term climate change like floods and drought?

UNIVERSITY OF MONTANA

Introductions

Teaching Assistant: Ethan Shafron

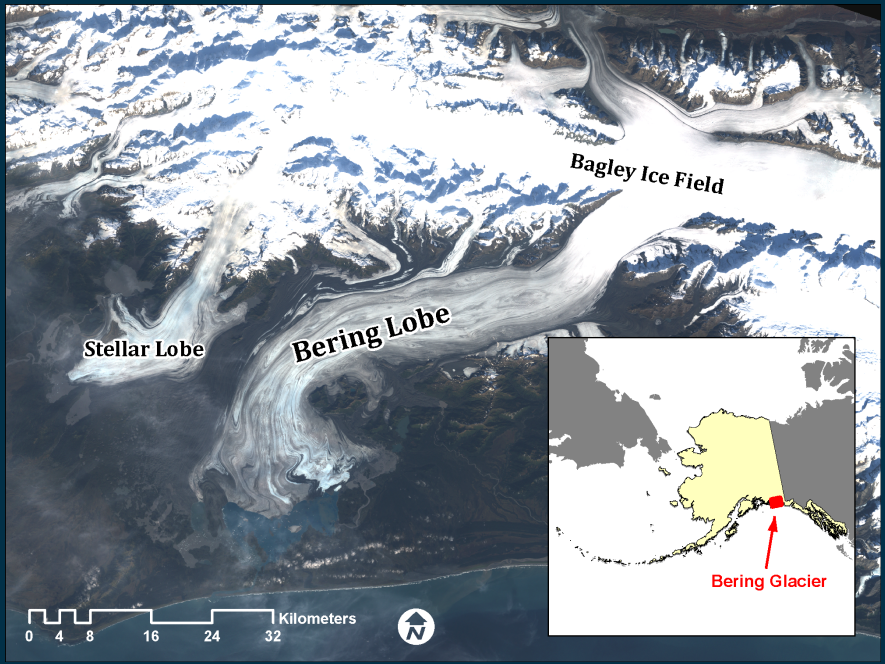
- Worked as remote sensing analyst at UVM, ASU, and UM
- Current PhD student at UM and intern with the imaging spectroscopy group at NASA's Jet Propulsion Lab
- Research: Mapping plant traits using hyperspectral imagery and linking plant carbon allocation to remote sensing

UNIVERSITY OF MONTANA

Outline

- GIS programming workflows: An example
- Some role-playing: How can programming improve your work?
- Thinking like a Computer Scientist
- Course overview

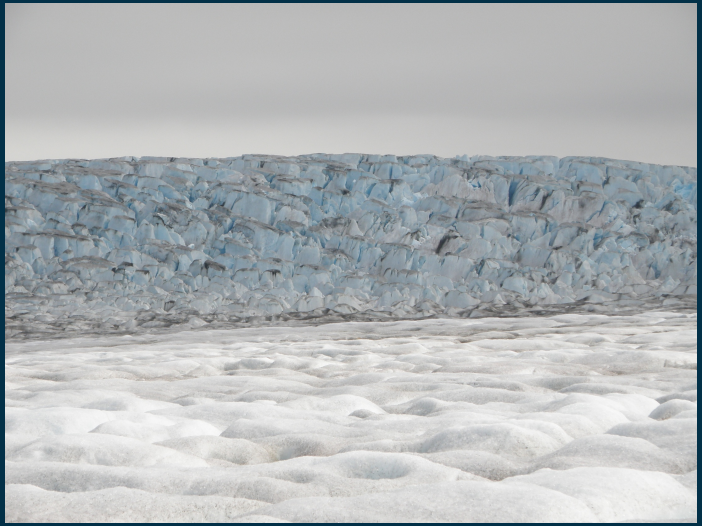
UNIVERSITY OF MONTANA


Bagley Ice Field
Stellar Lobe
Bering Lobe
Bering Glacier
0 4 8 16 24 32
Kilometers

Motivation: Ice Motion Tracking

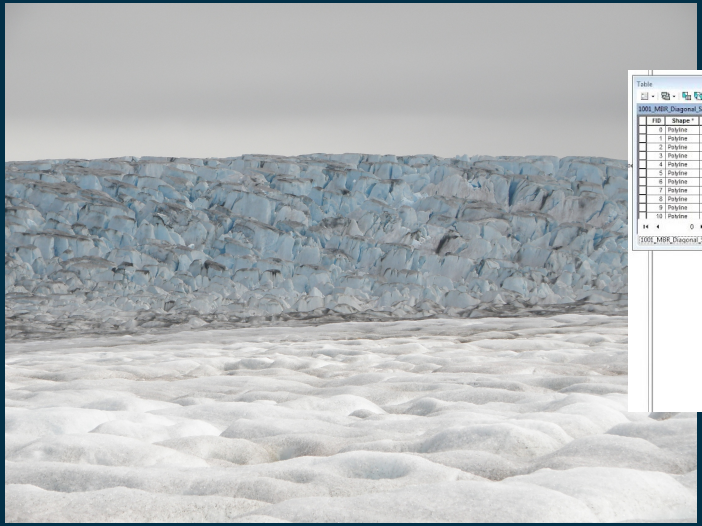
UNIVERSITY OF MONTANA

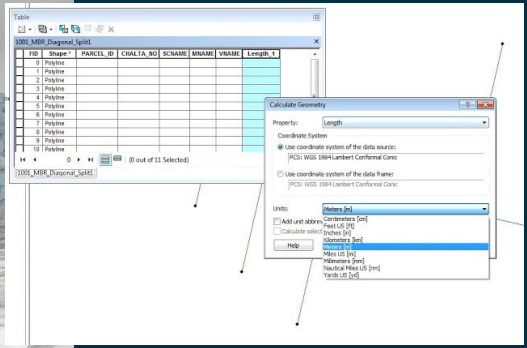
Motivation: Ice Motion Tracking (2)



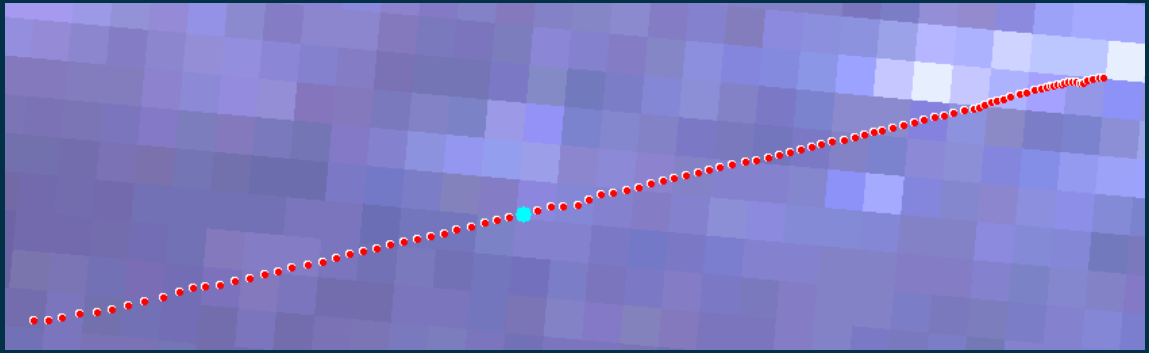
UNIVERSITY OF MONTANA

Motivation: Ice Motion Tracking (2)





Motivation: Ice Motion Tracking (3)



```
import scipy
scipy.signal.filtfilt(1, 2, gps_coords)
```

Live-Action Role-Playing: GIS Edition

Pair up! There are two roles: **Manager and Analyst.** Take 5 minutes in one role for Prompt A, then **switch roles** and read Prompt B. The Analyst should describe how they would achieve the goal using ArcGIS Desktop.

Questions to consider:

- How will the data be handled?
- What specific tools would you use, in what order?
- How long will it take?
- What are the possible sources of error?

UNIVERSITY OF MONTANA

So: Why Programming?

- Humans are terrible at repetitive tasks.

UNIVERSITY OF MONTANA

So: Why Programming?

- Humans are terrible at repetitive tasks.
- A computer program or script is a documentation of your workflow.

UNIVERSITY OF MONTANA

So: Why Programming?

- Humans are terrible at repetitive tasks.
- A computer program or script is a documentation of your workflow.
- Computer code is transferable and re-useable; it can be used to verify that your analysis was done correctly and to obtain the same result.

UNIVERSITY OF MONTANA

So: Why Programming?

- Humans are terrible at repetitive tasks.
- A computer program or script is a documentation of your workflow.
- Computer code is transferable and re-useable; it can be used to verify that your analysis was done correctly and to obtain the same result.
- **$$$:** U.S. average annual salary (2023):[1]
  - "GIS Analyst:" $72,530
  - "GIS Programmer:" $86,743

[1] ZipRecruiter.com

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist

Computational thinking is a *problem-solving* activity.[1]

[1] Cynthia Selby & John Woolard (2013)

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist

Computational thinking is a *problem-solving* activity.

Abstraction: Representing only what is essential.

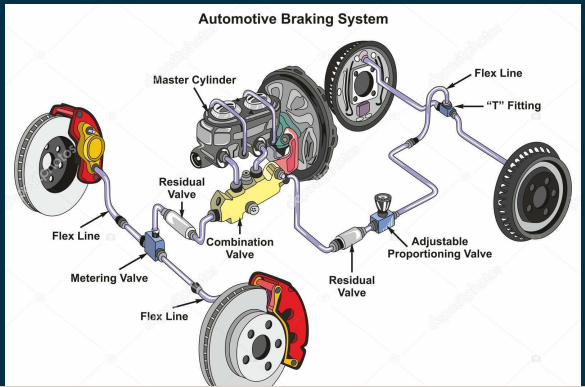
UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist

Computational thinking is a *problem-solving* activity.

Abstraction: Representing only what is essential.

Decomposition: Decomposing a complex problem or system into manageable parts.



Thinking Like a Computer Scientist

Computational thinking is a *problem-solving* activity.

Abstraction: Representing only what is essential.

Decomposition: Decomposing a complex problem or system into manageable parts.

Algorithms: Logical and ordered instructions for carrying out a task.

- Order and precedence; what is required before the next thing?
- Design the fewest number of steps; remove unnecessary steps

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist

Computational thinking is a *problem-solving* activity.

Abstraction: Representing only what is essential.

Decomposition: Decomposing a complex problem or system into manageable parts.

Algorithms: Logical and ordered instructions for carrying out a task.

Debugging and Continuous Improvement

- Detect and identify errors
- Start with an initial, acceptable solution
- Then iteratively refine the solution, as needed

Valerie Shute et al. (2017)

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist

Computational thinking is a *problem-solving* activity.

Abstraction: Representing only what is essential.

Decomposition: Decomposing a complex problem or system into manageable parts.

Algorithms: Logical and ordered instructions for carrying out a task.

Debugging and Continuous Improvement

Collaboration, Reflection, and Feedback

Valerie Shute et al. (2017)

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (2)

Don’t be afraid to experiment!

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (2)

Don’t be afraid to experiment!


```
pyplot.scatter(data)
```

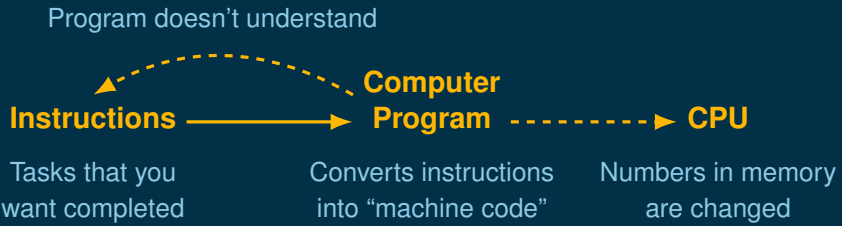
```
array.min()
```

You should be asking yourself:

- “Is there a pyplot.histogram() or pyplot.lineplot() function?”
- “Will array.max() calculate the maximum?”

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (3)



(CPU: Central Processing Unit)

Thinking Like a Computer Scientist (2)

UNIVERSITY OF MONTANA

Assembly Language

```
DOSSEG
.MODEL TINY
.DATA
TXT DB "Hello, world!$"
.CODE
START:
  MOV ax, @DATA
  MOV ds, ax
  MOV ah, 09h
  MOV dx, OFFSET TXT
  INT 21h
  MOV AX, 4C00h
  INT 21h
END START
```

UNIVERSITY OF MONTANA

Assembly Language

```
DOSSEG
.MODEL TINY
.DATA
TXT DB "Hello, world!$"
.CODE
START:
  MOV ax, @DATA
  MOV ds, ax
  MOV ah, 09h
  MOV dx, OFFSET TXT
  INT 21h
  MOV AX, 4C00h
  INT 21h
END START
```

2nd-Generation Language (C)

```
include <stdlib.h>
include <stdio.h>

int main(void)
{
  printf("Hello, world!\n");
  return EXIT_SUCCESS;
}
```

Assembly Language

```
DOSSEG
.MODEL TINY
.DATA
TXT DB "Hello, world!$"
.CODE
START:
  MOV ax, @DATA
  MOV ds, ax
  MOV ah, 09h
  MOV dx, OFFSET TXT
  INT 21h
  MOV AX, 4C00h
  INT 21h
END START
```

2nd-Generation Language (C)

```
include <stdlib.h>
include <stdio.h>

int main(void)
{
  printf("Hello, world!\n");
  return EXIT_SUCCESS;
}
```

3rd-Generation Language (Python)

```
print("Hello, world!\n")
```

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (3)

Compiled Languages

e.g., C, Java

- Code is *compiled* into machine code (binary) *before* the program is executed

Interpreted Languages

e.g., Python, R

- Code is translated into machine code automatically when the program is run

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (3)

Compiled Languages

e.g., C, Java

- Code is *compiled* into machine code (binary) *before* the program is executed
- Programmer must specify data types, how to allocate storage (memory), ...

Interpreted Languages

e.g., Python, R

- Code is translated into machine code automatically when the program is run
- Interpreter figures out what data types and memory are required

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (3)

Compiled Languages

e.g., C, Java

- Code is *compiled* into machine code (binary) *before* the program is executed
- Programmer must specify data types, how to allocate storage (memory), ...
- Compiled programs can be very fast: Programs executed as binary

Interpreted Languages

e.g., Python, R

- Code is translated into machine code automatically when the program is run
- Interpreter figures out what data types and memory are required
- Never as fast: Programs are executed as low-level instructions

UNIVERSITY OF MONTANA

Thinking Like a Computer Scientist (3)

Compiled Languages

e.g., C, Java

- Code is *compiled* into machine code (binary) *before* the program is executed
- Programmer must specify data types, how to allocate storage (memory), ...
- Compiled programs can be very fast: Programs executed as binary
- Can take a long time to write and debug a program

Interpreted Languages

e.g., Python, R

- Code is translated into machine code automatically when the program is run
- Interpreter figures out what data types and memory are required
- Never as fast: Programs are executed as low-level instructions
- Much easier to learn; programs can be written quickly

UNIVERSITY OF MONTANA

Who is a Computer Programmer?

Who is a Computer Programmer?

Hedy Lamarr (1914-2000)

- Invented frequency-hopping technology, initially for radio-guided torpedoes
- Same technology later used in cellular phones
- **Self-taught!** Collaborated with friend George Antheil to use a player-piano reel for timing the changes in frequency

UNIVERSITY OF MONTANA

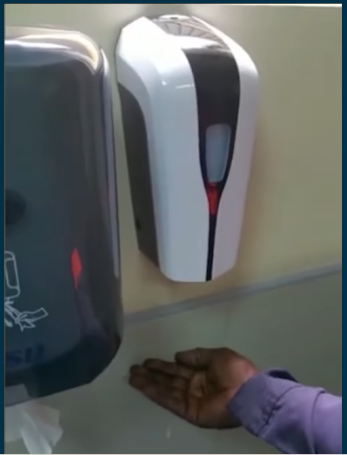
Who is a Computer Programmer? (2)

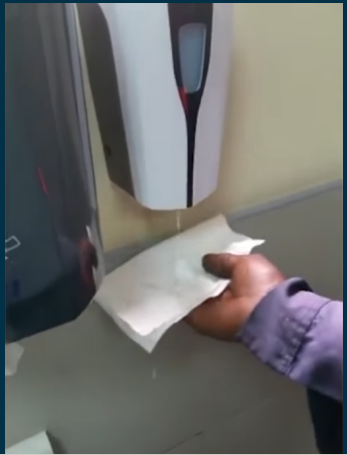
The most famous faces in the tech industry tend to look very similar...
and they're making the world less fair and less free.

UNIVERSITY OF MONTANA

Who is a Computer Programmer? (2)

The most famous faces in the tech industry tend to look very similar...
and they're making the world less fair and less free.





UNIVERSITY OF MONTANA

Who is a Computer Programmer? (2)

The most famous faces in the tech industry tend to look very similar…
and they're making the world less fair and less free.

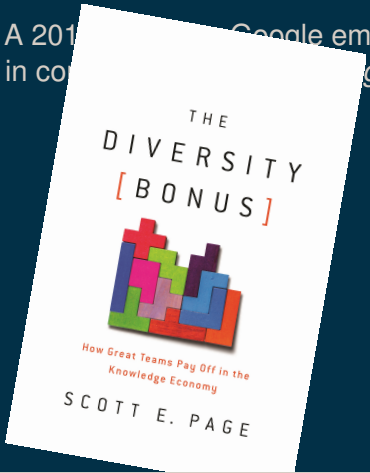
A 2017 memo by Google employee James Damore claimed that participation and success in computer science is *biologically determined.*

UNIVERSITY OF MONTANA

Who is a Computer Programmer? (2)

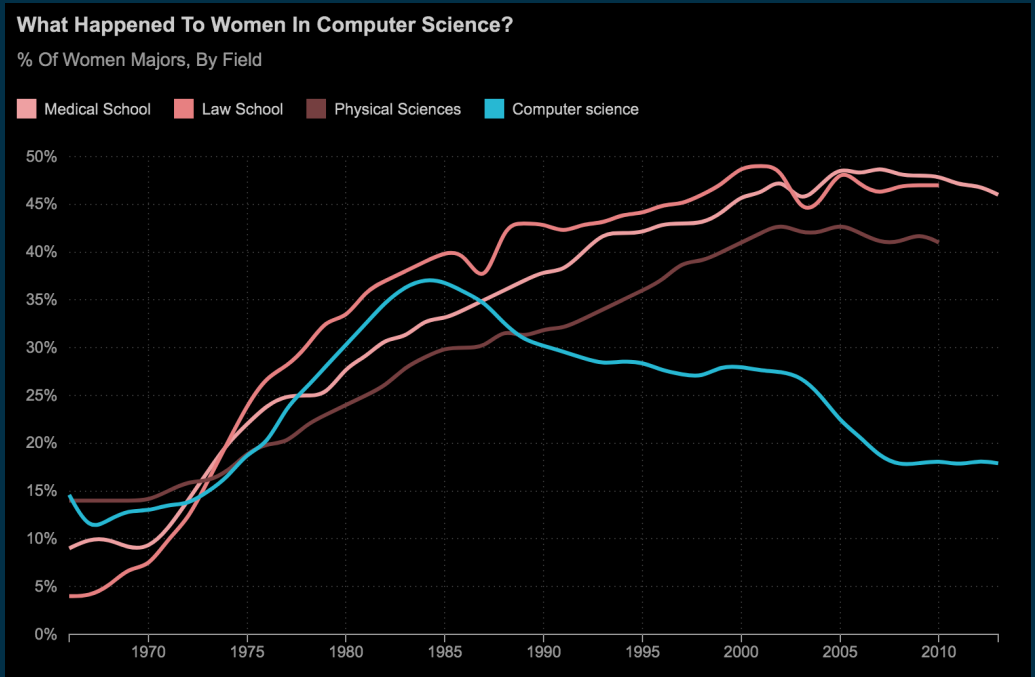
The most famous faces in the tech industry tend to look very similar…
and they're making the world less fair and less free.

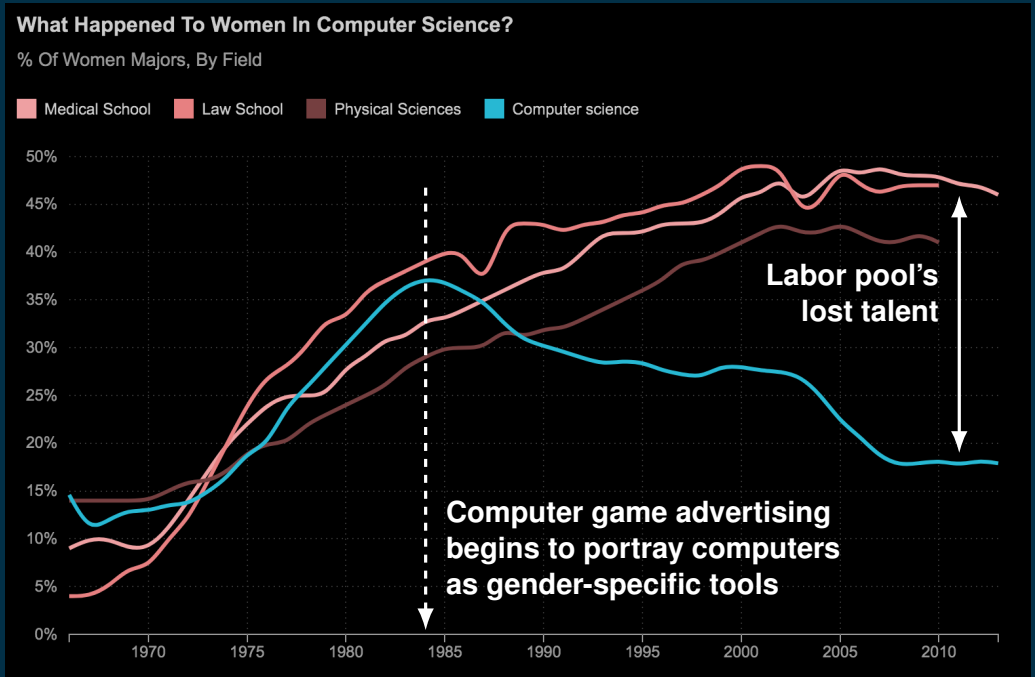
A 201 Google employee James Damore claimed that participation and success in co *gically determined.*



"As the initial pool of problem solvers becomes large, the best-performing [programmers] necessarily become similar...Their relatively greater ability is *more than offset* by their lack of problem-solving diversity."

UNIVERSITY OF MONTANA


What Happened To Women In Computer Science?
% Of Women Majors, By Field
Medical School
Law School
Physical Sciences
Computer science
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
1970
1975
1980
1985
1990
1995
2000
2005
2010


What Happened To Women In Computer Science?
% Of Women Majors, By Field
Medical School
Law School
Physical Sciences
Computer science
50%
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
1970
1975
1980
1985
1990
1995
2000
2005
2010
Labor pool's lost talent
Computer game advertising begins to portray computers as gender-specific tools

This is a challenging course, but you already have all the tools you need to be successful.

UNIVERSITY OF MONTANA

Course Overview

UNIVERSITY OF MONTANA

How You Will Learn

We'll use two learning strategies in this course:

1. **Hands-on-Keyboards** with real data
2. **Peer Programming** with real data

UNIVERSITY OF MONTANA

How You Will Learn

We'll use two learning strategies in this course:

1. **Hands-on-Keyboards** with real data
2. **Peer Programming** with real data

It's not Chemistry, it's Carpentry!

UNIVERSITY OF MONTANA

What You Will Learn

- **Python programming** (currently the most popular language in the world[1])
- **R for data analysis**

[1] https://www.tiobe.com/tiobe-index/

UNIVERSITY OF MONTANA

What You Will Learn (2)

How to choose the best language for the job:

Python

- **Higher performance**
- **Raster and array data processing**
- Batch processing of multiple files
- Machine learning
- Plotting large raster datasets
- Process-based or multi-criteria modeling (e.g., habitat suitability)
- Cellular automata/ Agent-based models
- Creating your own algorithm

R

- **Vector data analysis**
- Working with an Attribute Table
- **Plotting vector data and coarse-resolution raster data**
- Geostatistics
- Spatial point pattern analysis
- Spatial autoregressive models
- Using algorithms written by others

UNIVERSITY OF MONTANA

How You Will Be Evaluated

Lab Exercises: 60%

For graduate students or for undergraduate extra credit:

Midterm Project: 20%

Final Project: 20%

UNIVERSITY OF MONTANA

For Next Time

Make sure to sign up for *both* GPHY 491 and GPHY 489 (Lab).

Lab meets on Wednesdays at 9:00a in Stone Hall 106!

Please listen and read!

- NPR Planet Money Podcast: **"When Women Stopped Coding"**
- Dynamic Ecology blog post: **"Stereotype threat: A summary of the problem"**

UNIVERSITY OF MONTANA