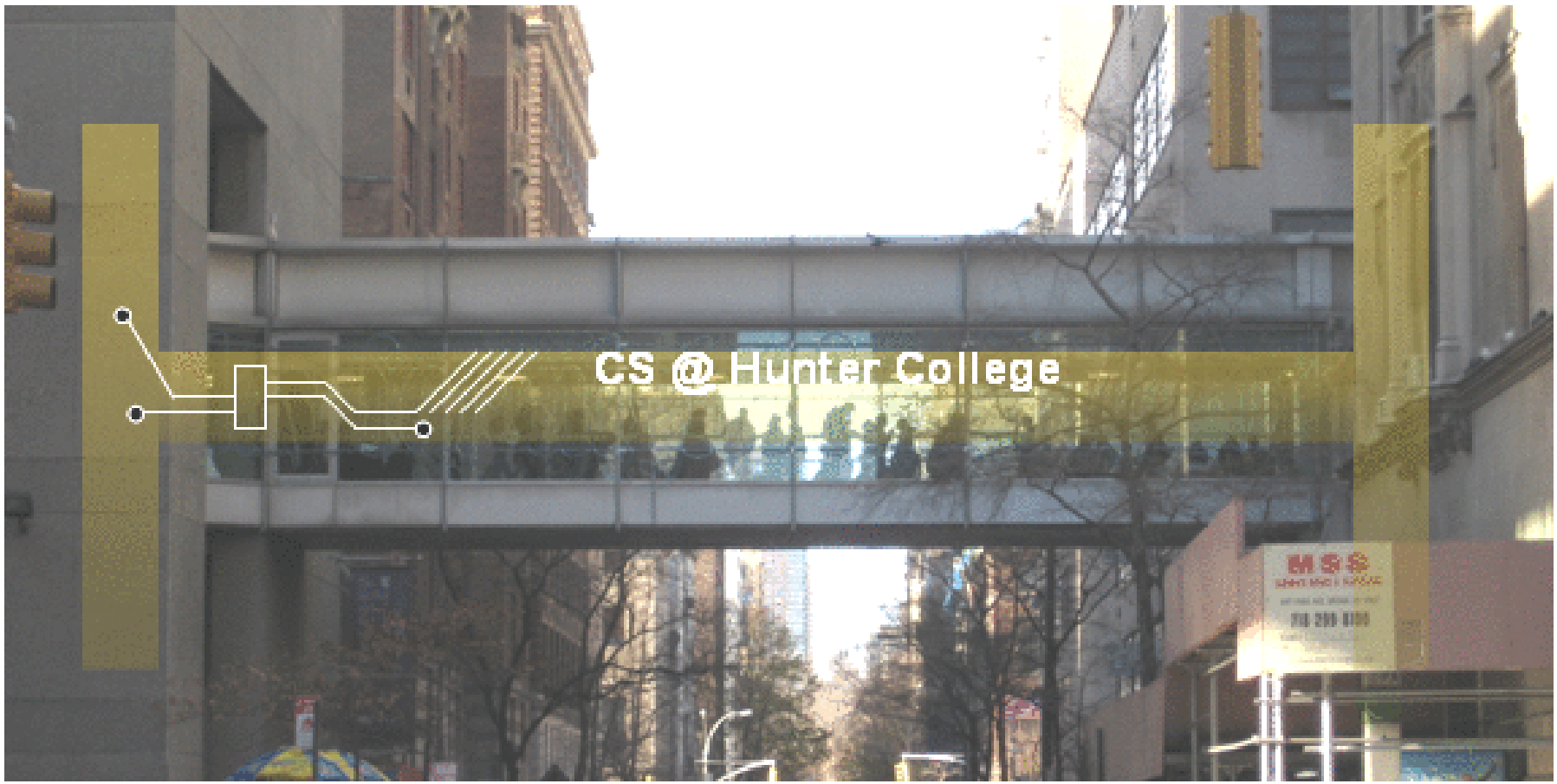
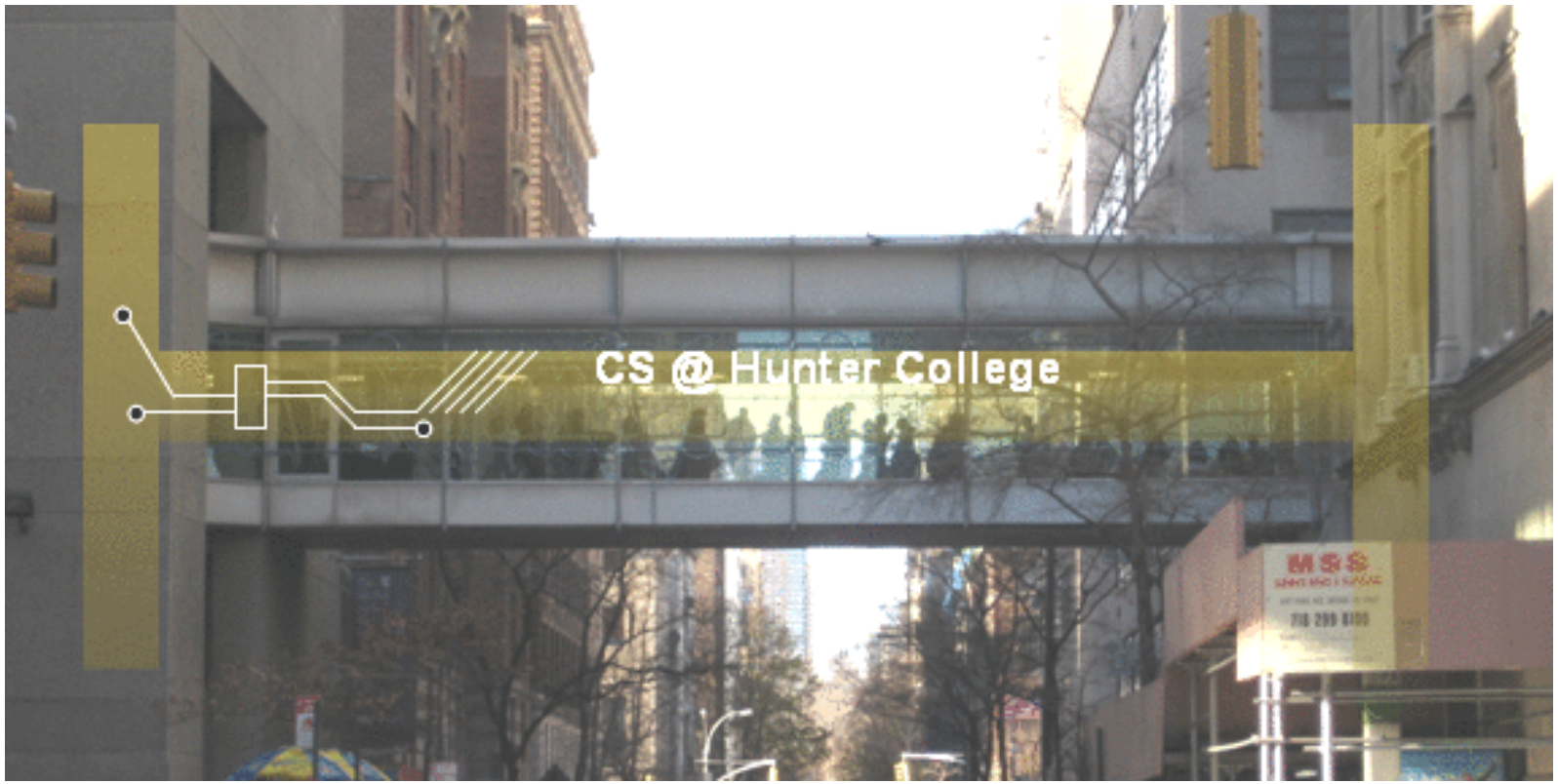
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Welcome



- This lecture will be recorded

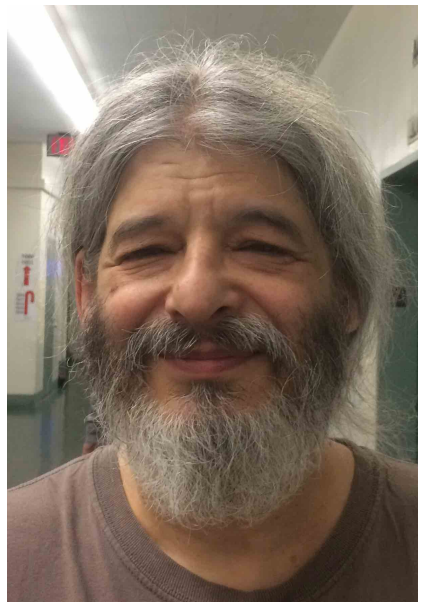
Introductions: Course Designers

Dr. Katherine St. John

Professor,
Interim Chair

Dr. William Sakas

Associate Professor,
Chair



Prof. Eric Schweitzer

Undergraduate Program
Coordinator

Introductions: Instructors

Katherine Howitt

Early College Initiative

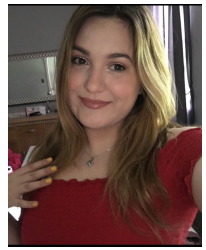
Dr. Tiziana Ligorio

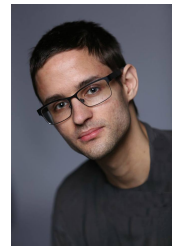
Large Lecture Course Coordinator

Introductions: Undergraduate Teaching Assistants


Aida Jevric


Arterio Rodrigues


Caiitlin Selca

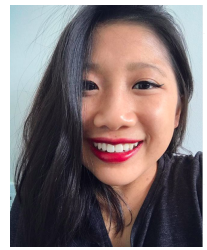

Illya Baburashvili


Leonardo Matone


Liulan Zheng


Lola Samigjonova

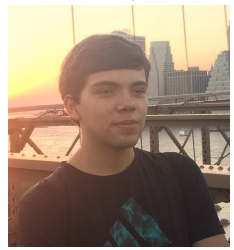

Mandy Yu


Nancy Ng


Nga Yu Lo


Owen Kunhardt

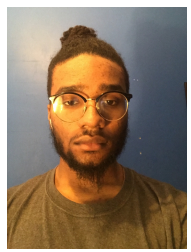

Patrick Chaca

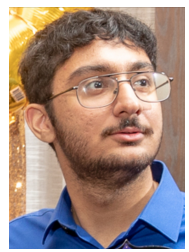

Ryan Chevarria


Sadab Hafiz


Shantel Dixon


Stephanie Yung


Tyler Robinson

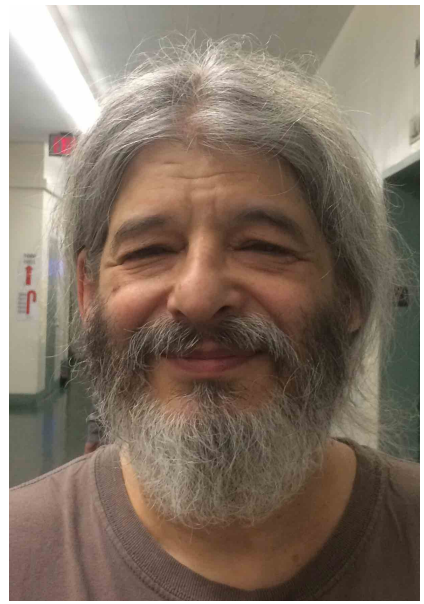

Yash Mahtani

Introductions: Advisors

Emely Peguero

Pre-majors & Early Majors

emely.pegueronova@hunter.cuny.edu



Eric Schweitzer

Undergraduate Program Coordinator

eschweit@hunter.cuny.edu

Where to find Course Content

- Course Website: https://huntercsci127.github.io/s21.html

Where to find Course Content

- Course Website: https://huntercsci127.github.io/s21.html
- Blackboard

Where to find Course Content

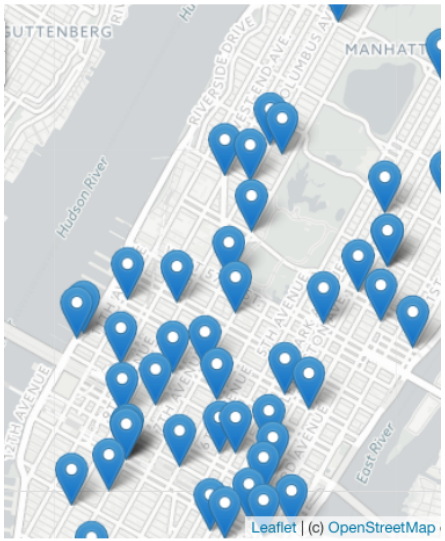
- Course Website: https://huntercsci127.github.io/s21.html
- Blackboard
- Gradescope (assessment)

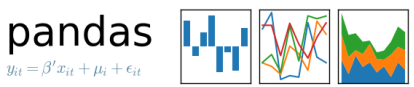
Syllabus

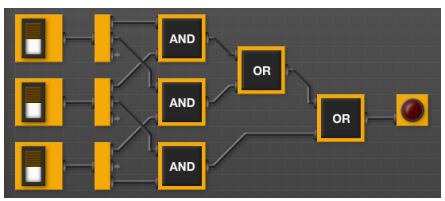
CSci 127: Introduction to Computer Science
Catalog Description: 3 hours, 3 credits: This course presents an overview of computer science (CS) with an emphasis on **problem-solving and computational thinking through 'coding'***: computer programming for beginners...*

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: Topics

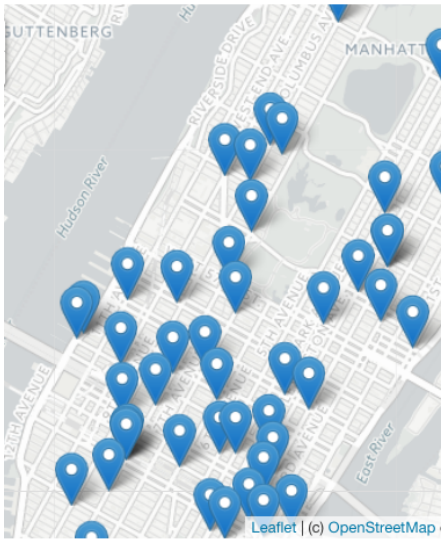


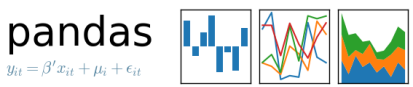


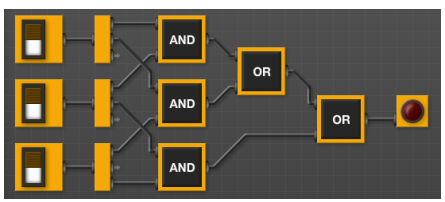


- **This course assumes no previous programming experience.**

Syllabus: Topics

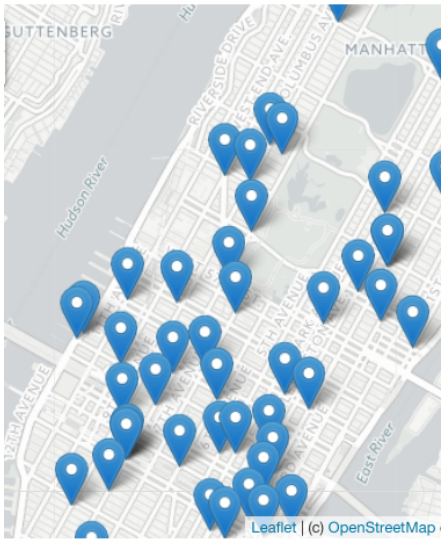


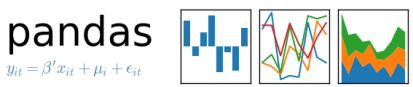


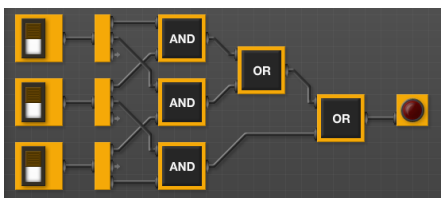


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:

Syllabus: Topics

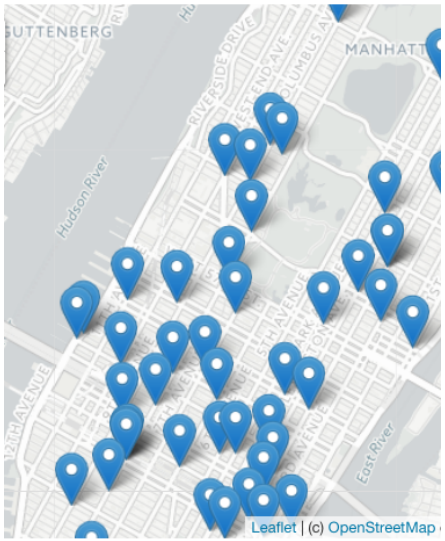


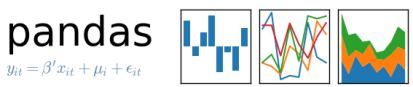


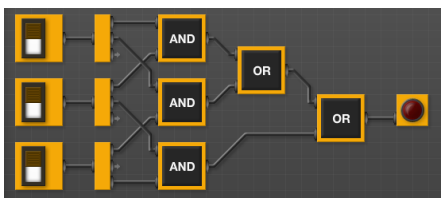


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,

Syllabus: Topics

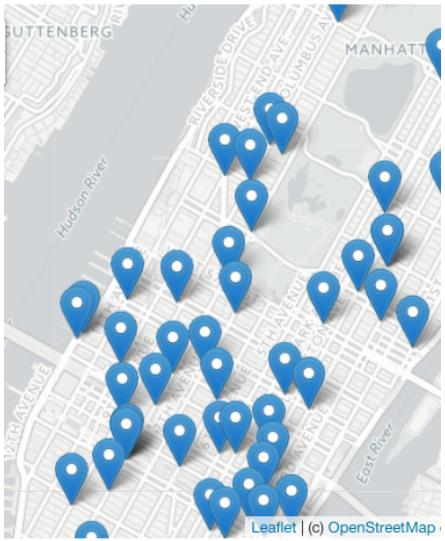


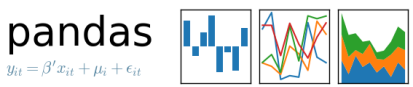


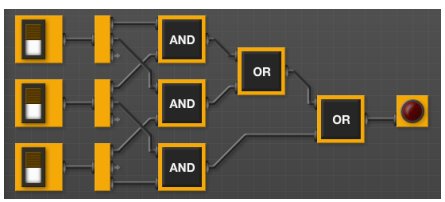


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),

Syllabus: Topics

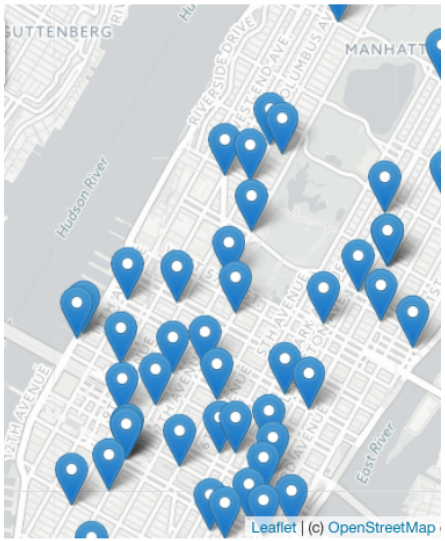


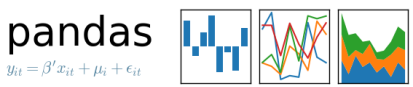


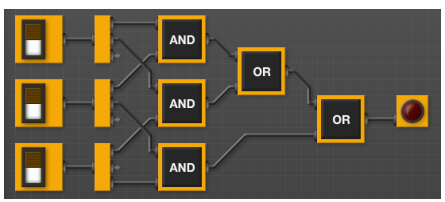


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:

Syllabus: Topics

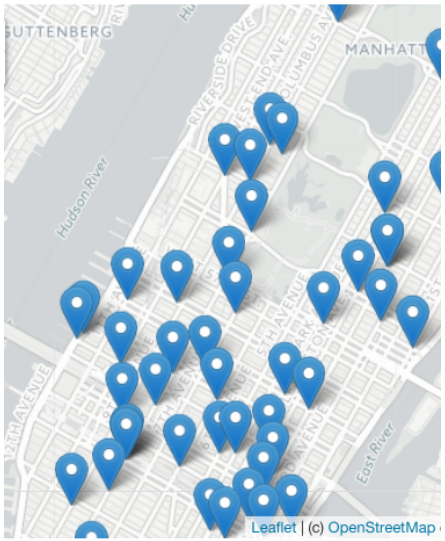


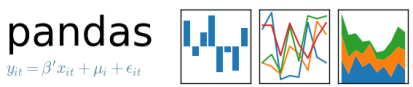


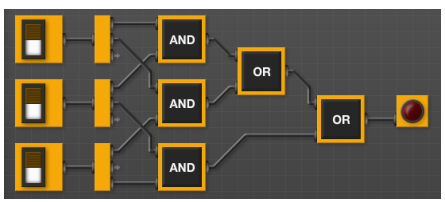


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,

Syllabus: Topics

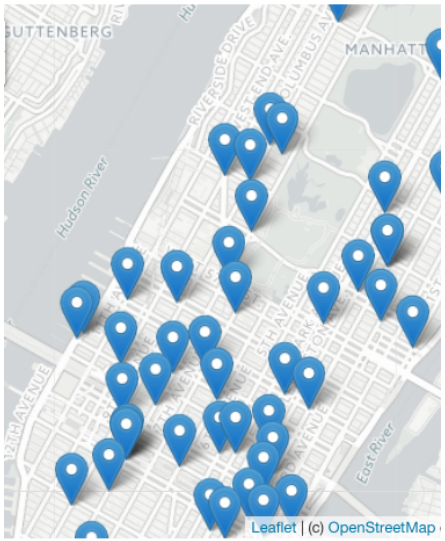


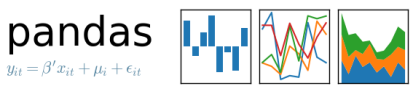


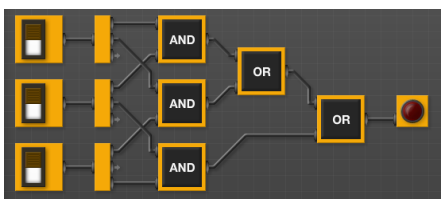


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

Syllabus: Topics

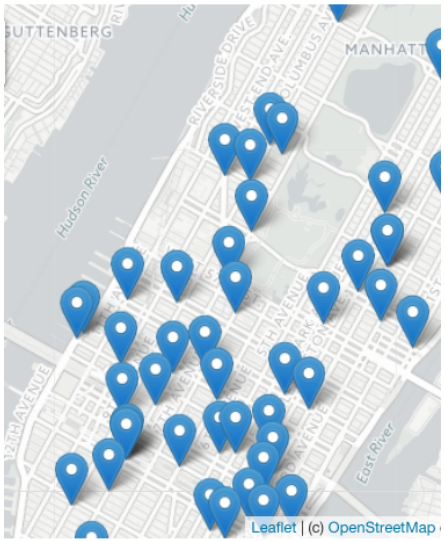


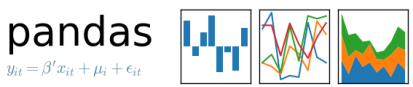


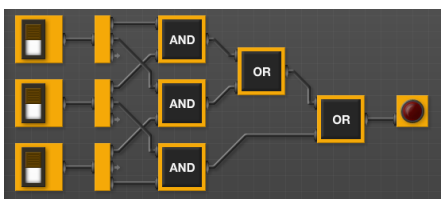


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

Syllabus: Topics

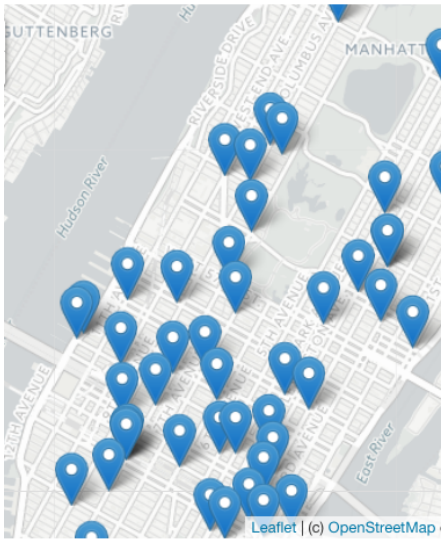






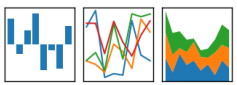
- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

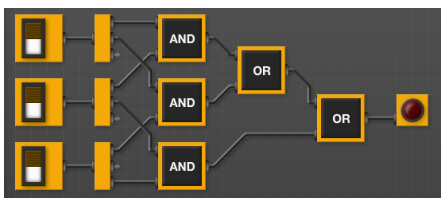
Syllabus: Topics



pandas

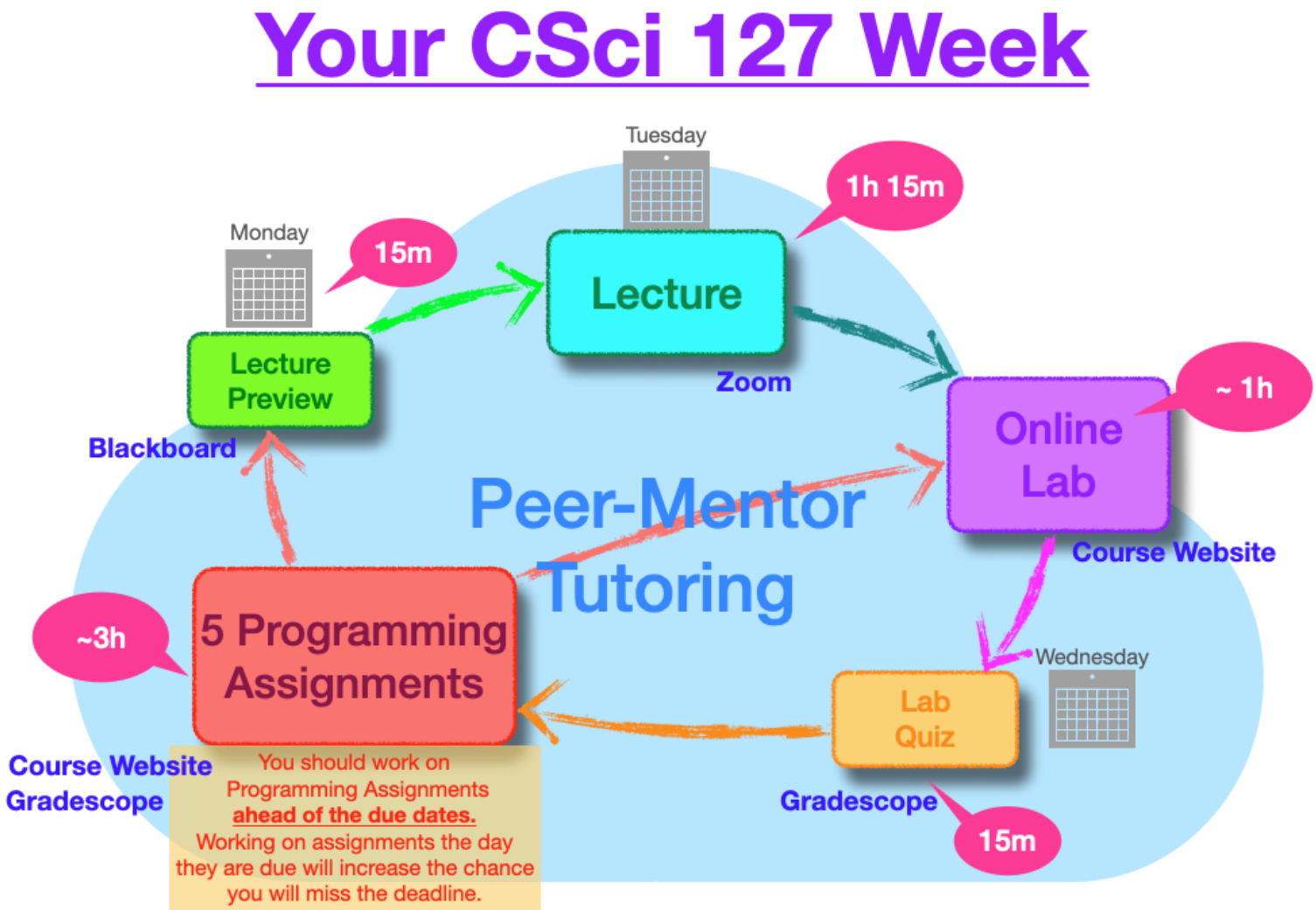
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



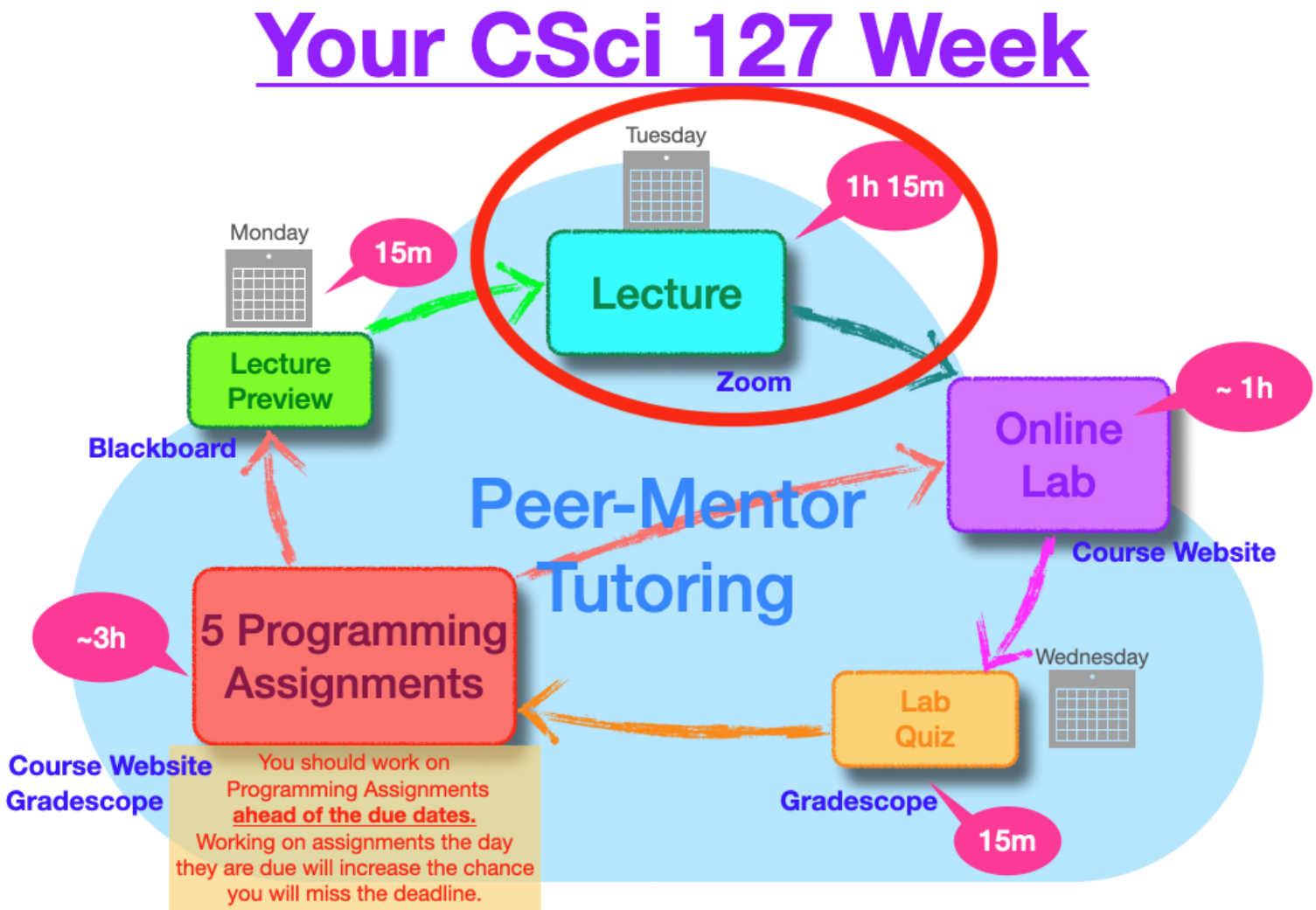


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

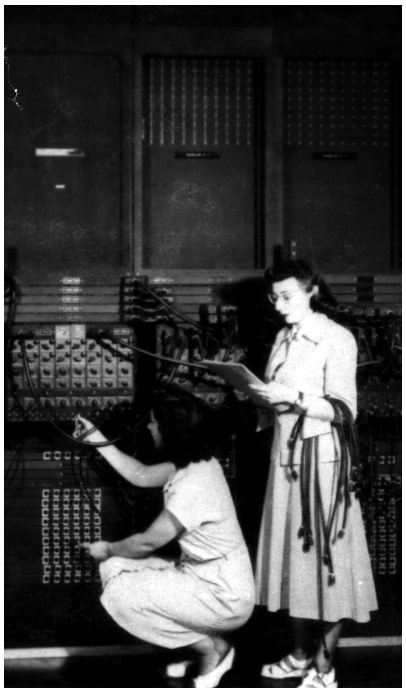
Course Structure



Course Structure



Lecture

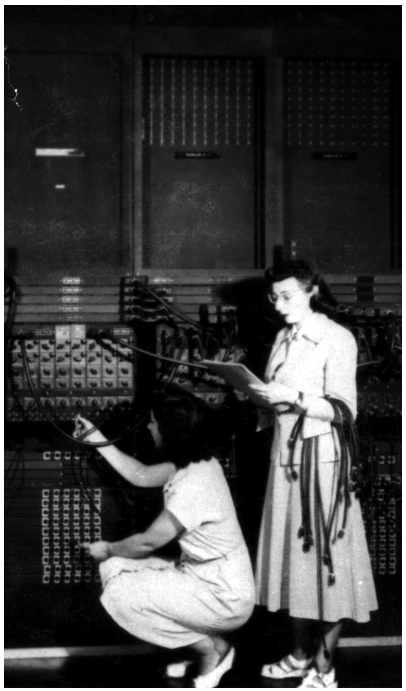


First "computers"

ENIAC, 1945.

- Tuesdays, 9:45-11:00am, on Zoom.

Lecture

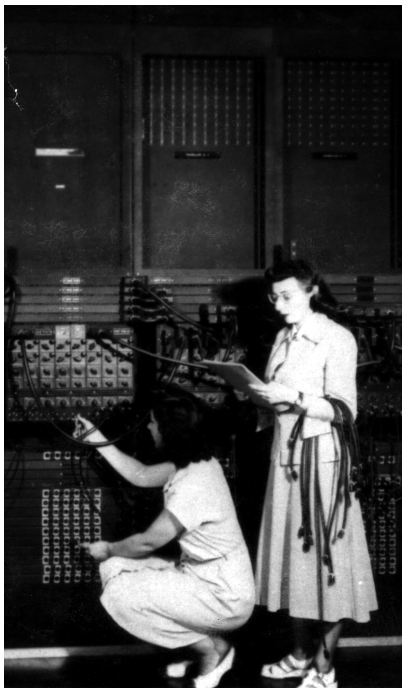


First "computers"

ENIAC, 1945.

- Tuesdays, 9:45-11:00am, on Zoom.
- Mix of explanation, challenges & active concept application.

Lecture

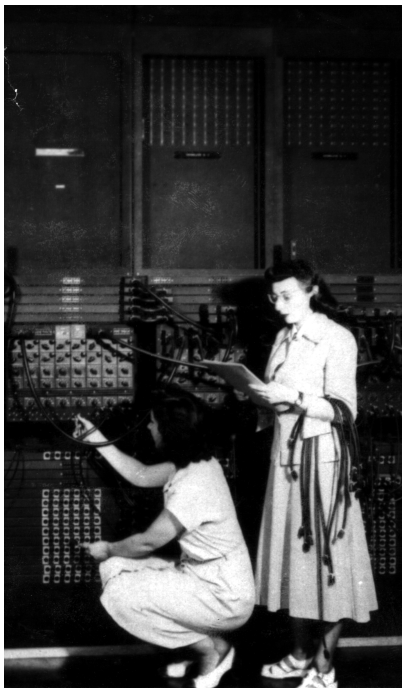


First "computers"

ENIAC, 1945.

- Tuesdays, 9:45-11:00am, on Zoom.
- Mix of explanation, challenges & active concept application.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).

Lecture

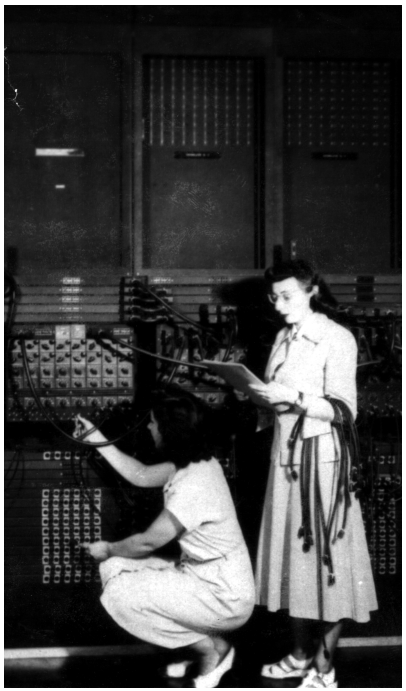


First "computers"

ENIAC, 1945.

- Tuesdays, 9:45-11:00am, on Zoom.
- Mix of explanation, challenges & active concept application.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).
- Lecture Quiz: on Gradescope during lecture.

Lecture

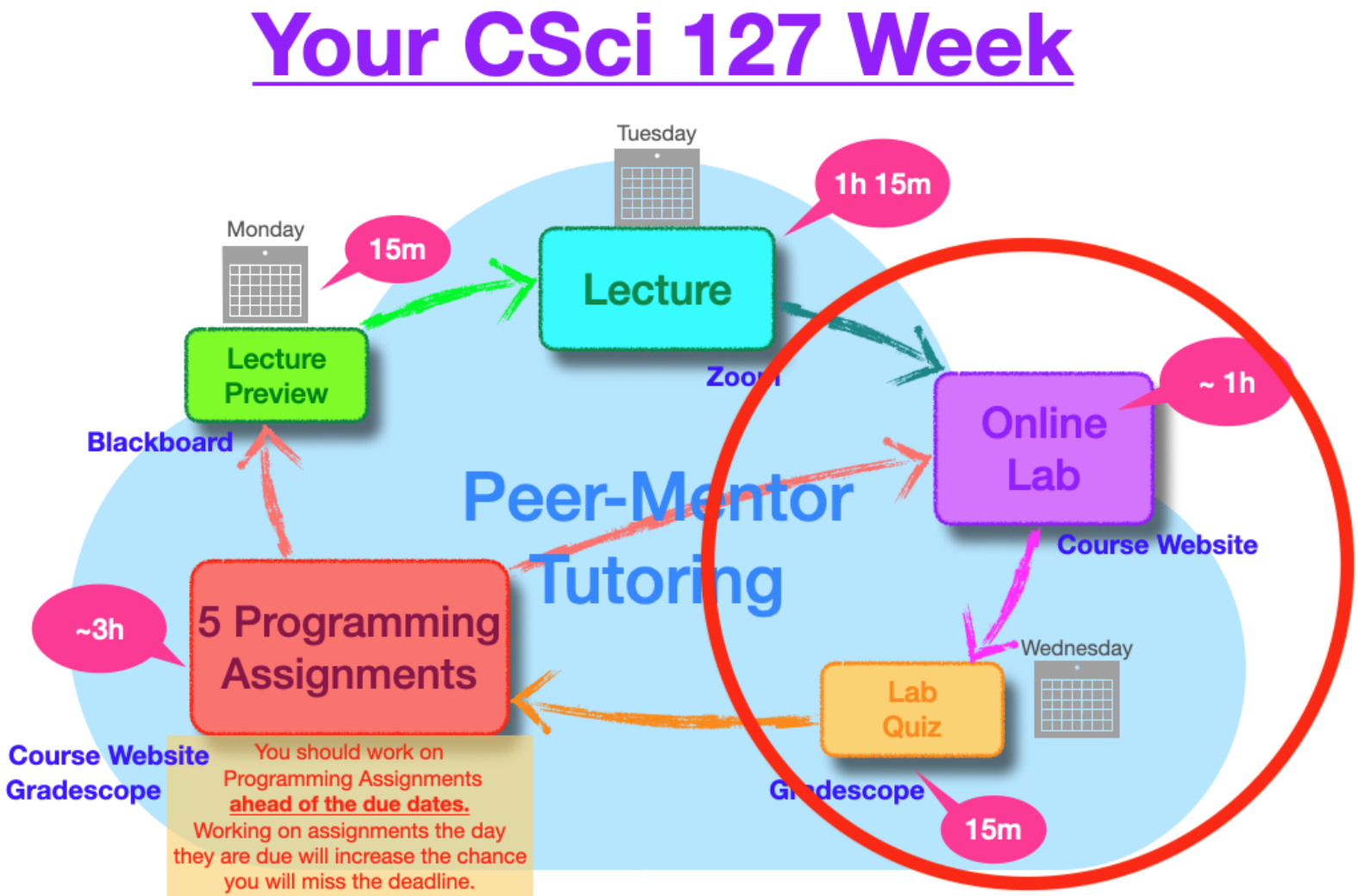


First "computers"

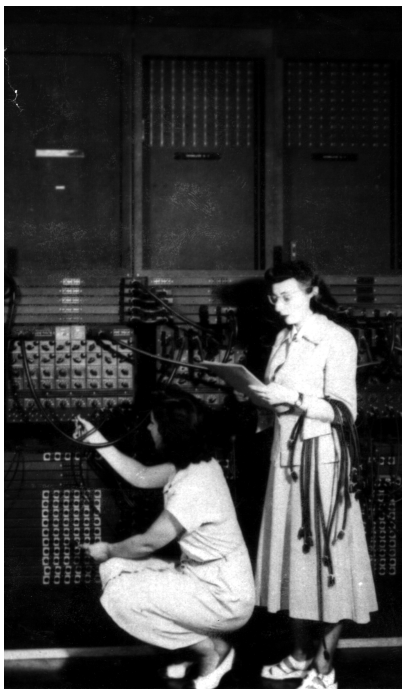
ENIAC, 1945.

- Tuesdays, 9:45-11:00am, on Zoom.
- Mix of explanation, challenges & active concept application.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).
- Lecture Quiz: on Gradescope during lecture.
- Ask questions in Q&A.

Course Structure



Online Lab & Quiz



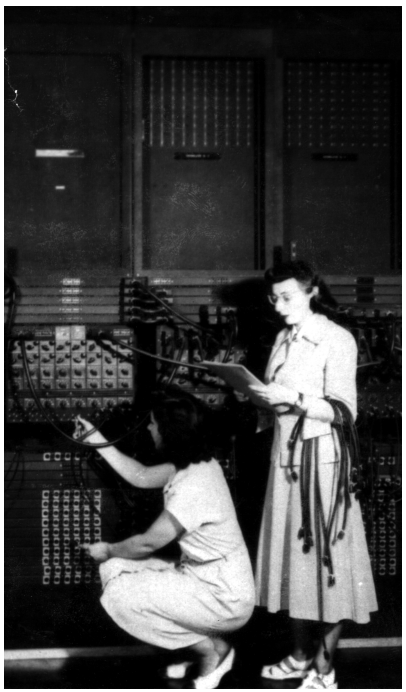
First "computers"

ENIAC, 1945.

Each Week:

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

Online Lab & Quiz



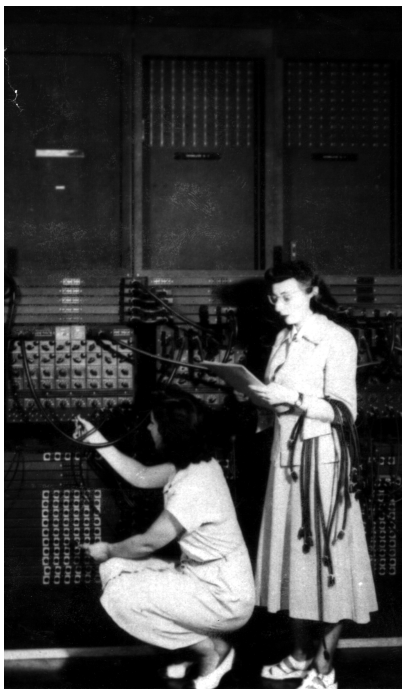
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.

Online Lab & Quiz



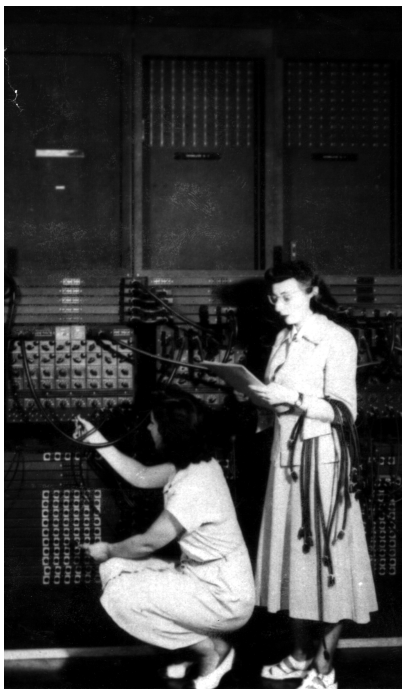
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.

Online Lab & Quiz



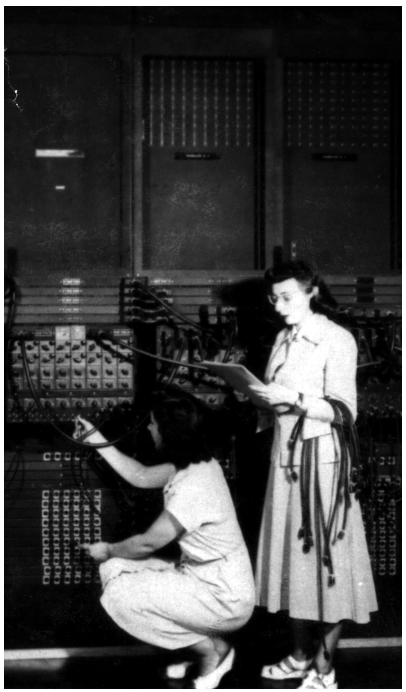
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.
- Alternatively, join a Lab Review session on Zoom (links on Blackboard / Synchronous Meetings)

Online Lab & Quiz



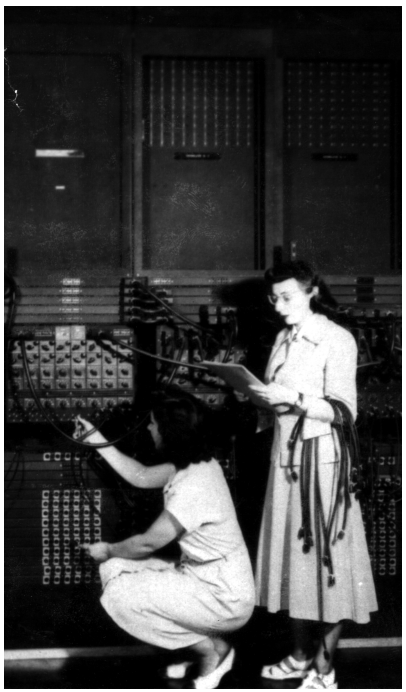
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.
- Alternatively, join a Lab Review session on Zoom (links on Blackboard / Synchronous Meetings)
- Lab content directly supports weekly programming assignments.

Online Lab & Quiz



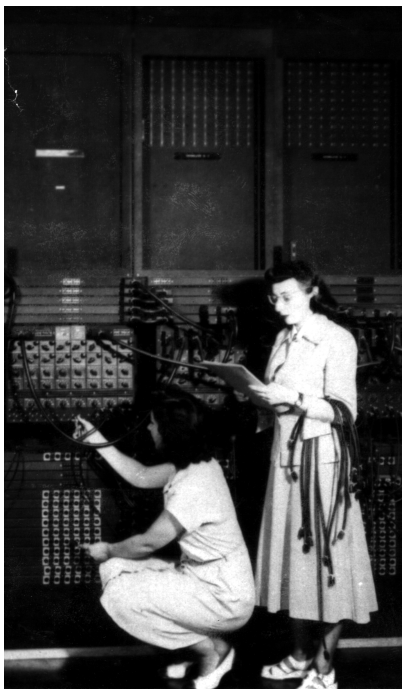
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.
- Alternatively, join a Lab Review session on Zoom (links on Blackboard / Synchronous Meetings)
- Lab content directly supports weekly programming assignments.
- **Lab Quiz** on Gradescope to assess your understanding of the Labs (**Due on Wedesdays 6pm**)

Online Lab & Quiz



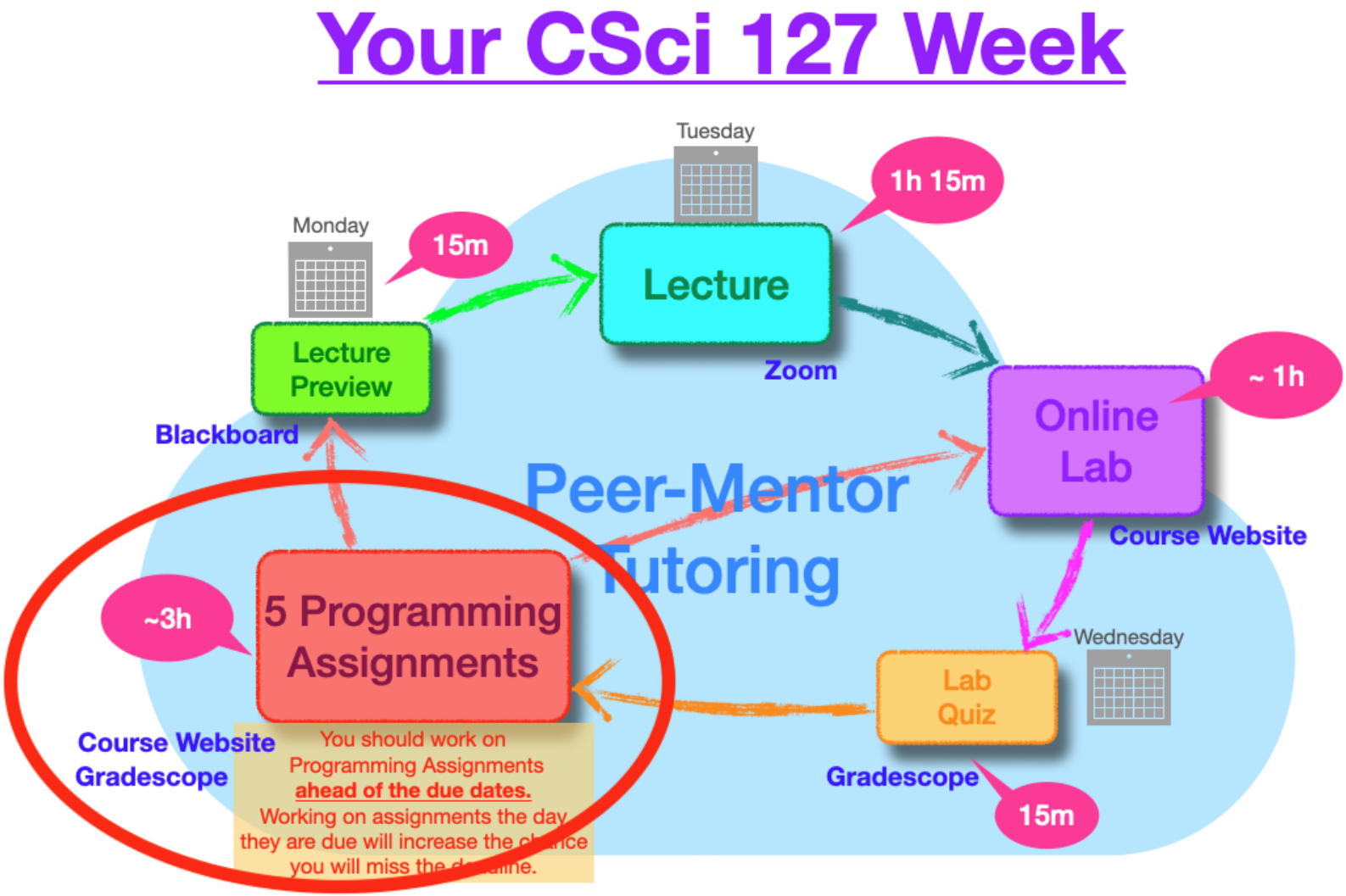
First "computers"

ENIAC, 1945.

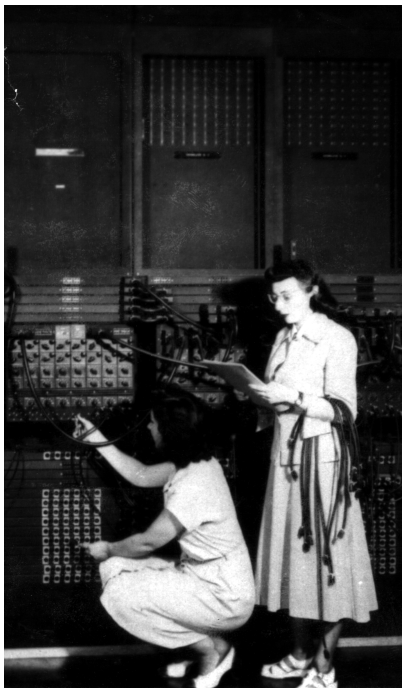
Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.
- Alternatively, join a Lab Review session on Zoom (links on Blackboard / Synchronous Meetings)
- Lab content directly supports weekly programming assignments.
- **Lab Quiz** on Gradescope to assess your understanding of the Labs (**Due on Wedesdays 6pm**)
- Labs found on course website (show)

Course Structure



Homework



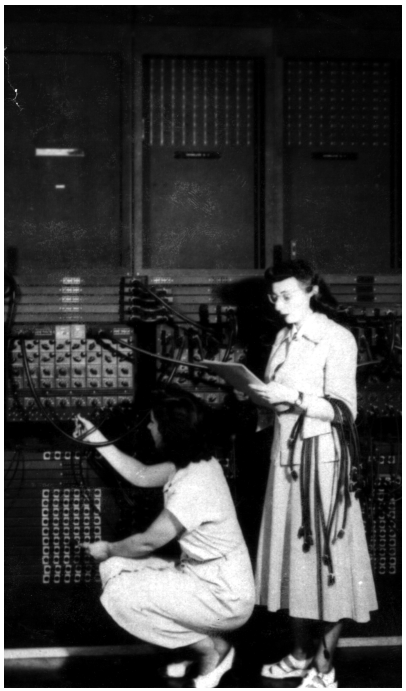
First "computers"

ENIAC, 1945.

Each Week:

- **5 Programming Assignments.**

Homework



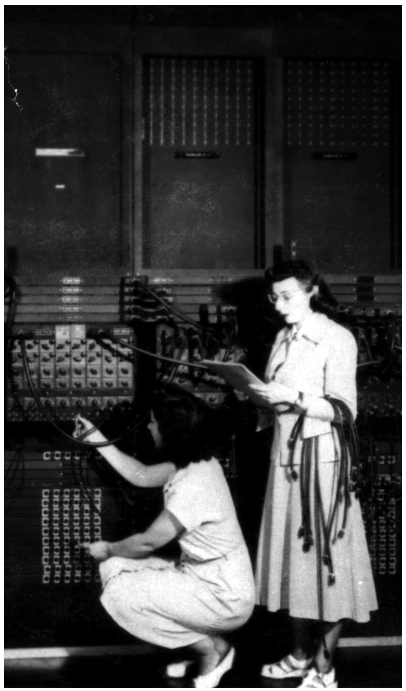
First "computers"

ENIAC, 1945.

Each Week:

- **5 Programming Assignments.**
- Description on Course Webpage.

Homework



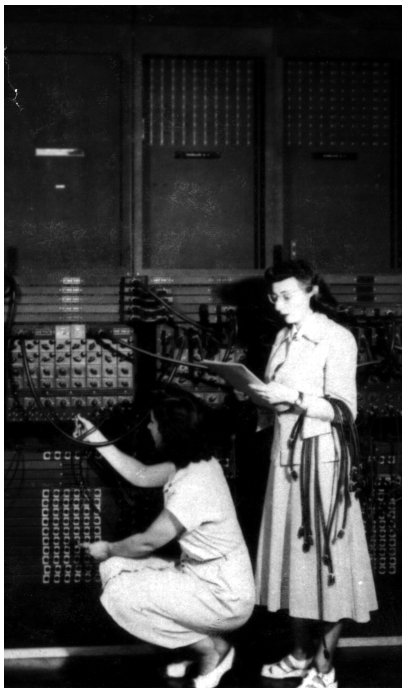
First "computers"

ENIAC, 1945.

Each Week:

- **5 Programming Assignments.**
- Description on Course Webpage.
- Implement and test on your computer.

Homework



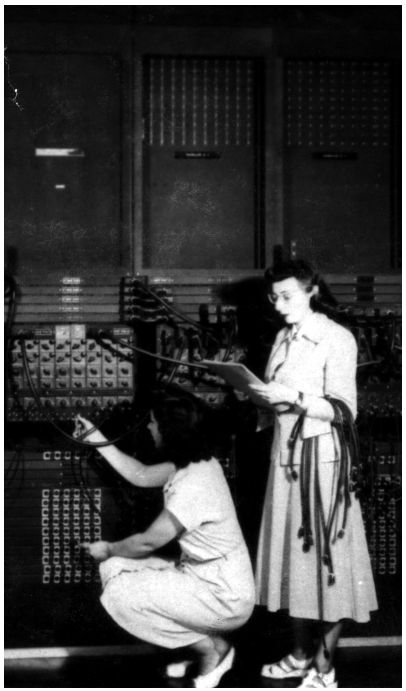
First "computers"

ENIAC, 1945.

Each Week:

- **5 Programming Assignments.**
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.

Homework



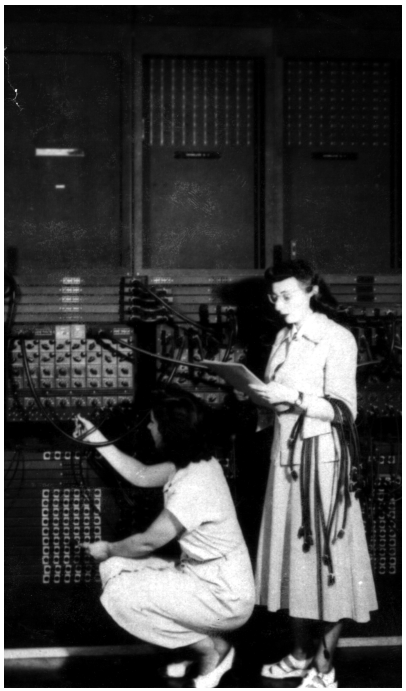
First "computers"

ENIAC, 1945.

Each Week:

- **5 Programming Assignments.**
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.
- Multiple submissions accepted.

Homework



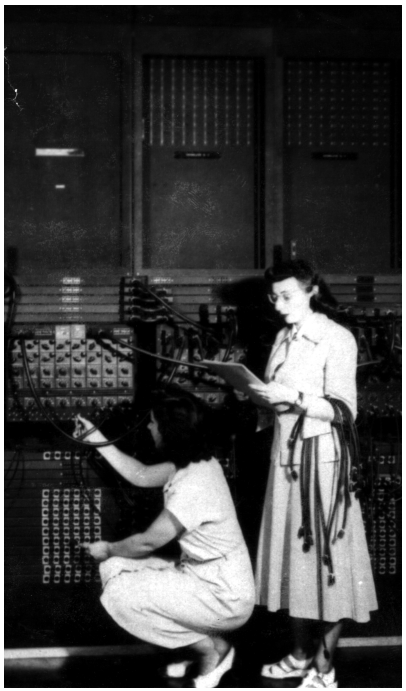
First "computers"

ENIAC, 1945.

Each Week:

- **5 Programming Assignments.**
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.
- Multiple submissions accepted.
- Watch Orientation Video: How to Write and Submit a Python Program (link on Blackboard)

Make Your Schedule!

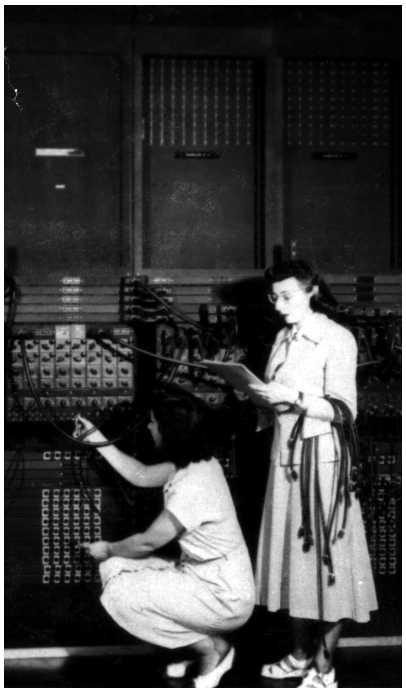


First "computers"

ENIAC, 1945.

- Schedule a regular time for the Online lab and quiz.

Make Your Schedule!

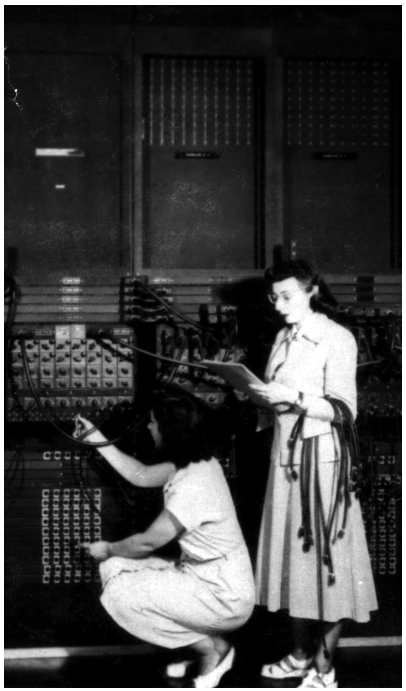


First "computers"

ENIAC, 1945.

- Schedule a regular time for the Online lab and quiz.
- Schedule a regular time for working on programming assignments.

Make Your Schedule!

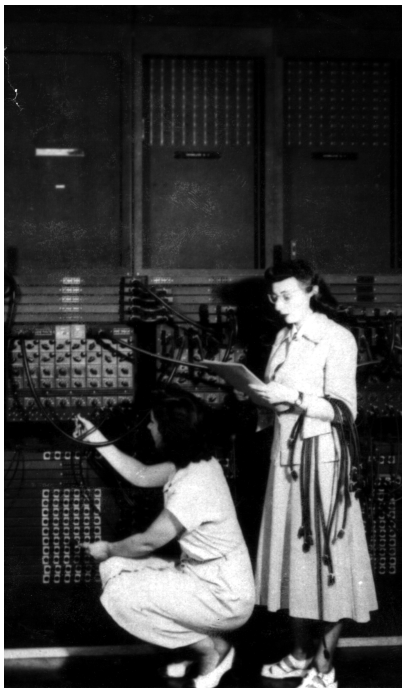


First "computers"

ENIAC, 1945.

- Schedule a regular time for the Online lab and quiz.
- Schedule a regular time for working on programming assignments.
- Schedule a regular time for taking the Lecture Preview

Make Your Schedule!

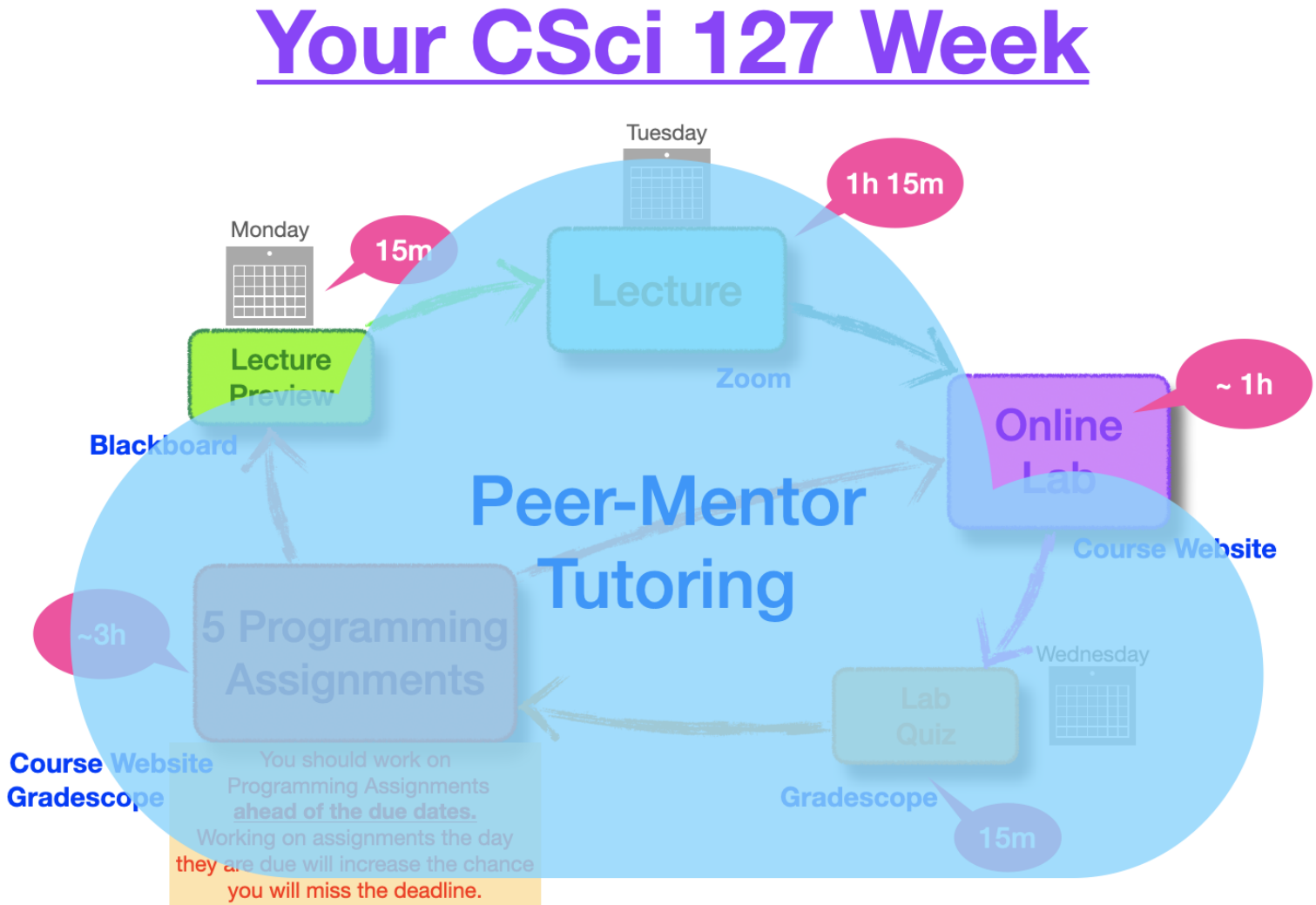


First "computers"

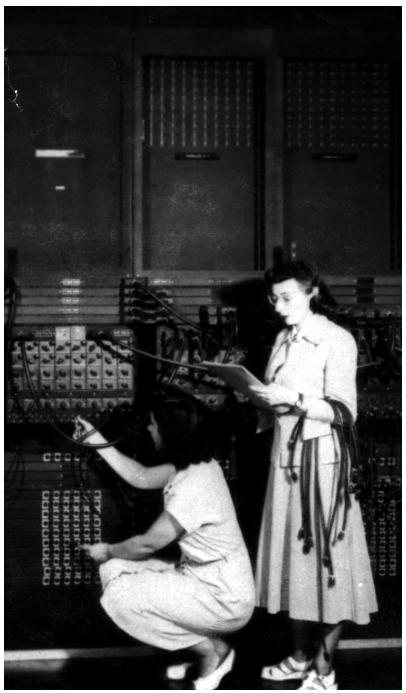
ENIAC, 1945.

- Schedule a regular time for the Online lab and quiz.
- Schedule a regular time for working on programming assignments.
- Schedule a regular time for taking the Lecture Preview
- Put them in your calendar now and then adjust if necessary.

Course Structure



Help and Support

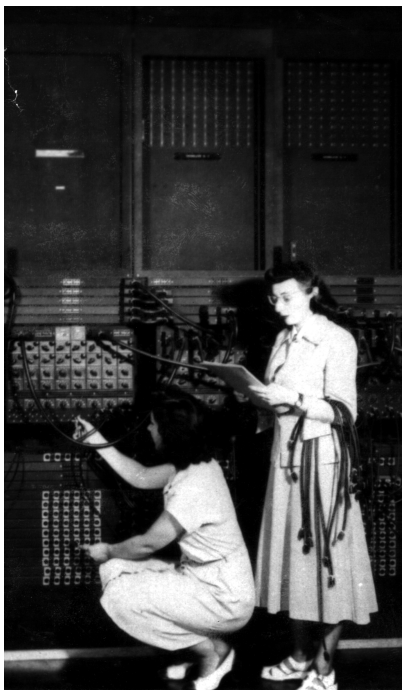


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Drop-in Tutoring**: UTA-lead group work to solve programming assignments

Help and Support

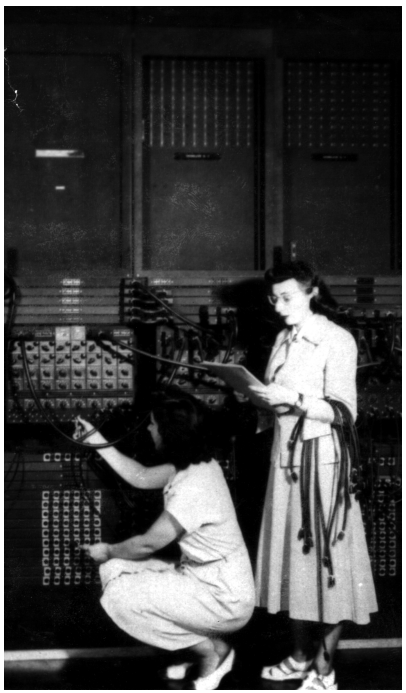


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Drop-in Tutoring**: UTA-lead group work to solve programming assignments
  - Link on Blackboard / Synchronous Meetings

Help and Support

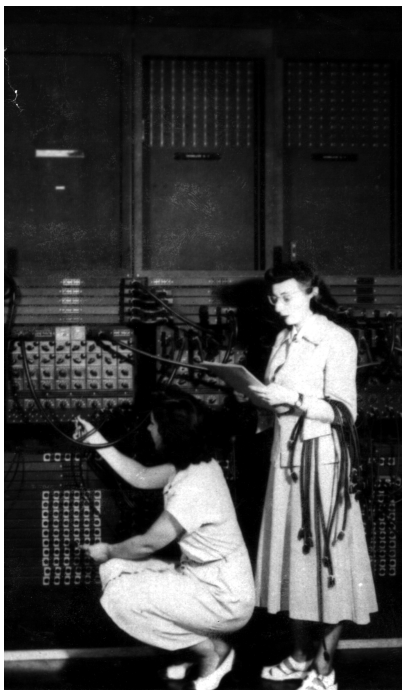


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Drop-in Tutoring**: UTA-lead group work to solve programming assignments
  - Link on Blackboard / Synchronous Meetings
  - **Discussion Board** on Blackboard

Help and Support

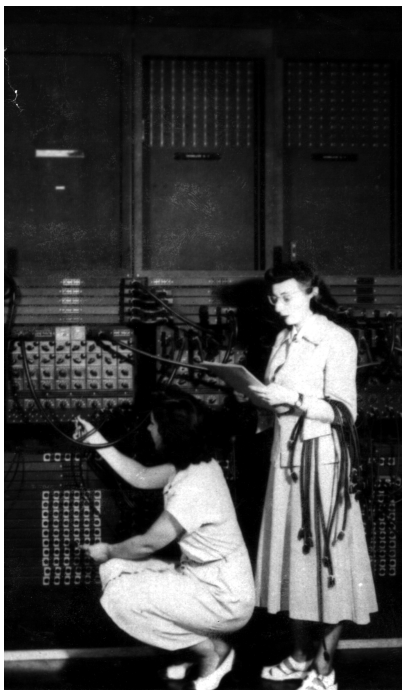


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Drop-in Tutoring**: UTA-lead group work to solve programming assignments
  - Link on Blackboard / Synchronous Meetings
  - **Discussion Board** on Blackboard
  - **Email: huntercsci127help@gmail.com**

Help and Support

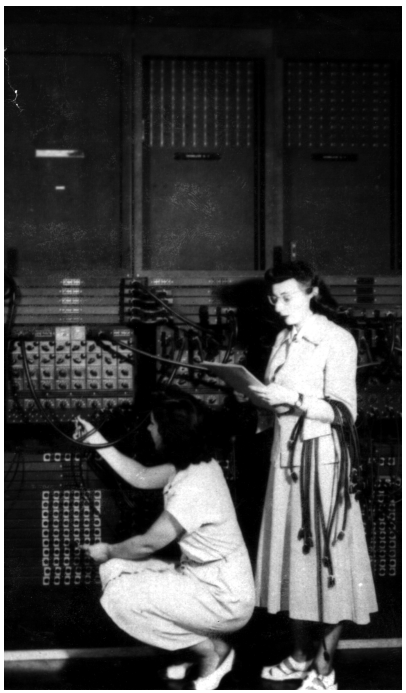


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Drop-in Tutoring**: UTA-lead group work to solve programming assignments
  - Link on Blackboard / Synchronous Meetings
  - **Discussion Board** on Blackboard
  - **Email: huntercsci127help@gmail.com**
  - All help available **Mo-Fr 11am-5pm** when classes are in session

Help and Support



First "computers"

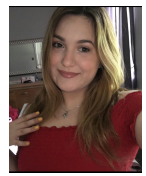
ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Drop-in Tutoring**: UTA-lead group work to solve programming assignments
  - Link on Blackboard / Synchronous Meetings
  - **Discussion Board** on Blackboard
  - **Email: huntercsci127help@gmail.com**
  - All help available **Mo-Fr 11am-5pm** when classes are in session
- Office Hours
  - Drop-in Hours: **Tuesday 11am-1pm**
  - Zoom link on Blackboard / Synchronous Meetings

Undergraduate Teaching Assistants

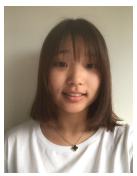

Aida Jevric


Arterio Rodrigues


Caiitlin Selca


Illya Baburashvili


Leonardo Matone


Liulan Zheng


Lola Samigjonova

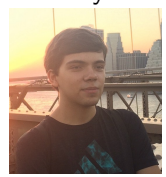

Mandy Yu


Nancy Ng


Nga Yu Lo


Owen Kunhardt

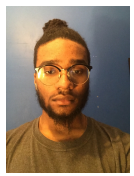

Patrick Chaca


Ryan Chevarria


Sadab Hafiz


Shantel Dixon


Stephanie Yung


Tyler Robinson

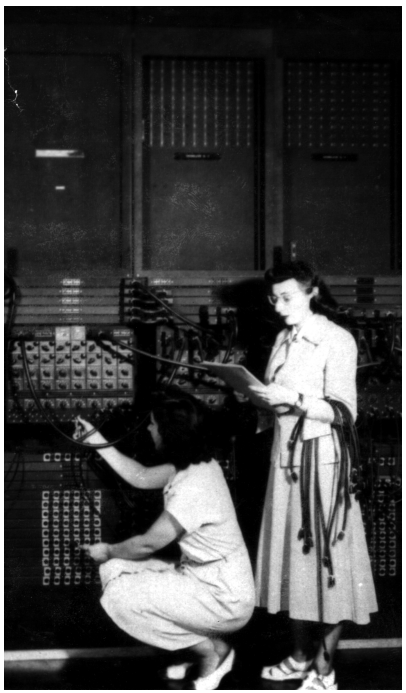

Yash Mahtani

Meet & Greet

Friday Feb 5, 5-6pm

Tutoring IS Important

Academic Dishonesty

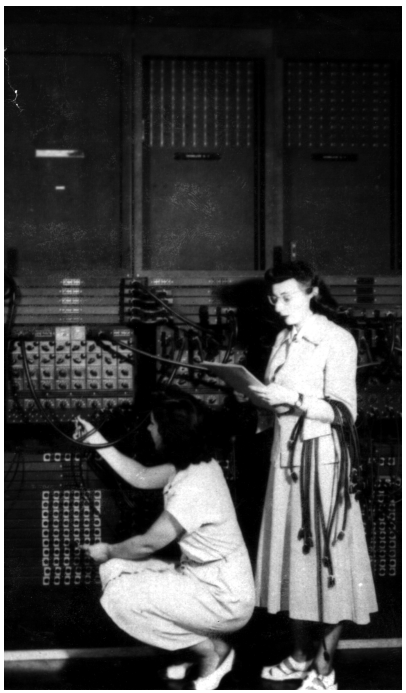


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*

Academic Dishonesty

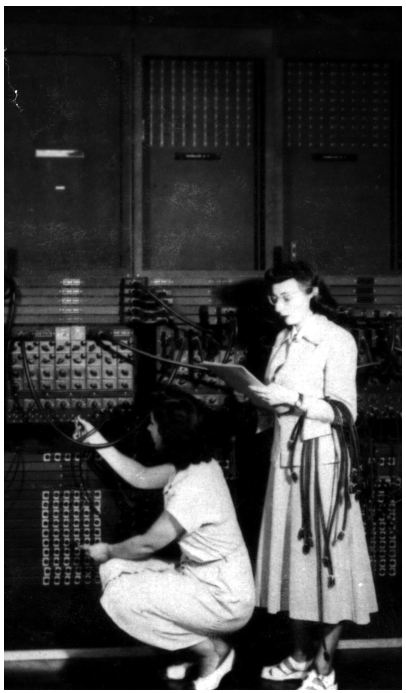


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*
- **Don't waste your time and money!**

Academic Dishonesty

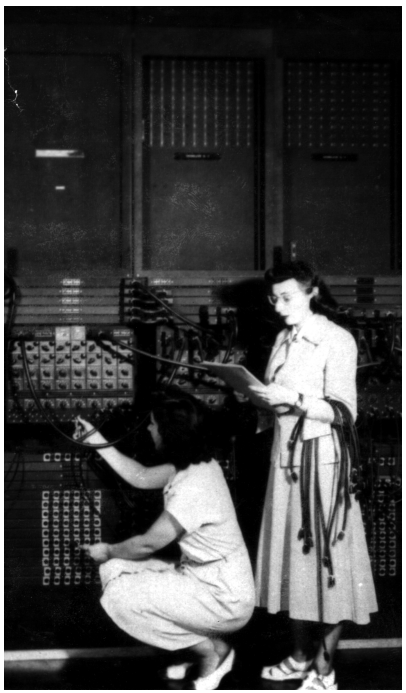


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*
- **Don't waste your time and money!**
- A few semesters down the road will be too late to catch up on core knowledge and **skills**.

Academic Dishonesty

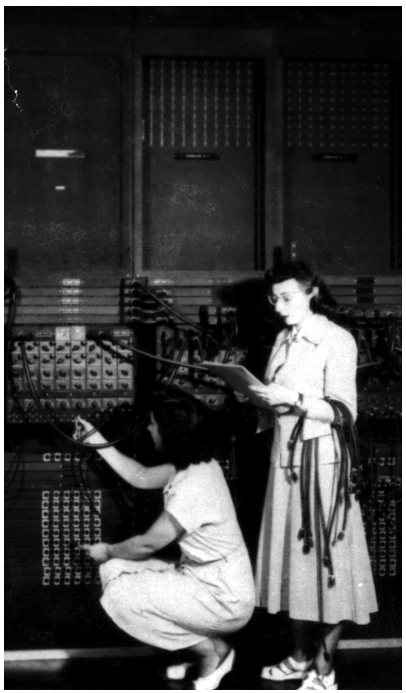


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*
- **Don't waste your time and money!**
- A few semesters down the road will be too late to catch up on core knowledge and **skills**.
- Cheating is immoral and it lowers the quality of our students and institution.

Academic Dishonesty

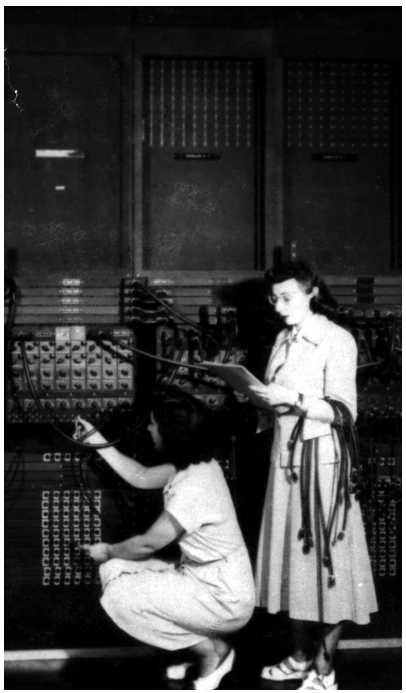


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*
- **Don't waste your time and money!**
- A few semesters down the road will be too late to catch up on core knowledge and **skills**.
- Cheating is immoral and it lowers the quality of our students and institution.
- Students that pose as experts often circulate bad/incorrect solutions

Academic Dishonesty

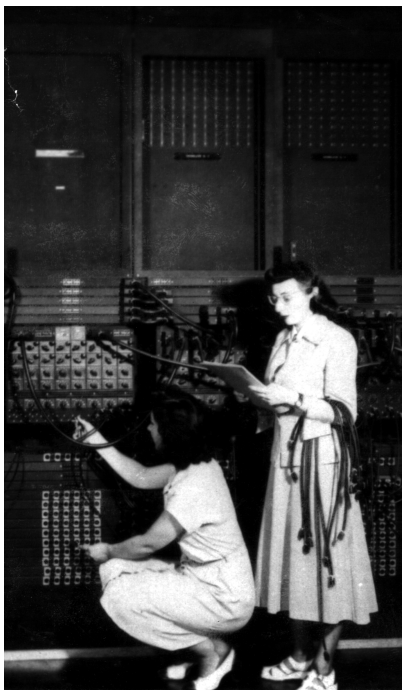


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*
- **Don't waste your time and money!**
- A few semesters down the road will be too late to catch up on core knowledge and **skills**.
- Cheating is immoral and it lowers the quality of our students and institution.
- Students that pose as experts often circulate bad/incorrect solutions
- Our UTAs are the true experts and equipped to help you learn and succeed!

Academic Dishonesty

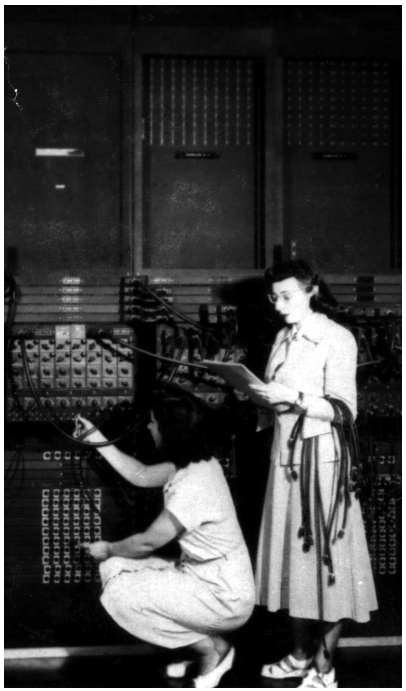


First "computers"

ENIAC, 1945.

- *The person who does the work gets the benefit! Learning is personal!!!*
- **Don't waste your time and money!**
- A few semesters down the road will be too late to catch up on core knowledge and **skills**.
- Cheating is immoral and it lowers the quality of our students and institution.
- Students that pose as experts often circulate bad/incorrect solutions
- Our UTAs are the true experts and equipped to help you learn and succeed!
- **All instances of academic dishonesty will be reported to the office of Student Affairs**

Communication

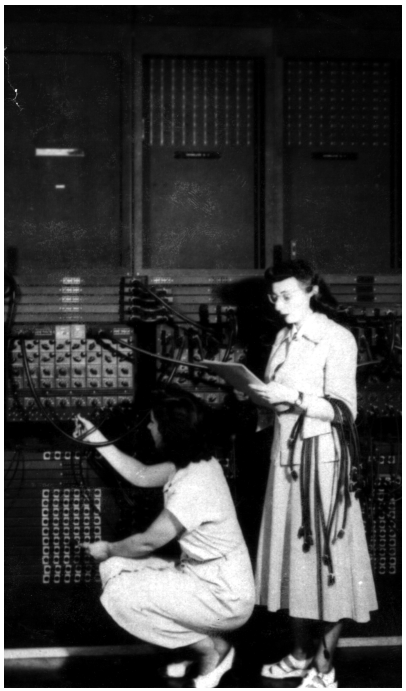


First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard

Communication

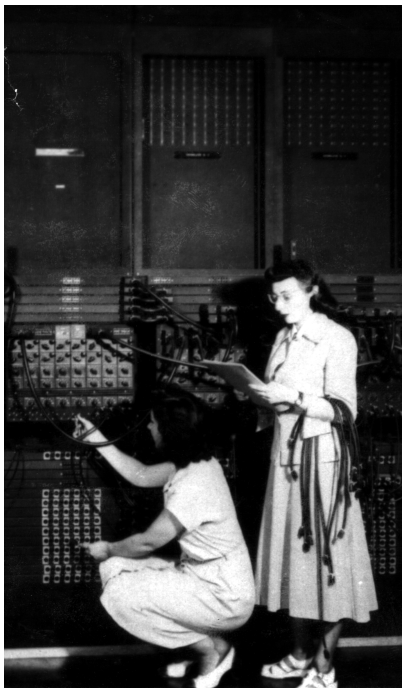


First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard
- Check your email account associated with Blackboard

Communication

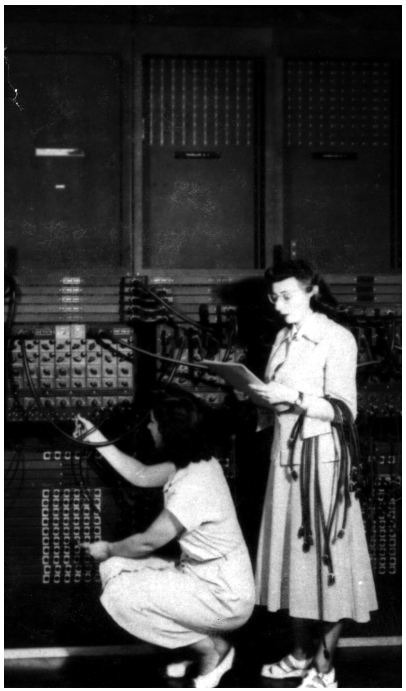


First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard
- Check your email account associated with Blackboard
- **Check your Spam folder**

Communication



First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard
- Check your email account associated with Blackboard
- **Check your Spam folder**
- Email studenthelpdesk@hunter.cuny.edu if you need to change it

How to Succeed in this Course

Each Week:

- Come to Lecture

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.
  - Pay attention during lecture.

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.
  - Pay attention during lecture.
  - Actively participate in lecture work: try to solve problems/challenges

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.
  - Pay attention during lecture.
  - Actively participate in lecture work: try to solve problems/challenges
- Read the Online Lab or join Lab Review on Zoom.

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.
  - Pay attention during lecture.
  - Actively participate in lecture work: try to solve problems/challenges
- Read the Online Lab or join Lab Review on Zoom.
- Take the weekly Lab Quiz.

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.
  - Pay attention during lecture.
  - Actively participate in lecture work: try to solve problems/challenges
- Read the Online Lab or join Lab Review on Zoom.
- Take the weekly Lab Quiz.
- Work on THIS WEEK'S Programming Assignments.

How to Succeed in this Course

Each Week:

- Come to Lecture
  - Take the lecture preview before lecture.
  - Pay attention during lecture.
  - Actively participate in lecture work: try to solve problems/challenges
- Read the Online Lab or join Lab Review on Zoom.
- Take the weekly Lab Quiz.
- Work on THIS WEEK'S Programming Assignments.
- Ask for help from our UTAs in Drop-in Tutoring or Discussion Board.

Philosophy (Or Why We Do What We Do)

Please see our Q&A on Blackboard

- Topics

Philosophy (Or Why We Do What We Do)

Please see our Q&A on Blackboard

- Topics
  - Grading

Philosophy (Or Why We Do What We Do)

Please see our Q&A on Blackboard

- Topics
  - Grading
  - Course Structure

Philosophy (Or Why We Do What We Do)

Please see our Q&A on Blackboard

- Topics
  - Grading
  - Course Structure
  - Help

Philosophy (Or Why We Do What We Do)

Please see our Q&A on Blackboard

- Topics
  - Grading
  - Course Structure
  - Help
- We will keep adding to it throughout the semester, look out for new content

Today's Topics

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

Today's Topics

- **Introduction to Python**
- Turtle Graphics
- Definite Loops (`for`-loops)
- 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, supportive community with hundreds of open source libraries and frameworks.

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, supportive community with hundreds of open source libraries and frameworks.
- 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, supportive community with hundreds of open source libraries and frameworks.
- 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).

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)

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!`
- We know that `Hello, World!` is a **string** (a sequence of characters) because it is surrounded by quotes

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!`
- We know that `Hello, World!` is a **string** (a sequence of characters) because it is surrounded by quotes
- 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,')
```

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,')
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.

Today's Topics

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

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)

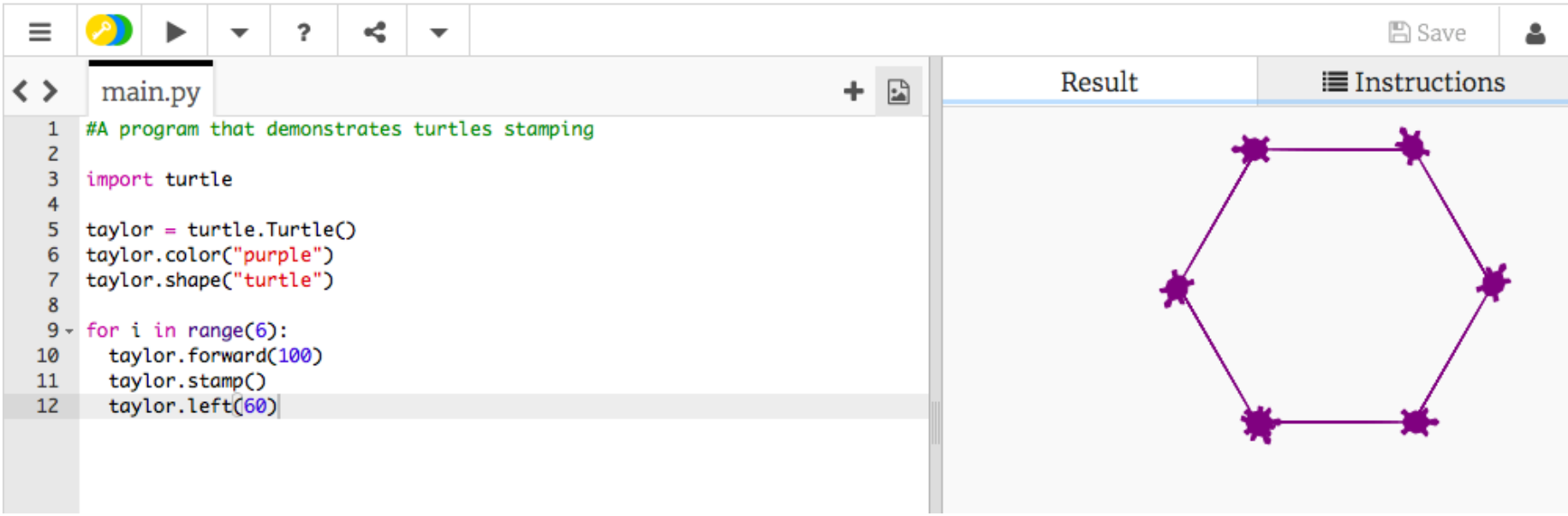
Turtles Introduction

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

Today's Topics

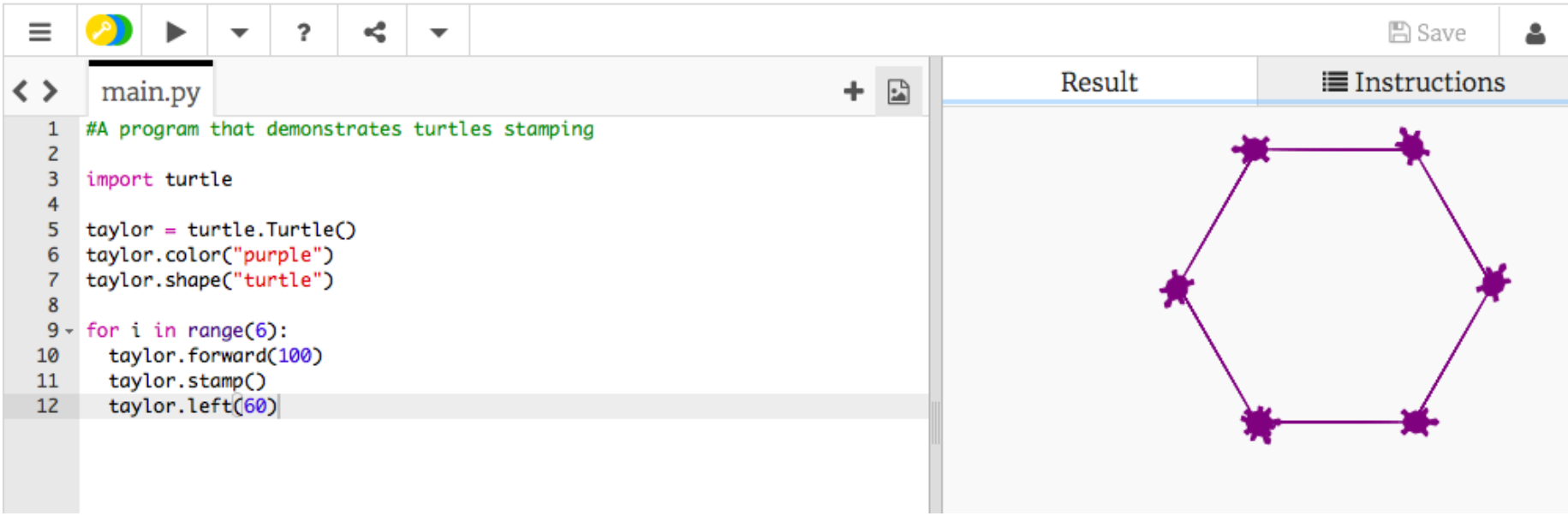
- Introduction to Python
- Turtle Graphics
- **Definite Loops (`for`-loops)**
- Algorithms

Turtles Introduction



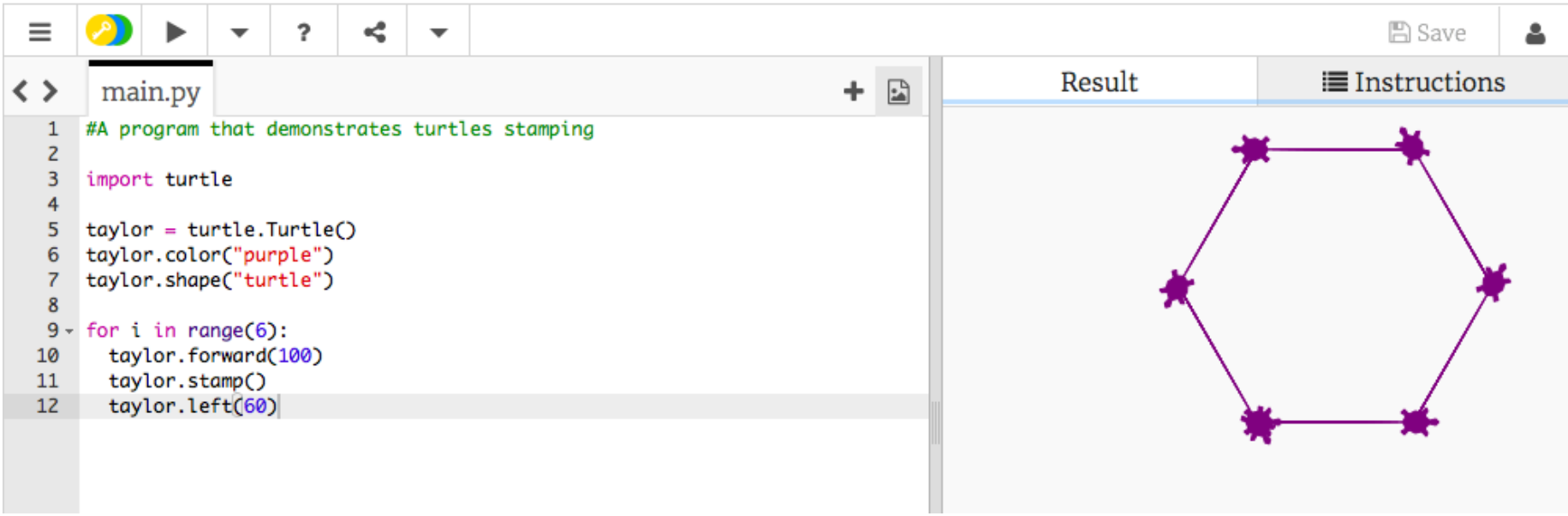
- Creates a turtle **variable**, called `taylor`.

Turtles Introduction



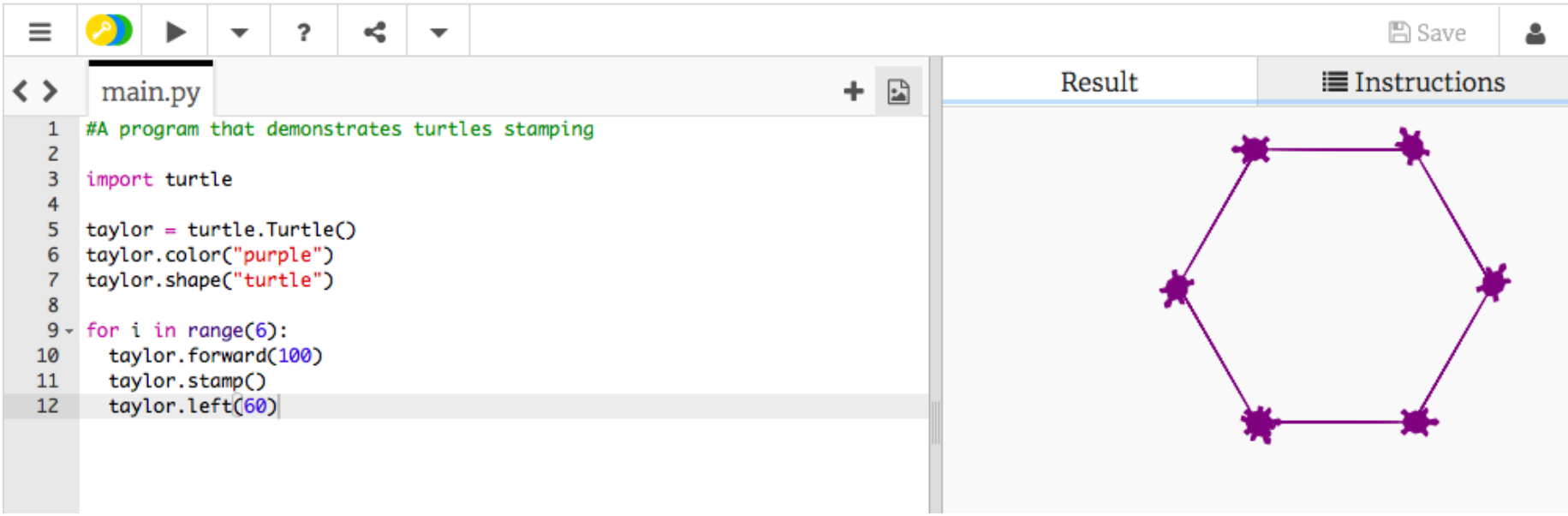
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).

Turtles Introduction



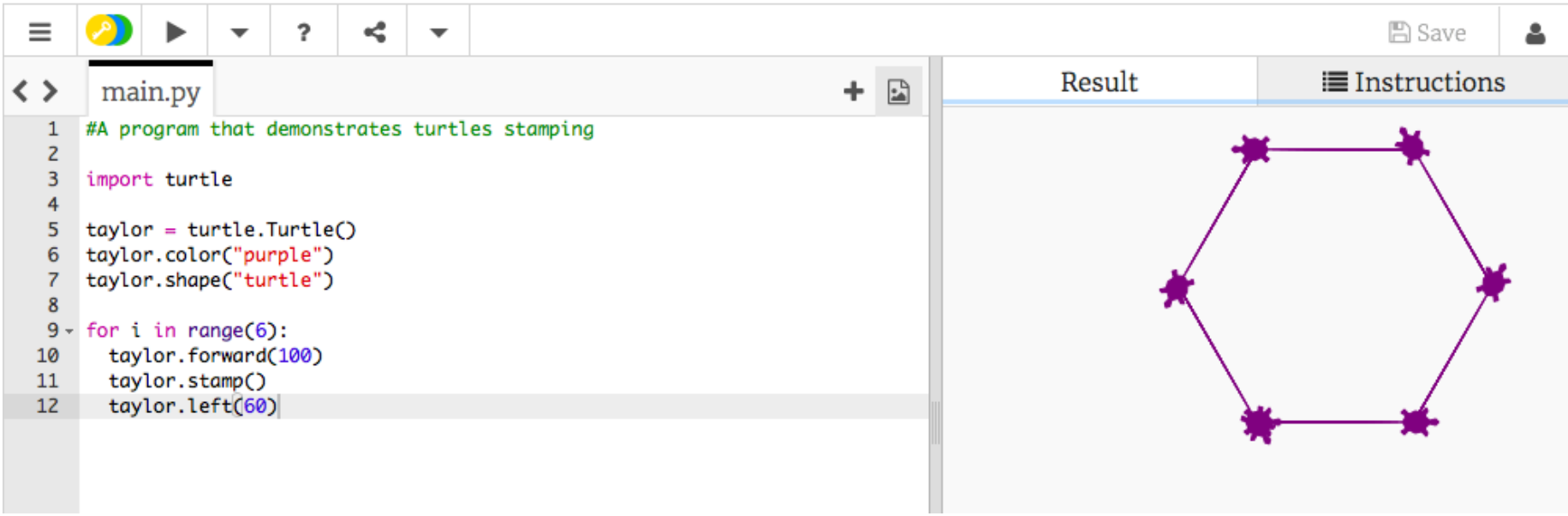
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:

Turtles Introduction



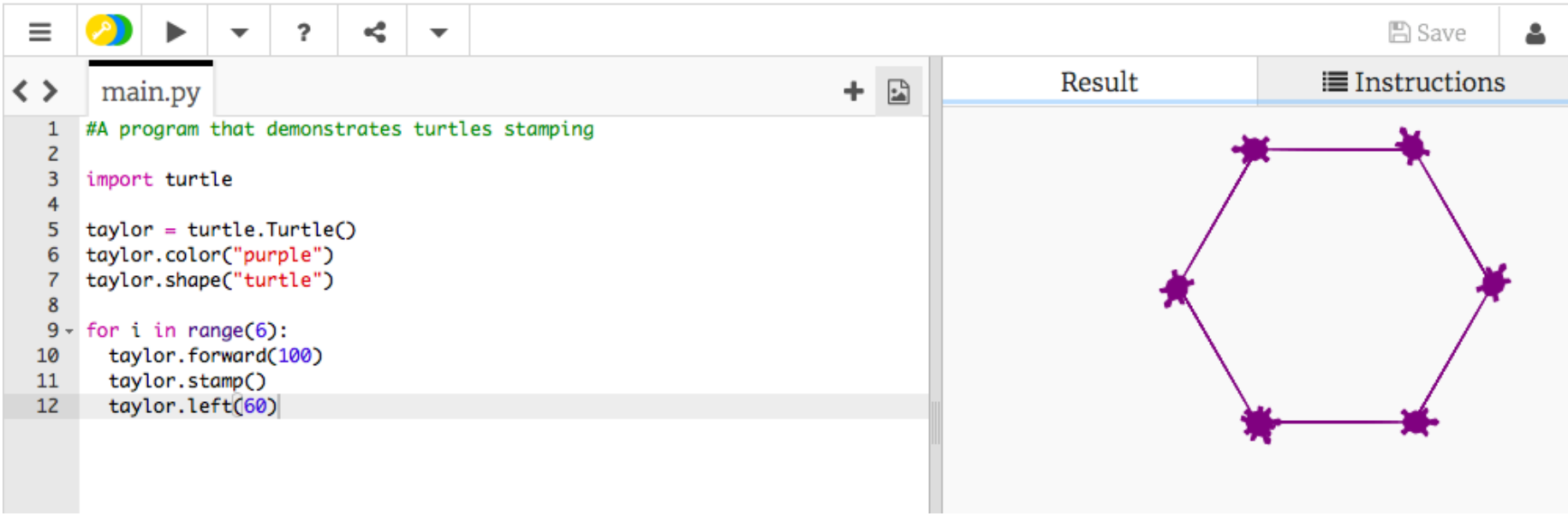
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
  - Move forward; stamp; and turn left 60 degrees.

Turtles Introduction



- Creates a turtle **variable**, 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"

Turtles Introduction



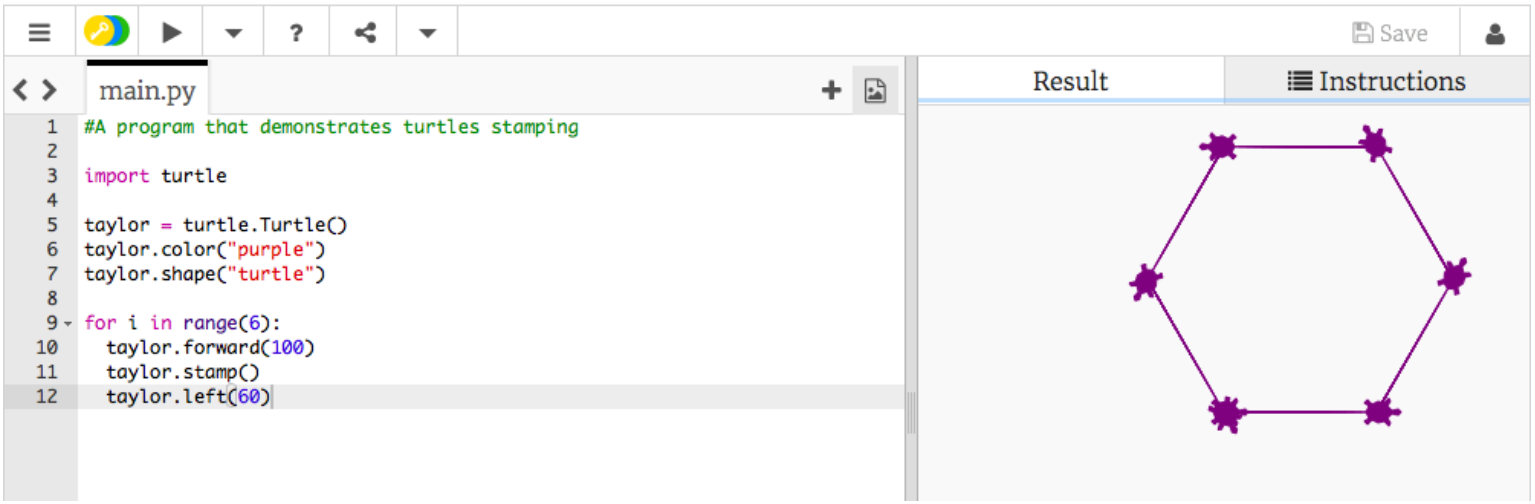
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
  - Move forward; stamp; and turn left 60 degrees.
- Repeats any instructions **indented** in the "loop block"
- This is a **definite** loop because it repeats a fixed number of times

Your Turn!!!

Try to solve this challenge:

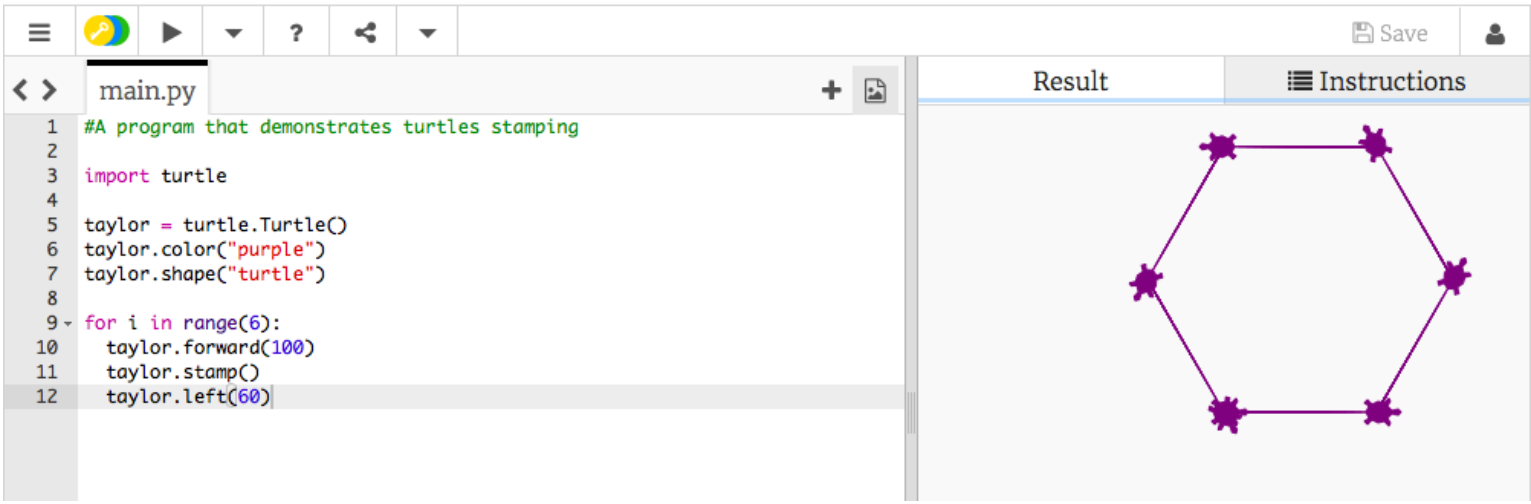
1. Write a program that will draw a 10-sided polygon.
2. Write a program that will repeat the line:
   `I'm lookin' for a mind at work!`
   three times.

Decagon Program



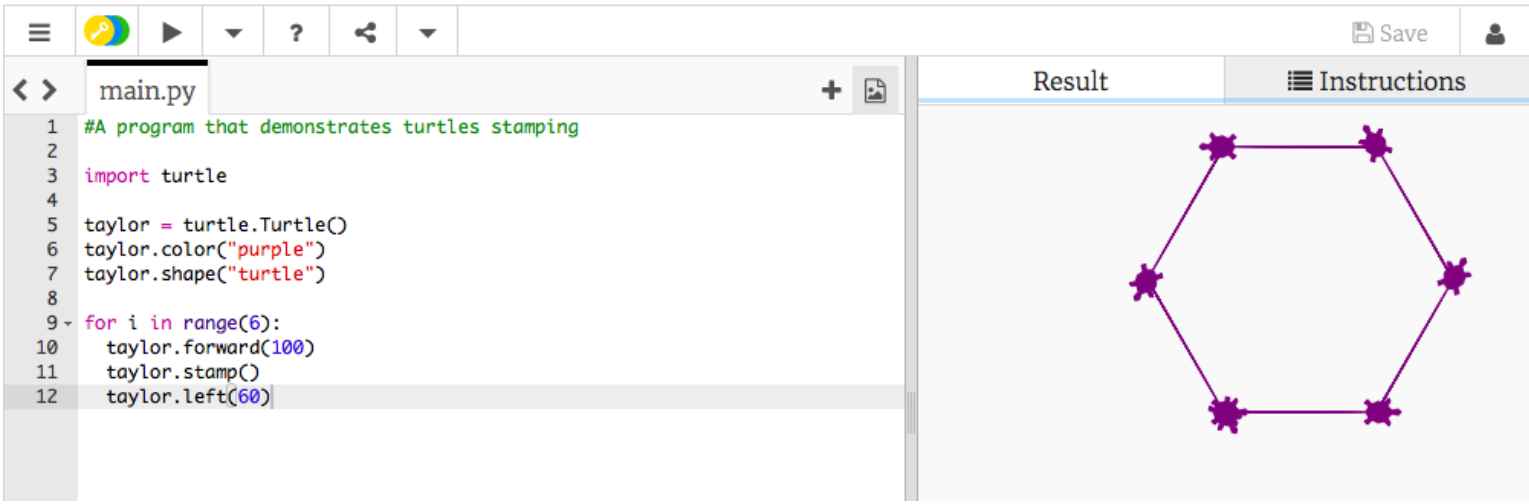
- Start with the hexagon program.

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!")
```

Lecture Quiz

Log-in to Gradescope

- Find Lecture 1 Quiz

Lecture Quiz

Log-in to Gradescope

- Find Lecture 1 Quiz
- Take the quiz

Lecture Quiz

Log-in to Gradescope

- Find Lecture 1 Quiz
- Take the quiz
- You have 3 minutes

Today's Topics

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

What is an Algorithm?

From our textbook:

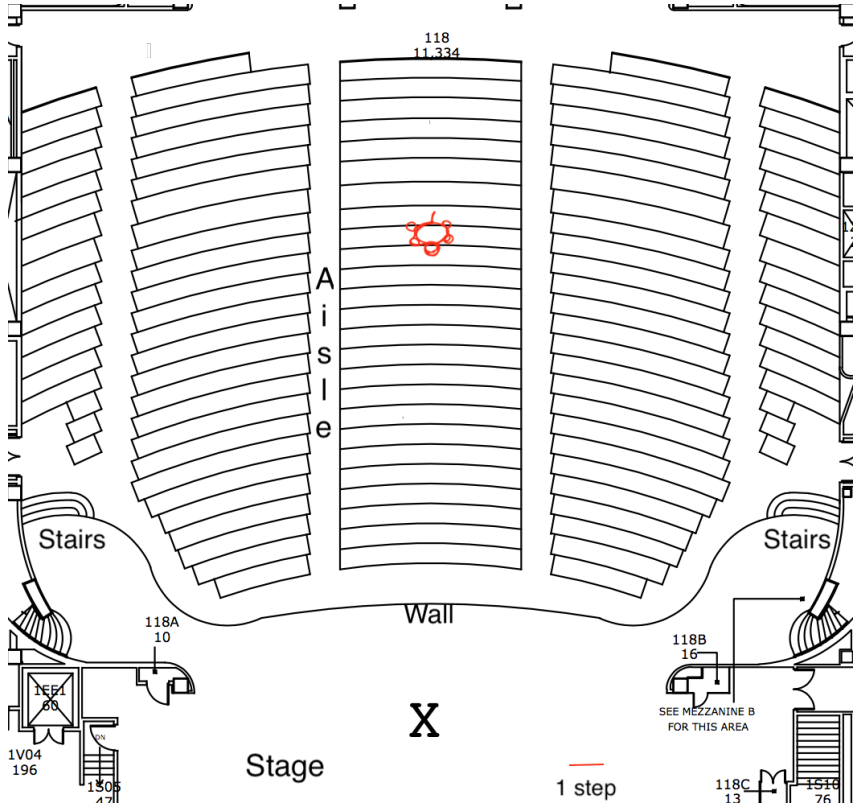
- An algorithm is a process or sequence of steps to be followed to solve a problem.

What is an Algorithm?

From our textbook:

- An algorithm is a process or sequence of steps 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 executed by a computer.

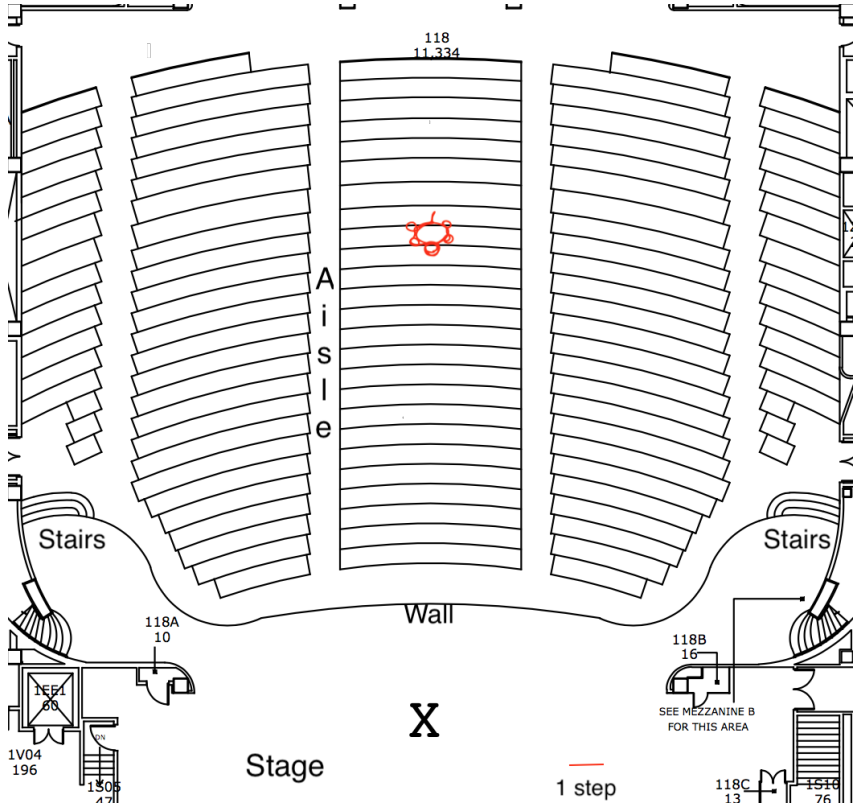
Your Turn!!!



Try to solve this challenge:

1. This is the floor plan of Assembly Hall at Hunter College.

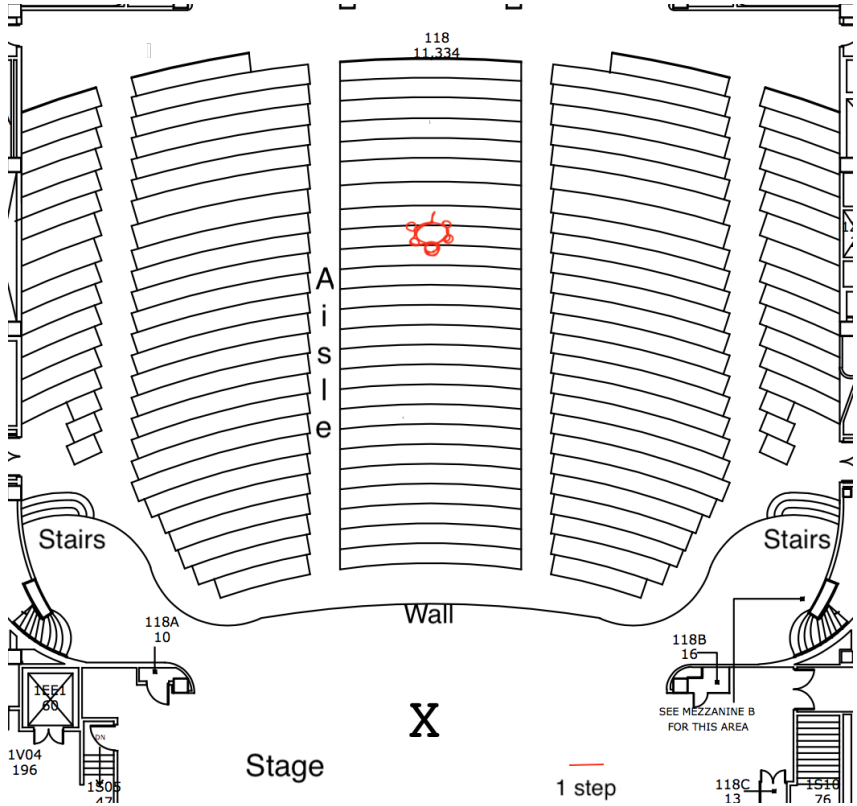
Your Turn!!!



Try to solve this challenge:

1. This is the floor plan of Assembly Hall at Hunter College.
2. Write an algorithm (step-by-step directions) to the red turtle to the `X` on Stage.

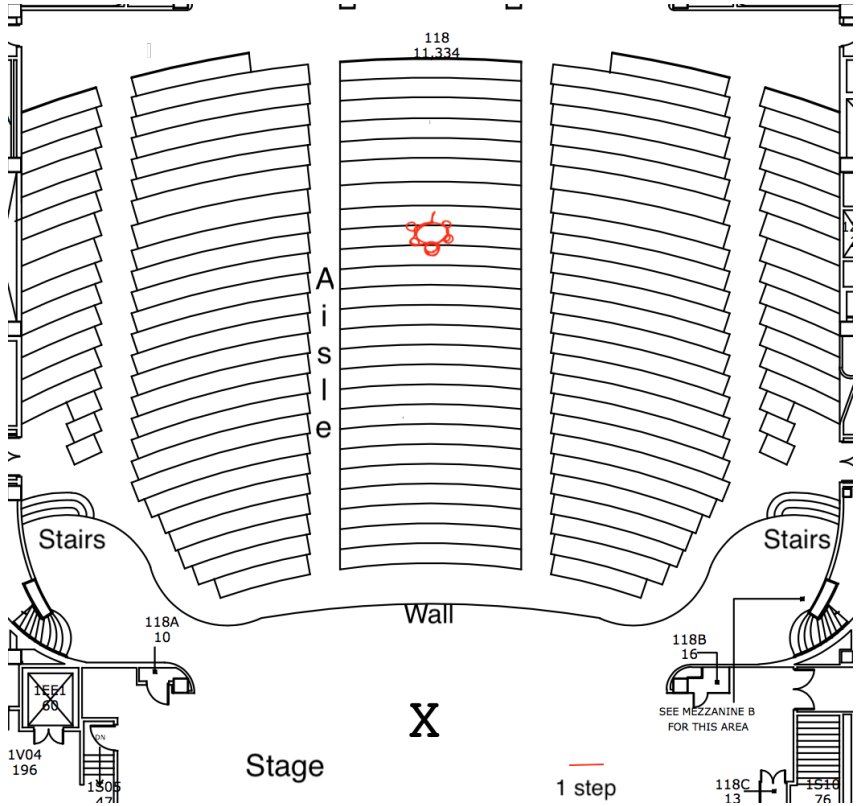
Your Turn!!!



Try to solve this challenge:

1. This is the floor plan of Assembly Hall at Hunter College.
2. Write an algorithm (step-by-step directions) to the red turtle to the `X` on Stage.
3. Basic Rules:

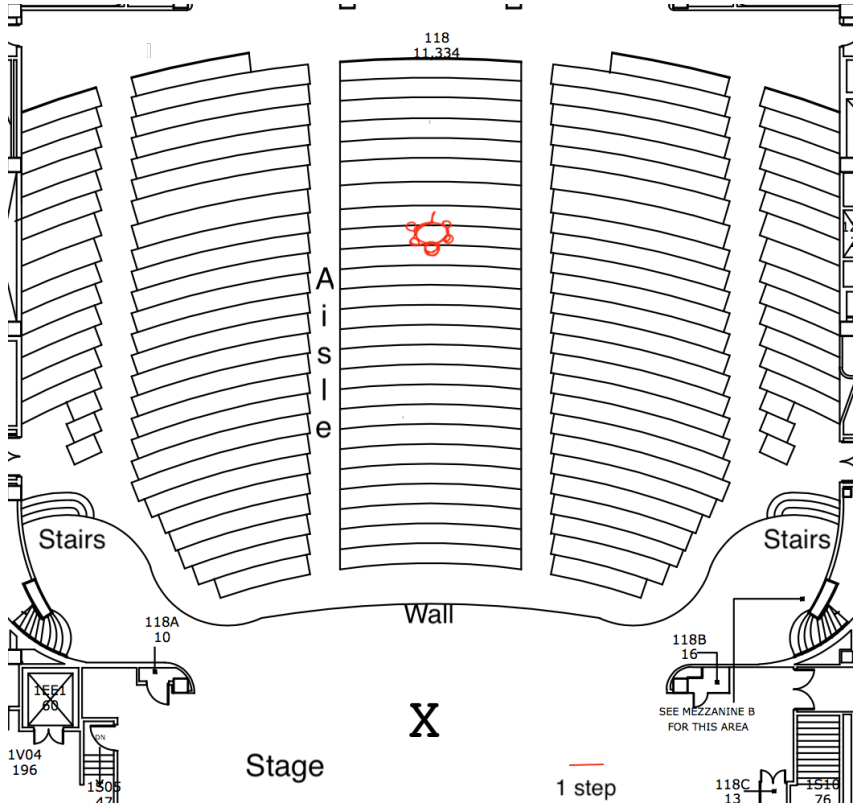
Your Turn!!!



Try to solve this challenge:

1. This is the floor plan of Assembly Hall at Hunter College.
2. Write an algorithm (step-by-step directions) to the red turtle to the `X` on Stage.
3. Basic Rules:
   - Use `turtle` commands.

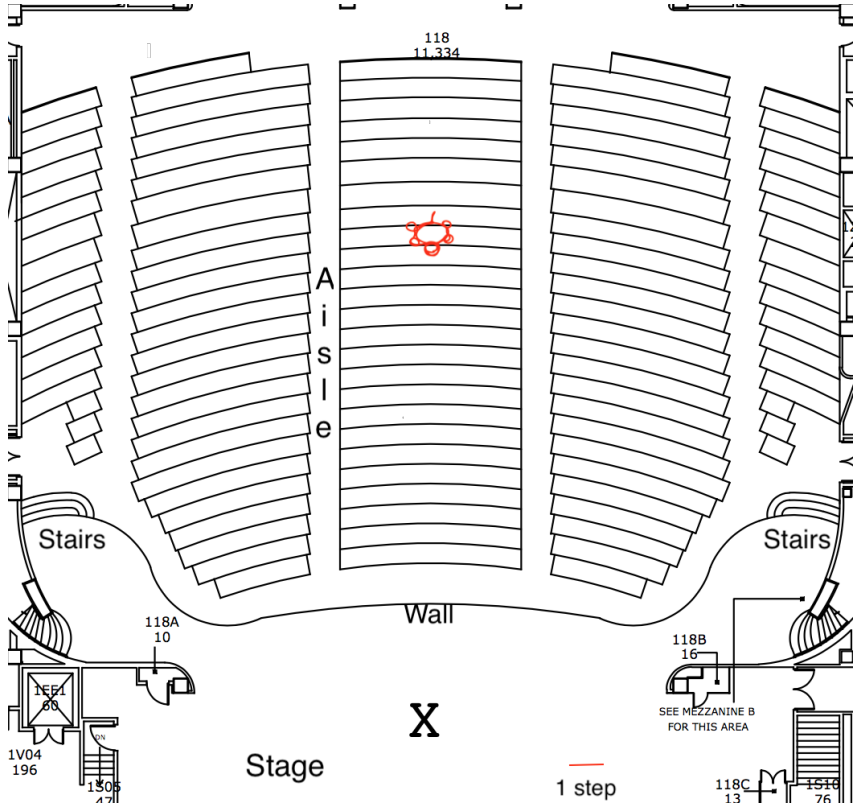
Your Turn!!!



Try to solve this challenge:

1. This is the floor plan of Assembly Hall at Hunter College.
2. Write an algorithm (step-by-step directions) to the red turtle to the `X` on Stage.
3. Basic Rules:
   - Use `turtle` commands.
   - Do not run turtles into walls, chairs, obstacles, etc.

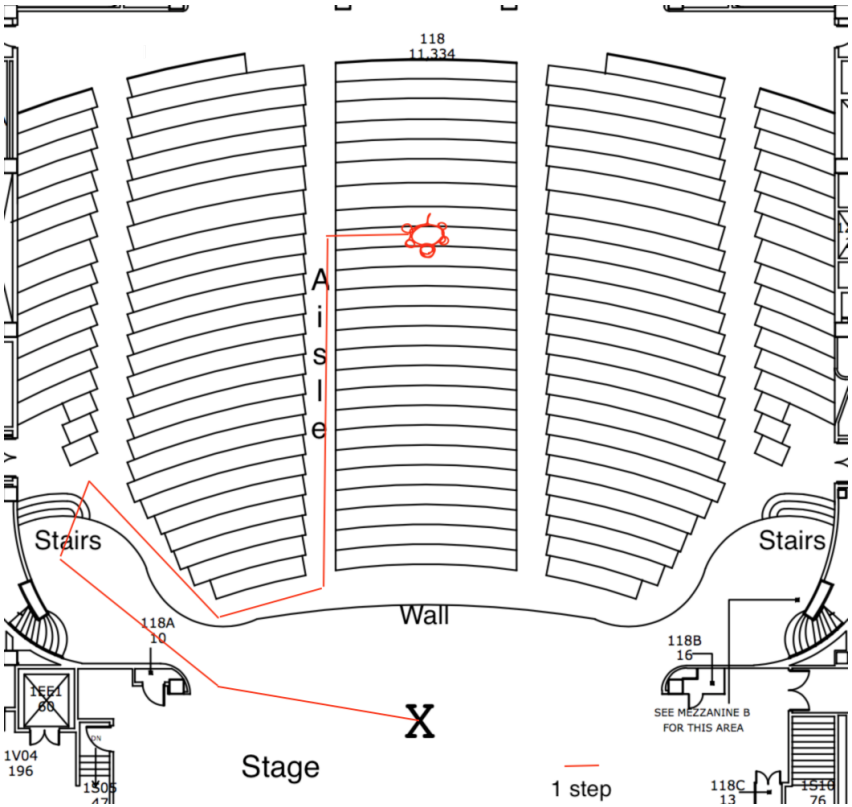
Your Turn!!!



Try to solve this challenge:

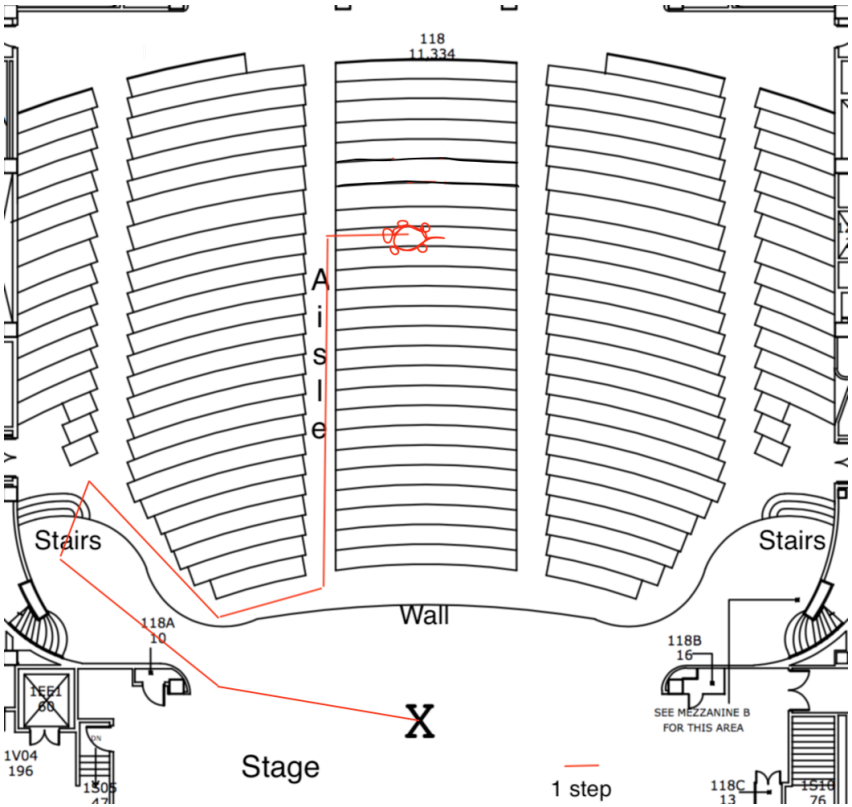
1. This is the floor plan of Assembly Hall at Hunter College.
2. Write an algorithm (step-by-step directions) to the red turtle to the `X` on Stage.
3. Basic Rules:
   - Use `turtle` commands.
   - Do not run turtles into walls, chairs, obstacles, etc.
   - Turtles cannot climb walls, must use stairs (walk forward on steps).

Your Turn!!!



One possible solution:

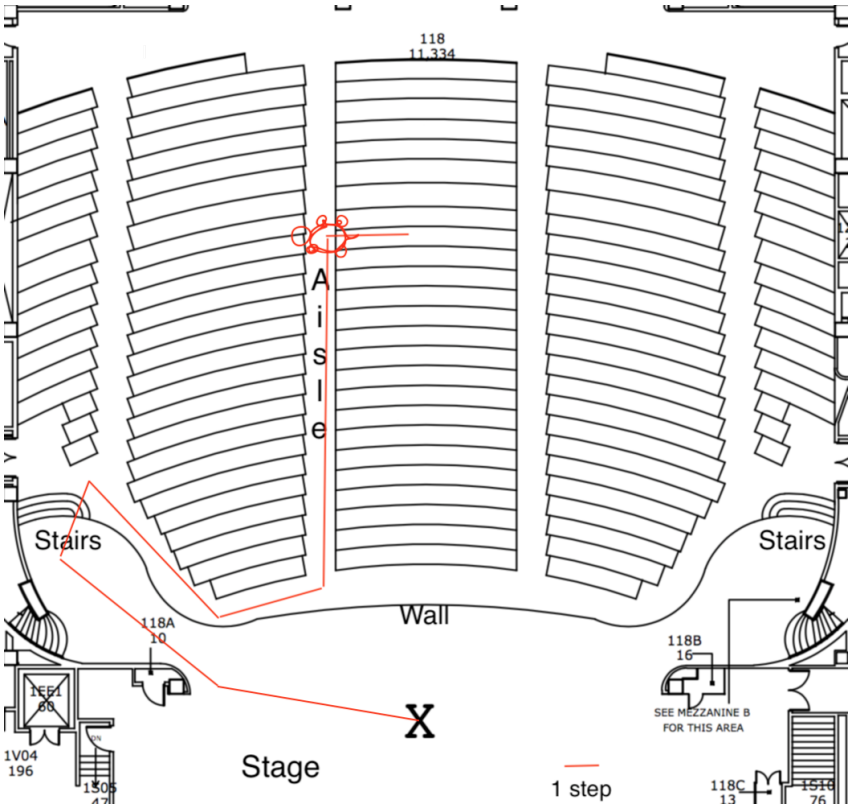
Your Turn!!!



One possible solution:

- Turn right 90 degrees.

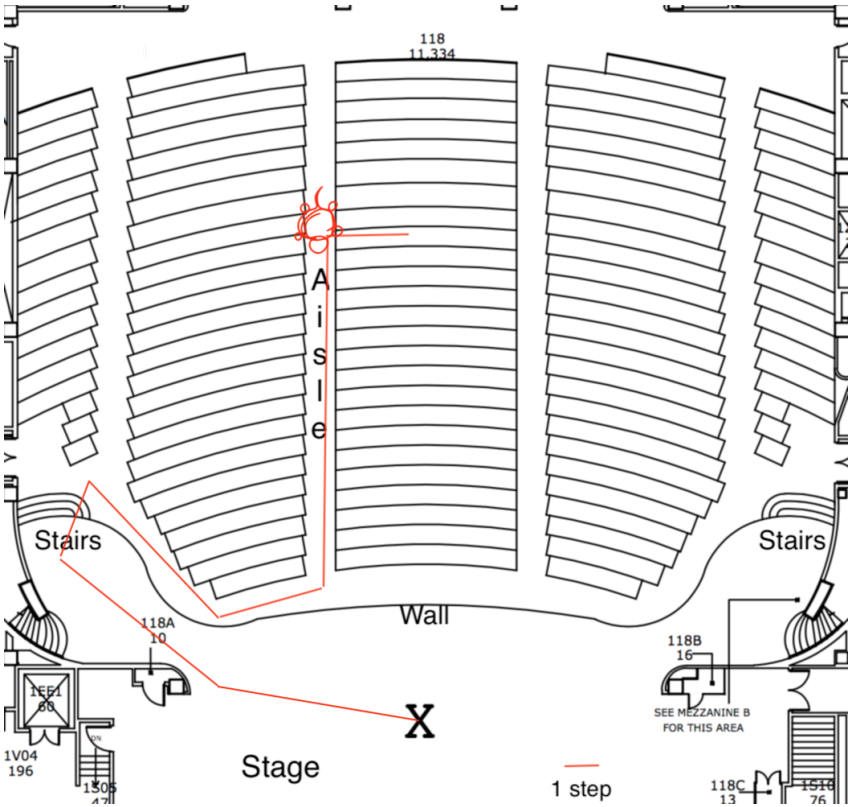
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.

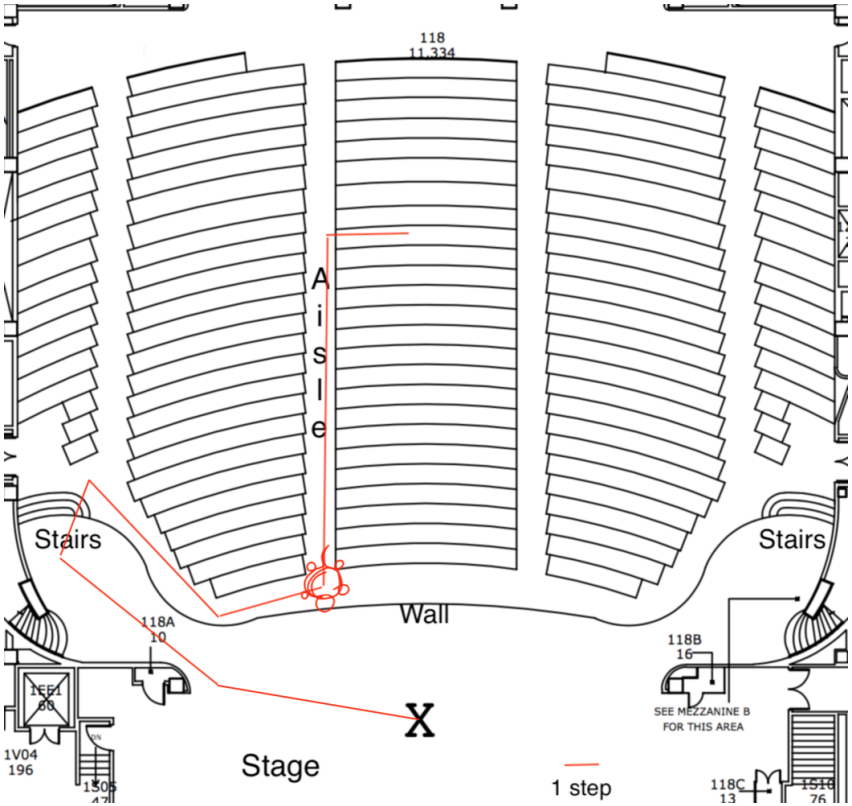
Your Turn!!!



- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.

One possible solution:

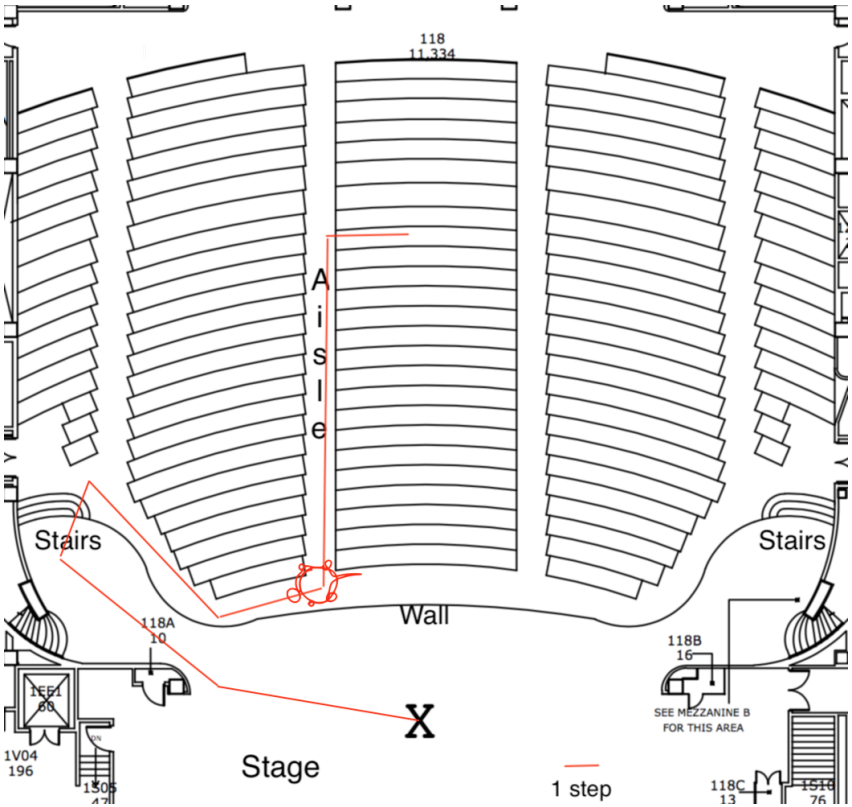
Your Turn!!!



- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.

One possible solution:

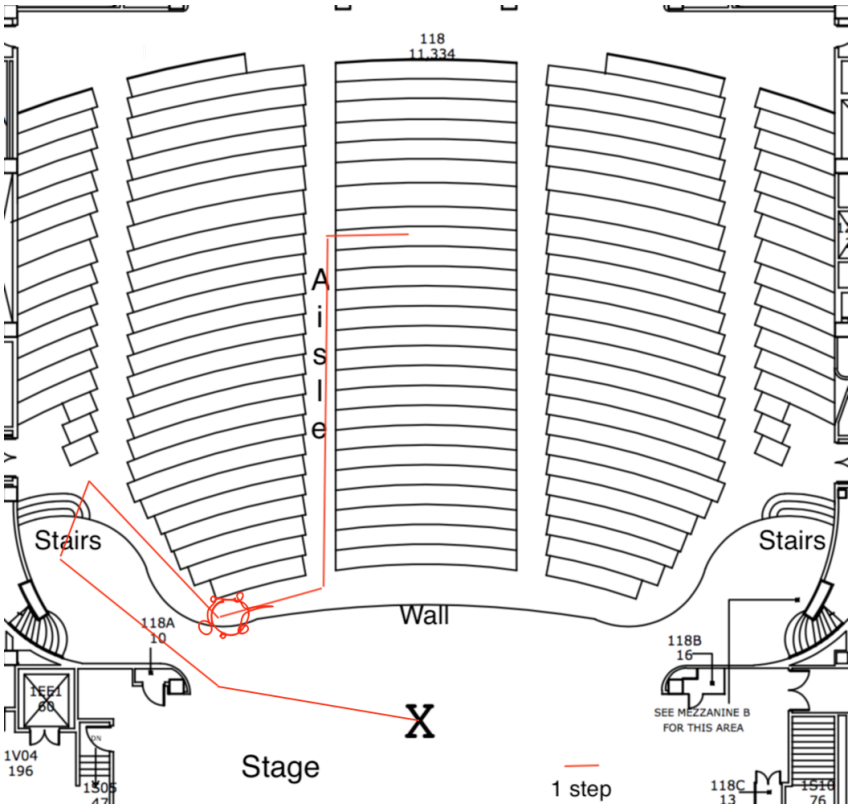
Your Turn!!!



- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees

One possible solution:

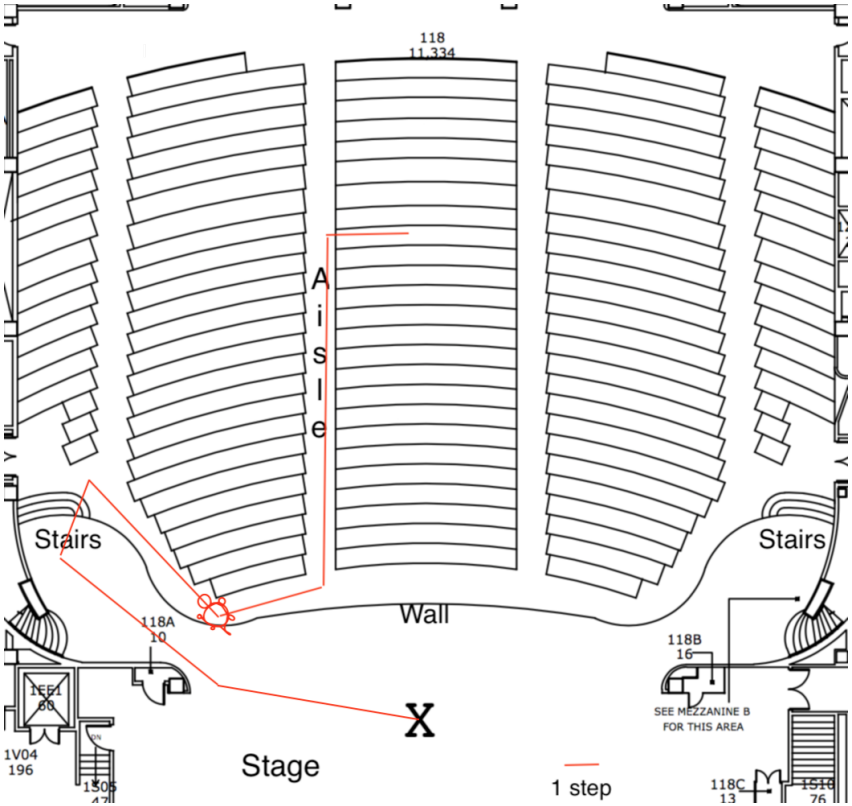
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.

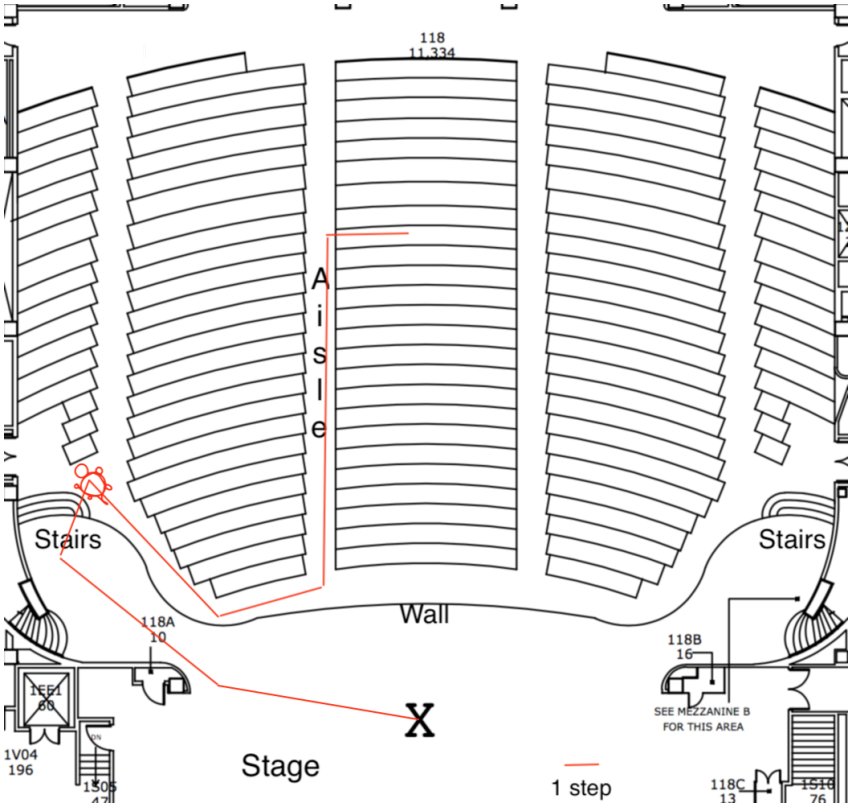
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.

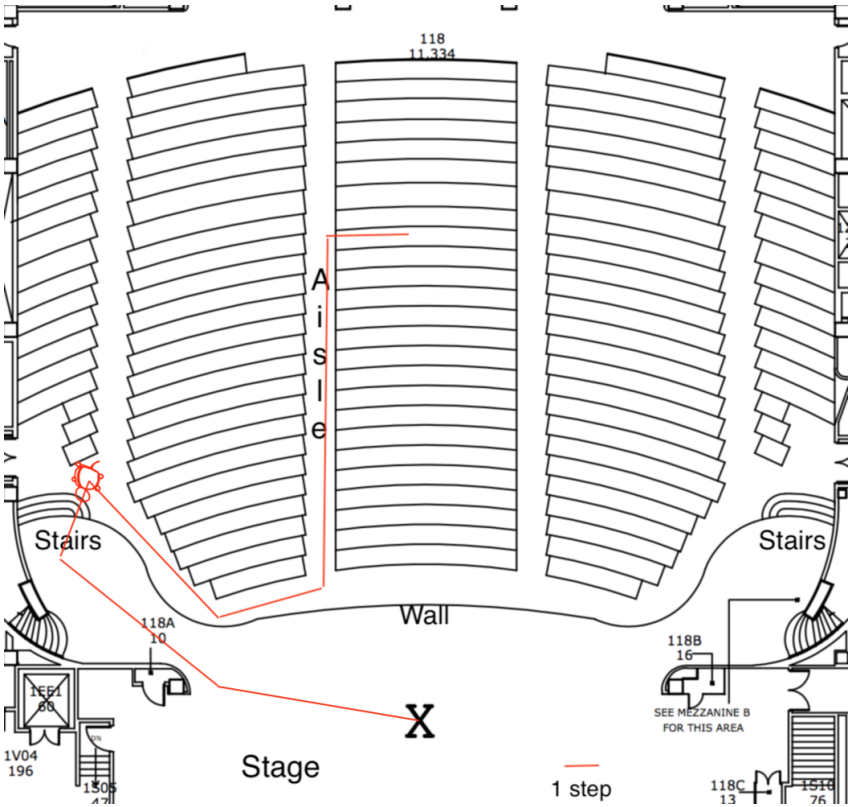
Your Turn!!!



- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.

One possible solution:

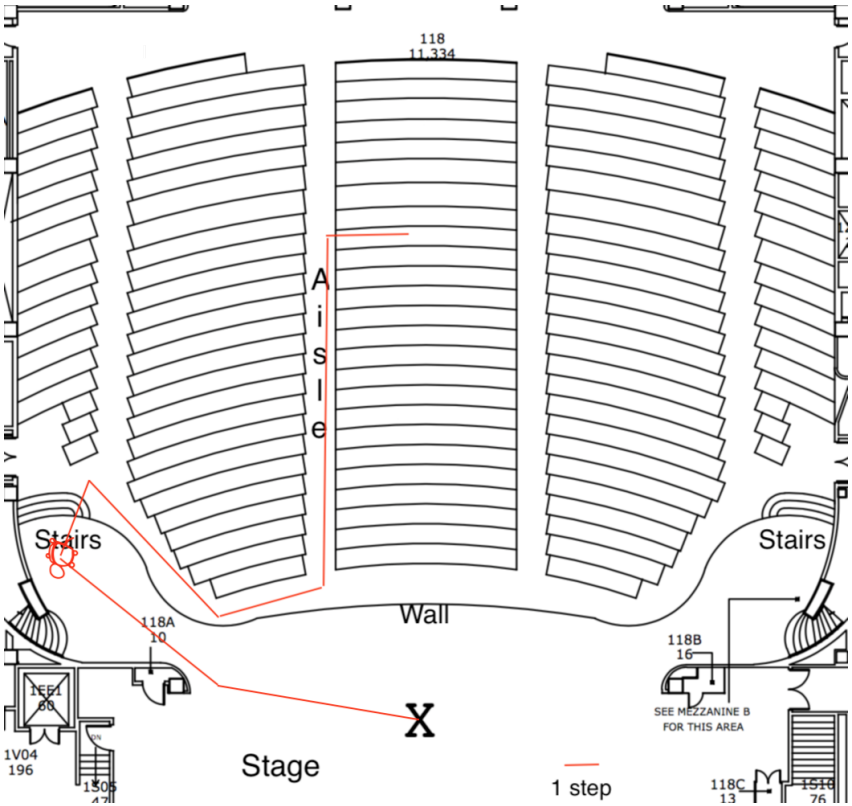
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.

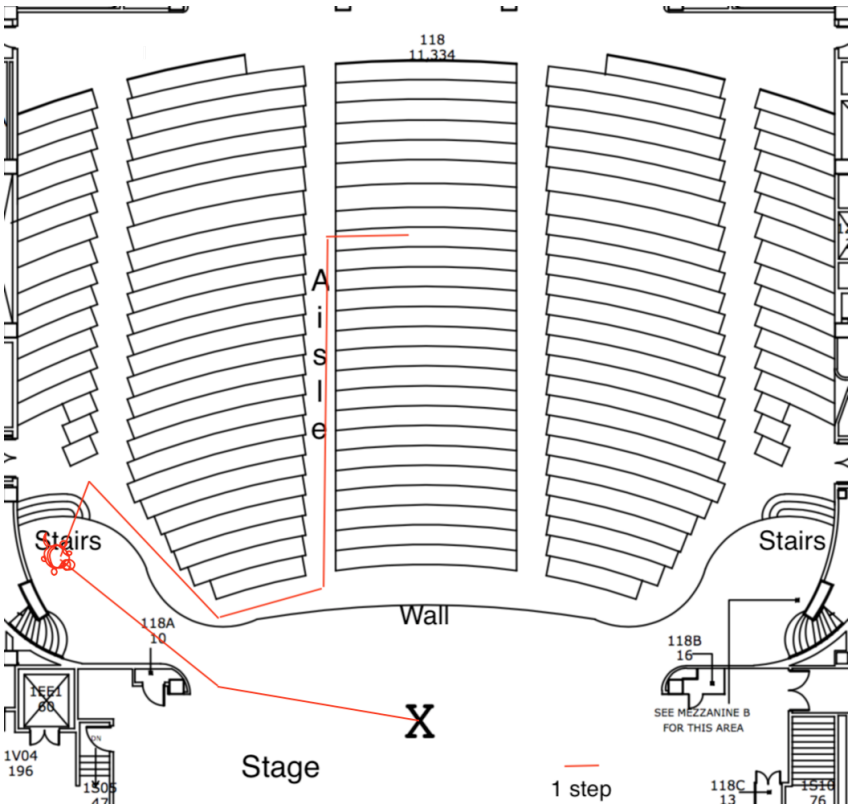
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.
- Walk forward 3 steps.

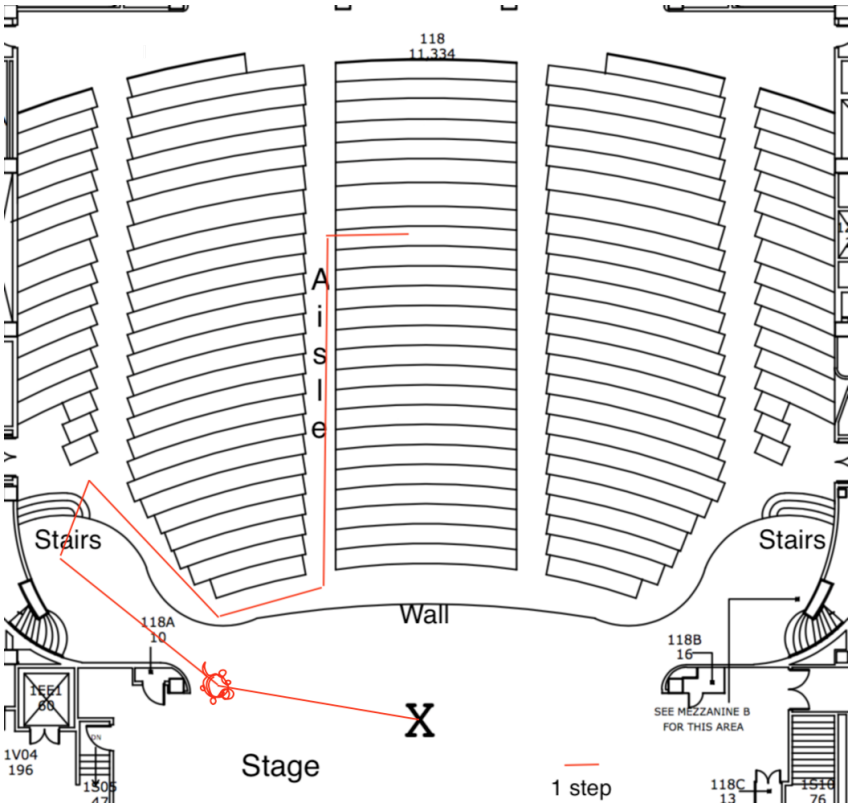
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.
- Walk forward 3 steps.
- Turn left 80 degrees.

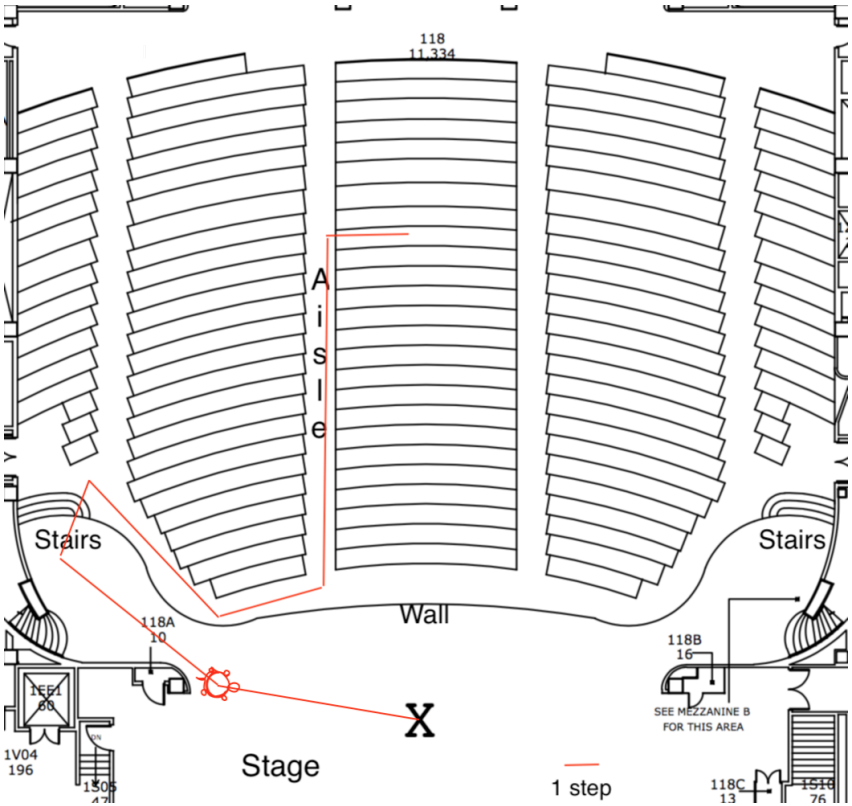
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.
- Walk forward 3 steps.
- Turn left 80 degrees.
- Walk forward 5 steps.

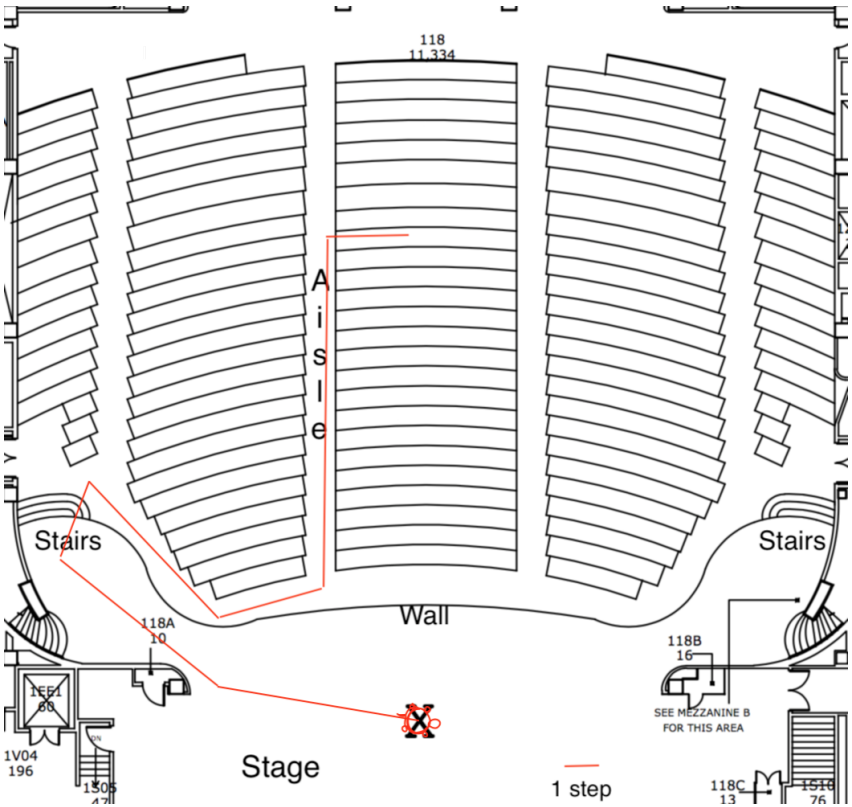
Your Turn!!!



One possible solution:

- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.
- Walk forward 3 steps.
- Turn left 80 degrees.
- Walk forward 5 steps.
- Turn left 30 degrees.

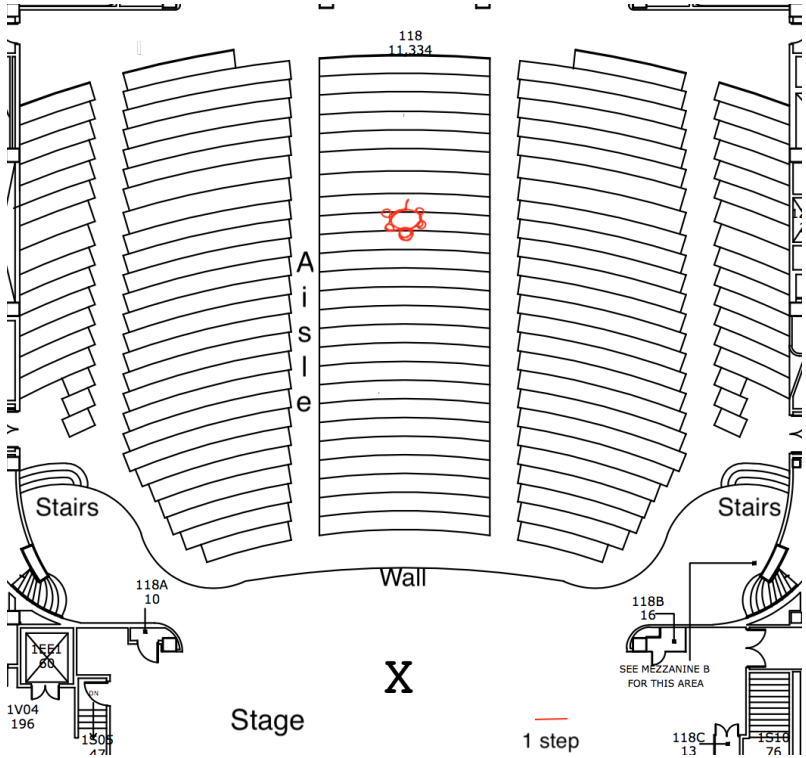
Your Turn!!!



One possible solution:

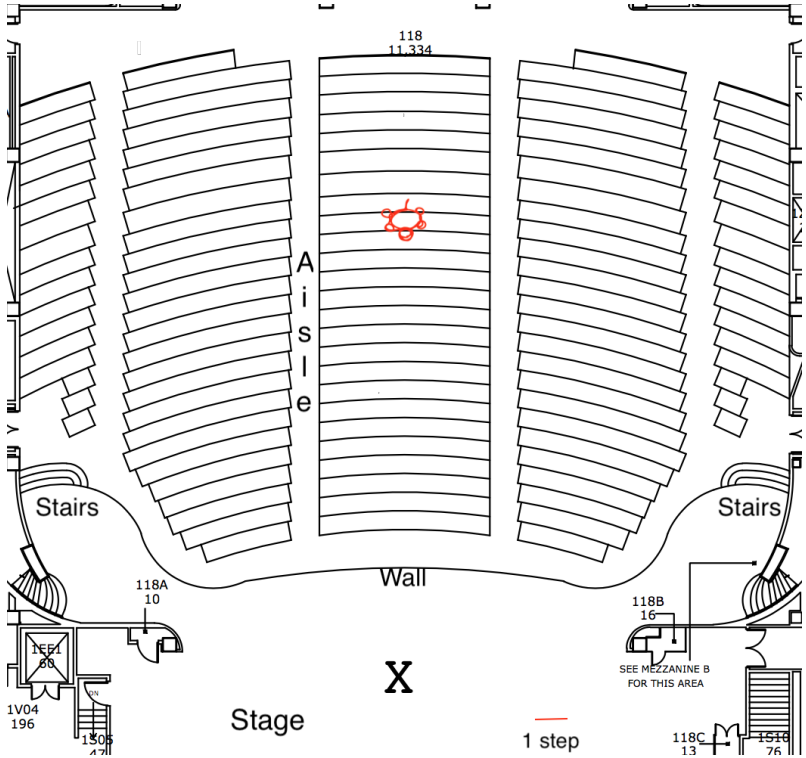
- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.
- Walk forward 3 steps.
- Turn left 80 degrees.
- Walk forward 5 steps.
- Turn left 30 degrees.
- Walk forward 6 steps. Reached X!!!

Your Turn!!!



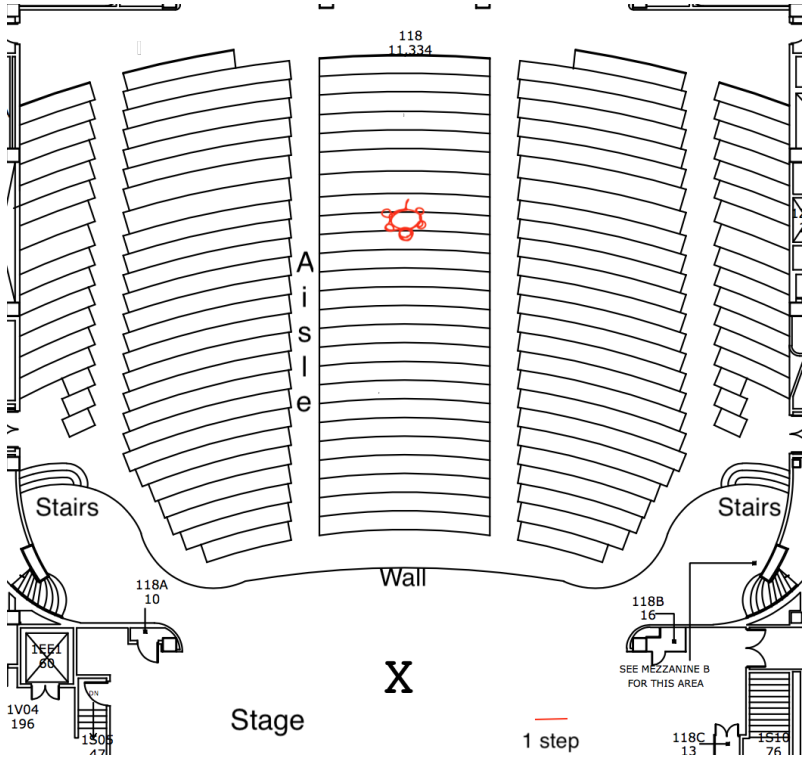
- For fun, post your algorithm on the "Turtle on Stage" forum in the Discussion Board on Blackboard

Your Turn!!!



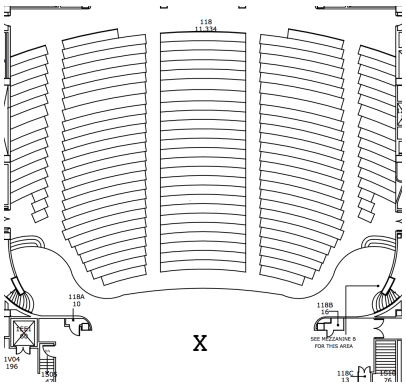
- For fun, post your algorithm on the "Turtle on Stage" forum in the Discussion Board on Blackboard
- "Test and Debug" other students' posted solutions and reply to their posts if you find a bug!

Your Turn!!!



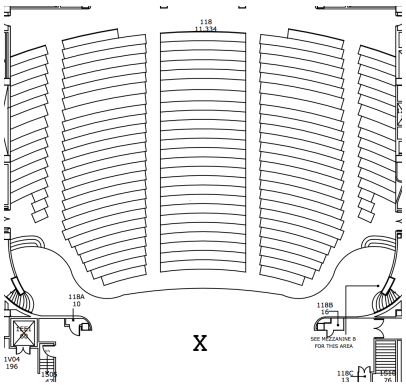
- For fun, post your algorithm on the "Turtle on Stage" forum in the Discussion Board on Blackboard
- "Test and Debug" other students' posted solutions and reply to their posts if you find a bug!
- Degrees the turtle turns are approximate, any good approximation is considered correct.

Recap



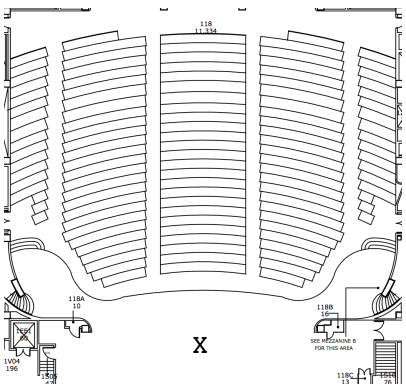
- Writing precise algorithms is difficult.

Recap



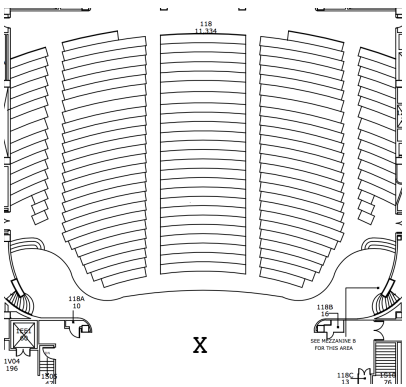
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



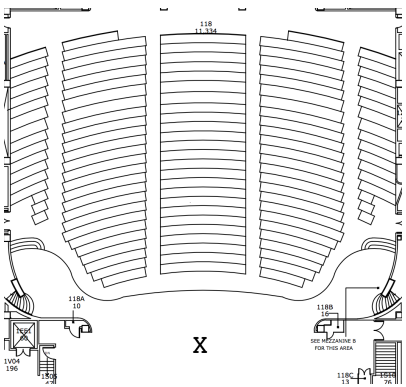
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,

Recap



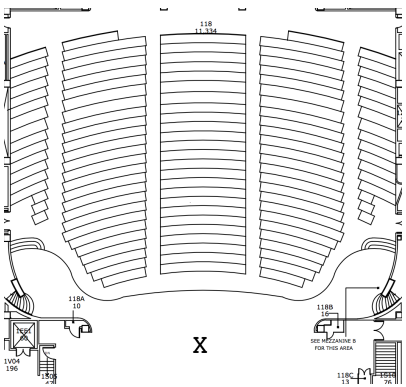
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,

Recap



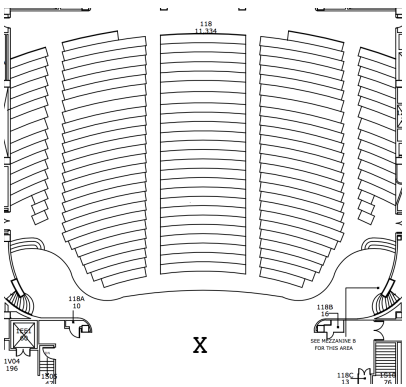
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &

Recap



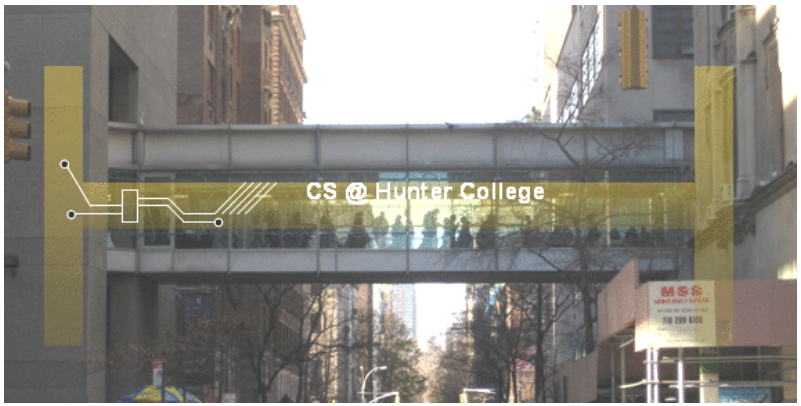
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &
  - variables containing turtles.

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.

Weekly Reminders!



Before next lecture, don't forget to:

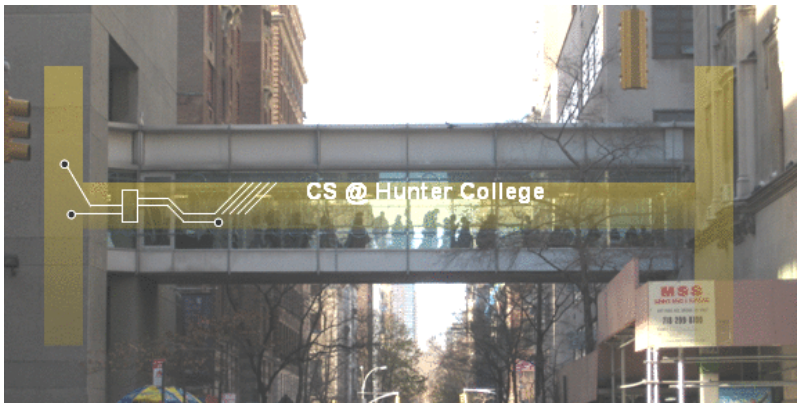
- Work on this week's Online Lab

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 1-5)

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 1-5)
- At any point, visit our Drop-In Tutoring 11am-5pm for help!!!

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Optional - attend Lab Review (Zoom links on Blackboard / Syncrhonous Meetings)
- Take the Lab Quiz on Gradescope by 6pm on Wednesday
- Submit this week's 5 programming assignments (programs 1-5)
- At any point, visit our Drop-In Tutoring 11am-5pm for help!!!
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)