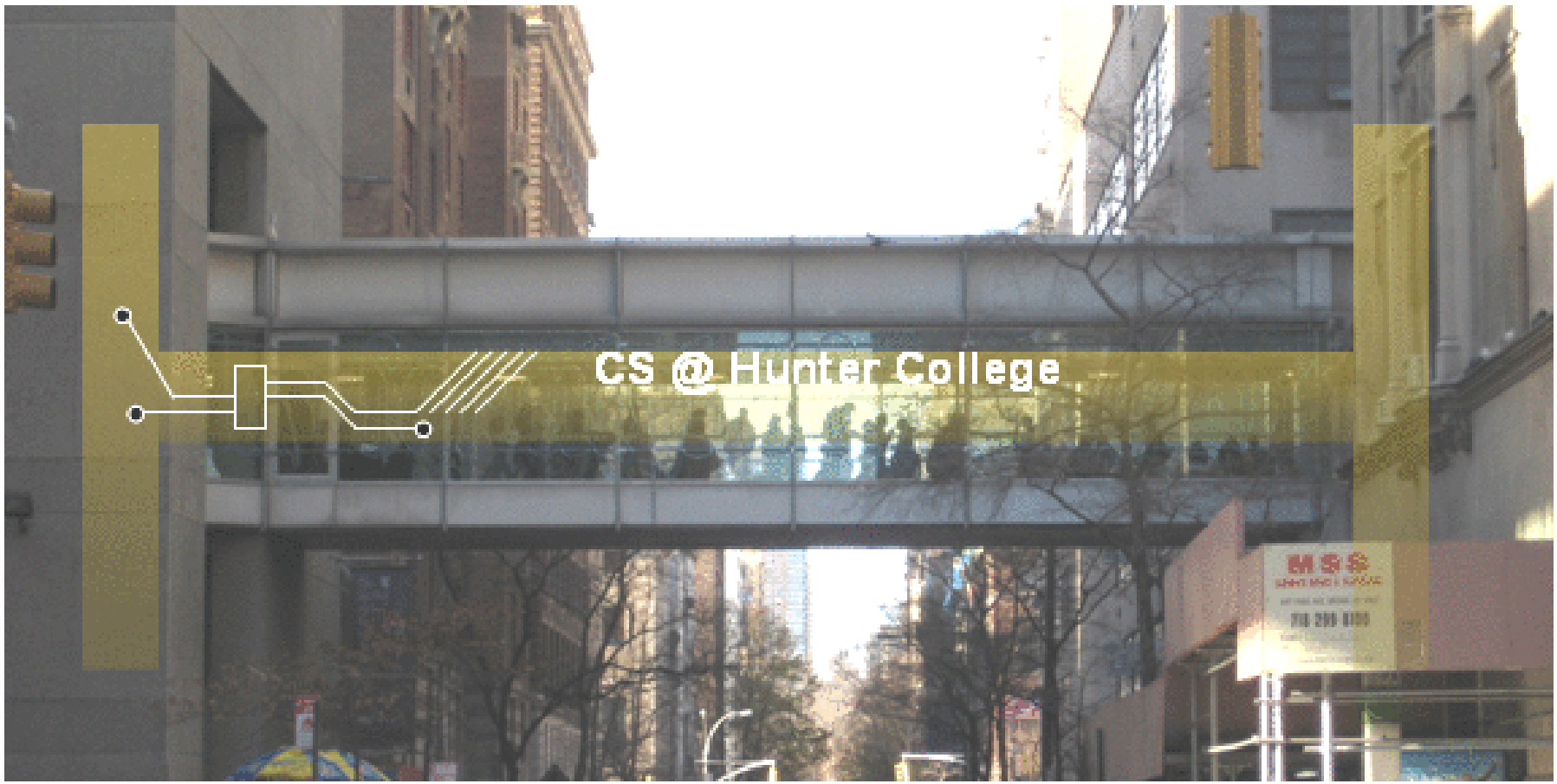
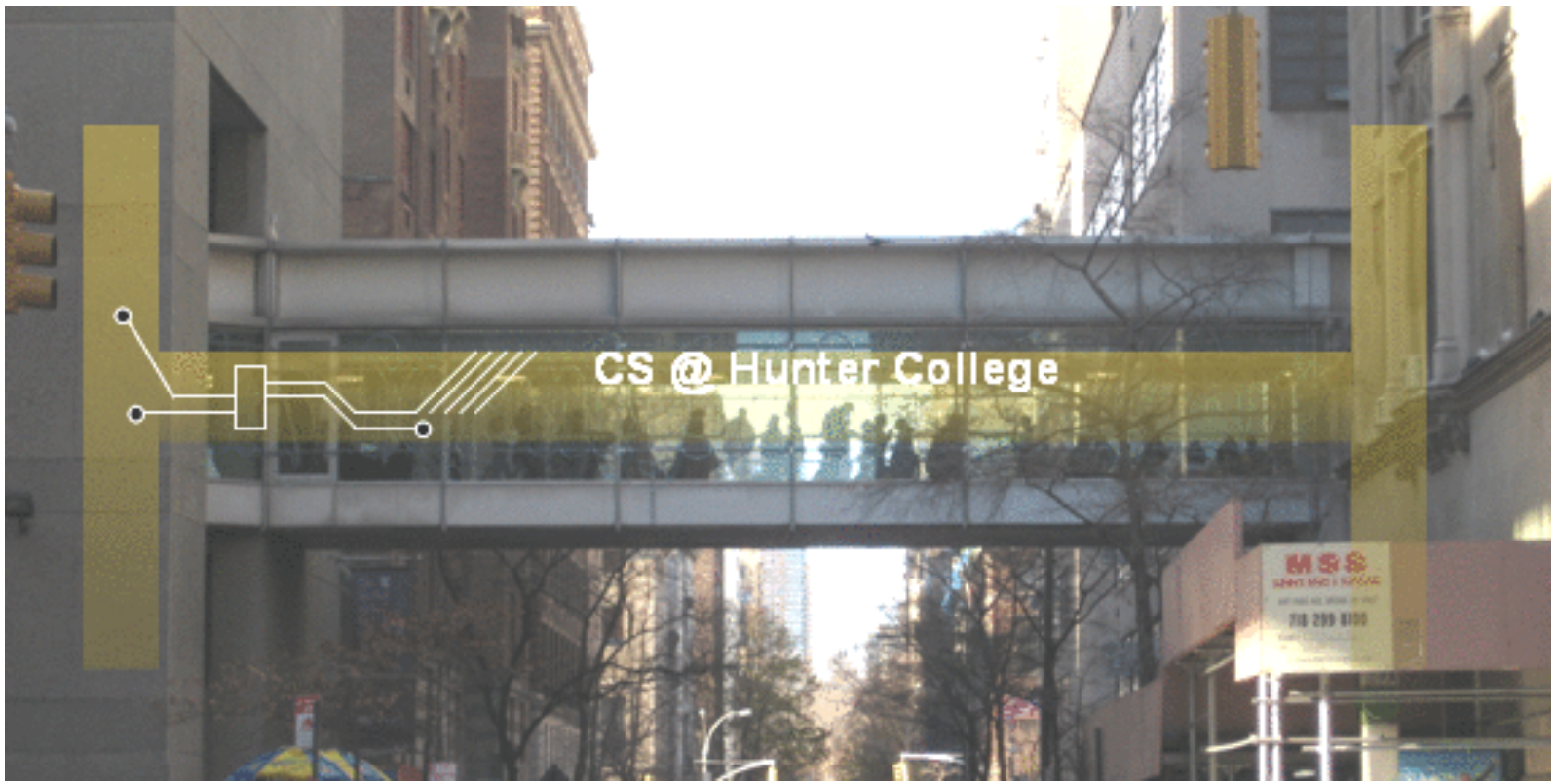
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Welcome



Acknowledgments

Thank you to the amazing support of:

President Raab

Dean Polsky
Arts & Science

Judy Spitz
WiTNY

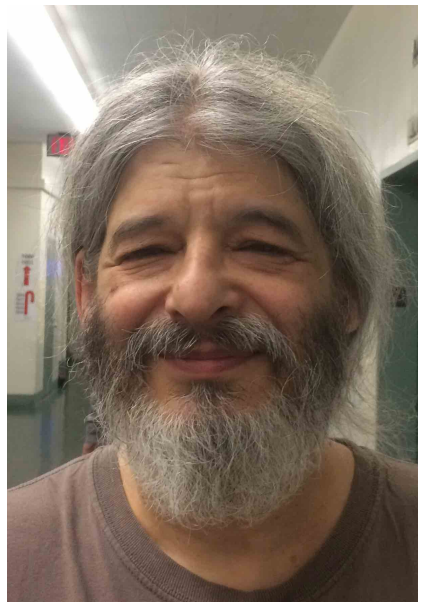
Introductions: Course Designers

Dr. Katherine St. John

Professor,
Course Coordinator

Dr. William Sakas

Associate Professor,
Chair



Prof. Eric Schweitzer

Undergraduate Program
Coordinator

Introductions: Instructors

Katherine Howitt

Tuesday Thursday
Lecture Lab

Raj Korpan

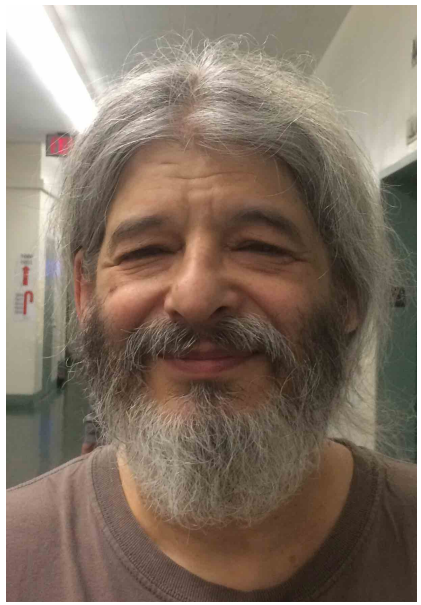
Monday Wednesday
Lecture Lab

Introductions: Undergraduate Teaching Assistants

Mandy Yu
Monday Wednesday

Tyler Robinson
Tuesday Thursday

Introductions: Advisors



Eric Schweitzer
Undergraduate Program
Coordinator

Justin Tojeira
Internships &
Upper Division

Syllabus

CSci 127: Introduction to Computer Science

Catalog Description: 3 hours, 3 credits: This course presents an overview of computer science (CS) with an emphasis on **problem-solving and computational thinking through 'coding'***: computer programming for beginners. Other topics include: organization of hardware, software, and how information is structured on contemporary computing devices. This course is pre-requisite to several introductory core courses in the CS Major. The course is also required for the CS minor. MATH 12500 or higher is strongly recommended as a co-req for intended Majors.*

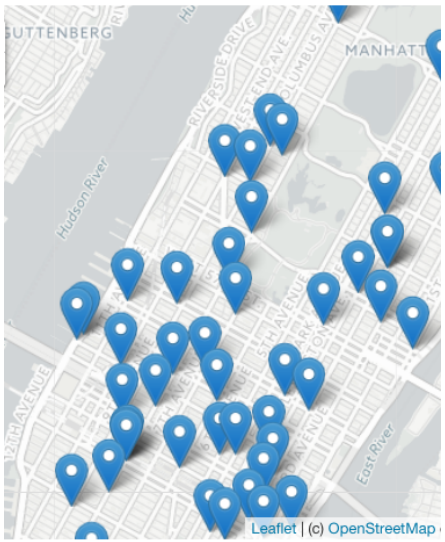
Syllabus

CSci 127: Introduction to Computer Science

Catalog Description: 3 hours, 3 credits: This course presents an overview of computer science (CS) with an emphasis on **problem-solving and computational thinking through 'coding'***: computer programming for beginners. Other topics include: organization of hardware, software, and how information is structured on contemporary computing devices. This course is pre-requisite to several introductory core courses in the CS Major. The course is also required for the CS minor. MATH 12500 or higher is strongly recommended as a co-req for intended Majors.*

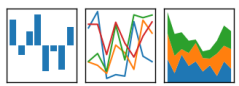
(Show syllabus webpage)

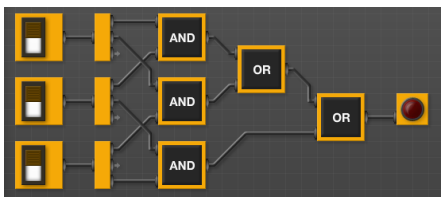
Syllabus: Topics



pandas

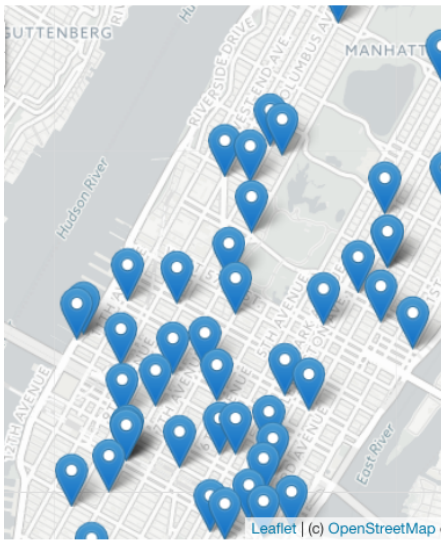
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

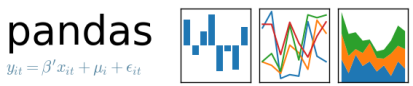


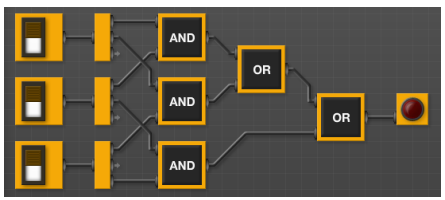


- **This course assumes no previous programming experience.**

Syllabus: Topics

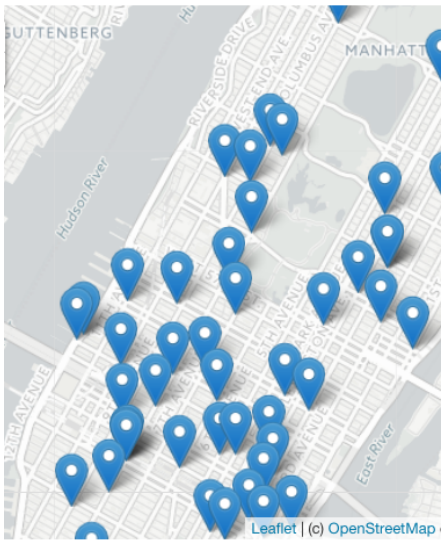


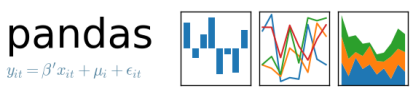


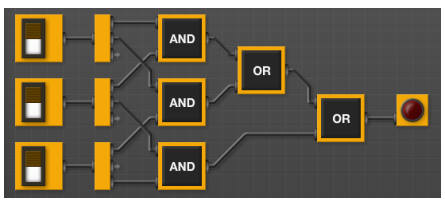


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."

Syllabus: Topics

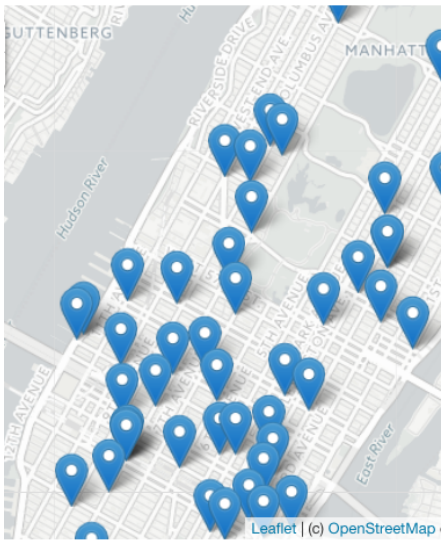


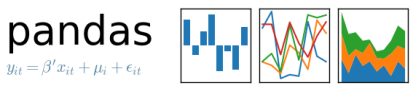


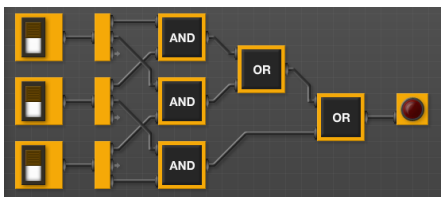


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:

Syllabus: Topics

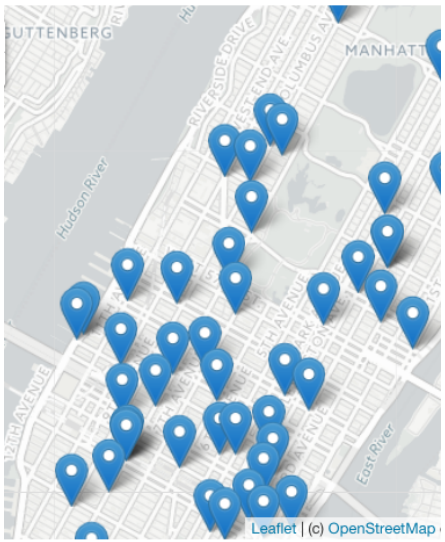


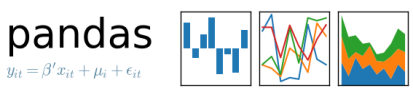


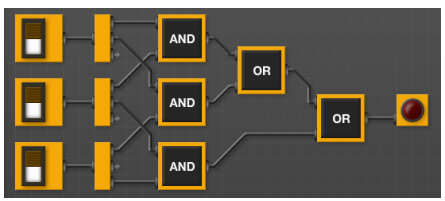


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,

Syllabus: Topics

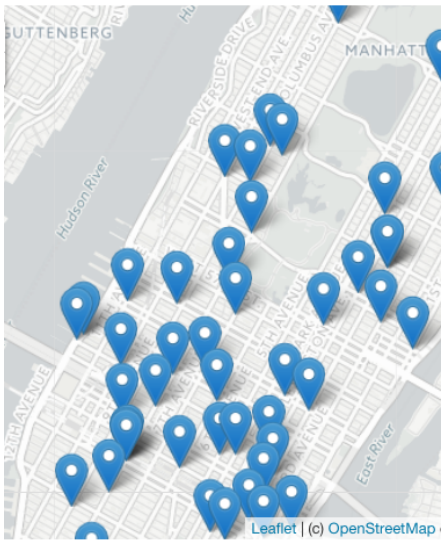






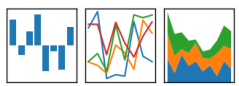
- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),

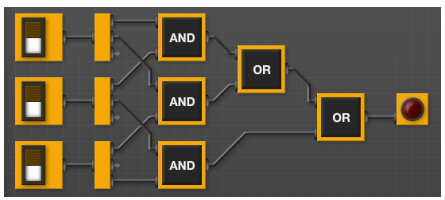
Syllabus: Topics



pandas

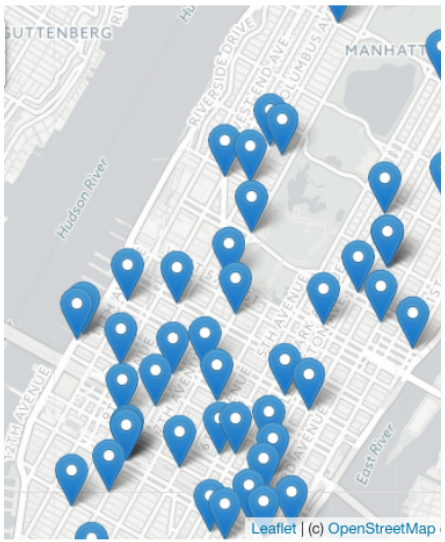
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

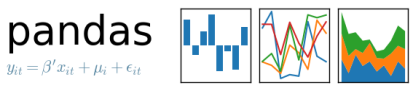


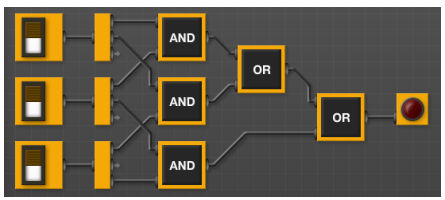


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:

Syllabus: Topics

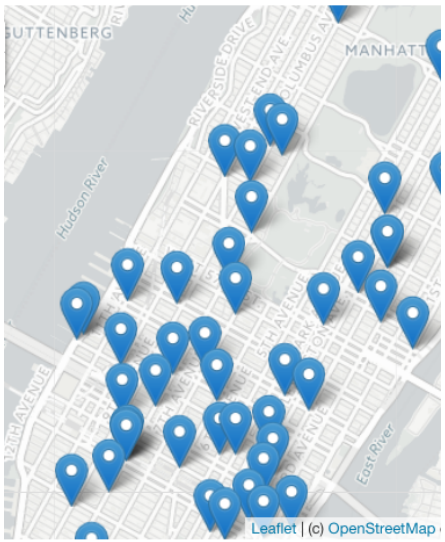


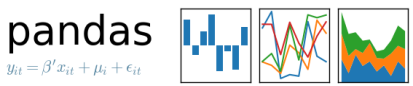


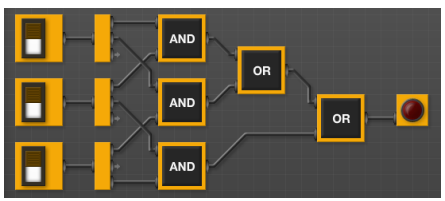


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,

Syllabus: Topics

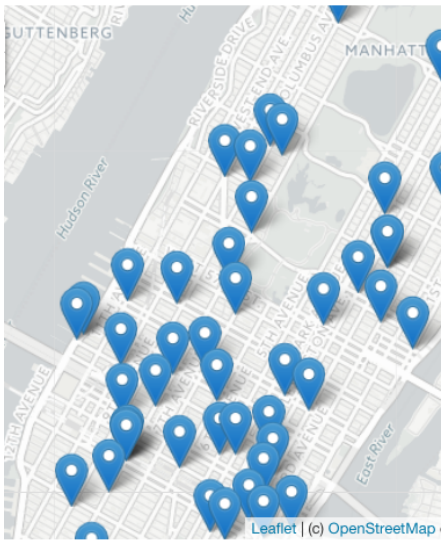


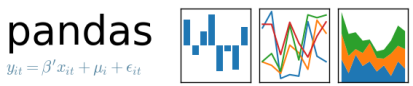


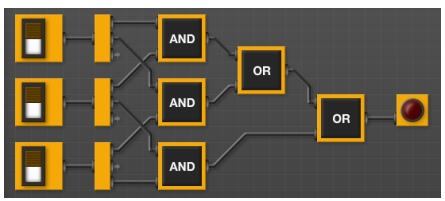


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    - ★ for Unix command line interface,

Syllabus: Topics

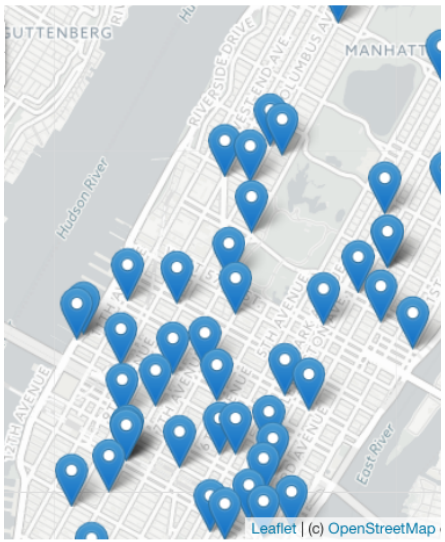






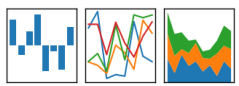
- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    - ★ for Unix command line interface,
    - ★ for the markup language for github,

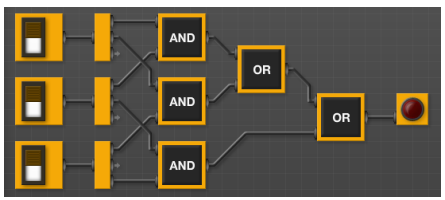
Syllabus: Topics



pandas

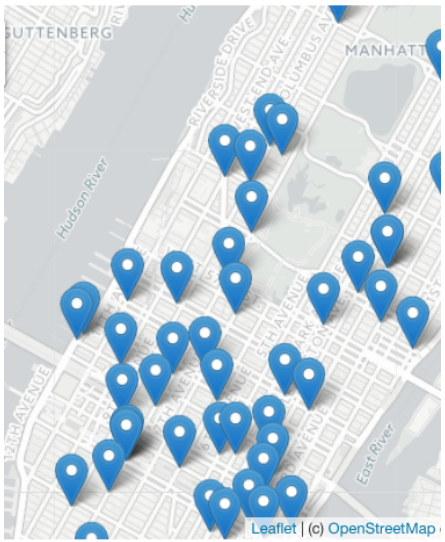
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

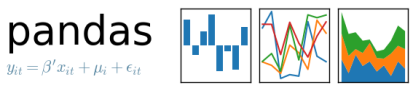


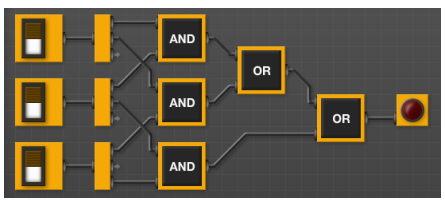


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    - ★ for Unix command line interface,
    - ★ for the markup language for github,
    - ★ for the simplified machine language, &

Syllabus: Topics





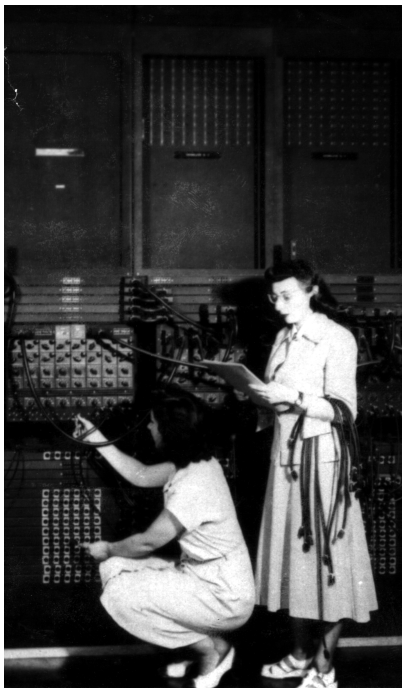


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    - ★ for Unix command line interface,
    - ★ for the markup language for github,
    - ★ for the simplified machine language, &
    - ★ for C++.

Class Structure

Lecture:

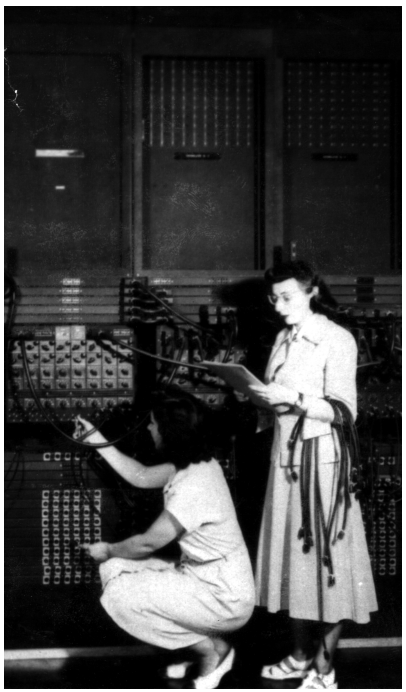
- Twice a week.



First "computers"

ENIAC, 1945.

Class Structure



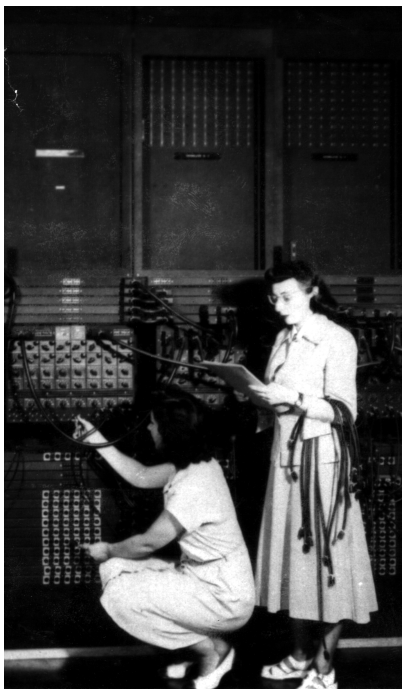
First "computers"

ENIAC, 1945.

Lecture:

- Twice a week.
- Mix of explanation, challenges, & interactive questions.

Class Structure



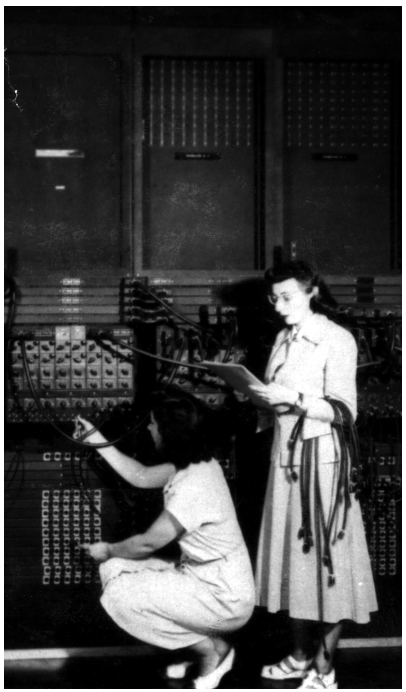
First "computers"

ENIAC, 1945.

Lecture:

- Twice a week.
- Mix of explanation, challenges, & interactive questions.
- Attendance: Mandatory until dismissal.

Class Structure



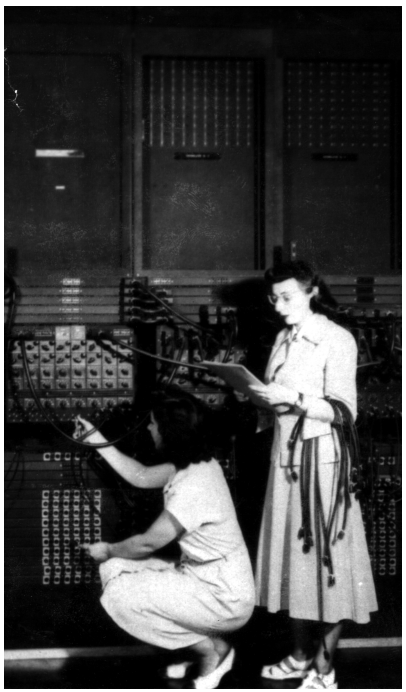
First "computers"

ENIAC, 1945.

Lecture:

- Twice a week.
- Mix of explanation, challenges, & interactive questions.
- Attendance: Mandatory until dismissal.
- Ask questions throughout lecture using chat, and aloud at challenges.

Class Structure



First "computers"

ENIAC, 1945.

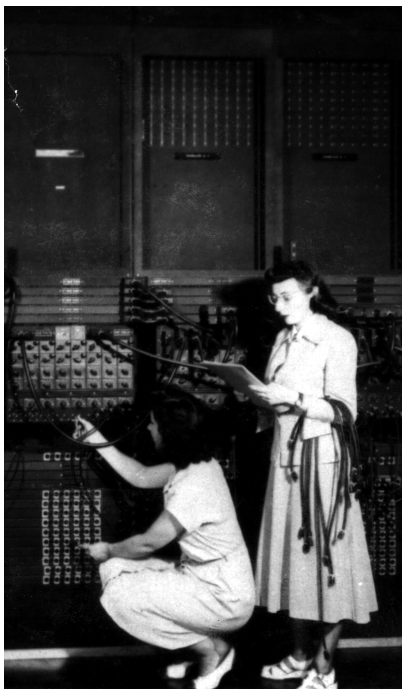
Lecture:

- Twice a week.
- Mix of explanation, challenges, & interactive questions.
- Attendance: Mandatory until dismissal.
- Ask questions throughout lecture using chat, and aloud at challenges.

Online Labs:

- **You must independently read through the weekly online Lab.**

Class Structure



First "computers"

ENIAC, 1945.

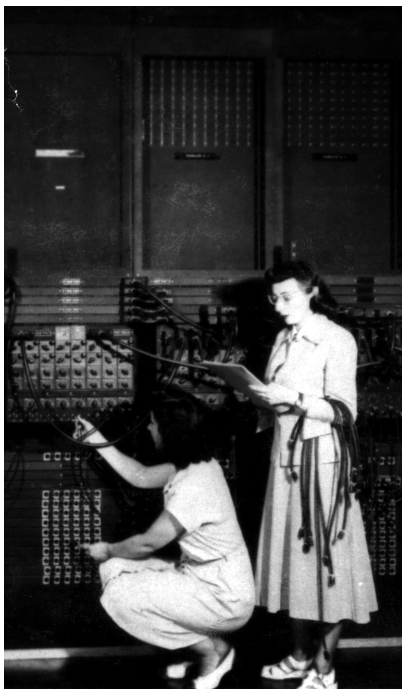
Lecture:

- Twice a week.
- Mix of explanation, challenges, & interactive questions.
- Attendance: Mandatory until dismissal.
- Ask questions throughout lecture using chat, and aloud at challenges.

Online Labs:

- **You must independently read through the weekly online Lab.**
- Instructor will foreshadow following lecture.

Class Structure



First "computers"

ENIAC, 1945.

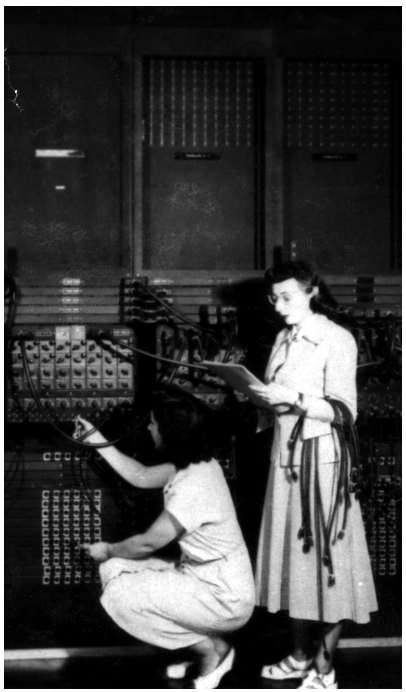
Lecture:

- Twice a week.
- Mix of explanation, challenges, & interactive questions.
- Attendance: Mandatory until dismissal.
- Ask questions throughout lecture using chat, and aloud at challenges.

Online Labs:

- **You must independently read through the weekly online Lab.**
- Instructor will foreshadow following lecture.
- Lab content will directly support your ability to solve programming assignments.

Class Structure



First "computers"

ENIAC, 1945.

Lecture:

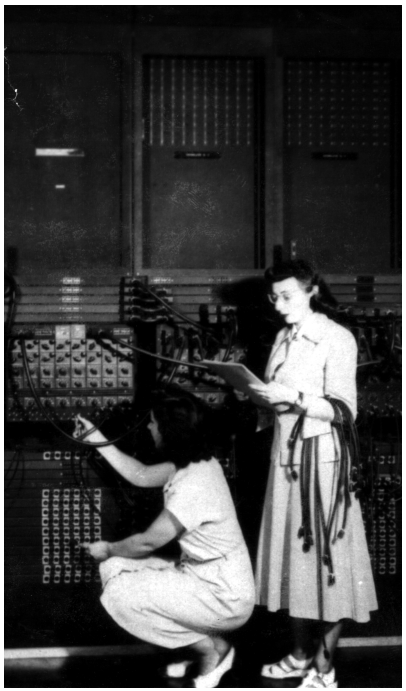
- Twice a week.
- Mix of explanation, challenges, & interactive questions.
- Attendance: Mandatory until dismissal.
- Ask questions throughout lecture using chat, and aloud at challenges.

Online Labs:

- **You must independently read through the weekly online Lab.**
- Instructor will foreshadow following lecture.
- Lab content will directly support your ability to solve programming assignments.
- Labs found on course website (show)

Class Structure

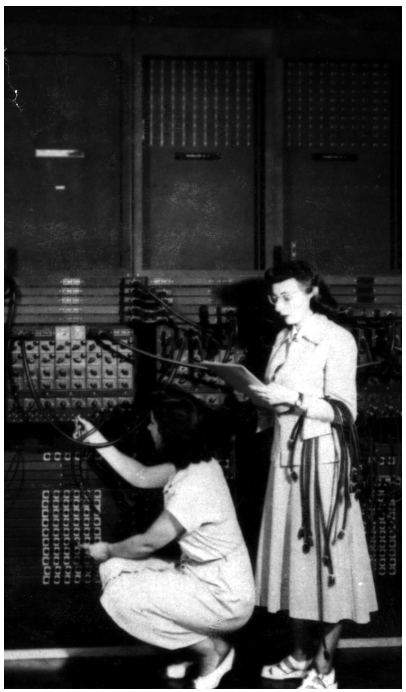
Quizzes:



First "computers"

ENIAC, 1945.

Class Structure



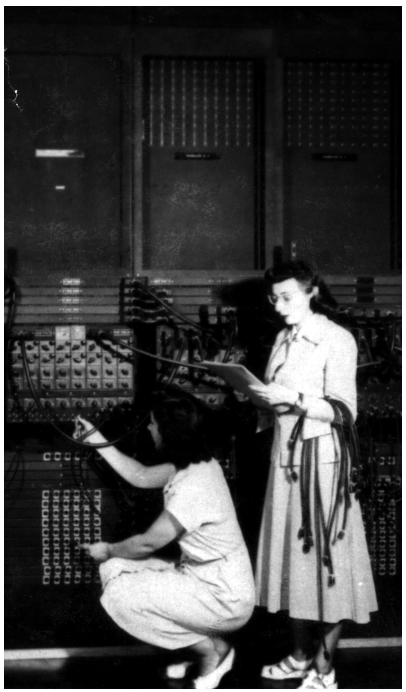
First "computers"

ENIAC, 1945.

Quizzes:

- Administered immediately following lecture.

Class Structure



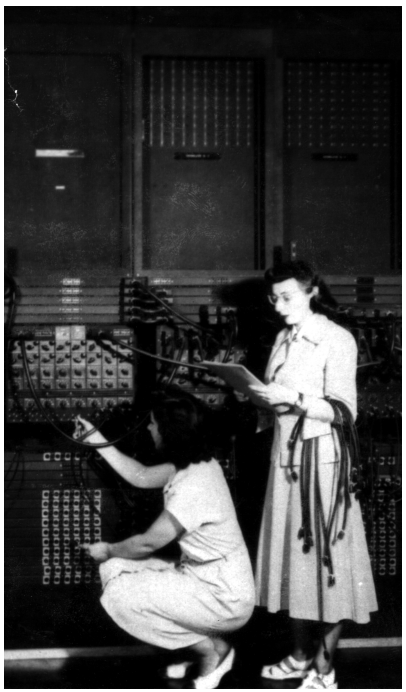
First "computers"

ENIAC, 1945.

Quizzes:

- Administered immediately following lecture.
- Gradescope Quiz: questions follow final exam style.

Class Structure



First "computers"

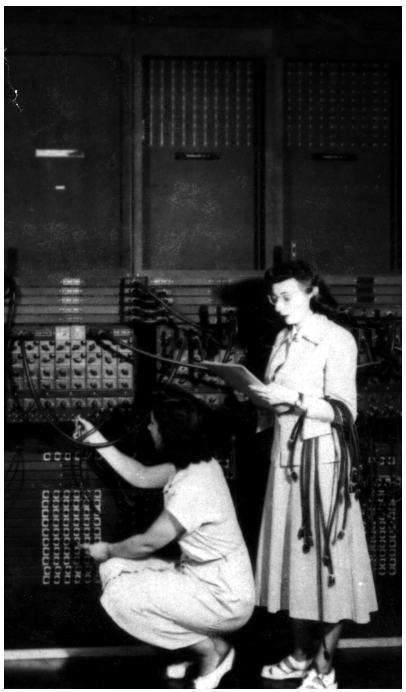
ENIAC, 1945.

Quizzes:

- Administered immediately following lecture.
- Gradescope Quiz: questions follow final exam style.
- 12 total quizzes

Software Platforms:

Class Structure



First "computers"

ENIAC, 1945.

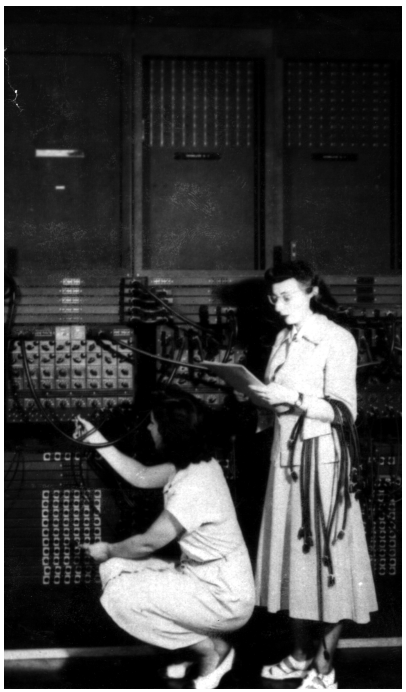
Quizzes:

- Administered immediately following lecture.
- Gradescope Quiz: questions follow final exam style.
- 12 total quizzes

Software Platforms:

- Blackboard
  - Email ICIT for access issues.
  - Change to your preferred email on CUNYFirst

Class Structure



First "computers"

ENIAC, 1945.

Quizzes:

- Administered immediately following lecture.
- Gradescope Quiz: questions follow final exam style.
- 12 total quizzes

Software Platforms:

- Blackboard
  - Email ICIT for access issues.
  - Change to your preferred email on CUNYFirst
- Gradescope
  - Email invite sent.
  - Match to Blackboard email.

Where to Go for Help

- Blackboard Discussion Board

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.
  - Moderated by instructor and TAs daily.

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.
  - Moderated by instructor and TAs daily.
  - Last checked at 6p.

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.
  - Moderated by instructor and TAs daily.
  - Last checked at 6p.
  - DO NOT post full code to board: TA or instructor will reach out via email or on gradescope to review full code.

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.
  - Moderated by instructor and TAs daily.
  - Last checked at 6p.
  - DO NOT post full code to board: TA or instructor will reach out via email or on gradescope to review full code.
- Email

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.
  - Moderated by instructor and TAs daily.
  - Last checked at 6p.
  - DO NOT post full code to board: TA or instructor will reach out via email or on gradescope to review full code.
- Email
  - Allow instructor 24 hours to review emails.

Where to Go for Help

- Blackboard Discussion Board
  - Different thread for each assignment.
  - Moderated by instructor and TAs daily.
  - Last checked at 6p.
  - DO NOT post full code to board: TA or instructor will reach out via email or on gradescope to review full code.
- Email
  - Allow instructor 24 hours to review emails.
  - Don't email last minute about an assignment!

How to Succeed in this Course

- Come to Lecture

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.
  - Actively participate: try to solve problems/challenges

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.
  - Actively participate: try to solve problems/challenges
- Read the Online Labs.

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.
  - Actively participate: try to solve problems/challenges
- Read the Online Labs.
- Work ahead on Programming Assignments.

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.
  - Actively participate: try to solve problems/challenges
- Read the Online Labs.
- Work ahead on Programming Assignments.
- Take the weekly Quiz.

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.
  - Actively participate: try to solve problems/challenges
- Read the Online Labs.
- Work ahead on Programming Assignments.
- Take the weekly Quiz.
- Ask for help from our UTAs in lecture, lab, and office hours.

How to Succeed in this Course

- Come to Lecture
  - Pay attention during lecture.
  - Actively participate: try to solve problems/challenges
- Read the Online Labs.
- Work ahead on Programming Assignments.
- Take the weekly Quiz.
- Ask for help from our UTAs in lecture, lab, and office hours.
- Use the Blackboard Discussion Board.

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?
  Yes, we are not a filter course. Our goal is for students to succeed in the course and in the CS major and minor.

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?
  Yes, we are not a filter course. Our goal is for students to succeed in the course and in the CS major and minor.
- What happens to my grade if I miss a lecture or quiz?

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?
  Yes, we are not a filter course. Our goal is for students to succeed in the course and in the CS major and minor.
- What happens to my grade if I miss a lecture or quiz?
  We replace missing or low grades on attendance and quizzes with your final exam grade.
  Attendance and quizzes only help your grade.

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?
  Yes, we are not a filter course. Our goal is for students to succeed in the course and in the CS major and minor.
- What happens to my grade if I miss a lecture or quiz?
  We replace missing or low grades on attendance and quizzes with your final exam grade.
  Attendance and quizzes only help your grade.
- Do I have to pass the final to pass the course?

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?
  Yes, we are not a filter course. Our goal is for students to succeed in the course and in the CS major and minor.
- What happens to my grade if I miss a lecture or quiz?
  We replace missing or low grades on attendance and quizzes with your final exam grade.
  Attendance and quizzes only help your grade.
- Do I have to pass the final to pass the course?
  Yes. *To demonstrate mastery, you must pass the final exam.*

Philosophy (Or Why We Do What We Do)

Grading:

- Do you curve grades?
  No, we grade on your mastery of the material and do not have a set number of A's, B's, C's that we curve grades to match (i.e. your demonstrated mastery over your relative performance to the class).
- Okay, then could everyone get an A?
  Yes, we are not a filter course. Our goal is for students to succeed in the course and in the CS major and minor.
- What happens to my grade if I miss a lecture or quiz?
  We replace missing or low grades on attendance and quizzes with your final exam grade.
  Attendance and quizzes only help your grade.
- Do I have to pass the final to pass the course?
  Yes. *To demonstrate mastery, you must pass the final exam.*

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.
  While this mimics some jobs, students (particularly those new to programming) master skills better with smaller challenges.

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.
  While this mimics some jobs, students (particularly those new to programming) master skills better with smaller challenges.
- Why weekly quizzes instead of midterms?

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.
  While this mimics some jobs, students (particularly those new to programming) master skills better with smaller challenges.
- Why weekly quizzes instead of midterms?
  Weekly quizzes increase pass rates and mastery of material.

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.
  While this mimics some jobs, students (particularly those new to programming) master skills better with smaller challenges.
- Why weekly quizzes instead of midterms?
  Weekly quizzes increase pass rates and mastery of material.
  Actively using knowledge increases your brain's ability to retain knowledge.

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.
  While this mimics some jobs, students (particularly those new to programming) master skills better with smaller challenges.
- Why weekly quizzes instead of midterms?
  Weekly quizzes increase pass rates and mastery of material.
  Actively using knowledge increases your brain's ability to retain knowledge.
- Why pre-testing (in the form of challenges)? Why do we get asked (ungraded) questions on stuff we have never seen before?

Philosophy (Or Why We Do What We Do)

Course Structure:

- Why 60 programs assignments? My friend only has to do 10.
  Traditionally, it's 10 long 'all-nighters' assignments.
  While this mimics some jobs, students (particularly those new to programming) master skills better with smaller challenges.
- Why weekly quizzes instead of midterms?
  Weekly quizzes increase pass rates and mastery of material.
  Actively using knowledge increases your brain's ability to retain knowledge.
- Why pre-testing (in the form of challenges)? Why do we get asked (ungraded) questions on stuff we have never seen before?
  While counter-intuitive, it gives a "mental scaffold" to store new material.

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*
- I'm stuck on a program. Where can I get help?

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*
- I'm stuck on a program. Where can I get help?
  - *Tutoring hours when classes are in session*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*
- I'm stuck on a program. Where can I get help?
  - *Tutoring hours when classes are in session*
  - *On-line help:*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*
- I'm stuck on a program. Where can I get help?
  - *Tutoring hours when classes are in session*
  - *On-line help:*
    - ★ *post on the discussion board*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*
- I'm stuck on a program. Where can I get help?
  - *Tutoring hours when classes are in session*
  - *On-line help:*
    - ★ *post on the discussion board*
    - ★ *schedule office hours appointment with instructor or TA*

Philosophy (Or Why We Do What We Do)

Help:

- What's the best way to master the concepts in this course?
  - *Most efficient way: do the programs (more programs on old finals).*
  - *Aim to complete programs when related ideas are covered in lab.*
  - *If stuck, there's reading and tutorials on webpage.*
- I'm stuck on a program. Where can I get help?
  - *Tutoring hours when classes are in session*
  - *On-line help:*
    - ★ *post on the discussion board*
    - ★ *schedule office hours appointment with instructor or TA*
    - ★ *email instructor and TA*

Introductions: Your Turn

- Introduce yourself to the class.
- Tell us your names & an interesting fact.

Today's Topics

- Introduction to Python
- Definite Loops (`for`-loops)
- Turtle Graphics
- Algorithms

Introduction to Python

- We will be writing programs– commands to the computer to do something.

Introduction to Python

- We will be writing programs– commands to the computer to do something.
- A **programming language** is a stylized way of writing those commands.

Introduction to Python

- We will be writing programs– commands to the computer to do something.
- A **programming language** is a stylized way of writing those commands.
- If you can write a logical argument or persuasive essay, you can write a program.

Introduction to Python

- We will be writing programs– commands to the computer to do something.
- A **programming language** is a stylized way of writing those commands.
- If you can write a logical argument or persuasive essay, you can write a program.
- Our first language, Python, is popular for its ease-of-use, flexibility, and extendibility.

Introduction to Python

- We will be writing programs– commands to the computer to do something.
- A **programming language** is a stylized way of writing those commands.
- If you can write a logical argument or persuasive essay, you can write a program.
- Our first language, Python, is popular for its ease-of-use, flexibility, and extendibility.
- The first lab goes into step-by-step details of getting Python running.

Introduction to Python

- We will be writing programs– commands to the computer to do something.
- A **programming language** is a stylized way of writing those commands.
- If you can write a logical argument or persuasive essay, you can write a program.
- Our first language, Python, is popular for its ease-of-use, flexibility, and extendibility.
- The first lab goes into step-by-step details of getting Python running.
- We'll look at the design and basic structure (no worries if you haven't tried it yet in lab).

First Program: `Hello, World!`

Demo in `pythonTutor`

First Program: Hello, World!

```
#Name:  Thomas Hunter
#Date:  September 1, 2017
#This program prints:  Hello, World!

print("Hello, World!")
```

First Program: Hello, World!

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)
```

First Program: Hello, World!

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)

print("Hello, World!")                         ← Prints the string "Hello, World!" to the screen
```

First Program: `Hello, World!`

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)

print("Hello, World!")                         ← Prints the string "Hello, World!" to the screen
```

- Output to the screen is: `Hello, World!`

First Program: `Hello, World!`

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)

print("Hello, World!")                         ← Prints the string "Hello, World!" to the screen
```

- Output to the screen is: `Hello, World!`
- Can replace `Hello, World!` with another string to be printed.

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
```

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
```

Spring18 here in Assembly Hall
Who is L-M Miranda?

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
```

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS ‘98
#This program prints intro lyrics

print(’Get your education,’)
print("don’t forget from whence you came, and")
```

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
print("don't forget from whence you came, and")
print("The world's gonna know your name.")
```

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
print("don't forget from whence you came, and")
print("The world's gonna know your name.")
```

- Each print statement writes its output on a new line.

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
print("don't forget from whence you came, and")
print("The world's gonna know your name.")
```

- Each print statement writes its output on a new line.
- Results in three lines of output.

Variations on Hello, World!

```
#Name:  L-M Miranda
#Date:  Hunter College HS '98
#This program prints intro lyrics

print('Get your education,')
print("don't forget from whence you came, and")
print("The world's gonna know your name.")
```

- Each print statement writes its output on a new line.
- Results in three lines of output.
- Can use single or double quotes, just need to match.

Turtles Introduction

- A simple, whimsical graphics package for Python.

Turtles Introduction

- A simple, whimsical graphics package for Python.
- Dates back to Logo Turtles in the 1960s.

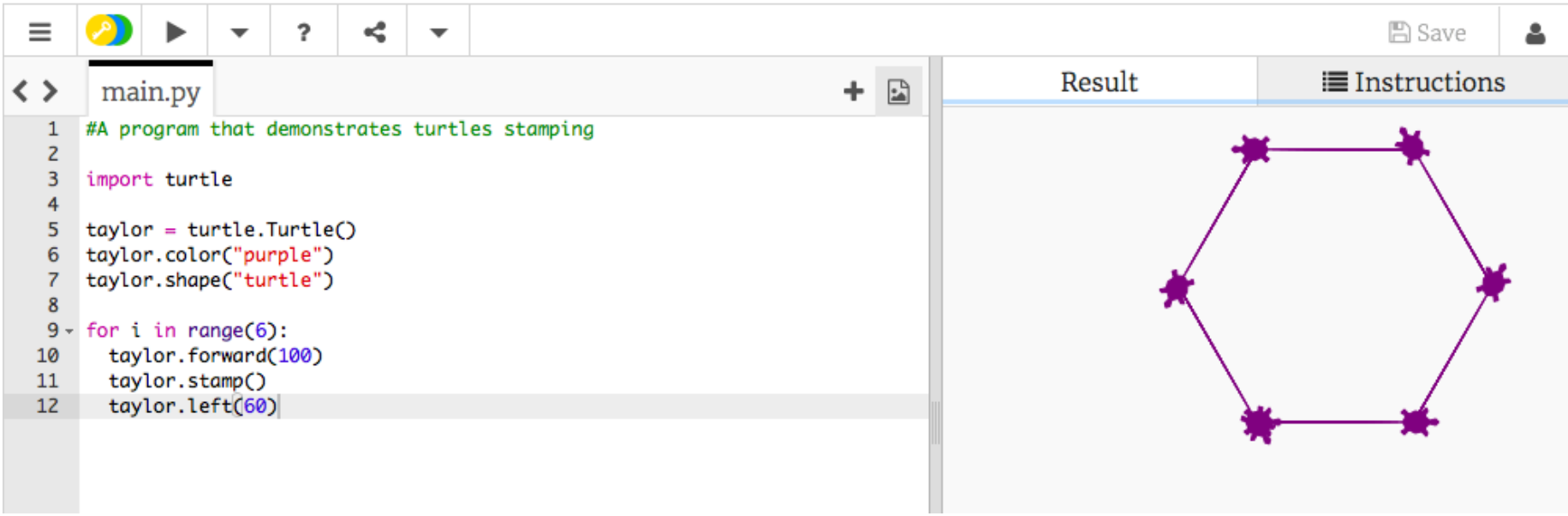
Turtles Introduction

- A simple, whimsical graphics package for Python.
- Dates back to Logo Turtles in the 1960s.
- (Demo from webpage)

Turtles Introduction

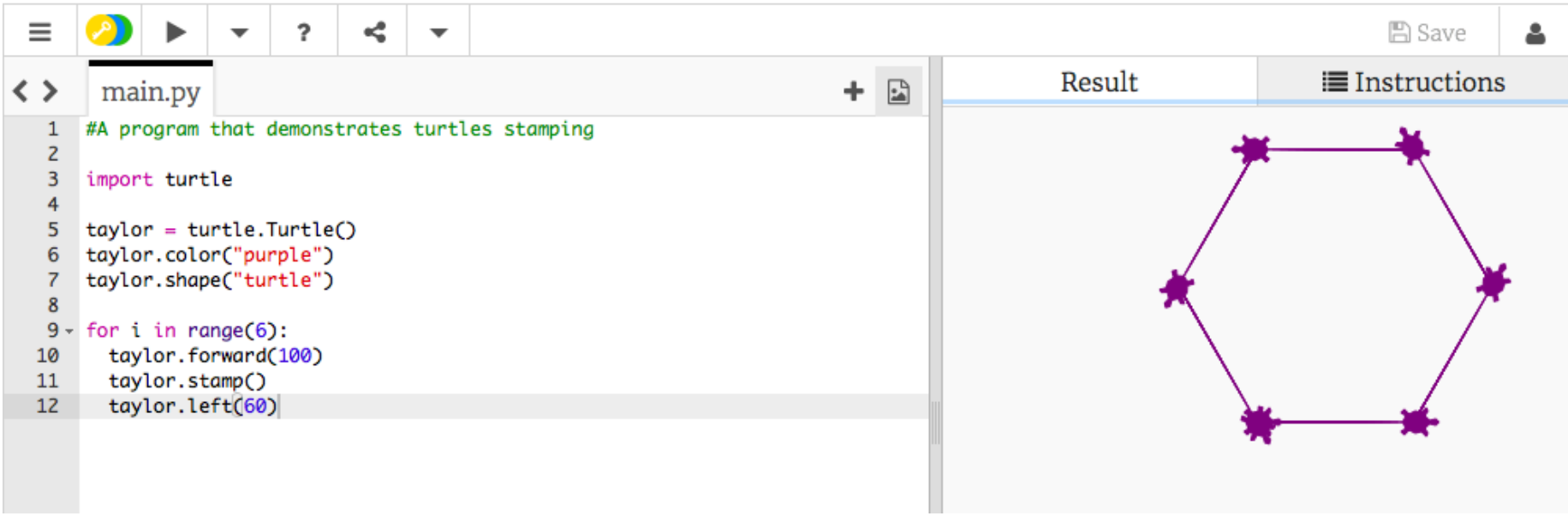
- A simple, whimsical graphics package for Python.
- Dates back to Logo Turtles in the 1960s.
- (Demo from webpage)
- (Fancier turtle demo)

Turtles Introduction



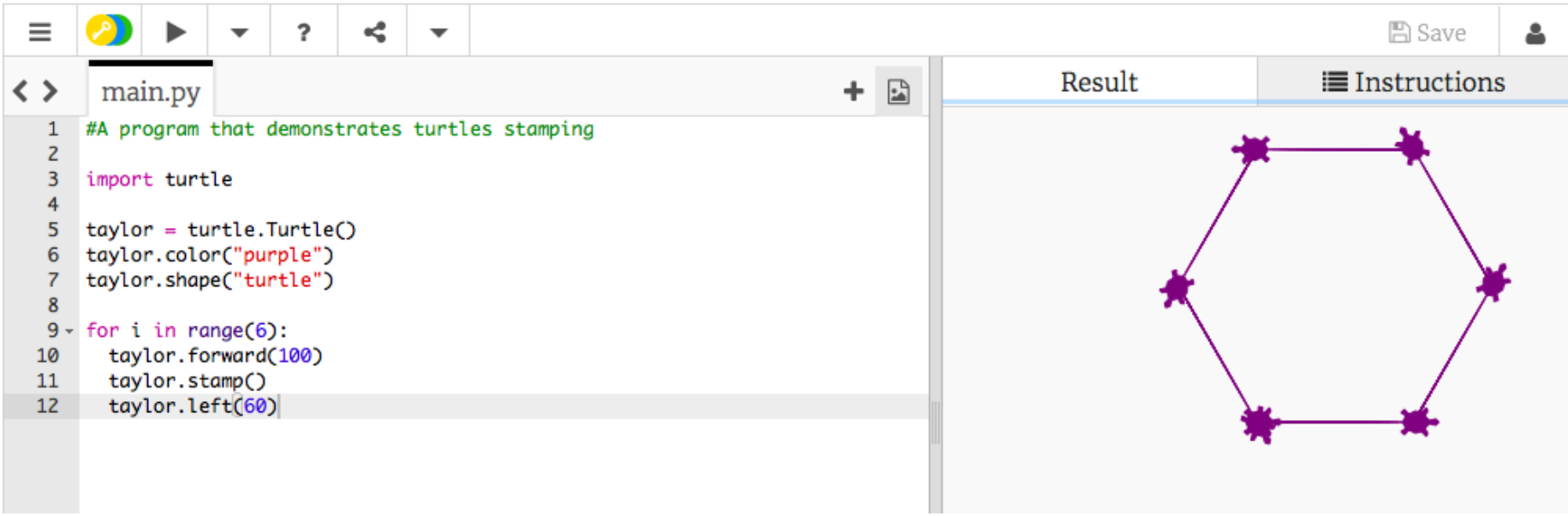
- Creates a turtle, called `taylor`.

Turtles Introduction



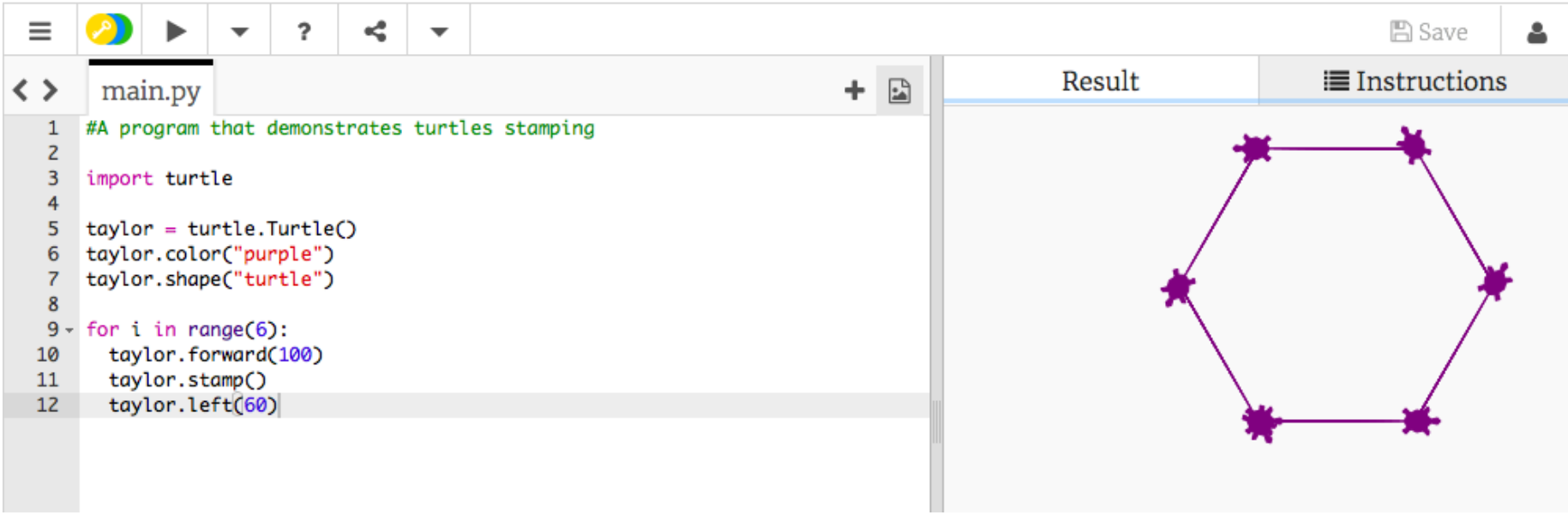
- Creates a turtle, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).

Turtles Introduction



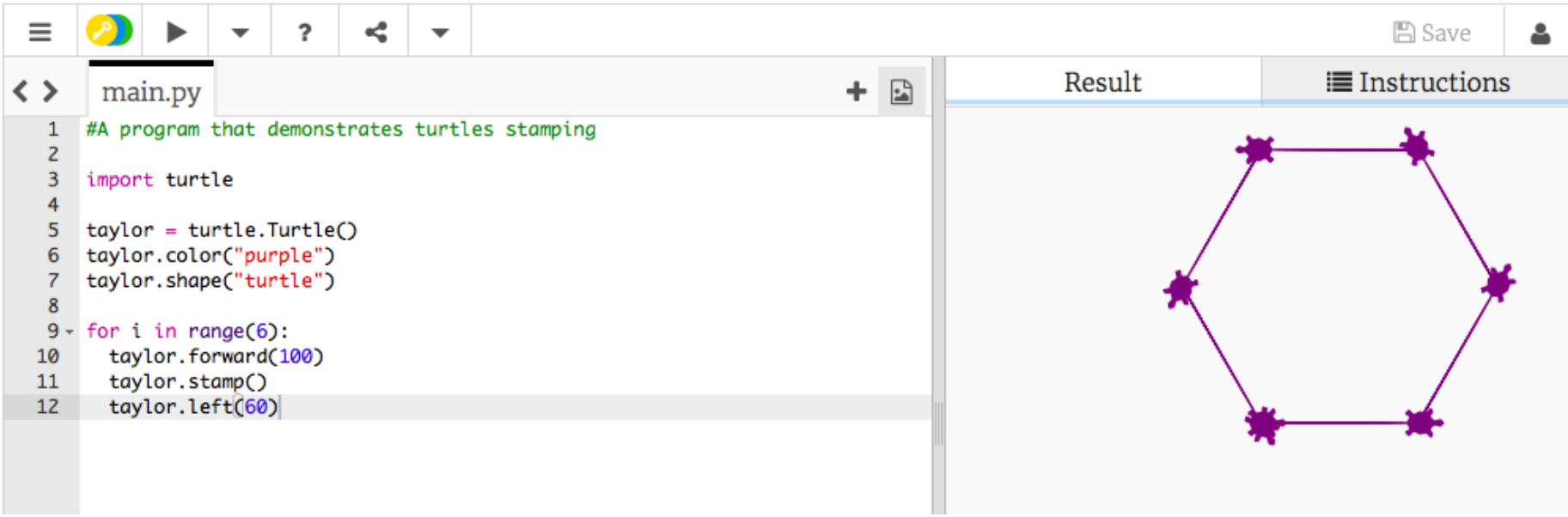
- Creates a turtle, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:

Turtles Introduction



- Creates a turtle, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
  - Move forward; stamp; and turn left 60 degrees.

Turtles Introduction



- Creates a turtle, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
  - Move forward; stamp; and turn left 60 degrees.
- Repeats any instructions **indented** in the "loop block"

Challenge Question

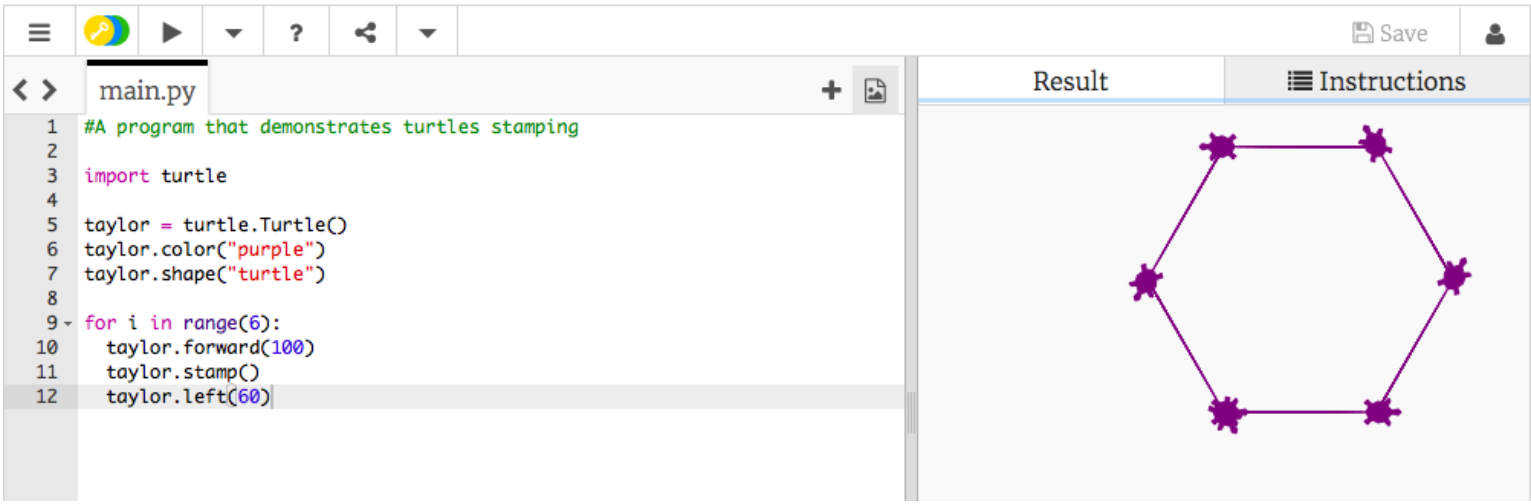
On a piece of paper (ungraded):

1. Write a program that will draw a 10-sided polygon.
2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

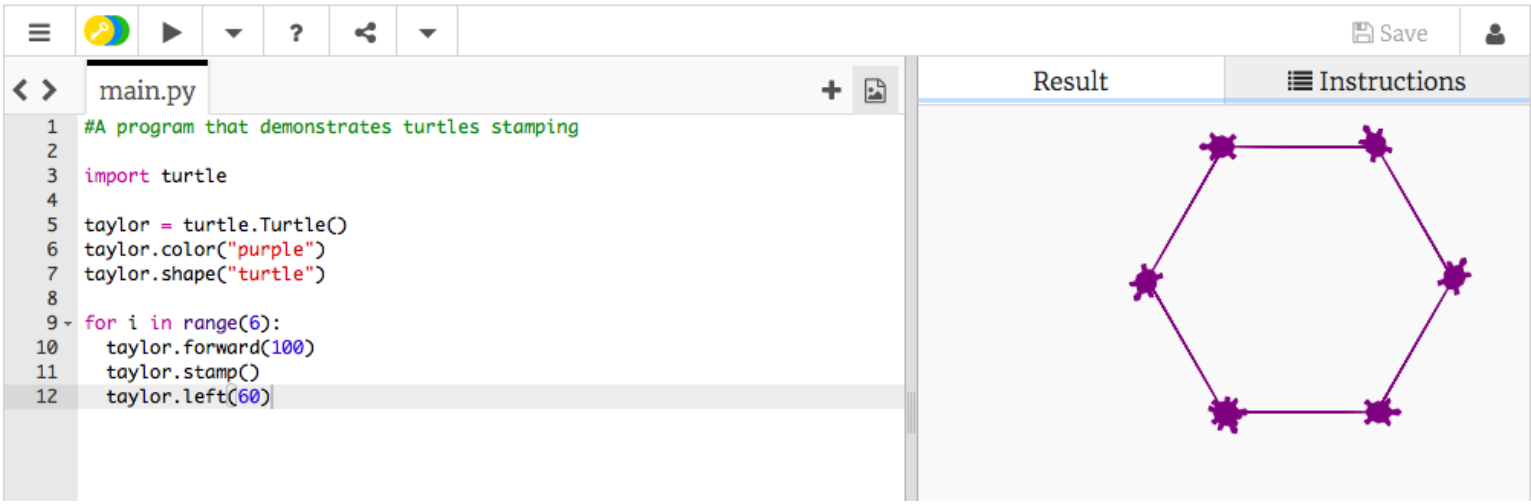
   three times.

Decagon Program



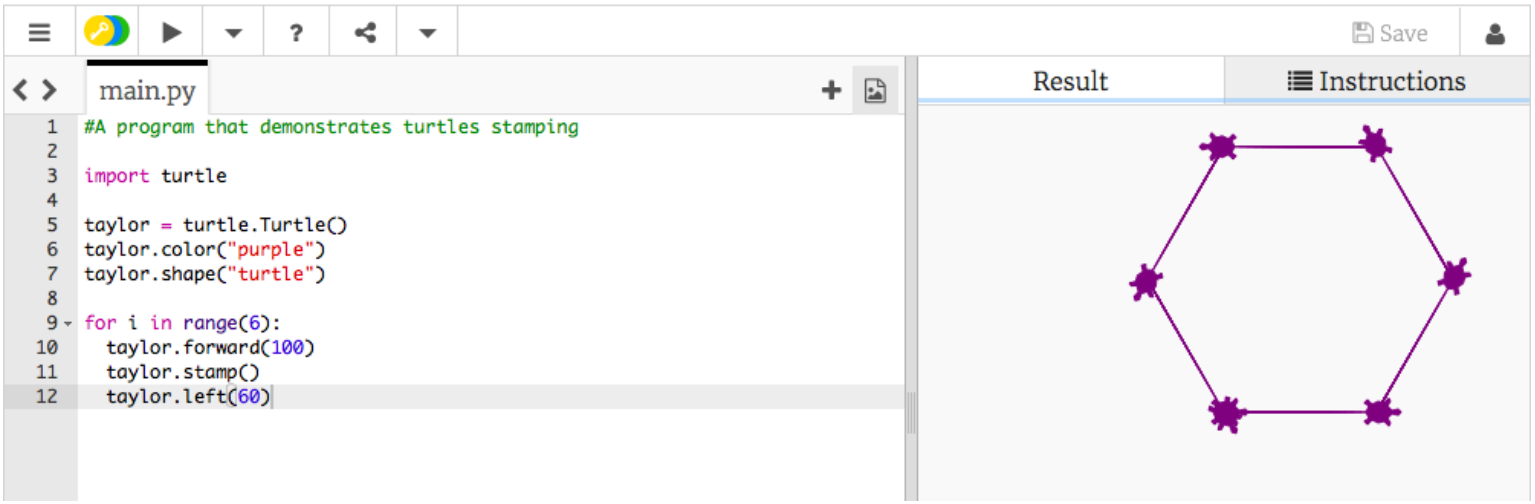
- Start with the hexagon program.

Decagon Program



- Start with the hexagon program.
- Has 10 sides (instead of 6), so change the `range(6)` to `range(10)`.

Decagon Program



- Start with the hexagon program.
- Has 10 sides (instead of 6), so change the `range(6)` to `range(10)`.
- Makes 10 turns (instead of 6),
  so change the `taylor.left(60)` to `taylor.left(360/10)`.

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

- Repeats three times, so, use `range(3)`:

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

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

- Repeats three times, so, use `range(3)`:

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

- Instead of turtle commands, repeating a print statement.

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

- Repeats three times, so, use `range(3)`:

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

- Instead of turtle commands, repeating a print statement.
- Completed program:

  ```
  # Your name here!
  for i in range(3):
      print("I'm lookin' for a mind at work!")
  ```

What is an Algorithm?

From our textbook:

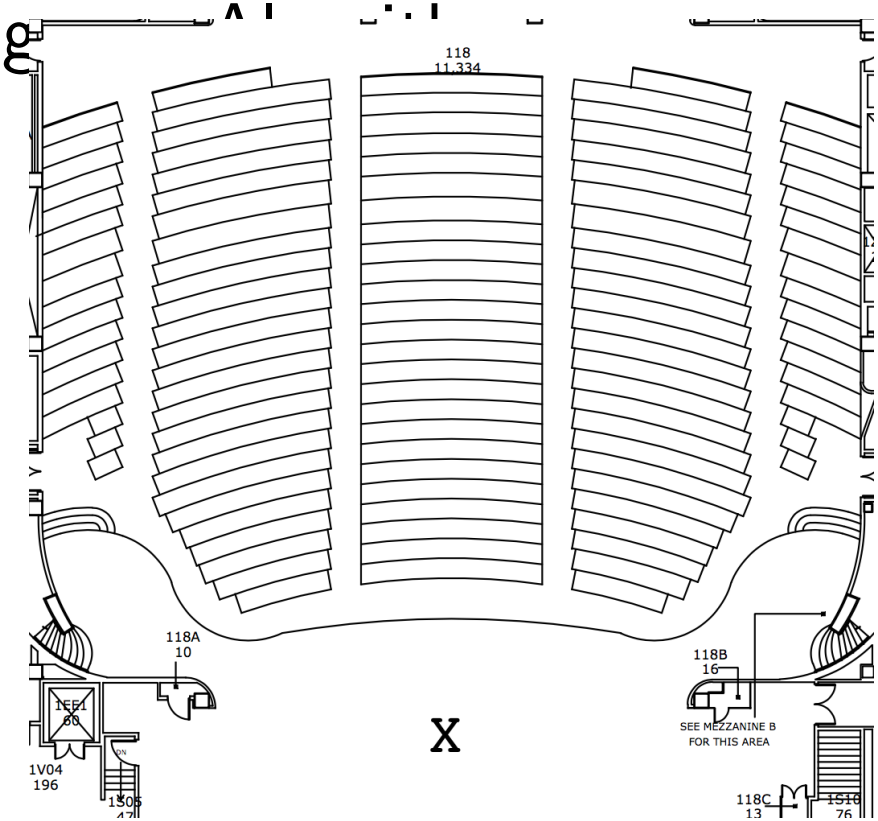
- An algorithm is a process or set of rules to be followed to solve a problem.

What is an Algorithm?

From our textbook:

- An algorithm is a process or set of rules to be followed to solve a problem.
- Programming is a skill that allows a computer scientist to take an algorithm and represent it in a notation (a program) that can be followed by a computer.

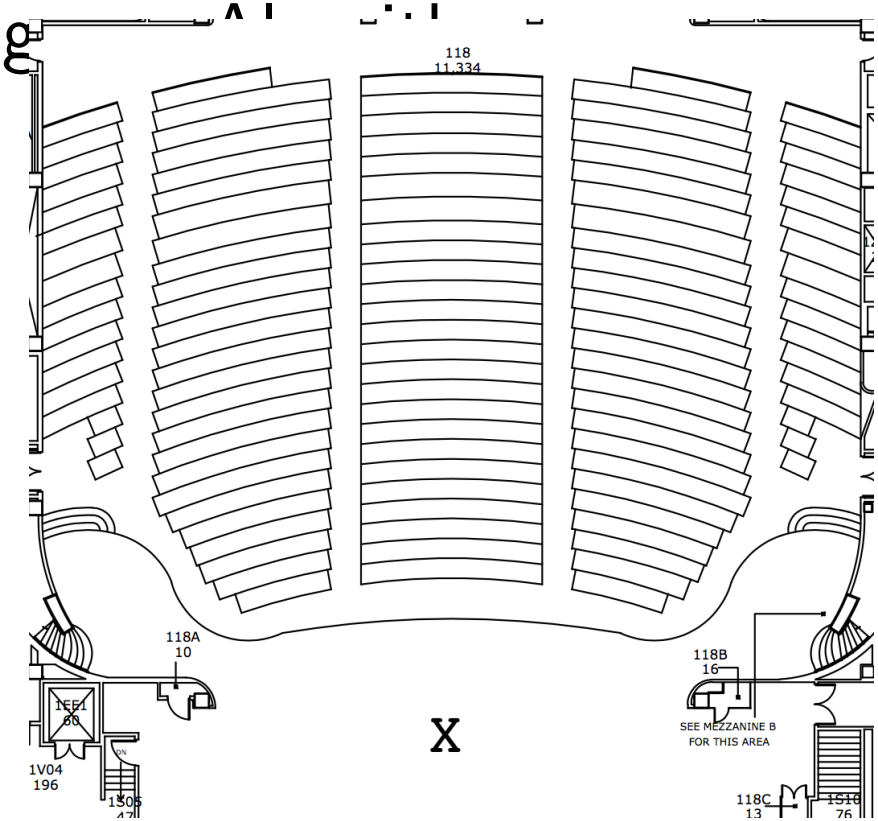
Challenge: Desig



On a piece of paper:

1. Choose a random location on this map.
2. Write an algorithm (step-by-step directions) to get to `X`.

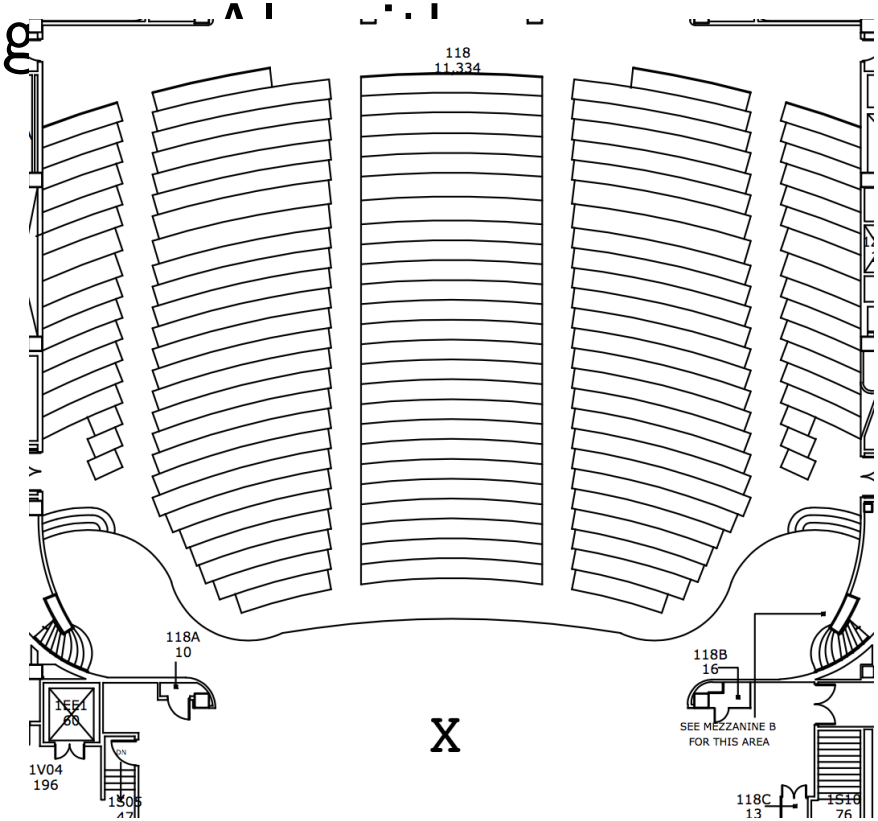
Challenge: Desig



On a piece of paper:

1. Choose a random location on this map.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:

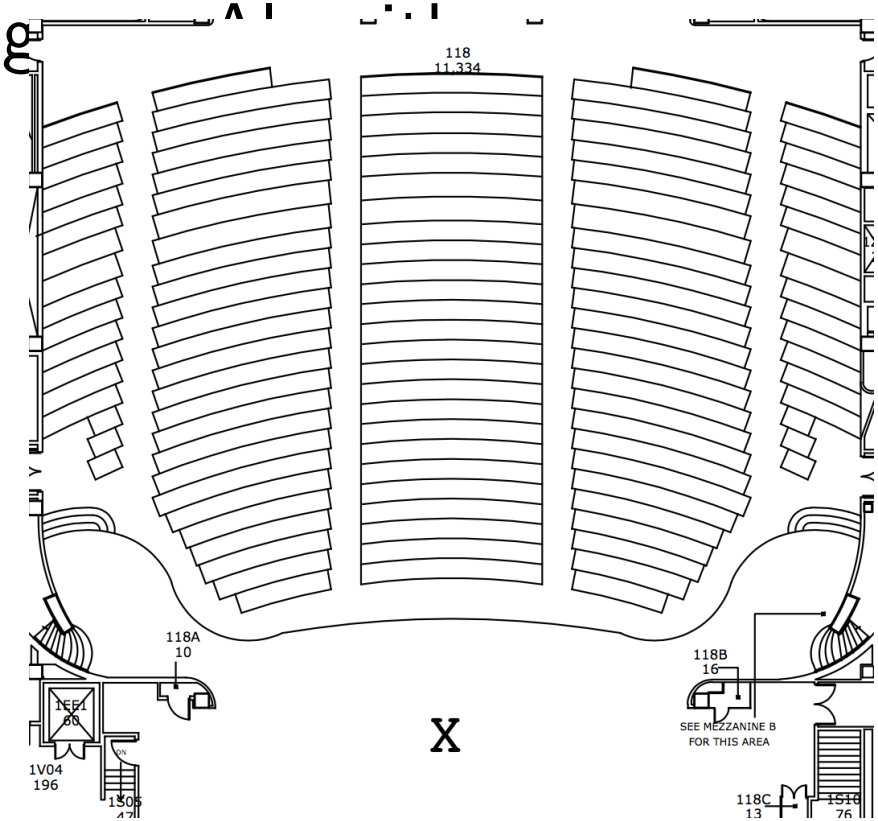
Challenge: Desig



On a piece of paper:

1. Choose a random location on this map.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:
   - Use `turtle` commands.

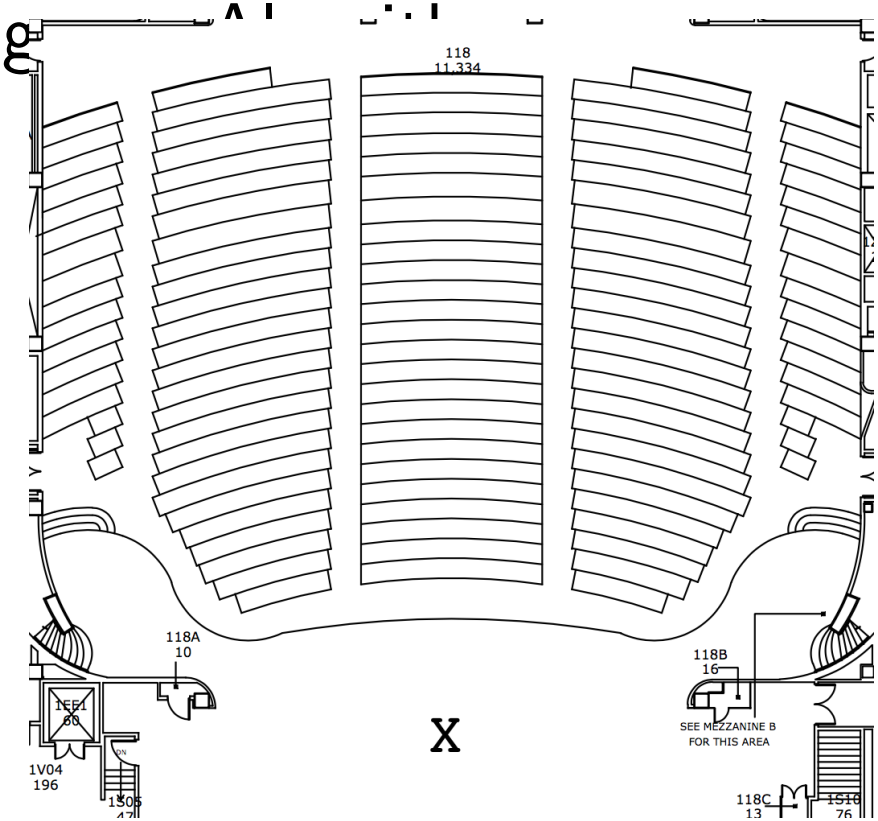
Challenge: Desig



On a piece of paper:

1. Choose a random location on this map.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:
   - Use `turtle` commands.
   - Do not run turtles into walls, chairs, obstacles, etc.

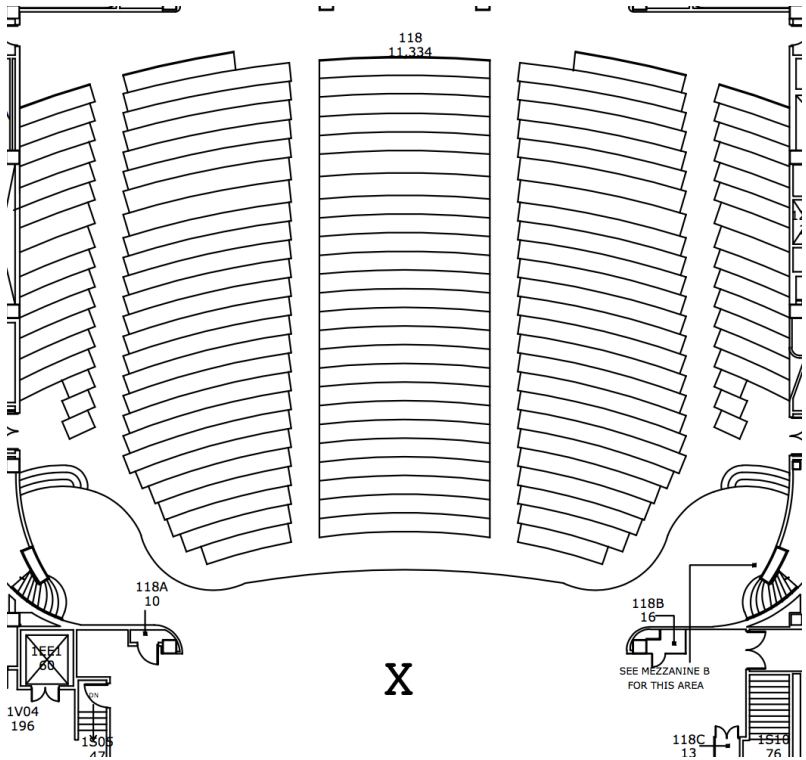
Challenge: Desig



On a piece of paper:

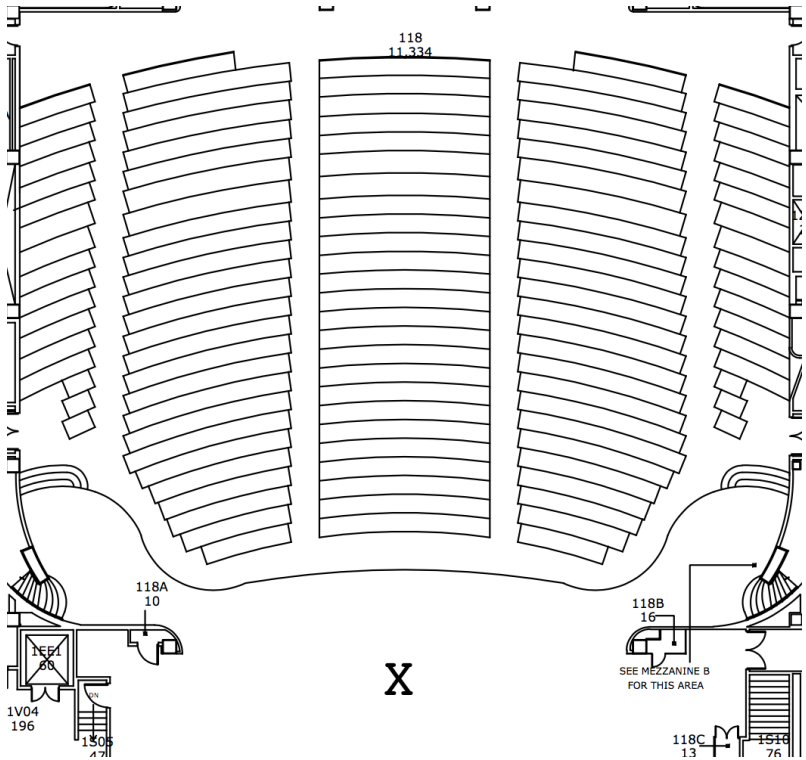
1. Choose a random location on this map.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:
   - Use `turtle` commands.
   - Do not run turtles into walls, chairs, obstacles, etc.
   - Turtles cannot climb walls, must use stairs.

Challenge



- Imagine someone needs to follow your directions exactly.

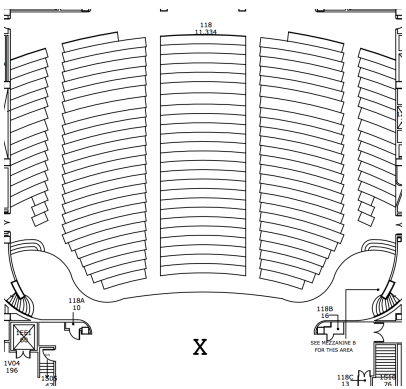
Challenge



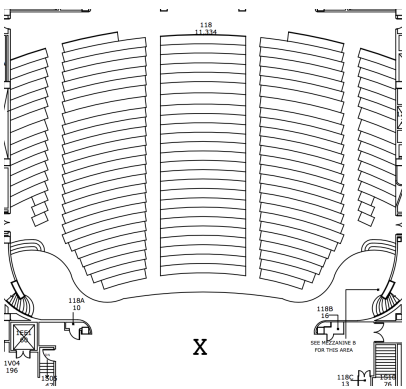
- Imagine someone needs to follow your directions exactly.
- Are there any changes needed to the directions (i.e. debug your work).

Recap

- Writing precise algorithms is difficult.

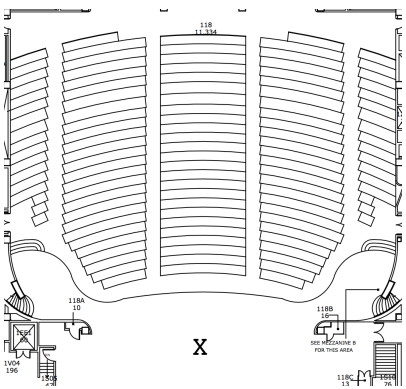


Recap



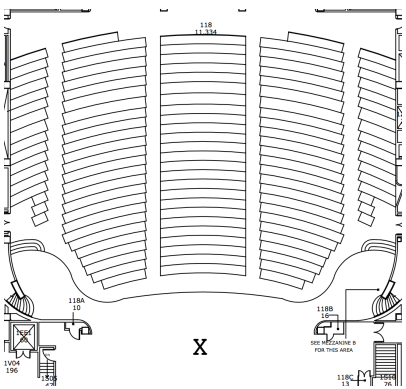
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



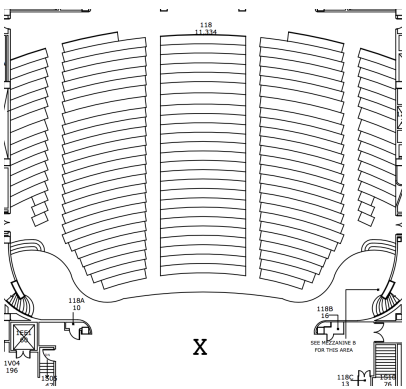
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,

Recap



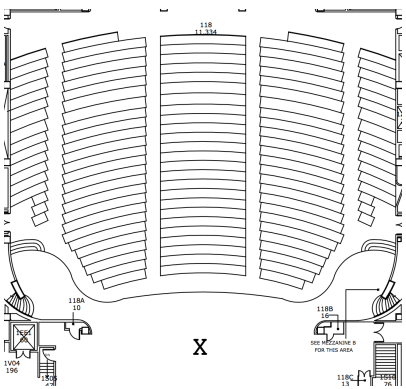
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,

Recap



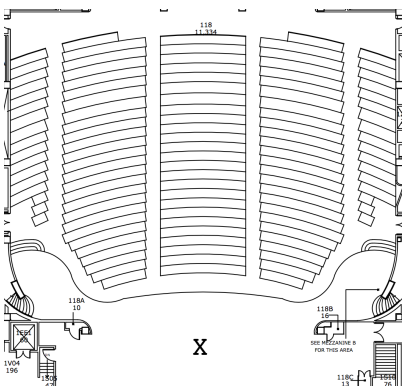
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - `strings`, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &

Recap



- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &
  - variables containing turtles.

Recap



- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &
  - variables containing turtles.
- Log in to Gradescope to complete Quiz 1.