

Algorithms and Data Structures for Data Science

Introduction

CS 277

January 17, 2023

Brad Solomon



UNIVERSITY OF
ILLINOIS
URBANA - CHAMPAIGN

Department of Computer Science



Learning Objectives

Get to know each other through brief introductions

Discuss class logistics and expectations

Begin reviewing programming and Python fundamentals

Who am I?

Data Structures!

Brad Solomon



Teaching Assistant Professor, Computer Science

2233 Siebel Center for Computer Science

Email: bradsol@illinois.edu

Office Hours:

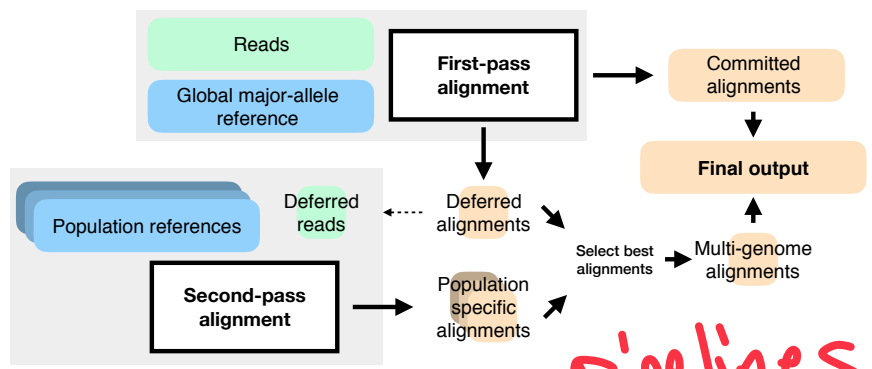
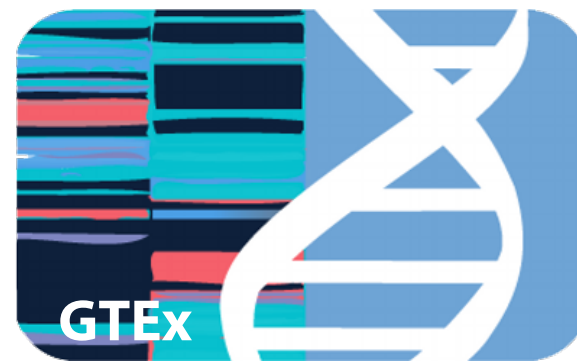
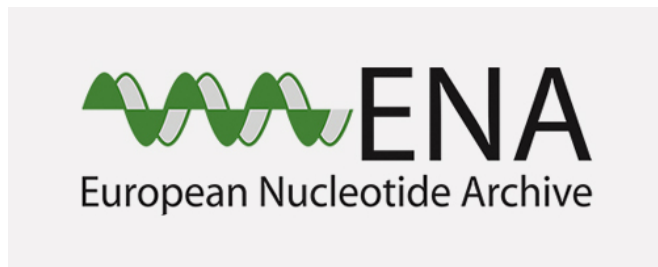
Thursdays, 11:00 - 12:00 PM

(Details are on the website)

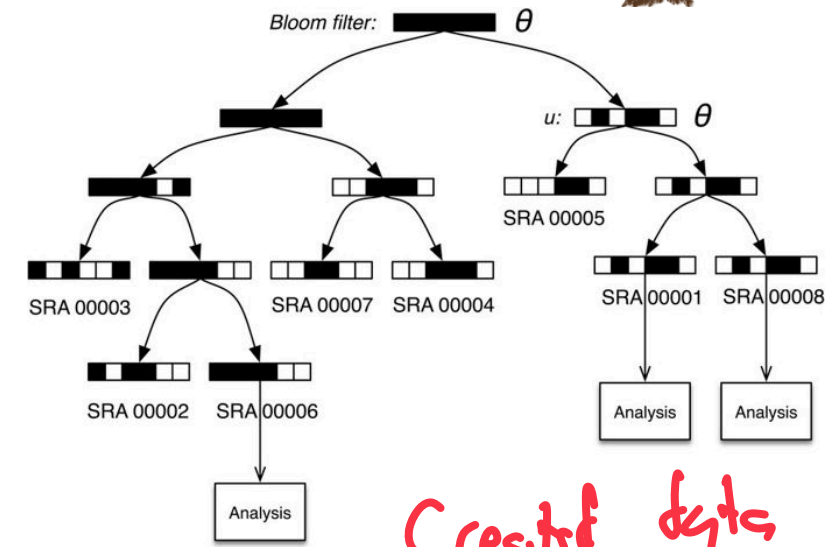
... can also make an appointment directly

Open door office policy!

Who am I?



Pipelines



Created data structures

Fast search of thousands of short read sequencing experiments. Brad Solomon and Carl Kingsford. *Nature Biotech* 2016

Reducing reference bias using multiple population reference genomes. Chen et al. *Genome Biology* 2021



Course Staff Introductions

Who are you?

Feel free to introduce yourself on Piazza:



<https://piazza.com/illinois/spring2024/cs277>

Freely talk with each other on Discord**



↳ Monday?

Introduce yourself when asking a question



Stop by my office hours at some point this semester!

What will you get out of this class?

Navigate, organize, and run moderately complex Python projects

The screenshot shows the JupyterLab interface for a workspace named 'lab_fundamentals'. The top bar includes the Jupyter logo, the workspace name, and a 'Logout' button. Below the top bar is a menu with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A 'Not Trusted' warning and 'Python 3 (ipykernel)' are also visible. The main area contains a code cell with the following content:

```
In [ ]: #grade

# INPUT:
# A string (checkin)
# A string (checkout)
# OUTPUT:
# A float storing the number of minutes between checkin and checkout time.
def getTotalTime(checkin, checkout):
    pass
```

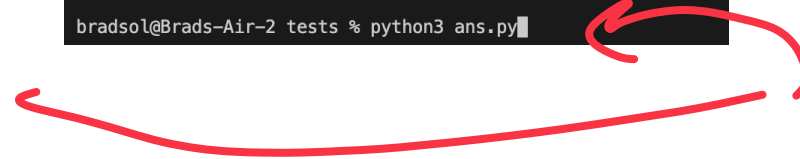
A hint is displayed above the code cell: "Hint: This function relies on your knowledge on how to convert variables from one type to another (Python will allow you to add two strings together but it certainly won't help you solve the problem if you do so). It also requires string manipulations either using built-in functions like `split` or writing your own string parse with the knowledge that the format is fixed in size (Ex: 8:00 AM would be 08:00:00 and 6:00 PM would be 18:00:00)."

```
1 # Part 1 - conditions
2 # getGrade
3 # INPUT:
4 # score is a number representing a student's score from
5 # OUTPUT:
6 # A single character with the letter grade based on the
7 def getGrade(score):
8     # ranges based on https://courses.grainger.illinois.edu
9     if score > 600:
10        return "D"
11    if score > 700:
12        return "C"
13    elif score > 800:
14        print("B")
15    elif score > 900:
16        return "A"
17    else:
18        return "F"
19
20 # Part 2 - Lists
21 # checkSorted
22 # Given a list, check if the number is sorted in ascending
```

The bottom of the screenshot shows a terminal window with the command: `bradso1@Brads-Air-2 tests % python3 ans.py`

The screenshot shows a file explorer view of a workspace. The workspace contains the following files and folders:

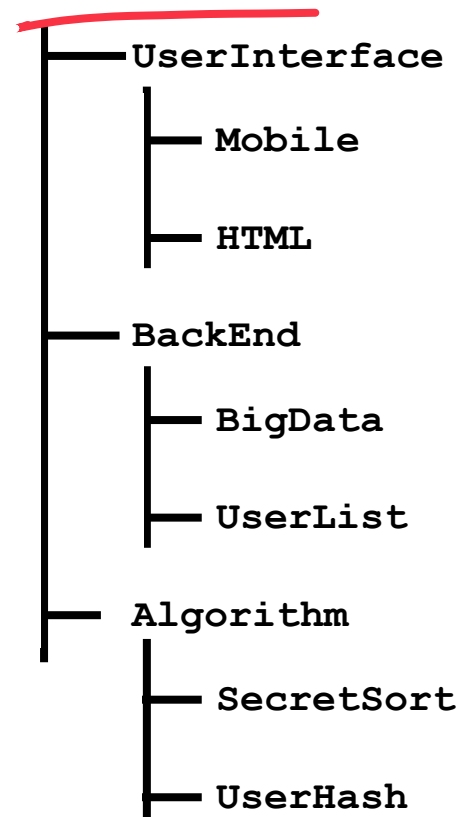
- workspace
 - data
 - output
 - test.gif
 - DataParse.py
 - DataParse.pyc
 - GraphVis.py
 - GraphVis.pyc
 - Grid2D.py
 - main.py



Why should you care?

Navigate, organize, and run moderately complex Python projects

TeamProject



```
#include <vector>

#include "util/coloredout.h"

#include "cs225/point.h"

using std::vector;
using std::string;
using std::ostream;
using std::cout;
using std::endl;
```

```
bradsol@Brads-Air-2 code % python3 debug.py
Use print statements, break statements, and return statements to debug errors!
We've given a few examples of print statements below.
Brad got 799 points and got a D
Harsh got 800 points and got a D

Now let's check ascending order
False

Traceback (most recent call last):
  File "debug.py", line 114, in <module>
    print(removeOdds(l1))
  File "debug.py", line 46, in removeOdds
    val = list_ld[i]
IndexError: list index out of range
```



FASTQ-to-FASTA

```
$ fastq_to_fasta -h
usage: fastq_to_fasta [-h] [-r] [-n] [-v] [-z] [-i INFILE] [-o OUTFILE]

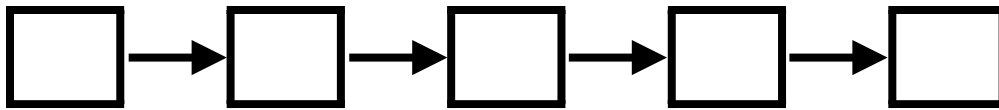
version 0.0.6
[-h] = This helpful help screen.
[-r] = Rename sequence identifiers to numbers.
[-n] = keep sequences with unknown (N) nucleotides.
      Default is to discard such sequences.
[-v] = Verbose - report number of sequences.
      If [-o] is specified, report will be printed to STDOUT.
      If [-o] is not specified (and output goes to STDOUT),
      report will be printed to STDERR.
[-z] = Compress output with GZIP.
[-i INFILE] = FASTA/Q input file. default is STDIN.
[-o OUTFILE] = FASTA output file. default is STDOUT.
```

Taken from FastX-Toolkit (hannonlab.cshl.edu)

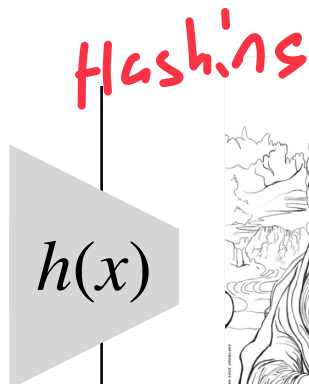
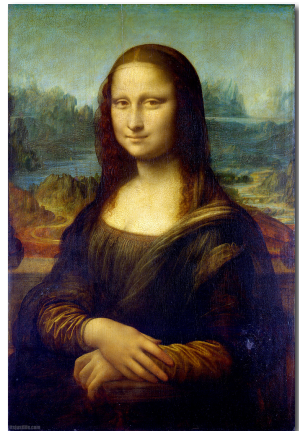
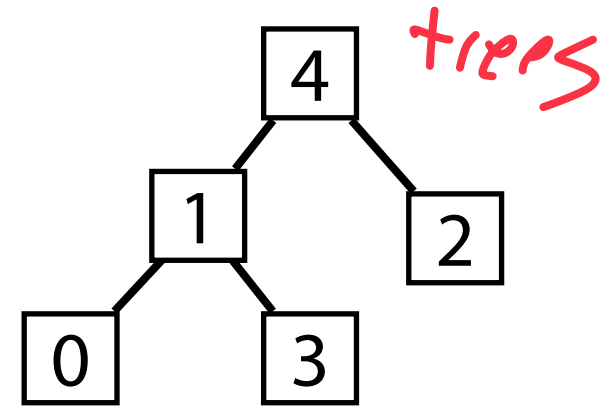


What will you get out of this class?

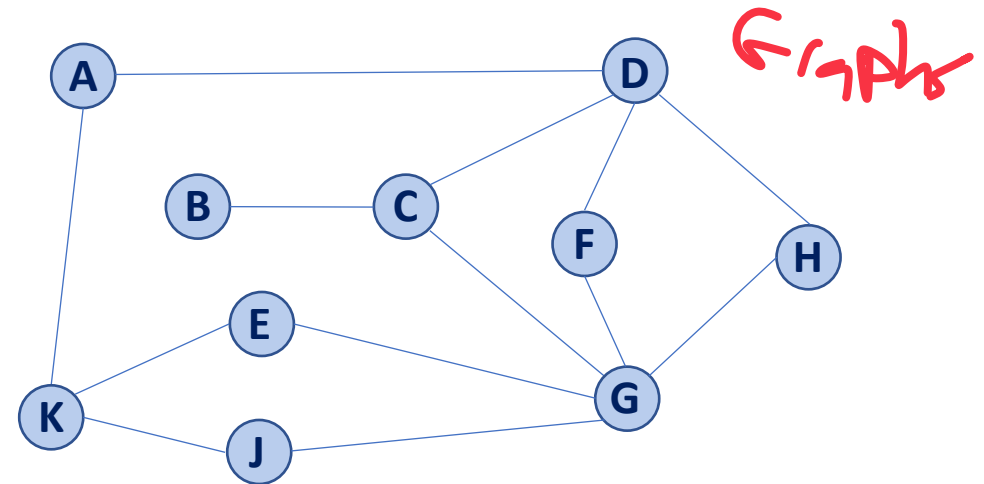
Understand foundational data structures and algorithms



Lists



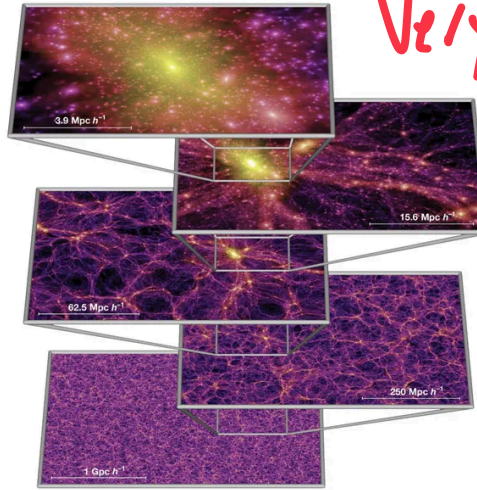
Hashing



Graphs

Why should you care?

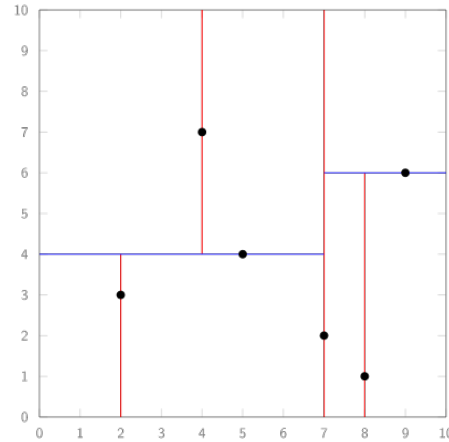
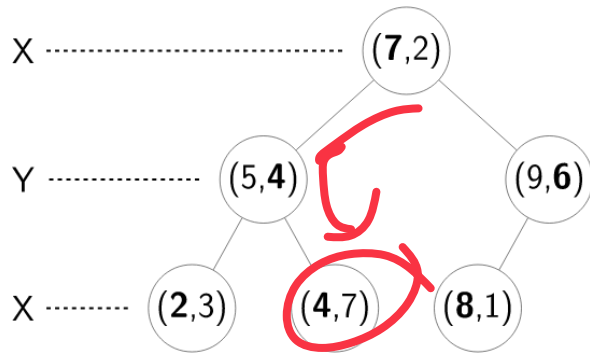
Understand foundational data structures and algorithms



Very Big

Petabytes

Trees are good



Internal Structures

```
Query: 161 atatcaccacgtcaaaggtgactccaactcca---ccactccattttgttcagataatgc 217
      |||
Sbjct: 481 atatcaccacgtcaaaggtgactccaact-tattgatagtgttttatgttcagataatgc 539
```



What will you get out of this class? *ultimate goal*

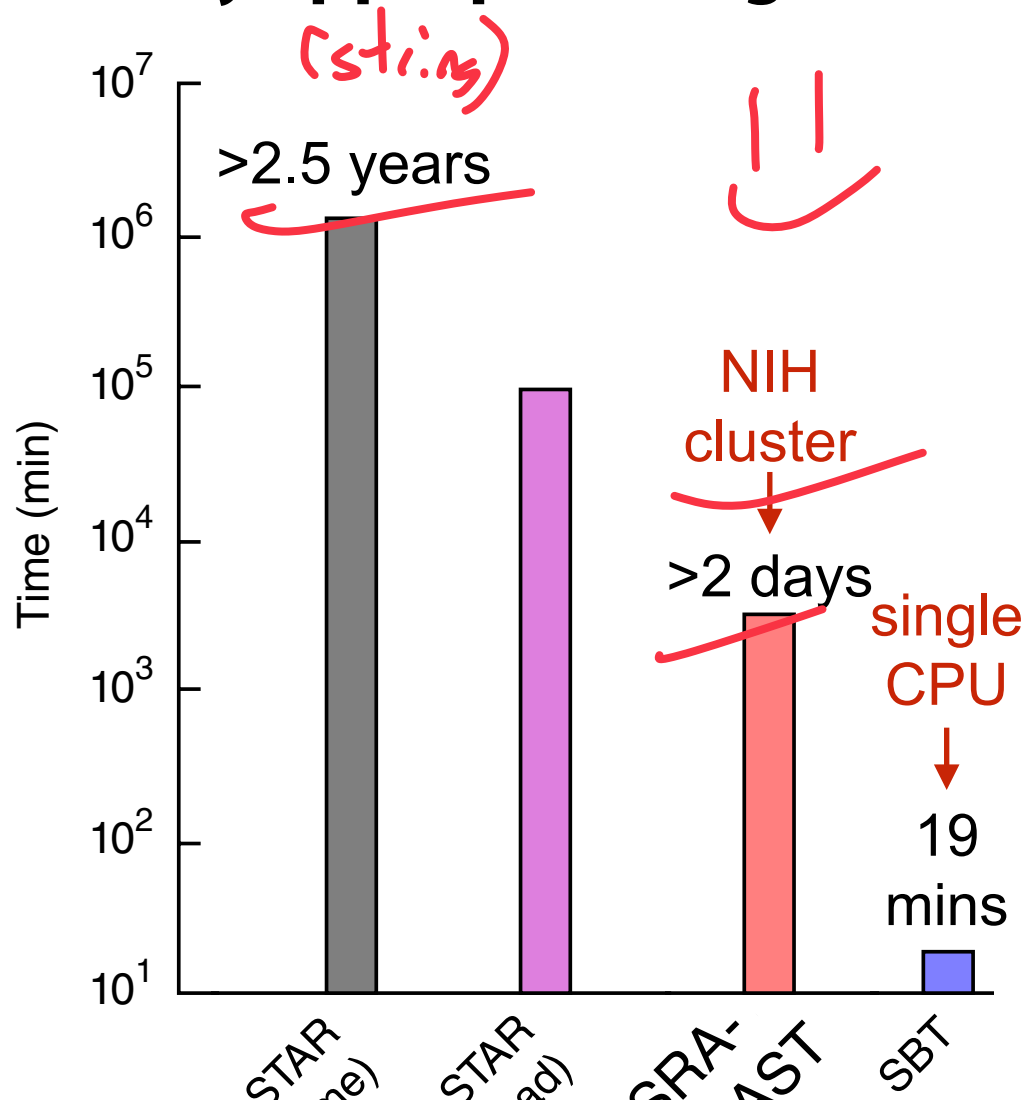
Justify appropriate algorithms for data science problems

Decompose problem into supporting data structures

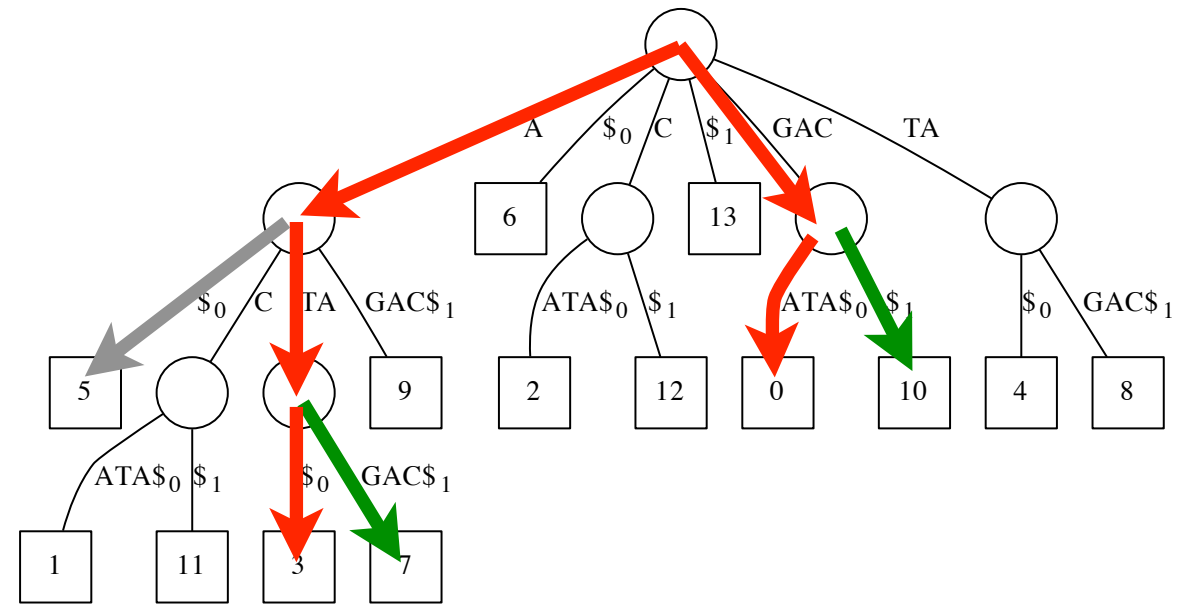
Analyze efficiency of implementation choices

Why should you care?

Justify appropriate algorithms for data science problems



Find all overlap: GACATA vs ATAGAC



CS 225 vs CS 277

The prerequisite requirements for CS 277 are very different

The learning goals and content are also very different.

CS 225 is necessary to enroll in many upper-level CS classes

Course Webpage



<https://courses.grainger.illinois.edu/cs277/sp2024/>



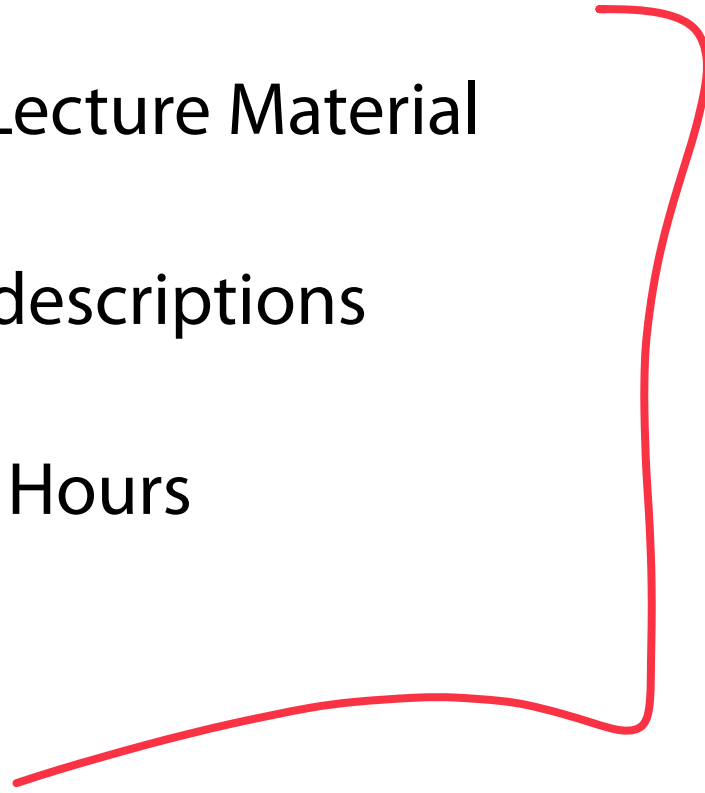
All course information and links can be found here!

Course Schedule and Lecture Material

Assignment links and descriptions

Piazza links and Office Hours

Syllabus



In-Lecture Course Expectations

Attendance is encouraged but not mandatory

Ask questions!



Participate in class exercises / labs



Out-of-lecture Course Expectations

Weekly assessments

Lab assignments are published **Friday** and due **Monday @ 11:59 PM**

Mini-projects deadlines are published with each project (~3 weeks)

Watch recorded lectures (if you missed in-person)

All lectures are published on Mediaspace:

<https://mediaspace.illinois.edu/channel/CS+277/225216063>

First assignment

Not be due Monday!

Grading

930 940 ?!?!
 A - A - A'?

Category	Contribution	Notes
<u>Mini-Projects</u>	300	75 points each x 4
<u>Labs</u>	300	25 points each x 12
<u>Exams</u> - 50	300	75 points each x 4
<u>Final</u>	<u>100</u>	+ 1 retake exam

Points	Grade
<u>900</u>	A-
<u>800</u>	B-
700	C-
<u>600</u>	D-
	F

50%

40%

↳ 1 hour + 50 minute retake

capture

!!

Mental Health

This class should be low-stress, medium work-load.

UIUC offers a variety of confidential services:

Counseling Center: 217-333-3704

610 East John Street Champaign, IL 61820

McKinley Health Center: 217-333-2700

1109 South Lincoln Avenue, Urbana, Illinois 61801

Diversity, Equity, and Inclusion



"If you witness or experience racism, discrimination, micro-aggressions, or other offensive behavior, you are encouraged to bring this to the attention of..."

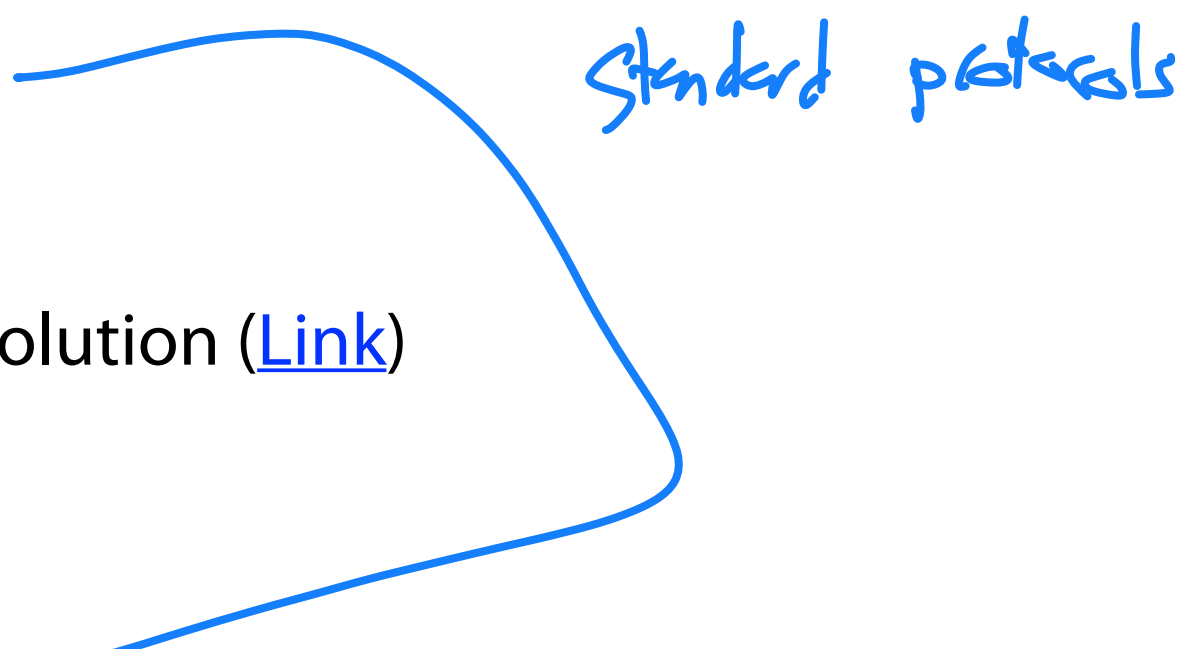
Course CAs / TAs

Faculty (me)

Campus Belonging Office ([Link](#))

The Office of Student Conflict Resolution ([Link](#))

CS CARES ([Link](#))



Class structure is under development!



Class size tripled from last iteration, course description changing

Frequent assessment will allow adjustments as needed

Learning Objective: Programming

Navigate, organize, and run moderately complex Python projects

Three sub-goals to be a better programmer:

- 1) Building up your 'programming toolbox' — *How many functional concepts*
↳ *Data Structures / Algorithms*
- 2) Thinking carefully about a problem
↳ *Break down a problem* ↳ *Debug a problem*
why I picked X
- 3) Choosing the right tool for the job → *Justification*

Programming Toolbox: Variables

What is a variable in Python? *Container / object*

Stores values

```
1 a="3"  
2 b=3  
3 c=3.0  
4 d=True  
5  
6 print(a + b)  
7  
8 print("3 + 3")  
9  
10 print(b + c)  
11  
12 print(c + d)  
13  
14 print(d)  
15  
16 print(d - d)  
17  
18
```

What information is necessary to define a variable?

↳ Name, value, data type

*Operation * ↪ absolutely true but...*

interface



Programming Toolbox: Variables

Everything in Python is an **object**

```
1 x = 1212
2
3
4
5
6
7
8
9
10
```

Var Name	X
----------	---



Type	integer
Value	1212
Ref Count	1

Programming Toolbox: Variables

Chunk of memory

Everything in Python is an **object**

Python

```
1 x = 1212
2
3 y = x
4
5
6
7
8
9
10
```

Var Name	X
----------	---

Var Name	Y
----------	---

Type	integer
Value	1212
Ref Count	1 2

Memory

↳ count of references

If ref count gets too big
↳ Problem?

If ref count = 0, what has happened?
- not declared
- overwritten

Programming Toolbox: Variables

Everything in Python is an **object**

```
1 x = 1212
2
3 y = x
4
5 y = 9000
6
7
8
9
10
```

Var Name	X
----------	---



Type	integer
Value	1212
Ref Count	1

Var Name	Y
----------	---



Type	integer
Value	9000
Ref Count	1

Programming Toolbox: Variables

Everything in Python is an **object**

garbage collected!

Type	integer
Value	1212
Ref Count	0

Type	integer
Value	12
Ref Count	1

Type	integer
Value	9000
Ref Count	1

```
1 x = 1212
2
3 y = x
4
5 y = 9000
6
7 x = 12
8
9
10
```

Var Name	X
----------	---



Var Name	Y
----------	---



Programming Choice: Data Type

Python has many built-in data types:

https://www.w3schools.com/python/python_datatypes.asp

```
1 string = "Hello World"
2
3 intv = 1
4
5 floatv = 1.0
6
7 listv = [1, 2.0, ":)"]
8
9 dictionary = {"Key" : "Value"}
10
11 boolean = True
12
13 setv = {1, 3, 5, 7, 9}
14
```



Programming Choice: Data Type

Which of the following will result in a variable `x` having the value 0.5?

```
1 # ** A **
2 x = 1 / 2
3
4 # ** B **
5 0.5
6
7 # ** C **
8 y = 1.0
9 x = y / 2
10
11 # ** D **
12 x == 0.5
13
14 # ** E **
15 2*x = 1
16
17 # ** F **
18 0.5 = x
```

int / int → 0.5 float! (one line of code)
← Not a declaration
float / int → float! x = 0.5 (works in other languages)
are not variable declarations

Of the valid solutions, which do we like the most?

Programming Choice: Data Type

Some objects are **mutable** — we can change their values after creation

```
1 x = [1,2,3,4,5]
2
3 print(id(x))
4
5 x[2]=0
6
7
8 print(id(x))
9
10 print(x)
```

0 1 2
↙

works!

[1, 2, 0, 4, 5]

Var Name	X
----------	---



Type	list
Value	...
Ref Count	1

—

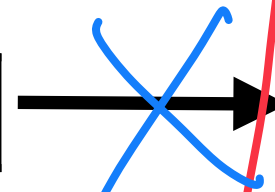
Programming Choice: Data Type

Some objects are **immutable** — you have to make a new object

```
1 x = "12345"  
2  
3 print(id(x))  
4  
5 x[2]=0  
6  
7  
8 print(id(x))  
9  
10 print(x)
```

x = "1212"

Var Name	X
----------	---



Type	String
Value	12345
Ref Count	1

cannot change!





Why do we care?

Imagine you have two datasets:

Dataset 1 is very large but fixed in size.

→ Immutable

Dataset 2 starts off small but grows to an unknown size.

→ Mutable

↳ grows to 1 MB

Would you rather have a **mutable** or **immutable** variable?

Small Scale doesn't matter

String

1 KB	TB
2 KB	TB
10 KB	TB

What do we care about when writing code?

↳ Simplicity (coding time / style) Organization / Readability

↳ consistency Speed of coding and correctness

↳ Speed of runtime / cycles on CPU Time

↓ ??

↳ Efficiency of computer resources Memory

What do we care about when writing code?



↳ My speed of parking



↳ How fast is system

↳ How big of a warehouse?

Tying it all back together...



For most simple programs or small datasets, efficiency doesn't really matter

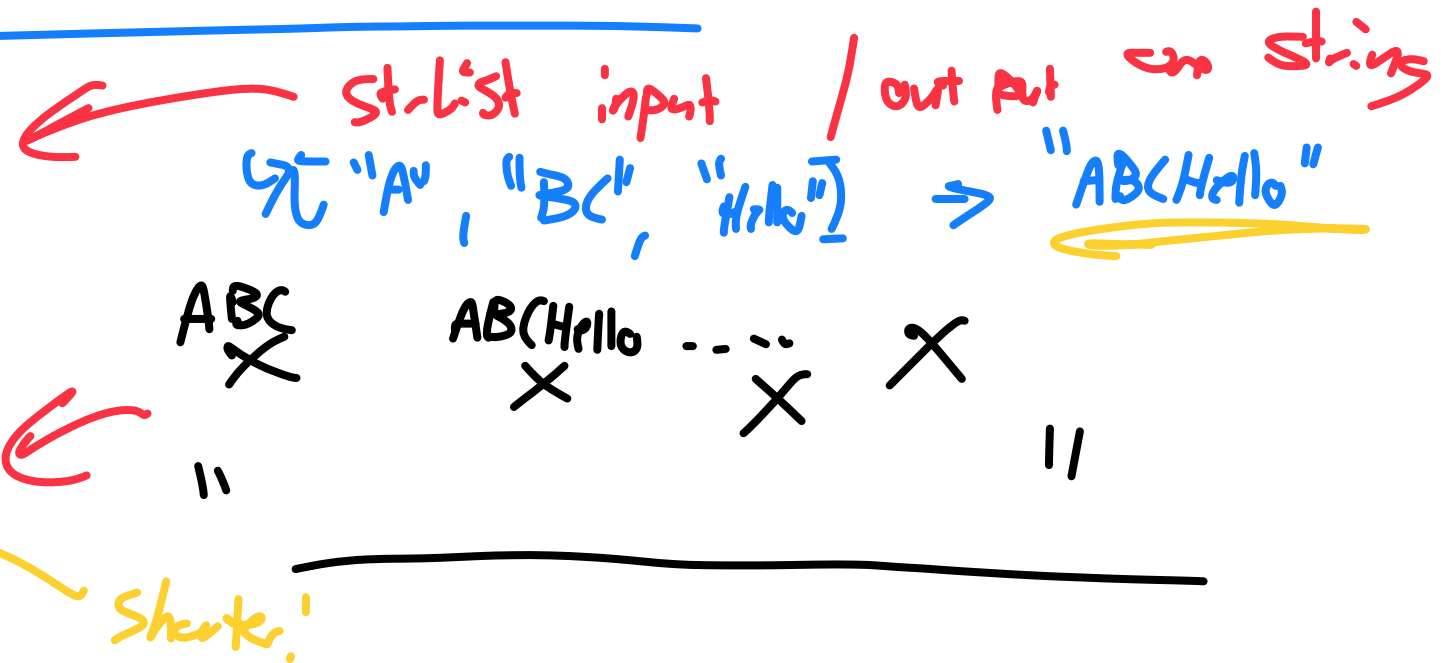
↳ time

↳ memory

But this will not always be true — especially in the data sciences!

```
1 def type1(strList):
2     out = ''
3     for s in strList:
4         out += s
5     return out
6
7 def type2(strList):
8     return ''.join(strList)
9
10
```

immutable



Programming Toolbox: Conditionals

Conditional statements control what blocks of code get run

```
1 num = 20
2
3 if num in [0,1,2,3,4]:
4     print("Top 5!")
5
6 elif num > 10:
7     print("num too large!")
8
9 elif num > 15:
10    print("will this ever get called?")
11
12 else:
13    print(num)
14
15
16
17
18
```

elif = else if

Programming Toolbox: Conditionals

Whats the difference?

```
1 num = 2
2
3 if num >= 1:
4     print("A")
5 elif num >= 2:
6     print("B")
7 elif num >= 3:
8     print("C")
```

linked

if

elif

else

```
1 num = 2
2
3 if num >= 1:
4     print("D")
5 if num >= 2:
6     print("E")
7 if num >= 3:
8     print("F")
```

each one indep

Programming Toolbox: Loops

We are often tasked with processing every item in a dataset.

We use loops to simplify our code structure.

```
1 for i in range(3):  
2     print(i)  
3  
4  
5  
6  
7  
8 count = 0  
9 while(count <= 2):  
10     print(count)  
11     count+=1  
12  
13  
14  
15  
16  
17  
18
```

For Loop: finite loop
↳ Know the number of iterations

While Loop: infinite loop
↳ Based on conditional

Programming Toolbox: Loops

There are a number of useful keywords for writing loops

```
1 count = 0
2
3 while(True):
4     if count % 2 == 0:
5         count+=1
6     else:
7         pass
8
9     if count > 10:
10        break
11    else:
12        count+=1
13        continue
14        count+=1
15    print('count: {}'.format(count))
16
17 print('count: {}'.format(count))
18
```

Pass: Placeholder

↳ Cleans up whitespace
skips line

Break: Escape the loop
move cur in code

Continue: Skip rest of current
iteration and continue at next
loop start

$i \rightarrow i+1$

What does this code print?



Count = 0 ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10
0 % 2 = 0

```
1 count = 0
2
3 while(True):
4     if count % 2 == 0:
5         count+=1
6     else:
7         pass
8
9     if count > 10:
10        break
11    else:
12        count+=1
13        continue
14        count+=1
15    print('count: {}'.format(count))
16
17 print('count: {}'.format(count))
18
```

3 ← 10
=

Print "11" once

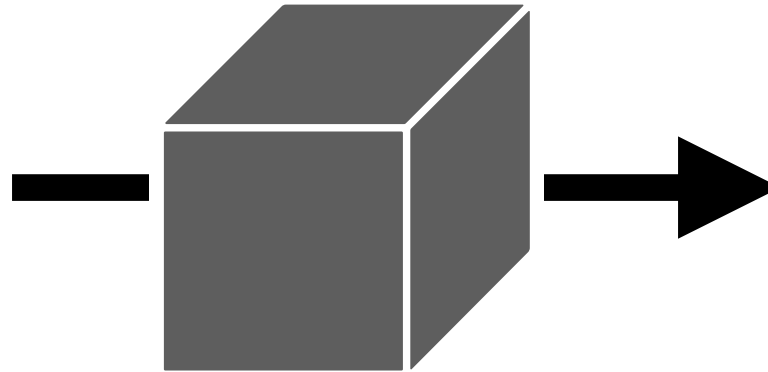


Programming Toolbox: Functions

Functions are the building blocks of programming

Input

Output



function

```
1 def type1(strList):  
2     out = ''  
3     for s in strList:  
4         out += s  
5     return out  
6  
7 def type2(strList):  
8     return ''.join(strList)  
9  
10
```

```
1 def mystery(inValue):  
2     return inValue + inValue  
3  
4  
5  
6  
7  
8  
9  
10
```


Programming Toolbox: Functions

You should always document the intended input and output.

```
1 # INPUT:
2 # A string (checkin)
3 # A string (checkout)
4 # OUTPUT:
5 # A float storing the number of minutes between checkin and checkout time.
6 def getTotalTime(checkin, checkout):
7
8
9
10 # Input:
11 # int/float (inValue)
12
13
14 # Double the value
15
16
17 def main(inValue):
18     return inValue + inValue
```

||
)

|||

arbitrary type with "+" implemented

|||

adding value to value

double

???

Programming Toolbox: Functions

Immutable variables created in a function have **local scope**

Mutable variables can be modified by functions



```
1 def scopeTest(inNum, inString, inList):
2     inNum = 3
3     inNum = 3
4     inString+="And After!"
5
6
7     inList.pop(-1)
8     inList.append(5)
9
10    x = 2
11    y = "Before! "
12    z = [1,2,3,4]
13
14    scopeTest(x,y,z)
15
16    print(x)
17    print(y)
18    print(z)
```

Programming Toolbox: Functions



Functions are **objects** (like everything in Python)

String "+"
float "+"
int "+"

```
1 # INPUT:
2 # Three integers (a, b, c)
3 # An optional function (f)
4 # OUTPUT:
5 # If f exists, return output of f(a, b, c). Else return defaultF(a,b,c)
6 def wrapperFunction(a, b, c, f=None):
7     if f == None:
8         return defaultF(a,b,c)
9     else:
10        return f(a,b,c)
11
12
13 if __name__ == '__main__':
14     wrapperFunction(5,3,2, add)
15
16
17     wrapperFunction(1,1,1, multiply)
18
```

In-Class Exercise: Slot Machine



Let's program the output of a slot machine!

\$500 = Four matching symbols

\$100 = Four of a color

\$50 = Three matching symbols in a row

\$10 = One of each symbol

Input is the state of machine
List contains strings
Output
→ int / float
cst (cents)

1) Get \$10

2) Break down our problem

↳ Look for 3 of a kind

↳ Identify counts of each object

↳ Look for 4 of a kind

↳ List
↳ List of cents

First Lab Friday!

Bring your laptop (first part will be going over installation instructions)

Lab will focus on how to break down a programming problem

Friday Foreshadowing: getTotalTime()

Given **HH:MM:SS** format, I want to know the exact difference between start and stop times in minutes. How would we approach this problem?