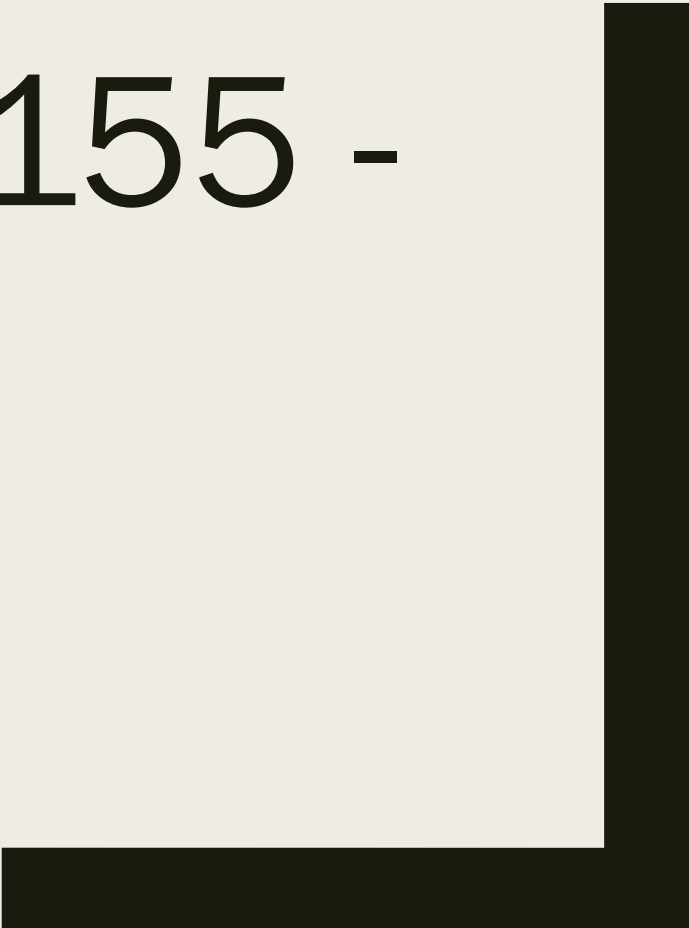




SP 19 – CSCI 3155 –

1 / 30

Spencer Wilson



Overview

- The New Deal
- Syllabus part 1
- Scala
- Road Map (more syllabus part 1)
- Tips and Tricks
- Coding

THE NEW DEAL



What's up with the course change?

- Enrollment in csci 3155 exceeded our projections by a LOT
- We couldn't secure a large enough lecture hall to host all the students in 1 section
- Many alternatives were considered
- Ultimately we created a second section to run at the same time as the main section with a different Instructor (Spencer)
- This was decided as the best possible solution to an issue that didn't really have any great solutions.

Who is this Spencer fellow anyhow?

- CU alum: BS Computer Science: Spring '17
- Associate Software Engineer at Northrop Grumman since September 2017
- TA/LC/CA/PLA experience in a variety of CSCI courses during undergrad
- Took CSCI 3155 Fa '15
- CAd CSCI 3155 Sp '16, Fa '16, Sp '17
- Instructed CSCI 3155 Su '17 and Su '18 (and soon Sp '19 and Su '19)

Thanks for having me

- I've seen this course evolve a lot over time
 - The course took a huge new direction this past Fall
 - I am so excited to learn this new direction and assist you all in your mastery of the course material
-
- The things I've learned in this course have indeed helped me at my job as a software engineer!
 - The topics made me a stronger programmer
 - The topics have made it easier for me to pick up new languages

SYLLABUS PART 1



Contacting me

- Piazza:

- *All correspondence with me should happen over piazza. (don't email me)*
- *You can send private messages directly to me in Piazza*

- Office: ECST 121

- Office hours:

- *After Class on Tuesdays*
- *Or by appointment. I am often available Tuesday and Thursday afternoons, and on weekends.*

Waitlist

- Obviously there were some scheduling issues...
- If you are still waitlisted, come talk to me after class OR get in touch with me on piazza
- I'll do anything that can to get you off the waitlist.

Course tools

- Moodle
 - *Homework*
 - *Grades*
 - *Lecture materials*
 - *Calendar*
- Piazza
 - *Correspondence with course staff*
 - *Discussions with your peers*
- Jupyter Notebooks
 - *Dev environment*
 - *Homework*
 - *Various course notes*

Classroom expectations: Don't be disruptive

- Coming in late? Come in through the back door.
- Laptops are encouraged!
 - *This is a course on programming. We WILL be coding in class*
 - *This is not a course on trolling reddit... If you are doing disruptive things on your computer, please sit in the back of the classroom*
- Silence your cell phones and your laptops

Classroom expectations: Continued

- I go by Spencer.
- I love it when people ask questions!
 - *But I prefer that you raise your hands*
 - *And I prefer that you wait to be prompted.*
 - *Also I want everyone to be able to ask questions, please try to capitalize all of class time*

Classroom Accommodations

- What is Disability Services?
- Where are they located?
- Please submit your DS accommodation letters to me by the end of class next Thursday 01/24

Clickers

- By show of hands, how many of us own a Clicker?

Course Textbooks

- Jupyter Notebooks: There will be a lot of material distributed to you via JN. You should absolutely read that material.
- Essentials of Programming Languages by Daniel P Friedman and Mitchell Ward: Great book! Probably won't help you with this course... It's written in scheme (where the car of the coulder of the list (5, 10, 15) is equal to 10)
- Programming in Scala (Ordersky, Spoon, Venners): Great book! This is a technical guide to the language. You might want to get used to reading books like this.

And so much more

- ... We'll look at these on Thursday
- Read the syllabus before class Thursday

SCALA



Programming Languages

- Each programming language is a tool
- Tools have different strengths and limitations
- What programming languages do you all use?
 - *Python*
 - *C*
 - *C++*
 - *Java*
 - *Matlab*
 - *Scala*
 - ...

Language Classifications

- These are rather grey
 - Declarative: Prolog...
 - Imperative: Python, C, Java...
 - Functional: Lisp, Scala...
 - Object Oriented: C++, Java...
-
- We won't talk about these really for the rest of the semester
 - There are many great videos on YouTube on this matter

Scala

- Scala is a child of Java
- It is a highly functional language
- It has a relatively low barrier to entry
- It can be compiled and it can be interpreted

First Exercise

- In your favorite language (or in English pros)
- Write a program that prints the numbers 1 through 100.
- It should print 1, then on a new line it would print 2, and so on until it prints 100, then it should stop.

Question 1: What construct did you use?

- A. For loop
- B. While loop
- C. A different loop
- D. A built in of the language
- E. Something else

Example with while loop (python)

```
def disp1to100():  
    x = 1  
    while x <= 100:  
        print(x)  
        x += 1 # the value of x has changed  
  
if __name__ == '__main__':  
    disp1to100()
```

Example with Builtin (python)

```
def disp1to100():
```

```
    [print(x) for x in range(1,101)] # list comprehension, I argue this mutates x
```

```
if __name__ == '__main__':
```

```
    disp1to100()
```


Functional Concept

- Functional languages have a grudge against mutable variables
- If something is mutable, then it can be really hard to share it:
 - *Between objects in the program*
 - *Between processors on a cluster*
 - *Between computers on a network*
- Imagine trying to program without any mutability...
- How would you do that? (rhetorical)

Example with while loop (python)

```
def disp1to100():  
    x = 1  
    while x <= 100:  
        print(x)  
        x += 1 # the value of x has changed  
  
if __name__ == '__main__':  
    disp1to100()
```

Exercise without Mutability

- In your favorite language
- Write a program that prints the numbers 1 through 100.
- It should print 1, then on a new line it would print 2, and so on until it prints 100, then it should stop.
- **Do not use mutability or builtins of the language**

Question 2: What construct did you use?

- A. A loop
- B. Recursion
- C. Other

Scala in this course

- We will learn a lot about this semester both its functional and non-functional features and how to use each
- But we'll probably focus more on the functional concepts
- We'll learn more about:
 - *Inductive Logic*
 - *Recursion*
 - *The true nature of functions*

Scala first program

- How do we code the previous exercise in scala?
 - What constructs do we need?
 - Let's begin with coding it in python
 - Identify the constructs used
 - And then translate it a bit
-
- This method can be rather useful when picking up a new language (it doesn't always work, because some languages are just too different and you can't translate easily between them, but here it works)

Not Mutable Solution (python)

```
def disp1to100():
    def helper(x):
        if x <= 100:
            print(x)
            helper(x + 1) # we didn't change the value of this x...
                           # we called the function with a new value for the argument
    helper(1)

if __name__ == '__main__':
    disp1to100()
```

Constructs

Construct	Python example
Function declaration	<code>def helper(x):</code>
Conditionals	<code>if x <= 100:</code>
Scope	<code><white space></code>
I/O	<code>print(x)</code>
Arithmetic	<code>x + 1</code>
Comparison	<code>x <= 100</code>
Function call	<code>helper(x + 1)</code>

Constructs

Construct	Python example	Scala translation
Function declaration	<code>def helper(x):</code>	<code>def helper(x:Int):Unit = {</code>
Conditionals	<code>if x <= 100:</code>	<code>if (x <= 100) {</code>
Scope	<code><white space></code>	<code>{ }</code>
I/O	<code>print(x)</code>	<code>println(x)</code>
Arithmetic	<code>x + 1</code>	<code>x + 1</code>
Comparison	<code>x <= 100</code>	<code>x <= 100</code>
Function call	<code>helper(x + 1)</code>	<code>helper(x + 1)</code>

Exercise without Mutability

- In scala: Write a program that prints the numbers 1 through 100. Do not use mutability

PYTHON:

```
def disp1to100():
```

```
    def helper(x):
```

```
        if x <= 100:
```

```
            print(x)
```

```
            helper(x + 1)
```

```
    helper(1)
```

```
if __name__ == '__main__':
```

```
    disp1to100()
```

Construct	Python example	Scala translation
Function declaration	def helper(x):	def helper(x:Int):Unit = {
Conditionals	if x <= 100:	if (x <= 100) {
Scope	<white space>	{ }
I/O	print(x)	println(x)
Arithmetic	x + 1	x + 1
Comparison	x <= 100	x <= 100
Function call	helper(x + 1)	helper(x + 1)

Scala solution

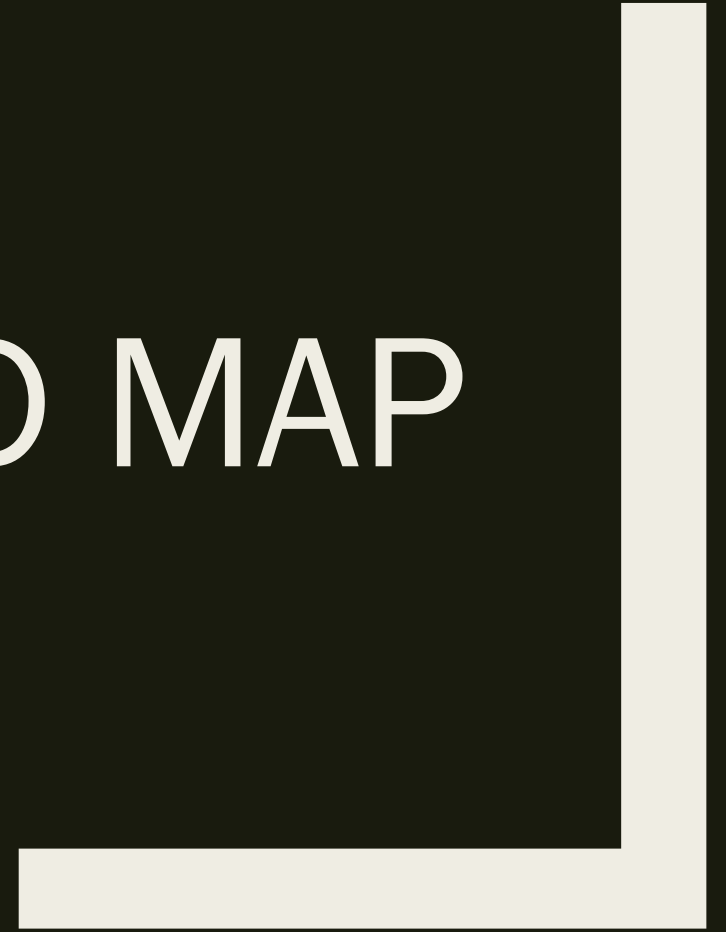
```
def disp1to100():Unit = {  
  def helper(x:Int):Unit = {  
    if (x <= 100) {  
      println(x)  
      helper(x + 1)  
    }  
  }  
  helper(1)  
}
```

```
disp1to100()
```

TODO:

- Get your programming environment set up ASAP
- You need to start experimenting with Scala!

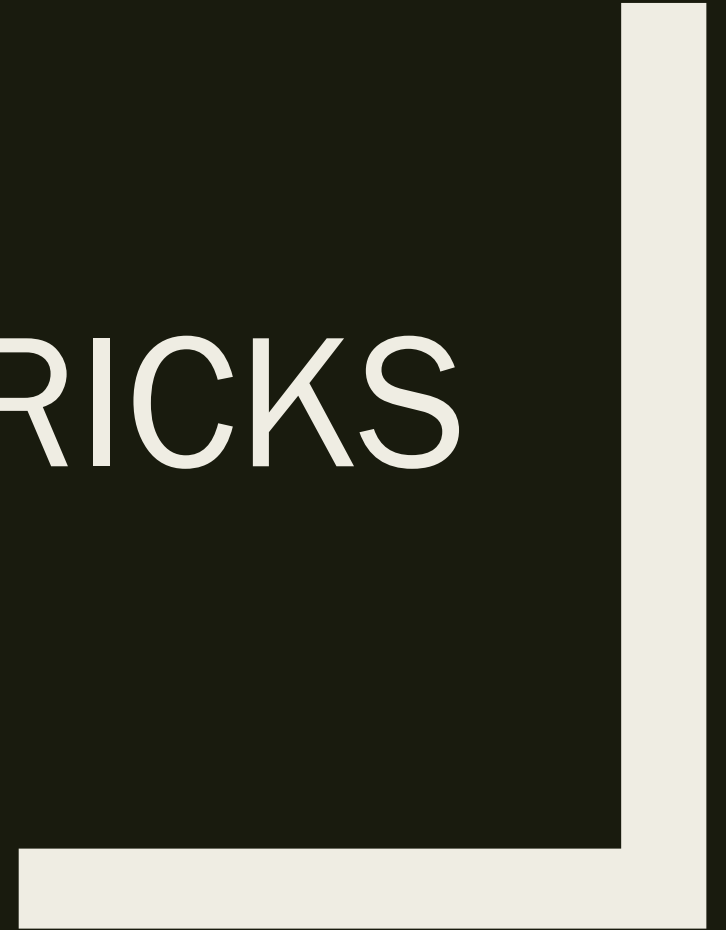
ROAD MAP



Where are we going in this class?

- Disclaimer, I don't totally know yet. I just work here.
- Learn the underlying concepts of Programming languages
 - *Learn topics that can be easily transferred between different languages*
 - *Functions*
 - *Objects*
- Learn some functional concepts and refine inductive thinking skills
- Learn cool data structures and operations for them
- Analyze, design and implement a useful programming language and interpreter
- ... more
- See syllabus for more details, and come to class Thursday to discuss

TIPS AND TRICKS



Environment

- Please! Get your programming environment set up ASAP so that you can start exploring scala, in this course we will be going through the basics of scala quite quickly. If you aren't prepared, you may fall behind

Get Help

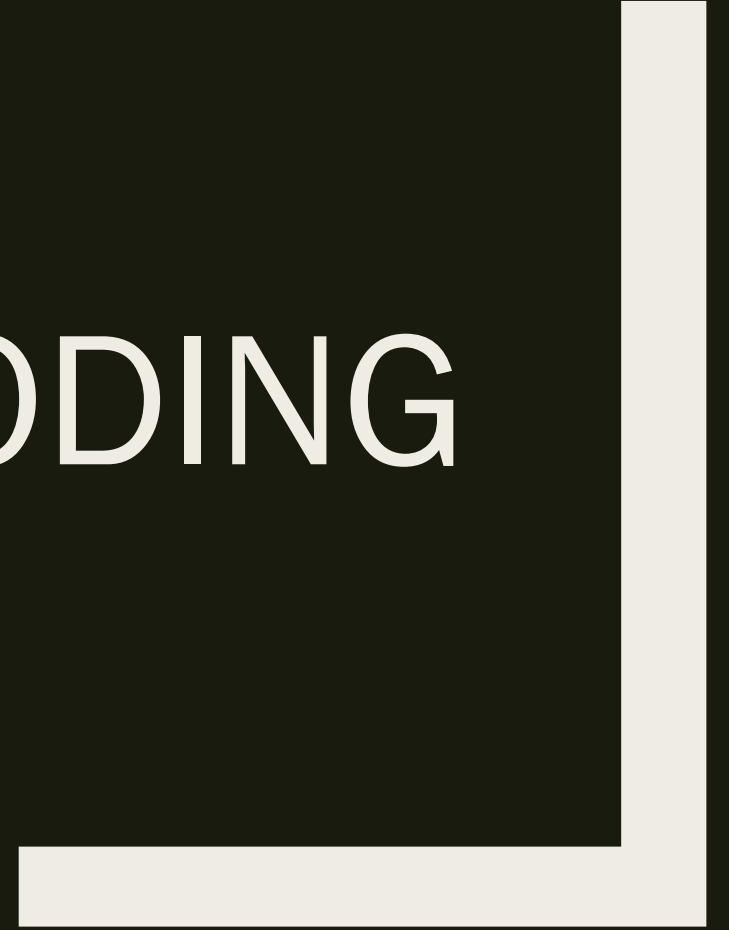
- Some of this stuff is HARD!
- If you are stuck, get some help!

- Your peers
- Your CAs
- Your TAs
- Your Instructors

Don't cheat

- Policies change by course... Our policy is:
- It is acceptable to work together, but you should not write solutions together. You shall write your own individual solution to each problem
- It is acceptable to look for general information online that will help you solve the problems in this course. You shall not look for exact solutions to the class problems online
- See the Syllabus for more information

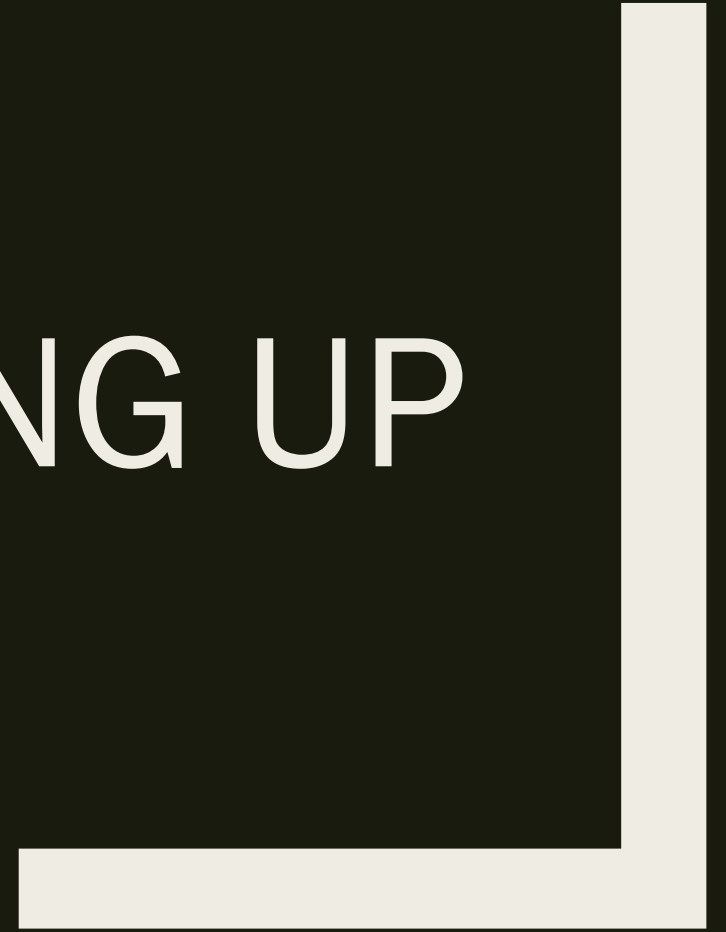
CODING



Code

- Code is posted or coming soon
- Let us code some things in Scala and learn more about the language.
- We'll do what we can in the rest of class today.
- TDD for factorial(n)
- If there is time: TDD for Roman Numeral string to Integer

WRAPPING UP



Overview

- The New Deal
- Syllabus part 1
- Scala
- Road Map (more syllabus part 1)
- Tips and Tricks
- Coding

TODO:

- Get your programming environment set up **before class Thursday (ASAP)**
 - *Jupyter*
 - *Almond*
- Get set up on the course tools:
 - *Moodle*
 - *Piazza*
 - *Calendar*
- Get yourself off the waitlist (come see me after class)
- Get your accommodations sheets to me ASAP (if applicable)