Algorithms and Data Structures for Data Science
Introduction and Python Basics

CS 277
Brad Solomon

August 23, 2021
Who am I?

Brad Solomon
Teaching Assistant Professor, Computer Science
2336 Siebel Center for Computer Science
Email: bradsol@illinois.edu

Office Hours:
Thursdays, 10:30-12:00 PM
Start with in-person — may move online
… can also make an appointment directly
Who is your CA?

**Harsh Deep**

Course Assistant

Email: hdeep2@illinois.edu

Will also answer questions on Campuswire

**Office Hours:**

Tuesday, 1 - 1:50 PM

Thursday 1 - 1:50 PM
Who are you?

Hope to get to know each of you this semester

You are encouraged to participate in class

What is your favorite type of ice cream?
Who are you?

Hope to get to know each of you this semester

You are encouraged to participate in class

You are encouraged to ask questions
Who are you?

Hope to get to know each of you this semester

You are encouraged to participate in class

You are encouraged to ask questions

There will be frequent surveys for feedback
What is this class about?

AGGCCACAGTGTATTATGACTG

AGGCCACAGTGGATTATGACTG

AAAAAAAAAGATGT-AAGTA

AAAAAAAAAGATGTAAAGTA

GAGG--TCAGATTTCACAGCCAC

GAGGGTCAGATTTCACAGCCAC

$T = \text{abaaba}$$

$h(x)$
What is this class about?

*Algorithms and Data Structures*

Logic and Relations | Lists, Stacks, Queues
Runtime Analysis | Trees
Searching and Sorting | Hashing and Hash Tables
Algorithm Design | Graphs

*Data Science*

Parsing and Processing | Storage and Compression
Visualization | Analysis
What will you get out of this class?

**Learning Objectives:**

Understand fundamental algorithms and data structures

Experience applying data structures, algorithms, and algorithm design principles to real world problems

Justify implementation choices based on theoretical or practical considerations

Build a foundation for future data science projects
Course website

https://courses.engr.illinois.edu/cs277/fa2021/

All course material is on the website:

Lectures

Assignments

Course Resources

Course Information
Course Campuswire (Link also on website)

https://campuswire.com/p/GB6460D77

Post questions and respond to polls

See course schedule

Meet and chat with peers
Course schedule

**Weekly Meeting:** MW 11:00 AM - 12:15 PM

**Daily Schedule:**

- Administrative Information
- 50-60 minute lecture

**Weekly Lab**: F 10:00 AM - 12:00 PM
Course workload

Course Total (1000 points):

12* lab assignments (10 points each)

6 machine problem sets (60 points each)

3 exams (100 points each)

1 final project (220 points)

Note: Course is brand new — may be reduced but will not be increased
Assignment policies

MPs and labs submitted through **Gradescope**

No extensions on labs; one lab dropped

Two ‘free’ 24-hour extensions on MPs

Additional extensions require a reason

Request an extension (link on website):

https://forms.gle/wEgY23HvqecBCdSS7
Course syllabus (link also on website)

https://courses.grainger.illinois.edu/cs277/fa2021/policy/syllabus/

Please read -- information includes:

- Course content
- Assessments and grading policies
- Late days
- Expectations
- Ethics
- etc
Mental Health

Your mental health is more important than class.

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…

Staff (Harsh Deep — the course CA)
Faculty (myself)
BART (https://bart.illinois.edu/)

For more details about the BART response policy, refer to https://bart.illinois.edu/procedures/
Learning Objectives

Demonstrate IDE-based programming

Review fundamentals of Python

Brainstorm what matters most when programming
Writing programs with a notebook

In this class, we will not use notebooks!
Writing programs with a IDE

See website for installation guide
Running Python in command line

```python
import sys

a = sys.argv[0]
b = sys.argv[1]
c = sys.argv[2]
d = sys.argv[3]

print(a)
print(b)
print(c)
print(d)
```

```
python3 Args.py 2 3,3 "red" Bob
```
Multi-file programs

mp_racing/
  code/
    main.py
    racingBot.py

data/
  track1_map.txt
  track1_obj.txt
  ...
...
Multi-file programs

(Coding) files in the same directory can be imported

<table>
<thead>
<tr>
<th>mod.py</th>
<th>main1.py</th>
</tr>
</thead>
</table>
| 1. def f1(x, y):
  2.     return x + y
| 1. from mod import f1, f2 |
| 2. def f2(x):
  3.     x += 1
  4.     return x |
| 4. |
| 5. def f3():
  6.     return 3 |
| 7. |
| 8. |

<table>
<thead>
<tr>
<th>main2.py</th>
<th>main3.py</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. from mod import *</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>main2.py</th>
<th>main3.py</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. import mod as m</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
</tr>
</tbody>
</table>
Python Fundamentals

Data types and Variables

Control Structures

Functions

Objects and Classes
Common data types

How do we know a variable’s type?
Dynamically-typed language

Variable types determined at runtime

```python
a="3"
b=3
c=3.0
d=True

print(a + b)
print("3 + 3")
print(b + c)
print(c + d)
print(d)
print(d - d)
```

`type.py`
Selection Control Structures

```python
num = 20

if num in [0,1,2,3,4]:
    print("Top 5!")

elif num > 10:
    print("num too large!")

elif num > 15:
    print("will this ever get called?")

else:
    print(num)
```

controlStructs.py
Repetition Control Structures

```python
for i in range(3):
    print(i)

for i in range(3, 5):
    print(i)

for i in range(0, 10, 3):
    print(i)
```

forLoop.py
Repetition Structures

```python
whileLoop.py

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

Functions give a rule for mapping input to output

\[ X + Y + Z = 10 \]

\[ X = 2 \]
\[ Y = 3 \]
\[ Z = 5 \]
def addTogether(x, y, z = None):
    if z == None:
        return x + y
    else:
        return x + y + z

if __name__ == '__main__':
    print(addTogether(2,3))
    print(addTogether(2,3,5))
Functions

Functions organize code with clear I/O

```python
# INPUT:
# file is the relative path of the string file being processed (string)
# bchar is the break character (string)

# OUTPUT:
# A list containing the complete collection of substrings formed by "splitting" at the break characters.

# NOTE:
# This includes potentially empty strings to the left or right of a break character in the file.
# You may assume the break character is always a single character
# You should NOT strip the whitespace for this function

def stringParseBreaks(file, bchar):
    pass
```
Functions

Functions can call other functions

def defaultFunction(a, b, c):
    return a * b * c

def wrapperFunction(a, b, c, f=None):
    if f == None:
        return defaultFunction(a, b, c)
    else:
        return f(a, b, c)

if __name__ == '__main__':
    wrapperFunction(5, 3, 2)
    wrapperFunction(1, 1, 1)
    wrapperFunction(1, 1, 1)
Variables created in a function have local scope

```python
def scopeTest(inNum, inString, inList):
    inNum = 3
    inString+="And After!"
    inList.pop(-1)
    inList.append(5)

    x = 2
    y = "Before! 
    z = [1,2,3,4]

    scopeTest(x,y,z)

    print(x)
    print(y)
    print(z)
```

```bash
scope.py
```
Python Fundamentals

Data types and Variables

Control Structures

Functions

Objects and Classes
Storing a CSV

<table>
<thead>
<tr>
<th>ID</th>
<th>Area</th>
<th>Beds</th>
<th>Baths</th>
<th>Zip</th>
<th>Year</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2222</td>
<td>3</td>
<td>3.5</td>
<td>32312</td>
<td>1981</td>
<td>250000</td>
</tr>
<tr>
<td>2</td>
<td>1628</td>
<td>3</td>
<td>2</td>
<td>32308</td>
<td>2009</td>
<td>185000</td>
</tr>
<tr>
<td>3</td>
<td>3824</td>
<td>5</td>
<td>4</td>
<td>32312</td>
<td>1954</td>
<td>399000</td>
</tr>
</tbody>
</table>
Object-Oriented Programming

An **object** is a conceptual grouping of variables and functions that make use of those variables.

**Variables:**

**Functions:**
A class is a blueprint for an object

```python
class Circle:
    def __init__(self, r):
        self.radius = r

    def __eq__(self, other):
        return (self is other)

    def circumference(self):
        return 2 * 3.14 * self.radius

    def area(self):
        return 3.14 * (self.radius)**2
```

circle.py
Each instance of an object has its own values

c1 = Circle(2)
c2 = Circle(2)
c3 = c1
c4 = Circle(3)

if c1.radius == c2.radius:
    print("Same radius!")
else:
    print("Different radius!")

c3.radius = 5

if c1.radius == c2.radius:
    print("Same radius!")
else:
    print("Different radius!")
Object-Oriented Programming

Distinct objects can still be identical in value

```python
x = [1, 2, 3]
y = x
z = [1, 2, 3]

print(x is y)
print(x is z)
print(x == y)
print(x == z)
```

is_vs_equal.py
Object-Oriented Programming

Everything in Python is an object

```
1  X = 1212
```

<table>
<thead>
<tr>
<th>Var Name</th>
<th>X</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1212</td>
</tr>
<tr>
<td>Ref Count</td>
<td>1</td>
</tr>
</tbody>
</table>
Object-Oriented Programming

Everything in Python is an object

```
1  X = 1212
```

<table>
<thead>
<tr>
<th>Var Name</th>
<th>X</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1212</td>
</tr>
<tr>
<td>Ref Count</td>
<td>1</td>
</tr>
</tbody>
</table>
Object-Oriented Programming

Everything in Python is an object

<table>
<thead>
<tr>
<th></th>
<th>X = 1212</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Y = X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var Name</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var Name</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1212</td>
</tr>
<tr>
<td>Ref Count</td>
<td>2</td>
</tr>
</tbody>
</table>
Object-Oriented Programming

Everything in Python is an object

<table>
<thead>
<tr>
<th></th>
<th>Var Name</th>
<th>Type</th>
<th>Value</th>
<th>Ref Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>integer</td>
<td>1212</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Y = X</td>
<td>integer</td>
<td>1213</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Y = Y + 1</td>
<td>integer</td>
<td>1213</td>
<td>1</td>
</tr>
</tbody>
</table>

```
X = 1212
Y = X
Y = Y + 1
```
Object-Oriented Programming

Everything in Python is an object

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X = 1212</td>
</tr>
<tr>
<td>2</td>
<td>Y = X</td>
</tr>
<tr>
<td>3</td>
<td>Y = Y + 1</td>
</tr>
<tr>
<td>4</td>
<td>X = 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>5</td>
</tr>
<tr>
<td>Ref Count</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Var Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1213</td>
</tr>
<tr>
<td>Ref Count</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1212</td>
</tr>
<tr>
<td>Ref Count</td>
<td>0</td>
</tr>
</tbody>
</table>
Object-Oriented Programming

Is Python’s OOP style good or bad?