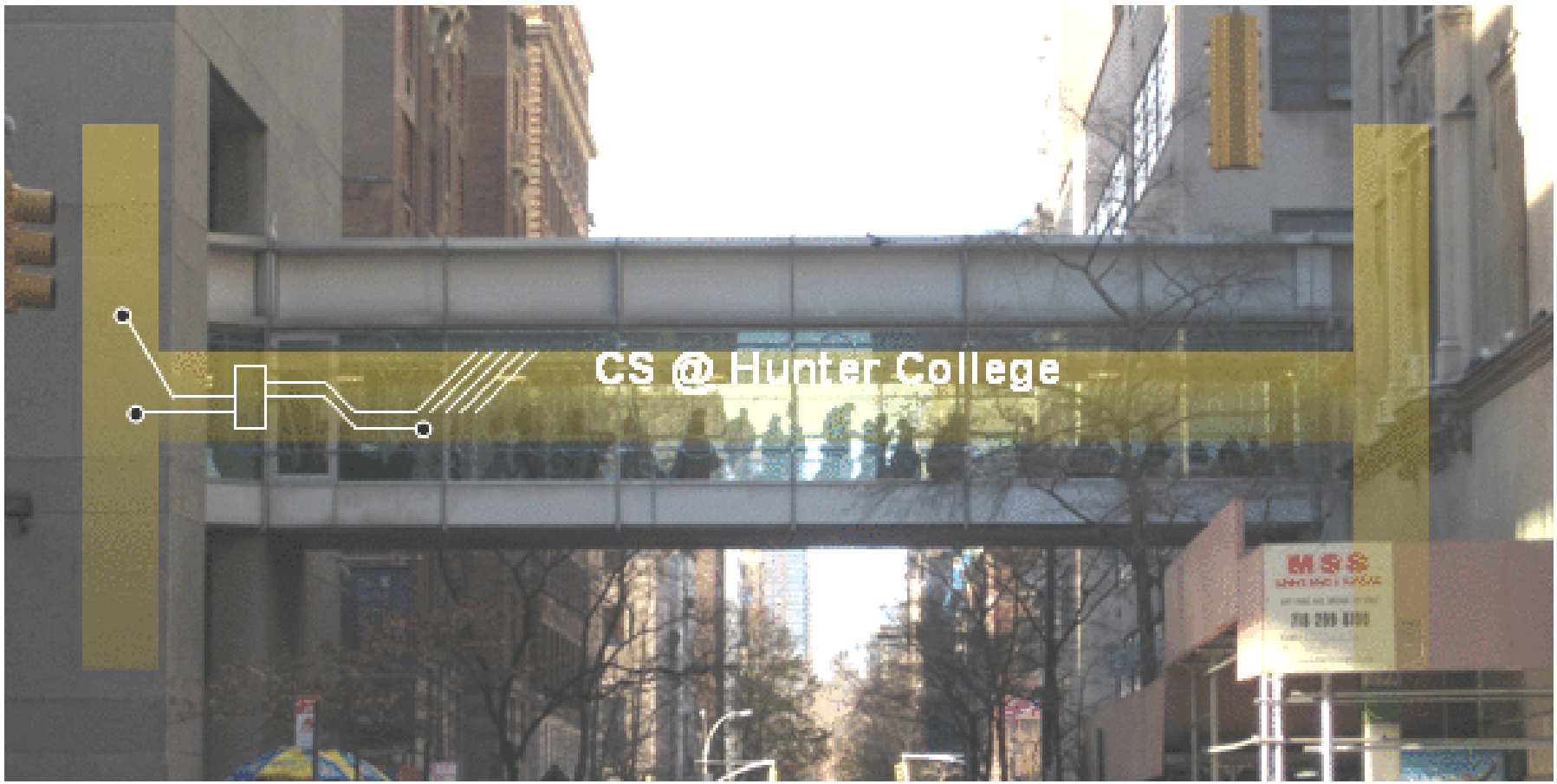
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

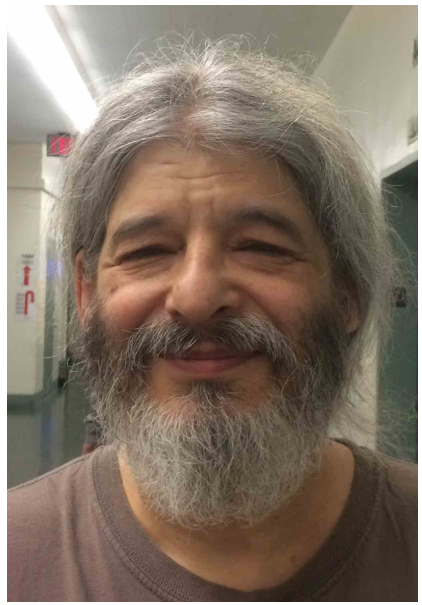
Introductions: Course Designers

Dr. Katherine St. John

Professor,

Dr. William Sakas

Associate Professor, Chair



Prof. Eric Schweitzer

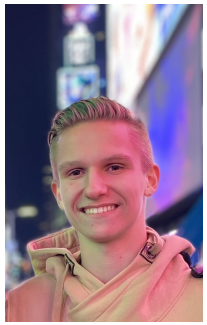
Undergraduate Program Coordinator

Introductions: Instructors

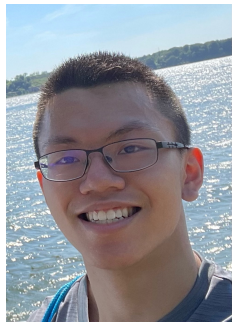
Dr. Tong Yi

Large Lecture
Course Coordinator

Introductions: Undergraduate Teaching Assistants



Adrian Mysliwiec



Alvin Wu

Amy Ng
lecture TA

Andy Li

Anika Sujana

Arsen Tumanian

Arshadul Monir
lecture TA

Arterio Rodrigues

Bode Chiu

Brendan South

Christopher Asma

Diana Luna

Farhin Bhuiyan

Filip Trzcinka

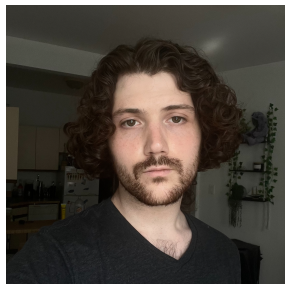
Georgina Woo
lecture TA

Introductions: Undergraduate Teaching Assistants:II

Ghazanfar Shahbaz

Hanz De Guzman

Hnin Lwin



Jeffrey Waters

Jessica Flores Olmos

Kazi Mansha
lecture TA

Kevin Perez

Maliha Tasnim
lecture TA

Manuel Reyes
lecture TA

Mashiyat Mahdi
lecture TA

Michelle Thaung

Moududur "Moody" Rahman
lecture TA

Omer Skaljic

Rita Chen

Roy Delgado

Introductions: Undergraduate Teaching Assistants: III

Ryan Vaz

Sheikh Fuad
lecture TA

Tyler Robinson

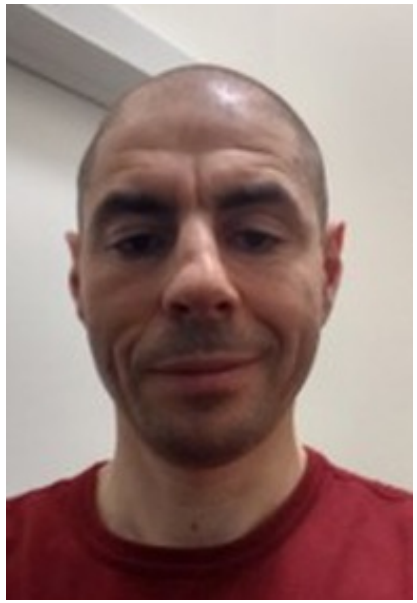
Yoomin Song

Zeeshan Ahmed Gondal

Introductions: Advisors

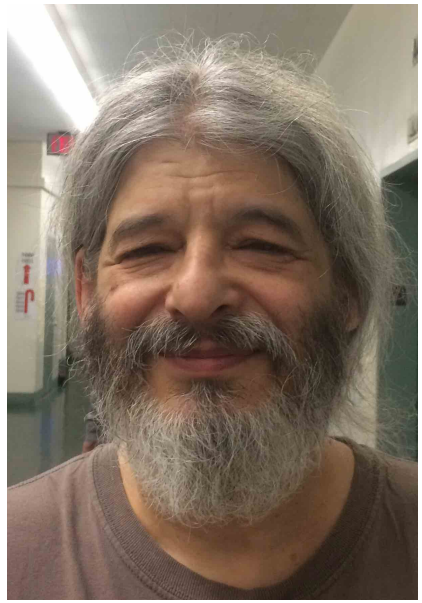
Justin Tojeira
CS Advisor

jtojeira@hunter.cuny.edu



Pavel Shostak
CS Advisor

ps57@hunter.cuny.edu



Eric Schweitzer
Undergraduate
Program Coordinator
eschweit@hunter.cuny.edu

Check https://www.hunter.cuny.edu/csci/advising/advising for details.

Where to find Course Content

- **Course Website:** https://huntercsci127.github.io/s23.html

Where to find Course Content

- **Course Website:** https://huntercsci127.github.io/s23.html
- **Blackboard:** Announcement, Discussion Board, Lecture Preview, Supplement course materials

Where to find Course Content

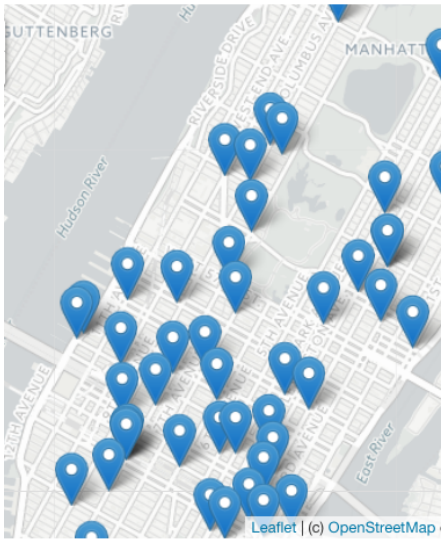
- **Course Website:** https://huntercsci127.github.io/s23.html
- **Blackboard:** Announcement, Discussion Board, Lecture Preview, Supplement course materials
- **Gradescope** (programming assignments submission)

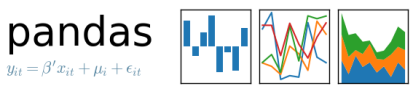
Syllabus

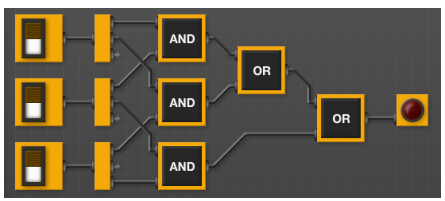
CSci 127: Introduction to Computer Science
Catalog Description: 3 hours, 3 credits: This course presents an overview of computer science (CS) with an emphasis on **problem-solving and computational thinking through 'coding'***: computer programming for beginners...*

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

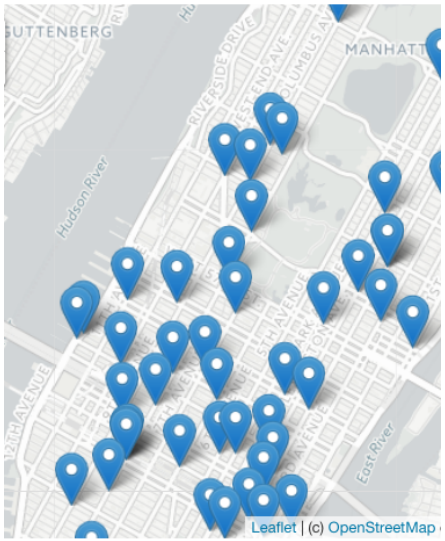


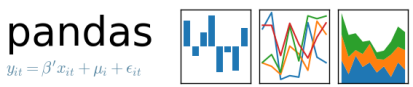


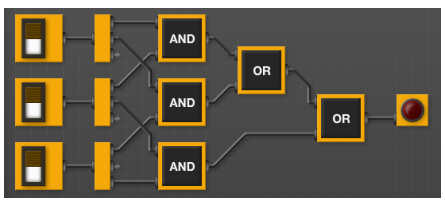


- **This course assumes no previous programming experience.**

Syllabus: Topics

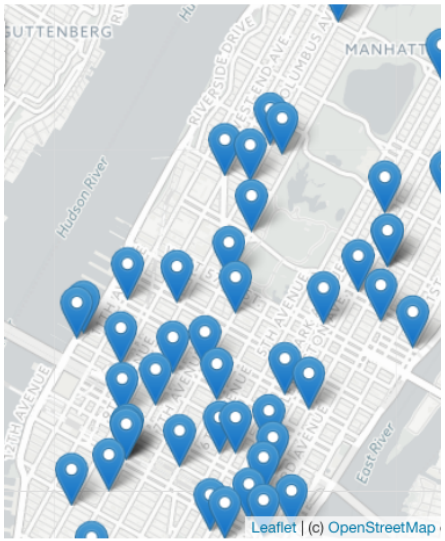


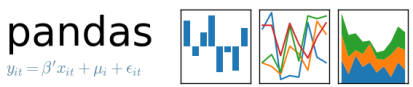


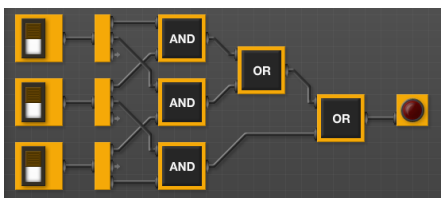


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:

Syllabus: Topics

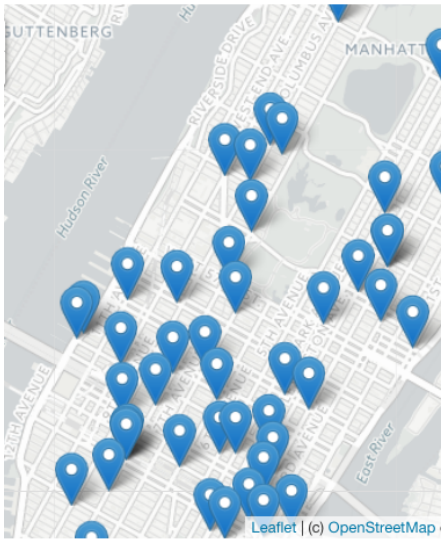


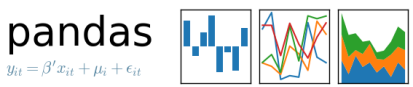


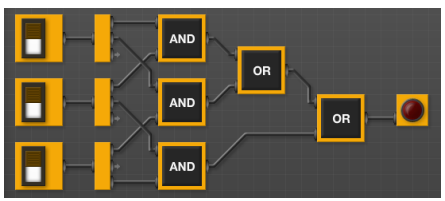


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,

Syllabus: Topics

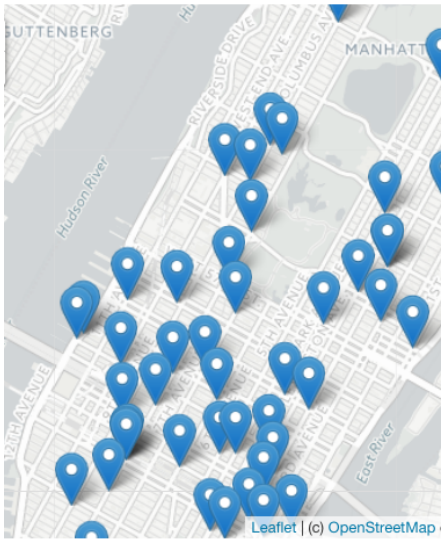


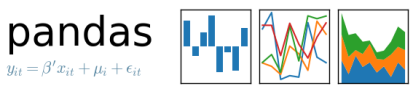


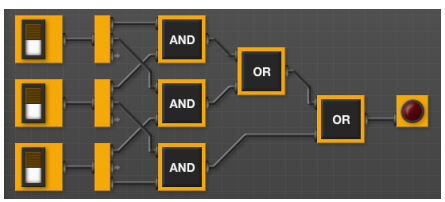


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),

Syllabus: Topics

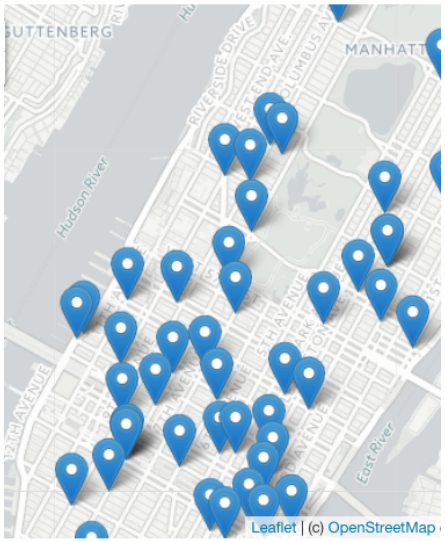


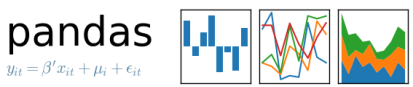


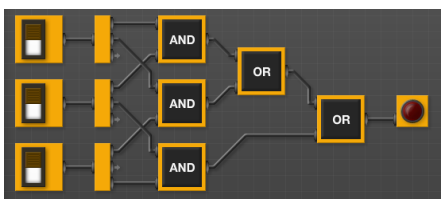


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:

Syllabus: Topics

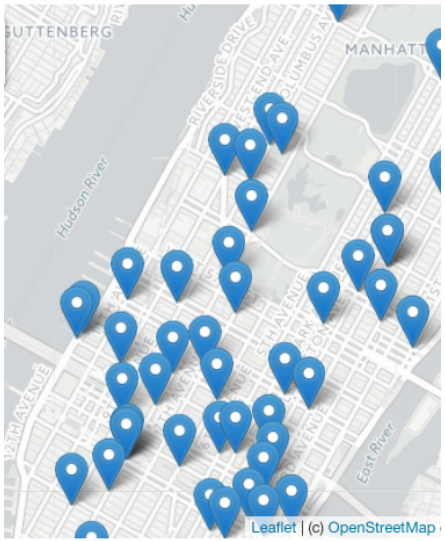


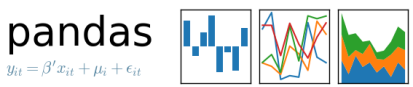


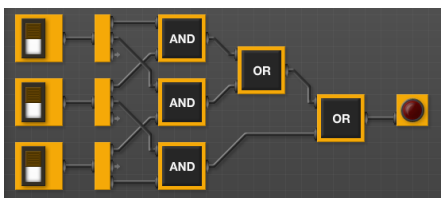


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,

Syllabus: Topics

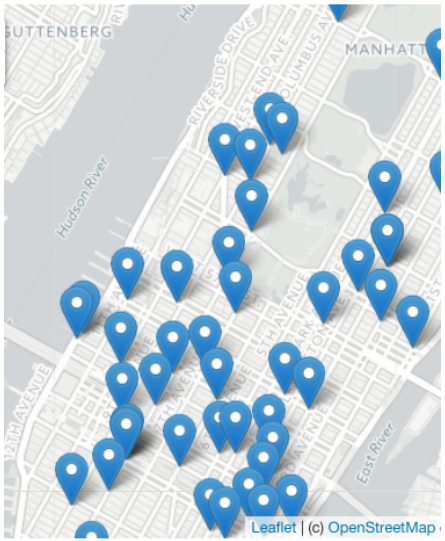


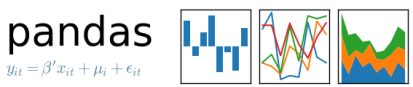


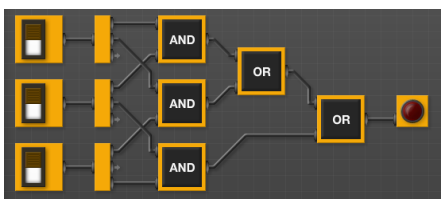


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

Syllabus: Topics

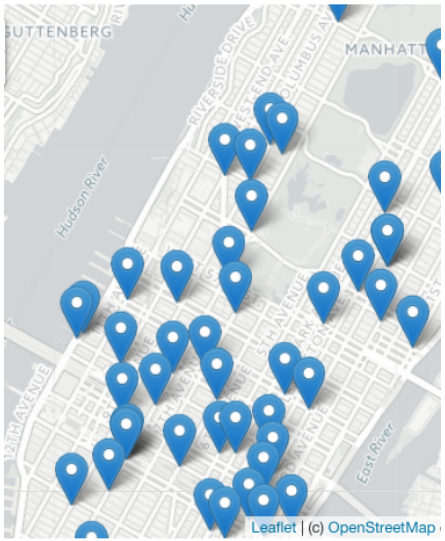






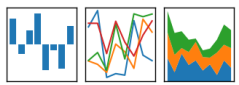
- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

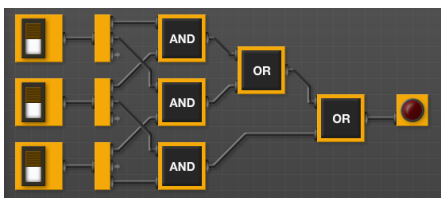
Syllabus: Topics



pandas

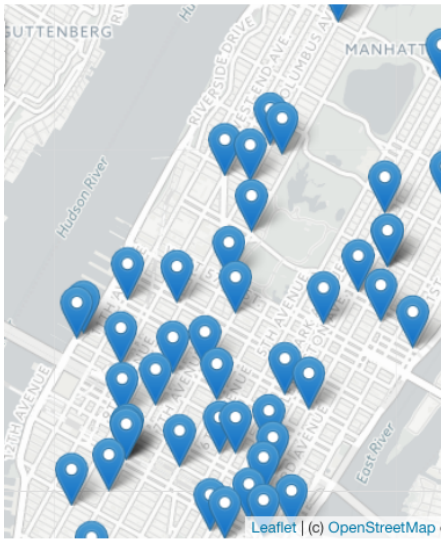
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



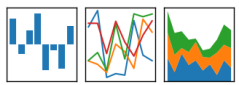


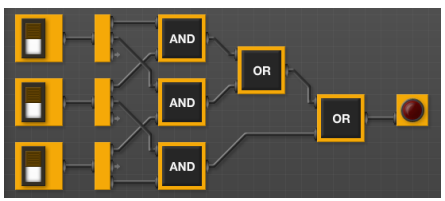
- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

Syllabus: Topics



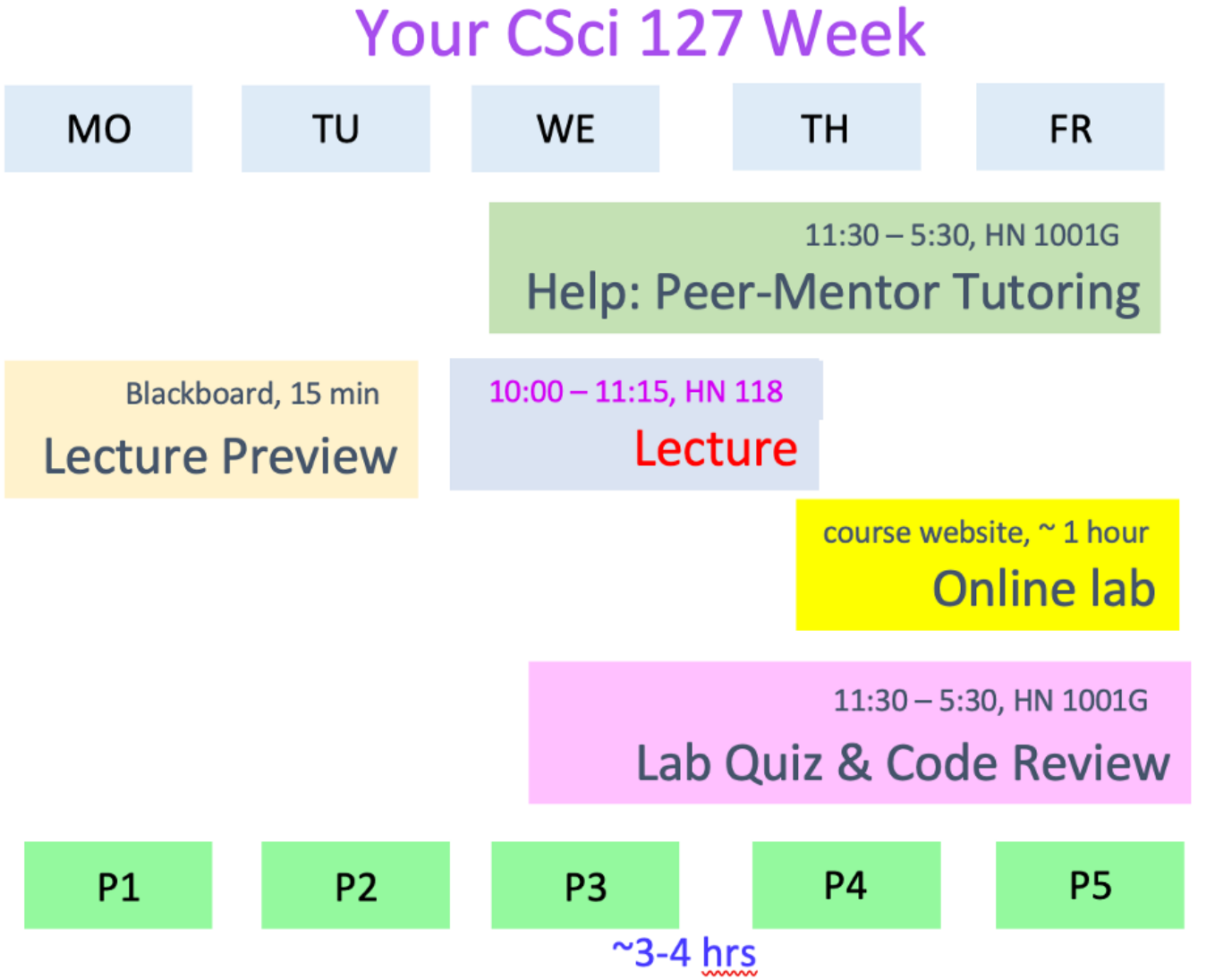
pandas





- **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: Overview



You should work on programming assignments ahead of due dates. Working on assignments on the days when they are due will increase the chance to miss deadline.

Course Structure: Lecture Preview

Category	**Lecture Preview**
Where	blackboard website, left pane
When	available on the Monday before lecture day, due at 10:15 AM on lecture day (**exception:** the first lecture preview is due at 10:15 AM on 2/1/23.)
Number of Tries	multiple tries before deadline
Expected time	15 minutes
Missing Assignments	No make up, will be replaced by the final grade.
Weight	5% of total grade
Note	also called online quiz

Course Structure: Lecture Slip

Category	**Lecture Slip**
Where	lecture room
When	distribute in each lecture, submitted to TAs before the end of lecture
Number of Tries	once in lecture
Missing Assignments	No make up, will be replaced by the final grade.
Weight	5% of total grade
Note	Need to write some meaningful things besides name and empl id. Grade for participation, not for correctness. But you need to try.

Course Structure: Online Lab

Category	**Online Lab**
Where	The labs are put online, for example, Lab 1 is in https://huntercsci127.github.io/s23/lab1.html. There are 13 labs, change 1 to the corresponding number to get that lab.
Expected Time	1 - 1.5 hours
Note	No submission is needed, you read a lab and work on its tasks before working on programming assignments and quizzes. This course is hybrid, besides in-person lecture and lab, you need to learn online labs.

Course Structure: Paper Quiz in In-Person Lab

Category	**Paper Quiz in In-Person Lab**
Where	North Building 1001 G
When	A total of 13 paper quizzes. Make appointments through navigate before due dates. (**exception:** Can walk in the week of 1/30/23 - 2/3/23 before navigate is ready.)
Due Date	Each quiz's due date is shown in deadlines for paper quizzes, code reviews, and programming assignments.
Number of Tries	at most one try before the deadline
Note	Close books, close notes, no electronic devices. Expect to take 15 minutes.
Missing Assignments	No make up, will be replaced by the final grade.
Weight	25% of total grade

Course Structure: Code Review in In-Person Lab

Category	**Code Review in In-Person Lab**
Where	North Building 1001 G
When	A total of 12 code reviews. Make appointments through navigate before due dates. (**exception:** Can walk in the week of 1/30/23 - 2/3/23 before navigate is ready.)
Due Date	Each code review's due date is shown in deadlines for paper quizzes, code reviews, and programming assignments.
Number of Tries	at most one try before the deadline
Expected time	15 minutes
Missing Assignments	No make up, will be replaced by the final grade.
Weight	5% of total grade

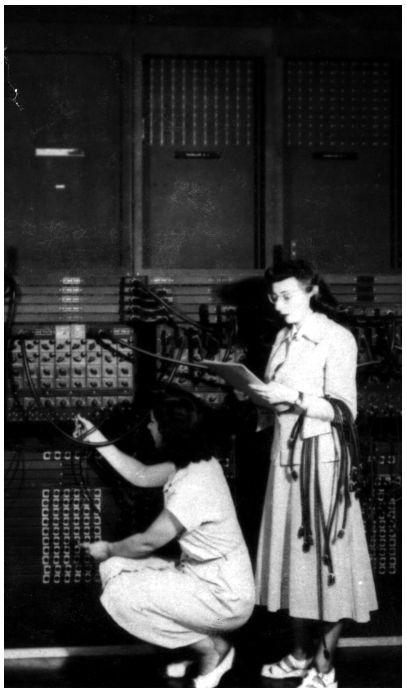
Course Structure: Programming Assignments

Category	**Programming Assignments**
Where	Questions are posted in programming assignments web page and submit to gradescope.
Due Date	Each programming assignment's due date is shown in deadlines for paper quizzes, code reviews, and programming assignments.
Number of Tries	as many times as you like before the deadline
Missing Assignments	No make up, CANNOT be replaced by the final grade. Will drop the lowest 5 programming assignments.
Weight	30% of total grade
Note	Start early. Do not wait until the last day.

Course Structure: Final Exam

Category	**Final Exam**
Where	North Building 118 (lecture room)
Due Date	9 - 11 AM, May 22, 2023, Monday.
Number of Tries	at most once
Importance	**NO midterm. You must pass the final to pass the course.** Furthermore, to take CS 135 or above, you need to pass the final and get a total grade of C or above.
Weight	30% of total grade
Note	If you need extra time, take the exam in accessibility office. Prepare enough time to come to the lecture room, public transportation might not be reliable.

1&2 - Lecture

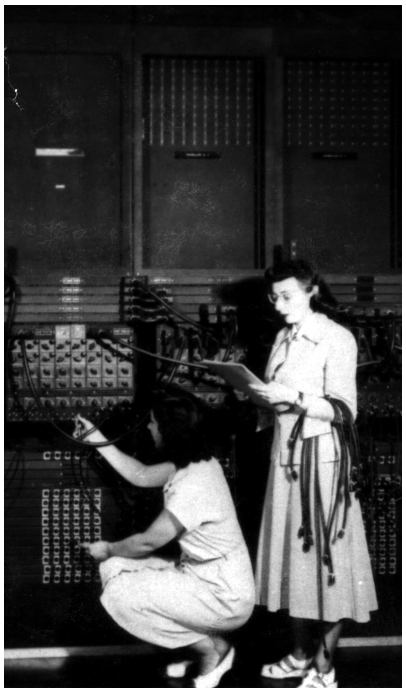


First "computers"

ENIAC, 1945.

- Tuesdays, 10:00 -11:15am, In person: 118 HN, Assembly Hall

1&2 - Lecture

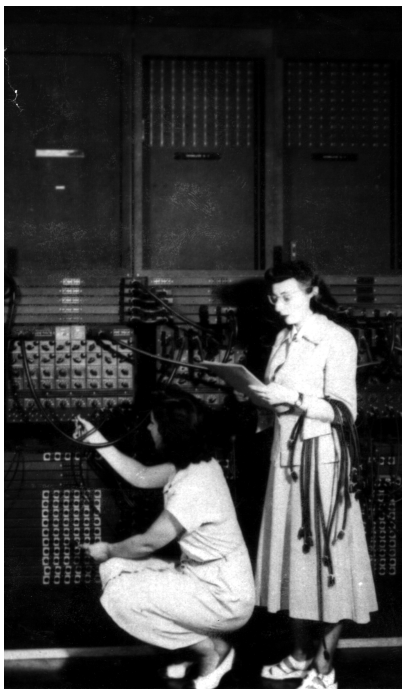


First "computers"

ENIAC, 1945.

- Tuesdays, 10:00 -11:15am, In person: 118 HN, Assembly Hall
- Mix of explanation, challenges & group work.

1&2 - Lecture

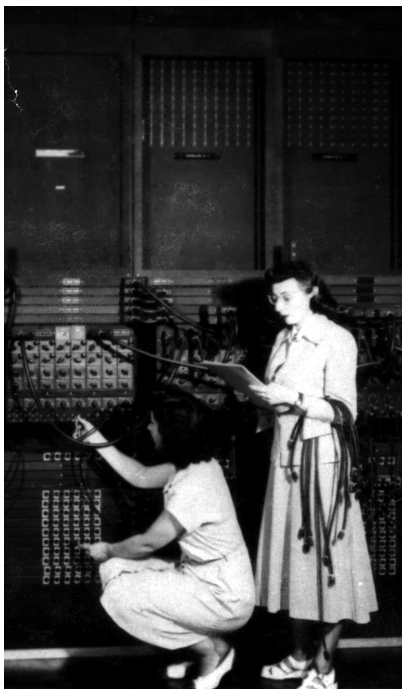


First "computers"

ENIAC, 1945.

- Tuesdays, 10:00 -11:15am, In person: 118 HN, Assembly Hall
- Mix of explanation, challenges & group work.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).

1&2 - Lecture

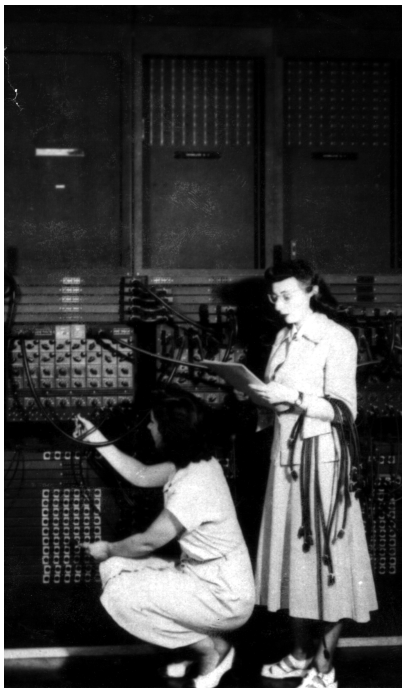


First "computers"

ENIAC, 1945.

- Tuesdays, 10:00 -11:15am, In person: 118 HN, Assembly Hall
- Mix of explanation, challenges & group work.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).
- Lecture Slips: group challenges during lecture.

1&2 - Lecture

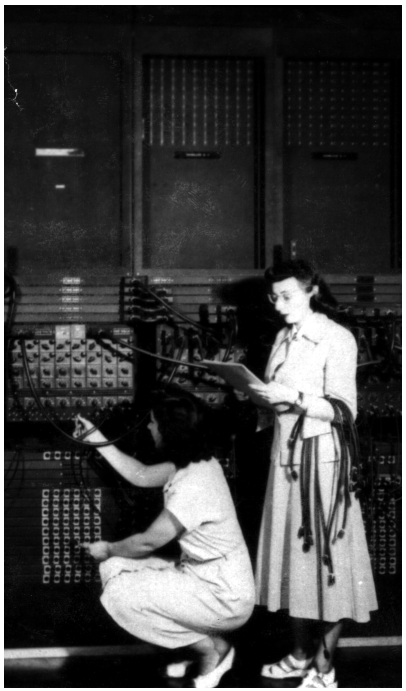


First "computers"

ENIAC, 1945.

- Tuesdays, 10:00 -11:15am, In person: 118 HN, Assembly Hall
- Mix of explanation, challenges & group work.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).
- Lecture Slips: group challenges during lecture.
- Ask questions during group work.

3 - Online Lab



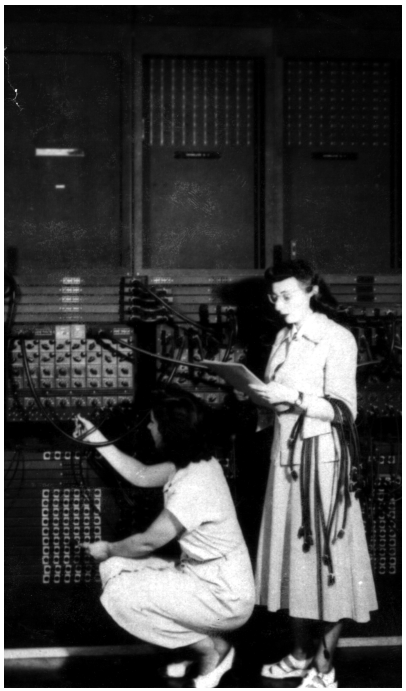
First "computers"

ENIAC, 1945.

Each Week:

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

3 - Online Lab



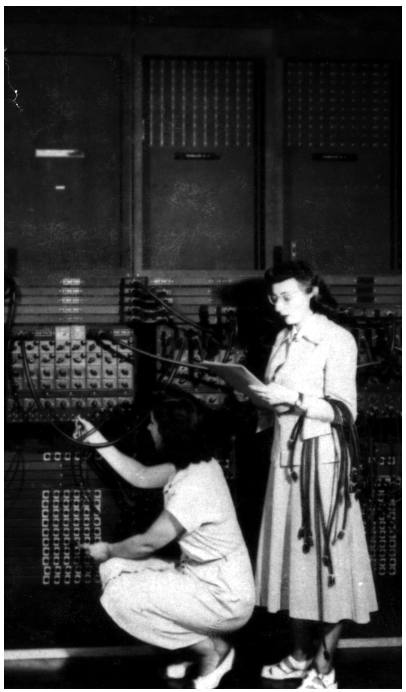
First "computers"

ENIAC, 1945.

Each Week:

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

3 - Online Lab



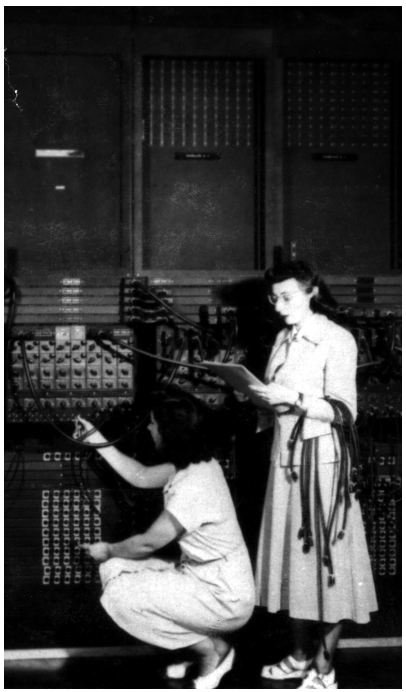
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.
- Lab content directly supports weekly programming assignments.

3 - Online Lab



First "computers"

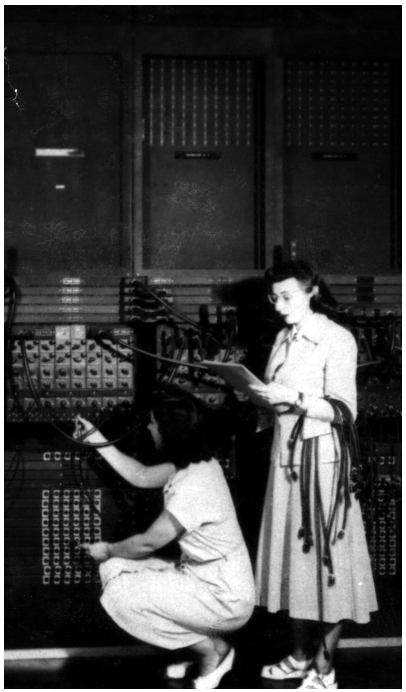
ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Set aside about 1 hour each week, preferably at the same time, add it to your schedule.
- Lab content directly supports weekly programming assignments.
- Labs found on course website.

4 -In-person Quiz & Code Review

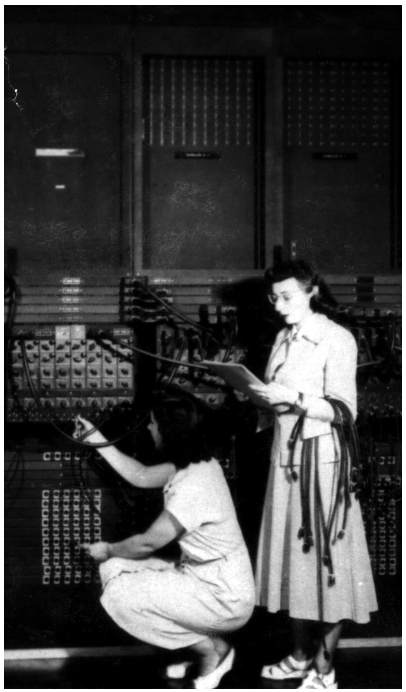
- **Every week you must take a paper quiz** in Lab 1001G Hunter North



First "computers"

ENIAC, 1945.

4 -In-person Quiz & Code Review

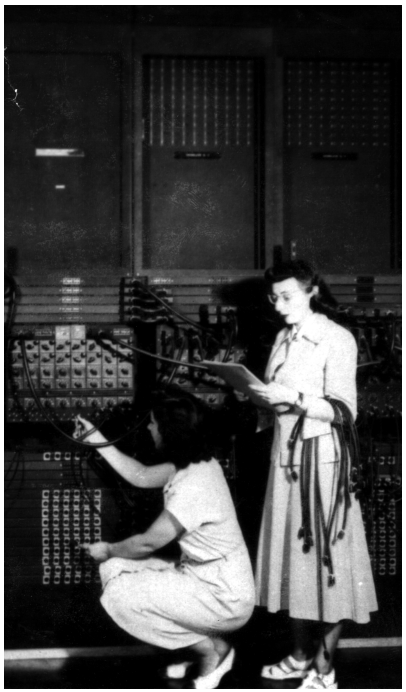


First "computers"

ENIAC, 1945.

- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content

4 -In-person Quiz & Code Review

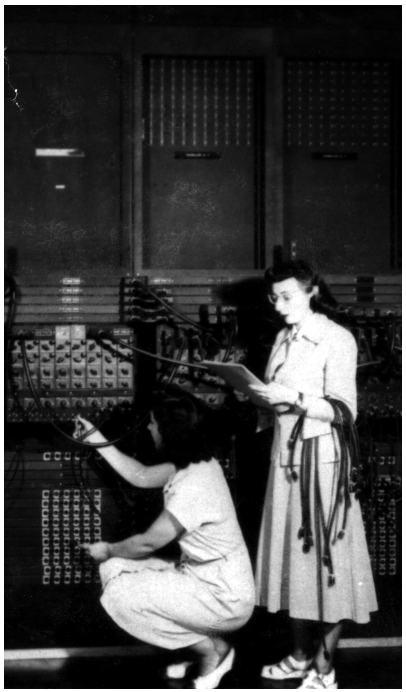


First "computers"

ENIAC, 1945.

- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content
- **Every week you must take a code review** in Lab 1001G Hunter North

4 -In-person Quiz & Code Review

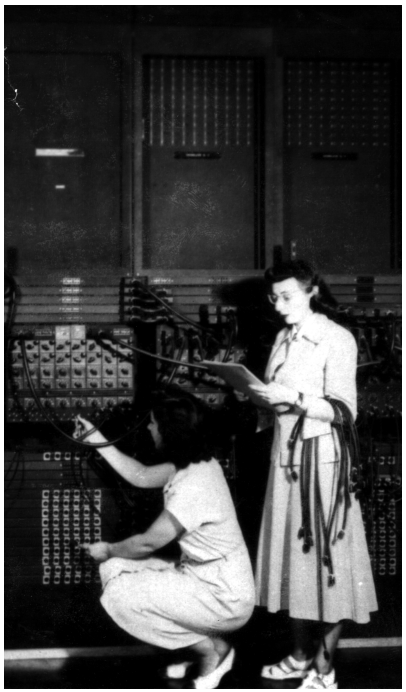


First "computers"

ENIAC, 1945.

- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content
- **Every week you must take a code review** in Lab 1001G Hunter North
- You **must make an appointment** for taking quiz and code review (two separate appointments, you can make them back to back)

4 -In-person Quiz & Code Review

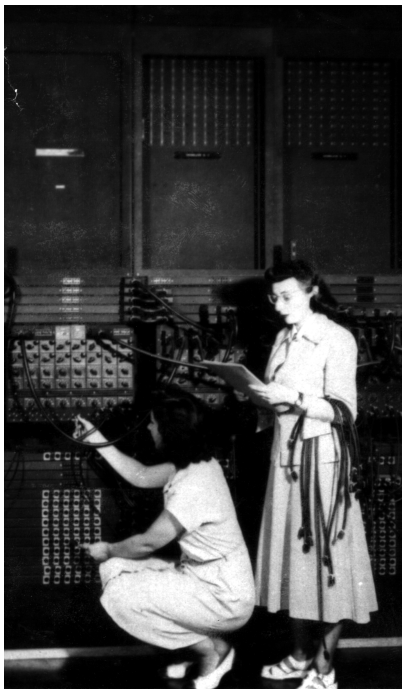


First "computers"

ENIAC, 1945.

- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content
- **Every week you must take a code review** in Lab 1001G Hunter North
- You **must make an appointment** for taking quiz and code review (two separate appointments, you can make them back to back)
- There is limited availability, plan ahead and don't miss your appointments!

4 -In-person Quiz & Code Review

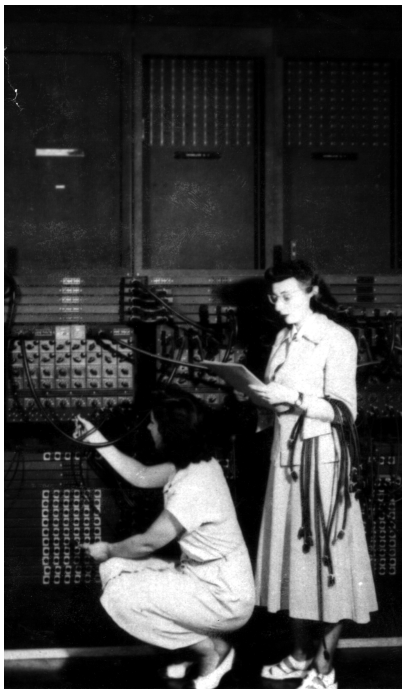


First "computers"

ENIAC, 1945.

- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content
- **Every week you must take a code review** in Lab 1001G Hunter North
- You **must make an appointment** for taking quiz and code review (two separate appointments, you can make them back to back)
- There is limited availability, plan ahead and don't miss your appointments!
- Links to make appointments will be available on Blackboard

4 -In-person Quiz & Code Review

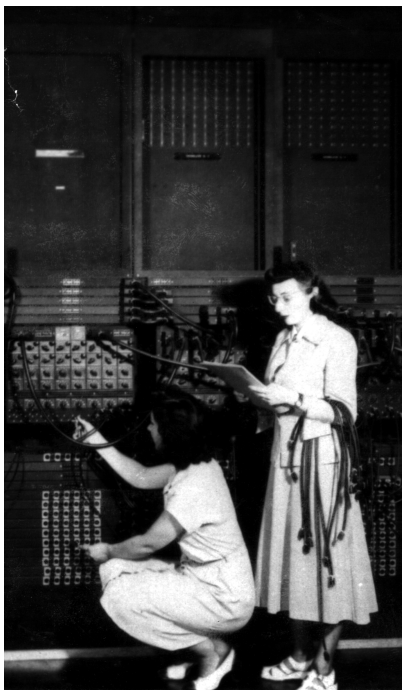


First "computers"

ENIAC, 1945.

- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content
- **Every week you must take a code review** in Lab 1001G Hunter North
- You **must make an appointment** for taking quiz and code review (two separate appointments, you can make them back to back)
- There is limited availability, plan ahead and don't miss your appointments!
- Links to make appointments will be available on Blackboard
- Quiz and code review topics and due dates can also be found on the course website

Programming Assignments



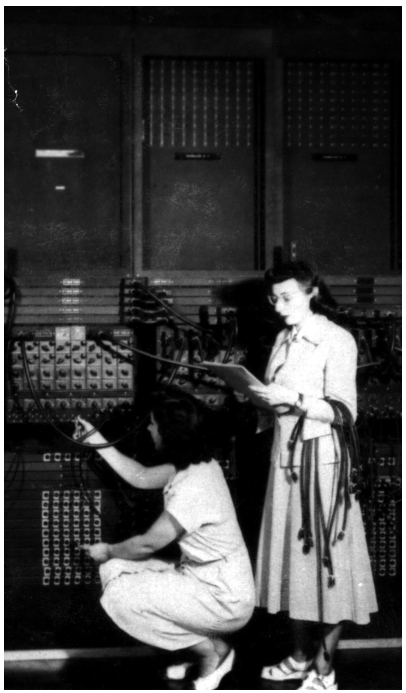
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!

Programming Assignments



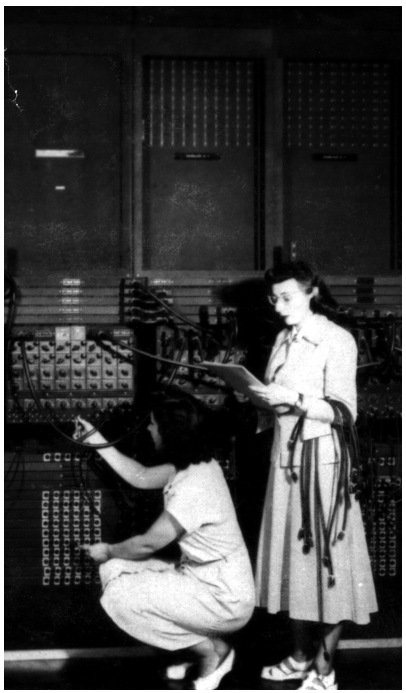
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**

Programming Assignments



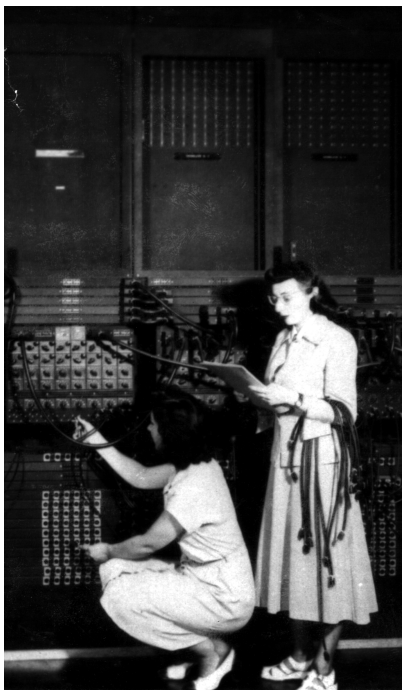
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!

Programming Assignments



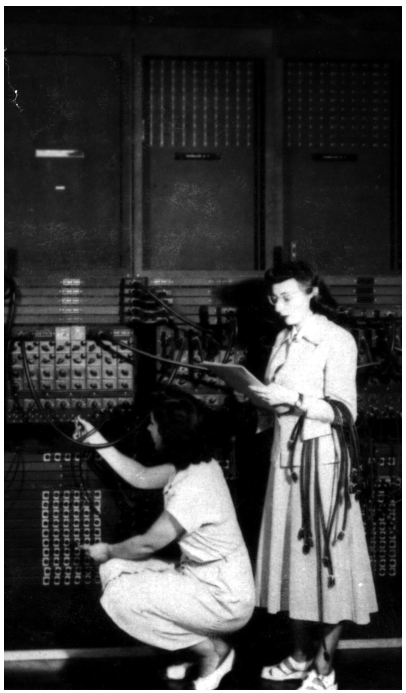
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.

Programming Assignments



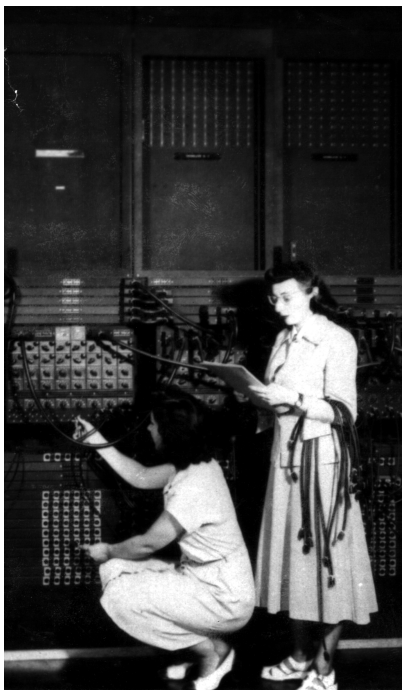
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.

Programming Assignments



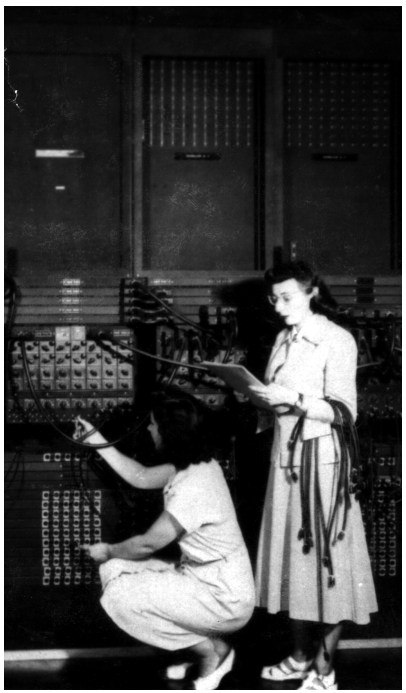
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.

Programming Assignments



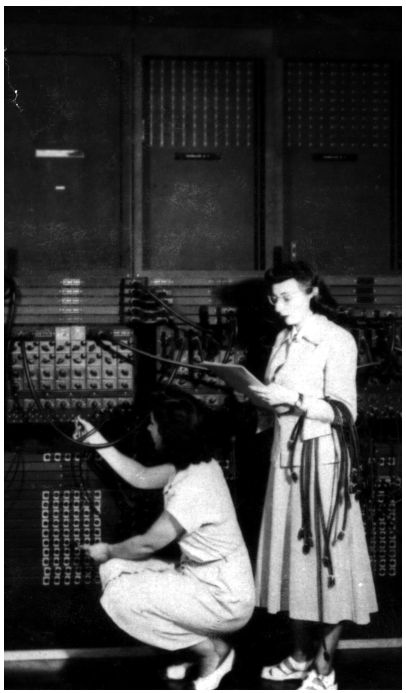
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.
- Multiple submissions accepted.

Programming Assignments



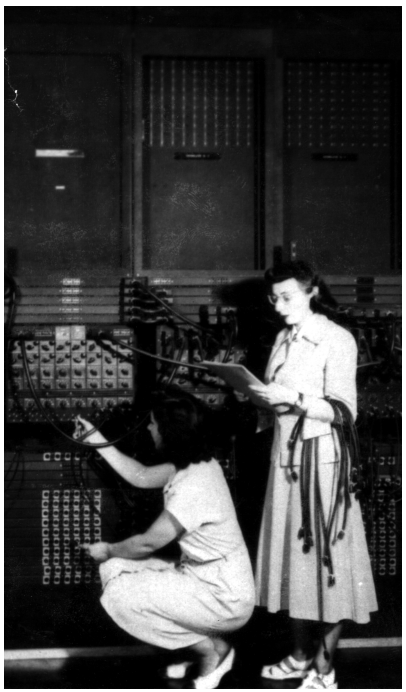
First "computers"

ENIAC, 1945.

Each Week:

- Starting February 9, there will be one program due each day!
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.
- Multiple submissions accepted.
- For help to run and submit programming assignments, please visit the 1001G lab.

Make Your Schedule!

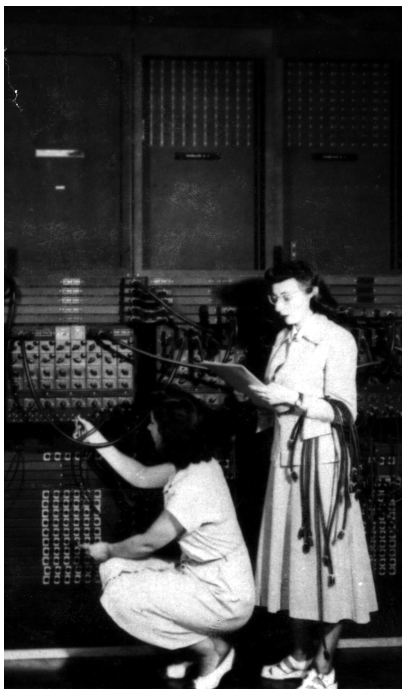


First "computers"

ENIAC, 1945.

- This is a hybrid course: there is some work you must do independently outside of class meetings.

Make Your Schedule!

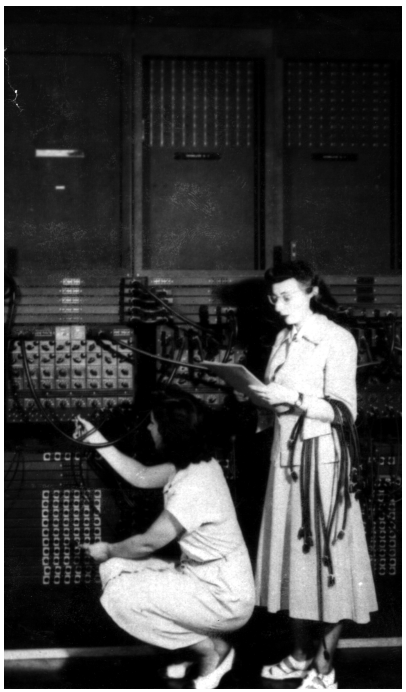


First "computers"

ENIAC, 1945.

- This is a hybrid course: there is some work you must do independently outside of class meetings.
- Schedule a regular time for the **Online lab, for example, lab 1 is in https://huntercsci127.github.io/s23/lab1.html**.

Make Your Schedule!

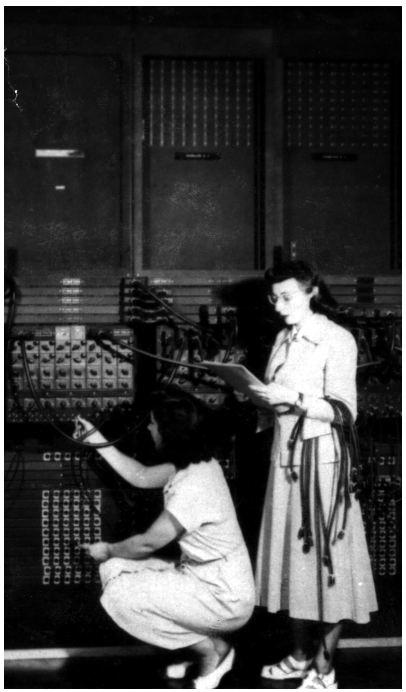


First "computers"

ENIAC, 1945.

- This is a hybrid course: there is some work you must do independently outside of class meetings.
- Schedule a regular time for the **Online lab, for example, lab 1 is in https://huntercsci127.github.io/s23/lab1.html**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!

Make Your Schedule!

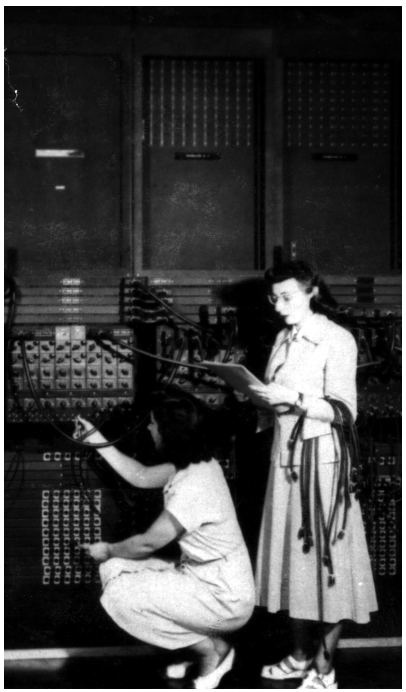


First "computers"

ENIAC, 1945.

- This is a hybrid course: there is some work you must do independently outside of class meetings.
- Schedule a regular time for the **Online lab, for example, lab 1 is in https://huntercsci127.github.io/s23/lab1.html**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!
- Schedule a regular time for working on **programming assignments**.

Make Your Schedule!

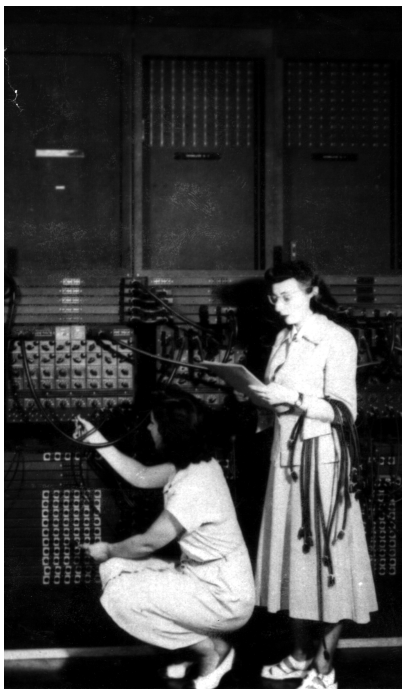


First "computers"

ENIAC, 1945.

- This is a hybrid course: there is some work you must do independently outside of class meetings.
- Schedule a regular time for the **Online lab, for example, lab 1 is in https://huntercsci127.github.io/s23/lab1.html**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!
- Schedule a regular time for working on **programming assignments**.
- Schedule a regular time for taking the **Lecture Preview**

Make Your Schedule!

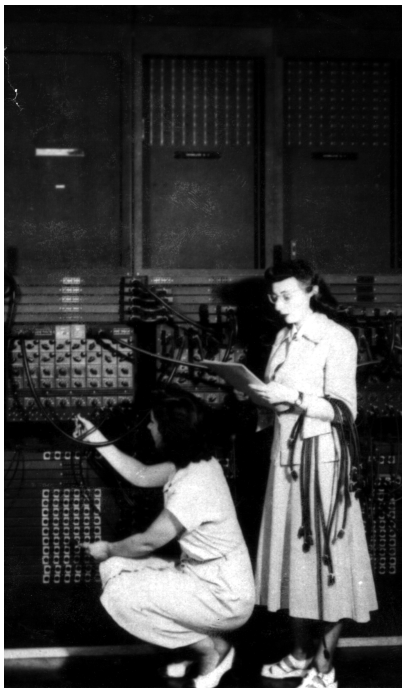


First "computers"

ENIAC, 1945.

- This is a hybrid course: there is some work you must do independently outside of class meetings.
- Schedule a regular time for the **Online lab, for example, lab 1 is in https://huntercsci127.github.io/s23/lab1.html**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!
- 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.

Help and Support

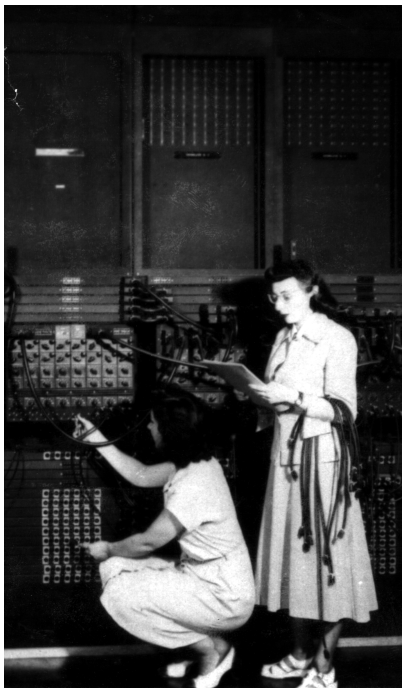


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001G Hunter North

Help and Support

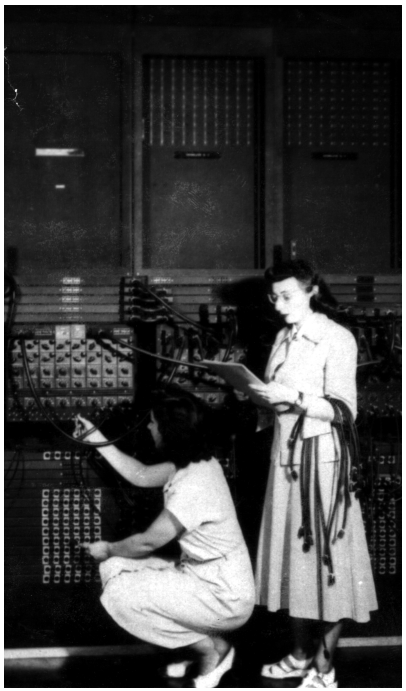


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001G Hunter North
  - Schedule an appointment for tutoring, links will be available on Blackboard

Help and Support

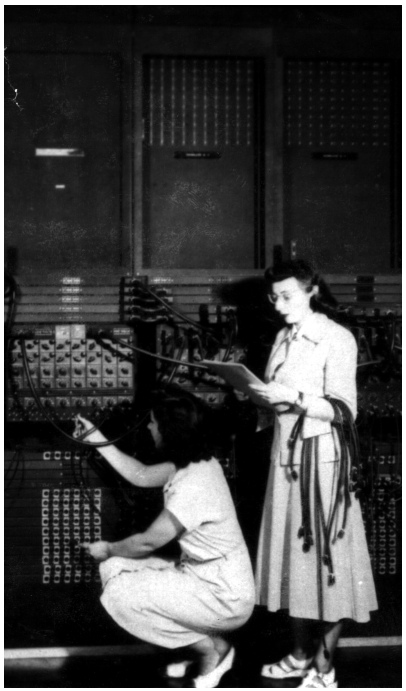


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001G Hunter North
  - Schedule an appointment for tutoring, links will be available on Blackboard
  - **Discussion Board** on Blackboard

Help and Support

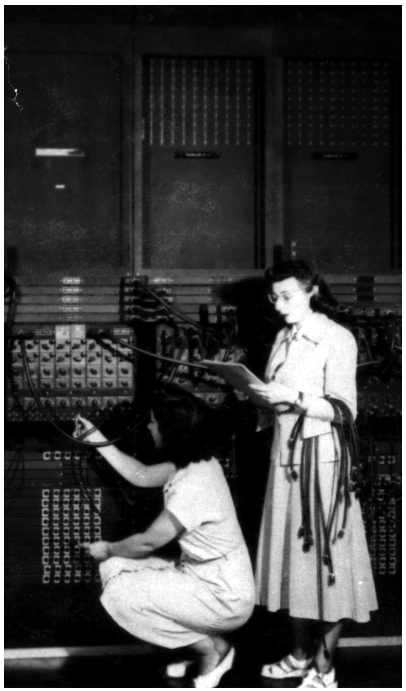


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001G Hunter North
  - Schedule an appointment for tutoring, links will be available on Blackboard
  - **Discussion Board** on Blackboard
  - **Email: cs127uta@hunter.cuny.edu**

Help and Support

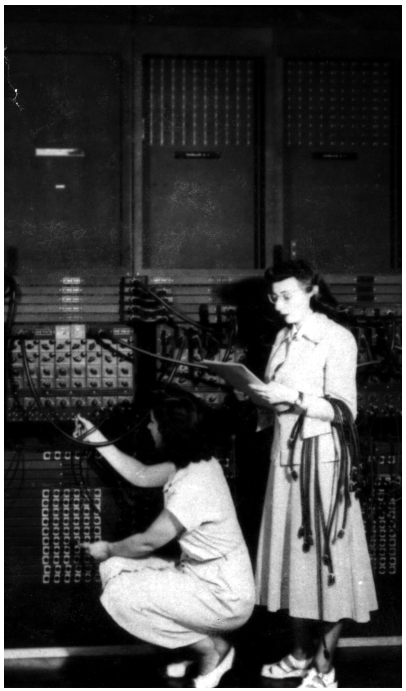


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001G Hunter North
  - Schedule an appointment for tutoring, links will be available on Blackboard
  - **Discussion Board** on Blackboard
  - **Email: cs127uta@hunter.cuny.edu**
  - All help available **Mo-Fr 11:30am-5:30pm** when classes are in session

Help and Support



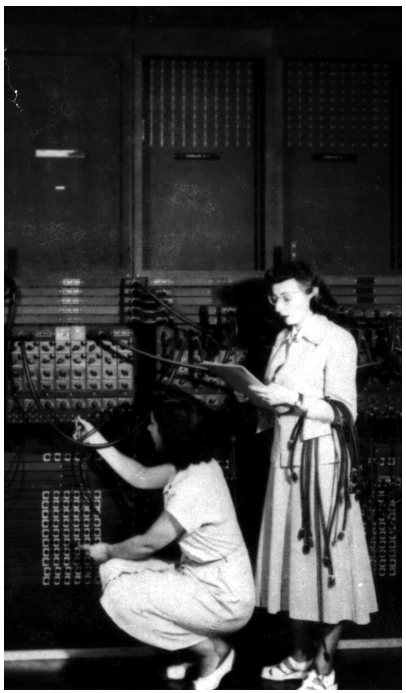
First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001G Hunter North
  - Schedule an appointment for tutoring, links will be available on Blackboard
  - **Discussion Board** on Blackboard
  - **Email: cs127uta@hunter.cuny.edu**
  - All help available **Mo-Fr 11:30am-5:30pm** when classes are in session
- Office Hours with Dr. Tong Yi
  - Drop-in Hours: **Tuesday 12-2pm**
  - By appointment: email ty680@hunter.cuny.edu

Benefits of Tutoring and Code Review

Academic Dishonesty

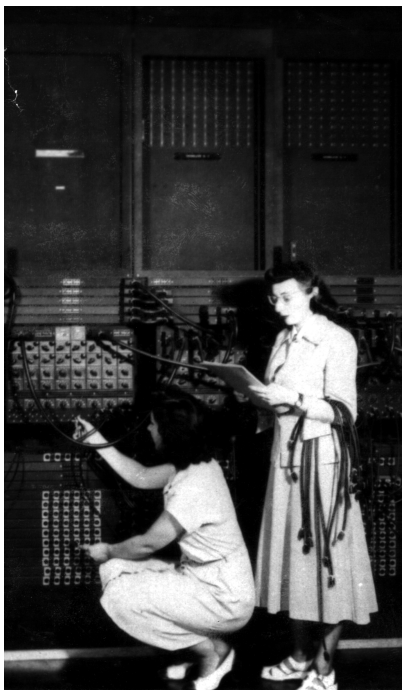


First "computers"

ENIAC, 1945.

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

Academic Dishonesty

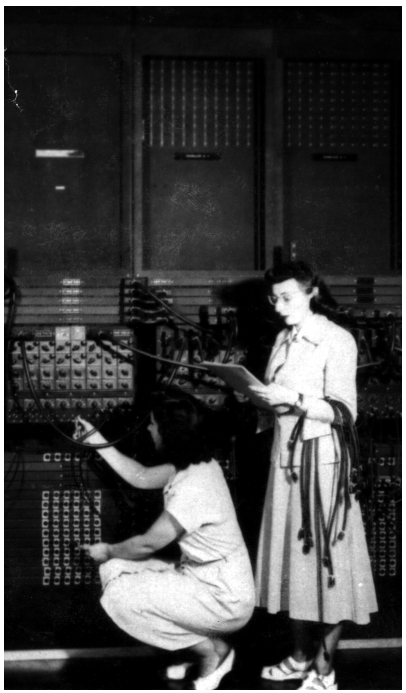


First "computers"

ENIAC, 1945.

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

Academic Dishonesty

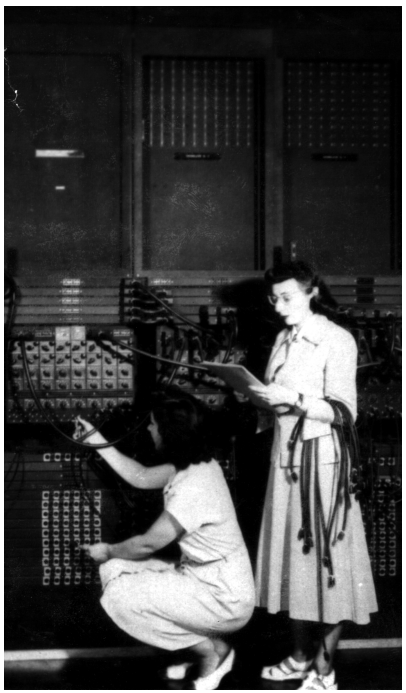


First "computers"

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

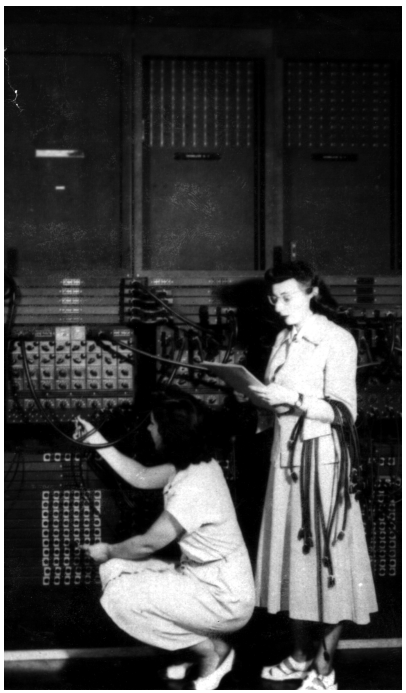


First "computers"

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

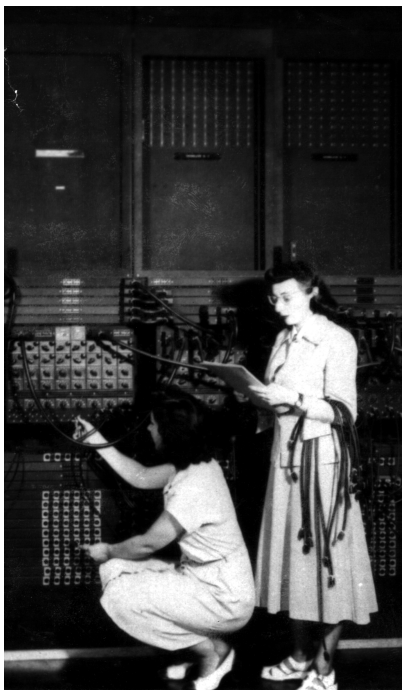


First "computers"

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

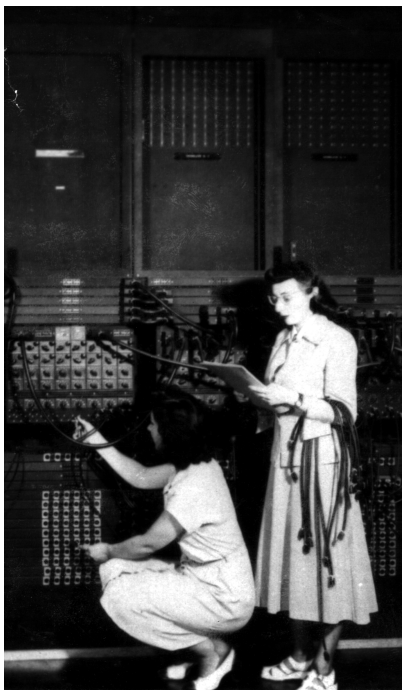


First "computers"

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

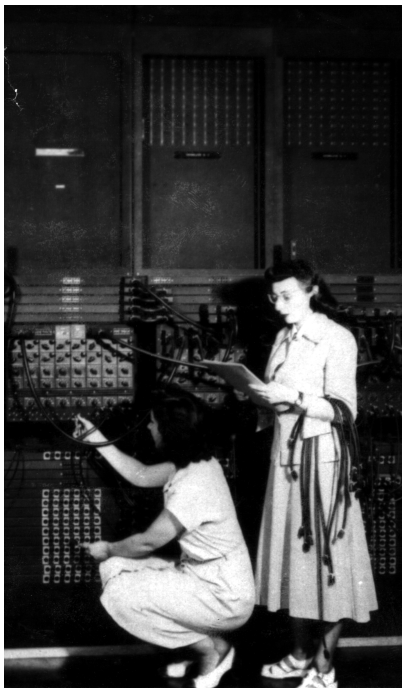


First "computers"

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

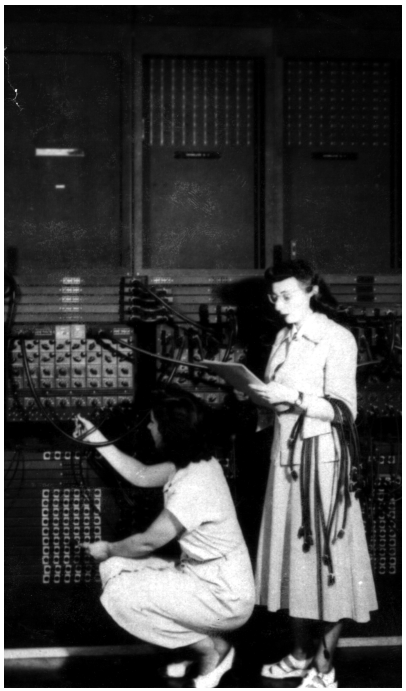


First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard

Communication

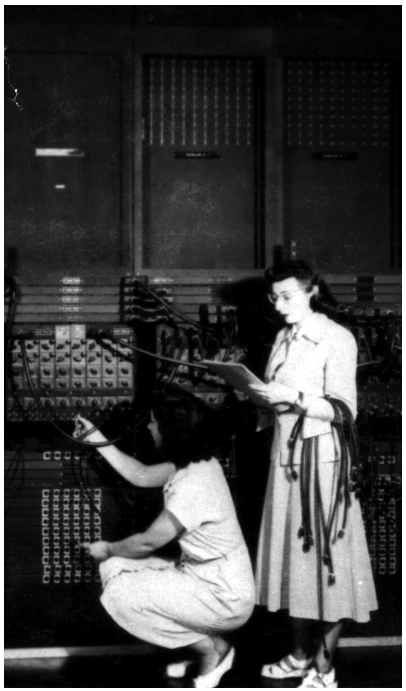


First "computers"

ENIAC, 1945.

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

Communication

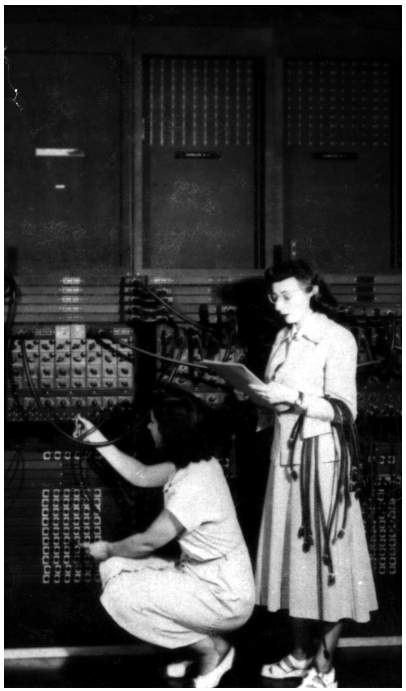


First "computers"

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**
- Instructions for changing your email on Blackboard announcements

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:  Aug 31, 2022
#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

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)

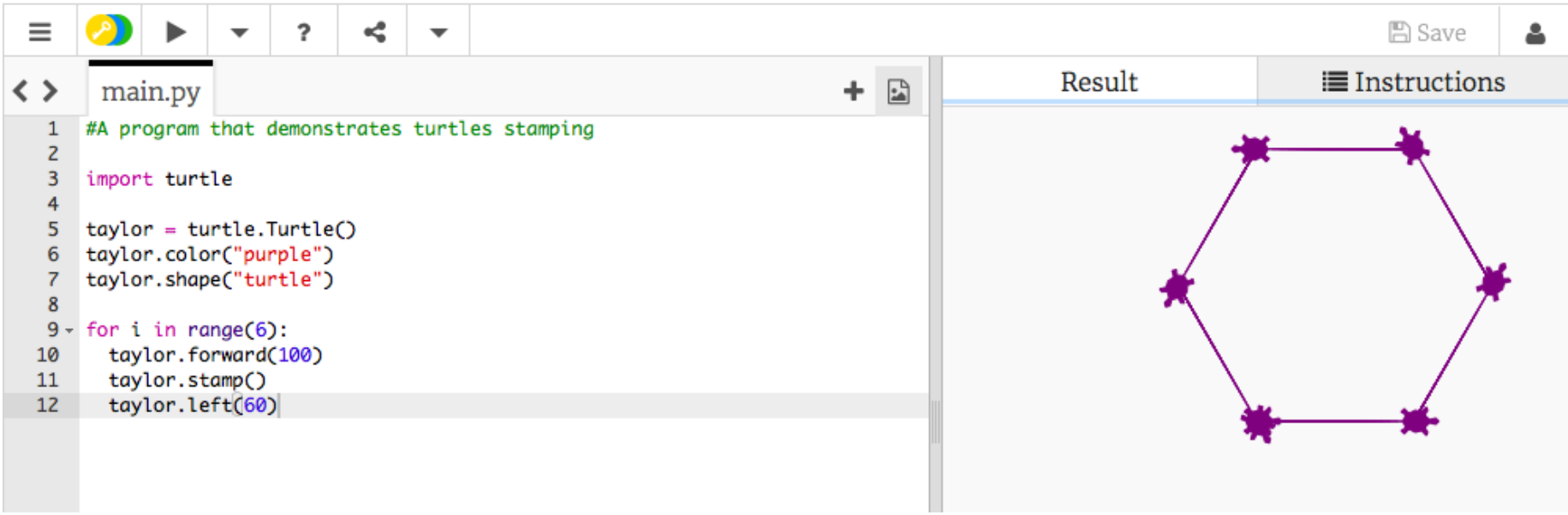
Turtles Introduction

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

Today's Topics

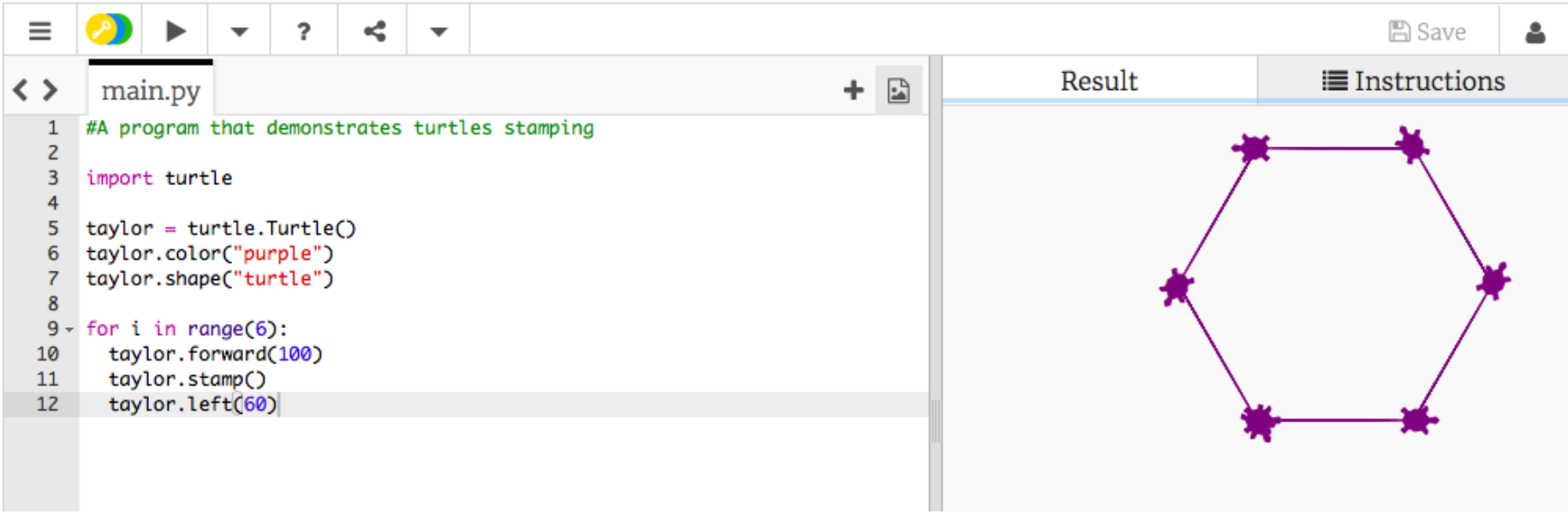
- Introduction to Python
- Turtle Graphics
- **Definite Loops (`for`-loops)**
- Algorithms

Turtles Introduction



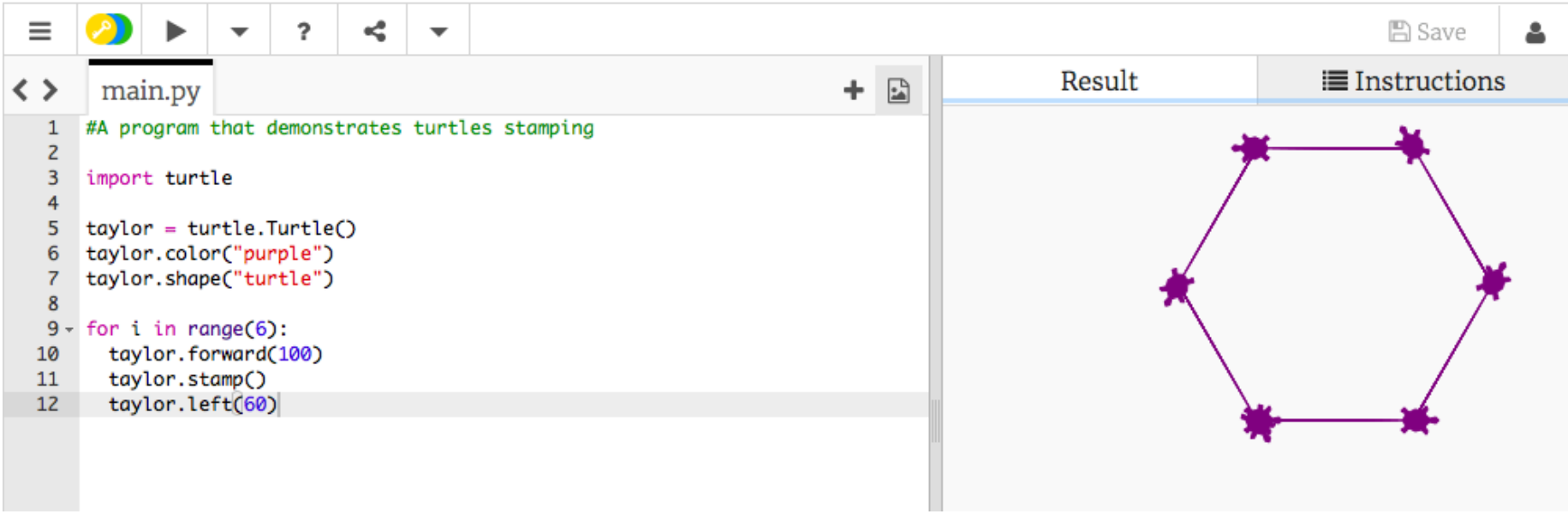
- Creates a turtle **variable**, called `taylor`.

Turtles Introduction



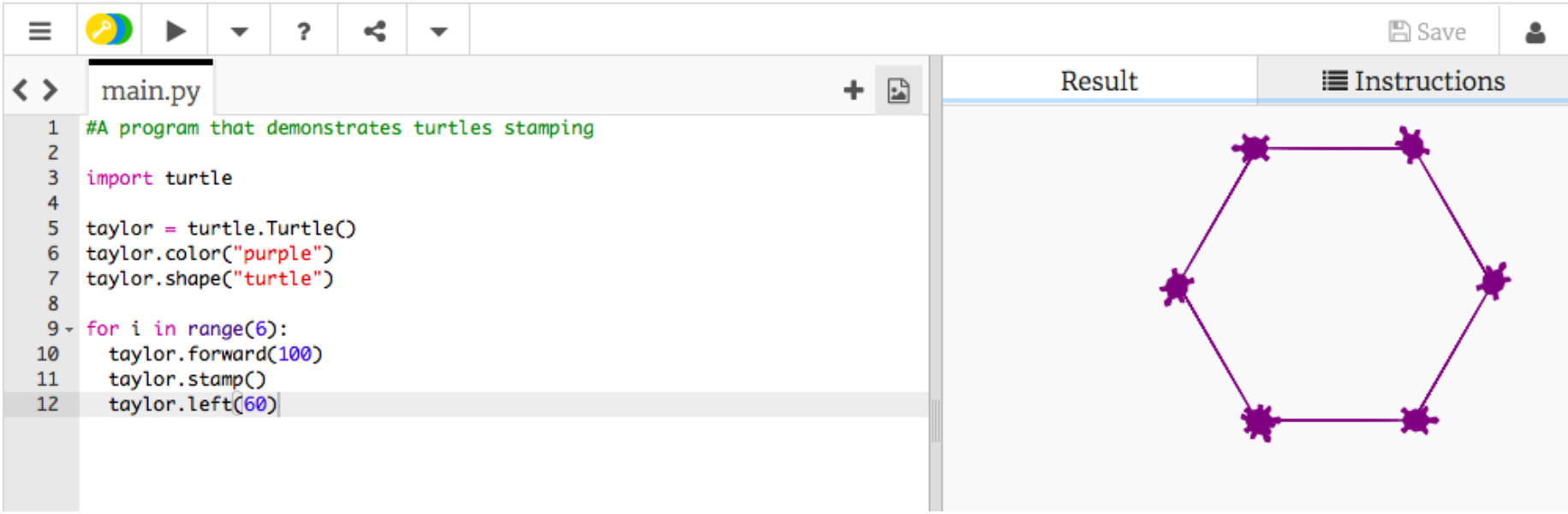
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).

Turtles Introduction



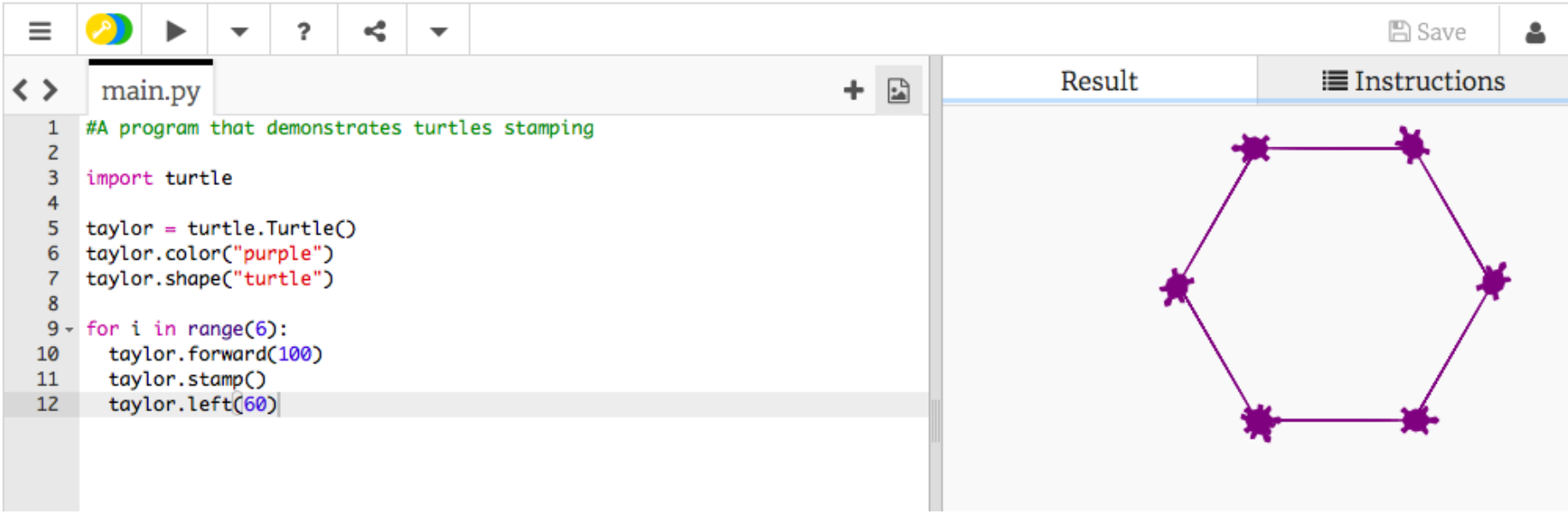
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:

Turtles Introduction



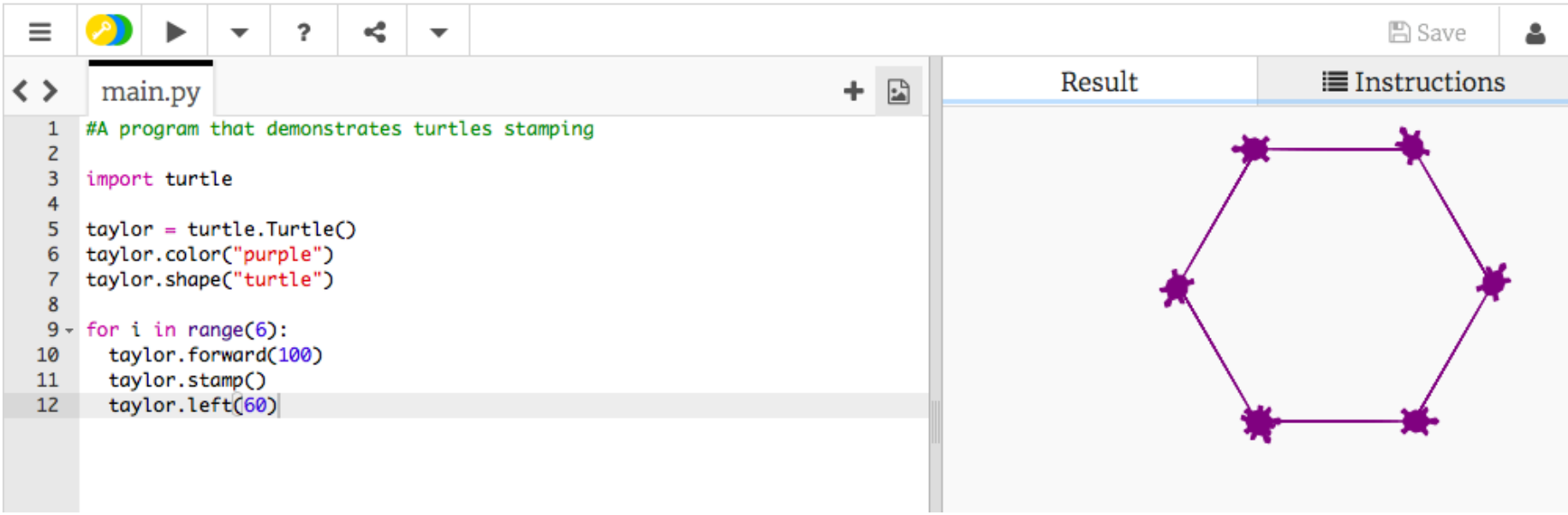
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
  - Move forward; stamp; and turn left 60 degrees.

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

Group Work

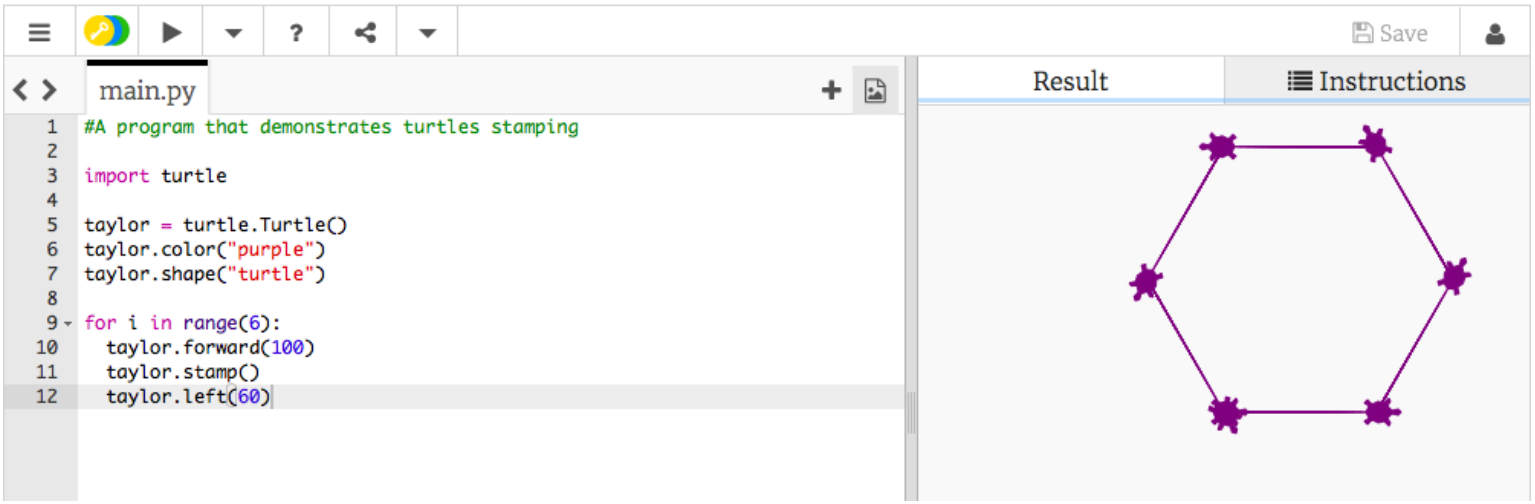
Working in pairs or triples:

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!
   ```

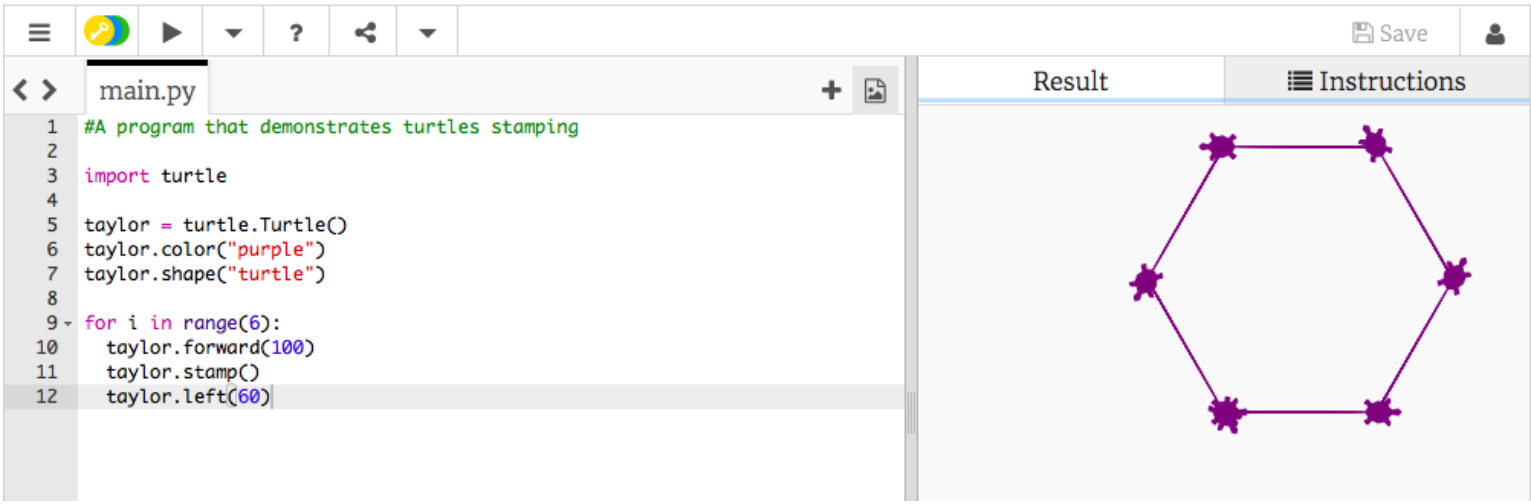
   three times.

Decagon Program



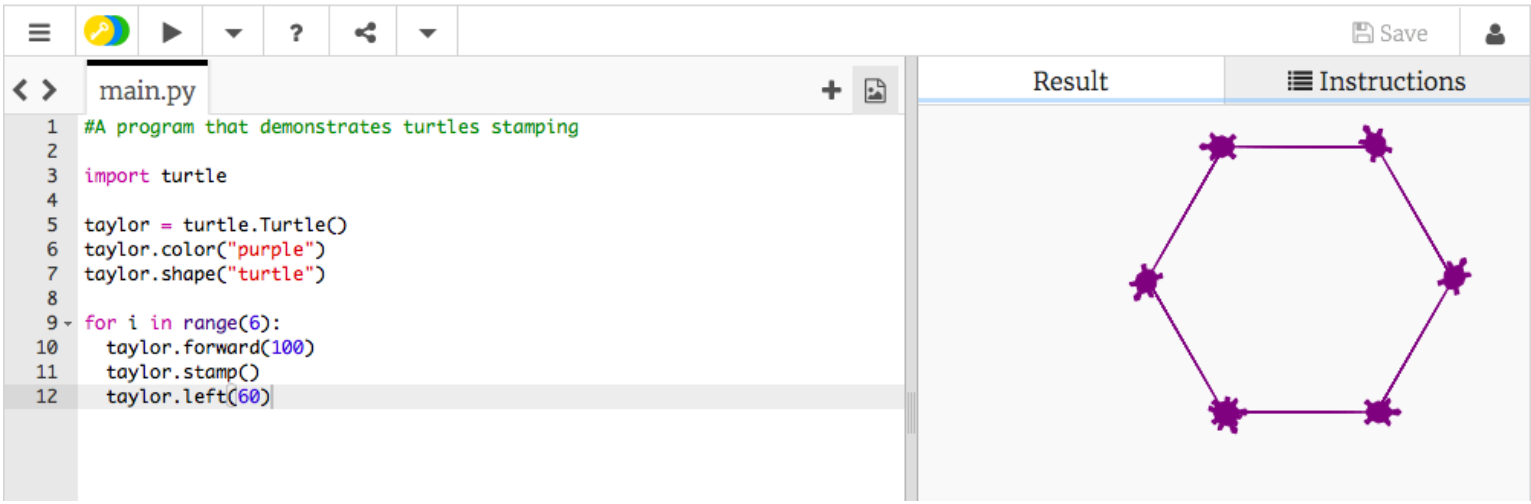
- Start with the hexagon program.

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!")
  ```

Today's Topics

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

What is an Algorithm?

From our textbook:

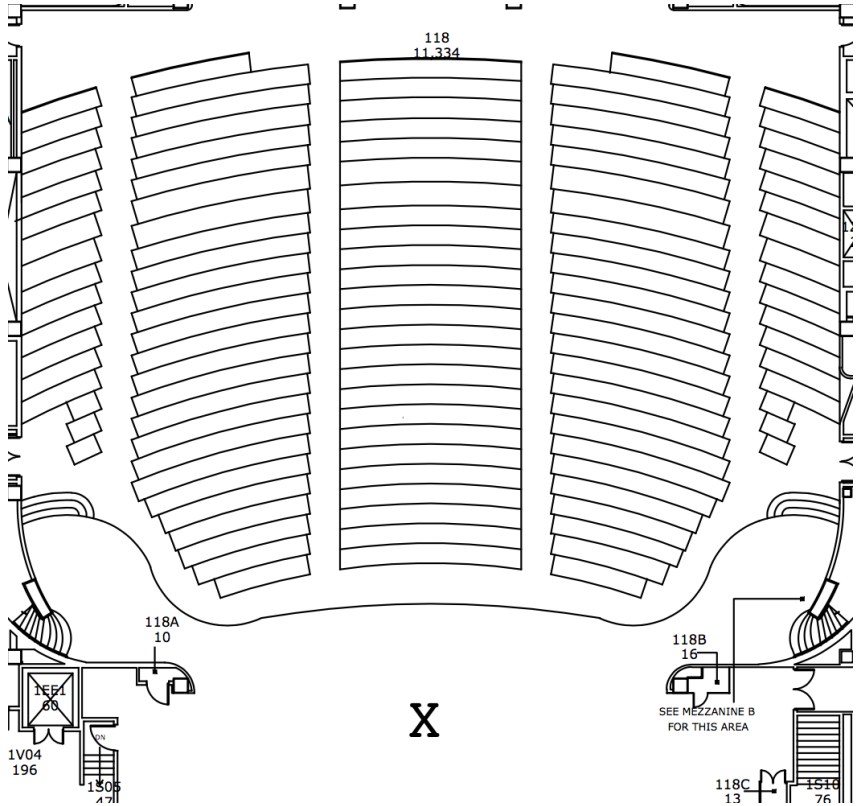
- An algorithm is a process or sequence of steps to be followed to solve a problem.

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.

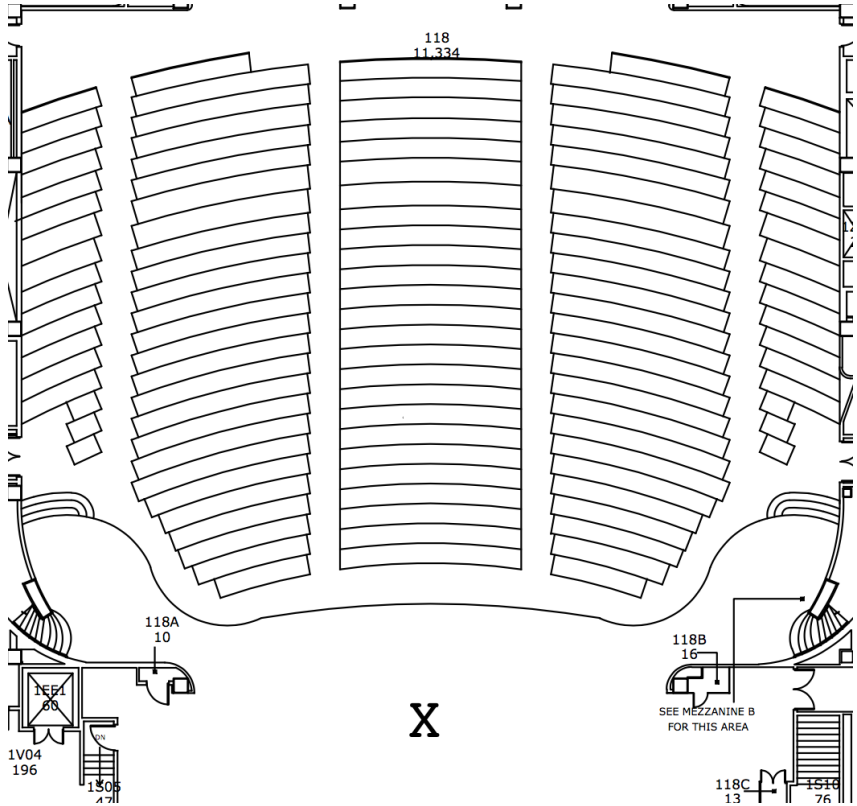
Group Work



Working in pairs or triples:

1. On the floorplan, mark your current location.
2. Write an algorithm (step-by-step directions) to get to `X`.

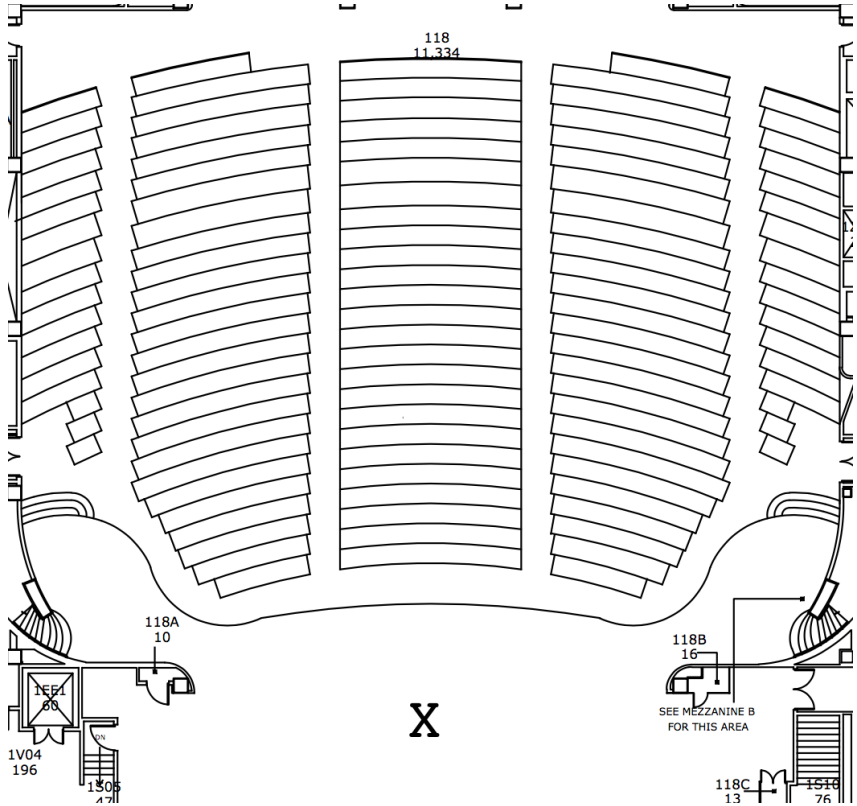
Group Work



Working in pairs or triples:

1. On the floorplan, mark your current location.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:

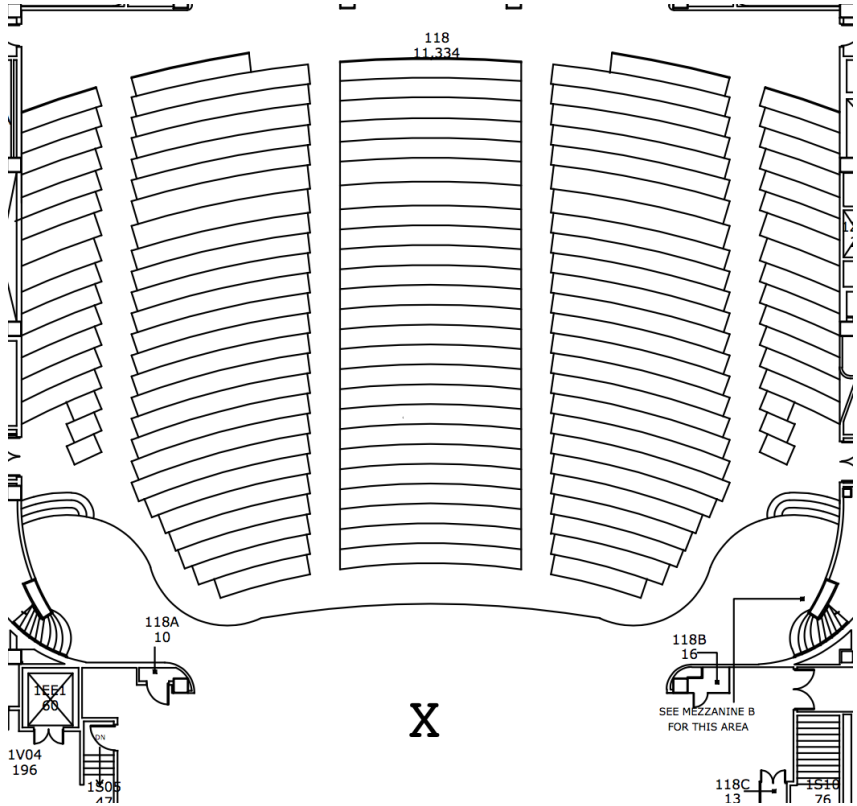
Group Work



Working in pairs or triples:

1. On the floorplan, mark your current location.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:
   - Use `turtle` commands.

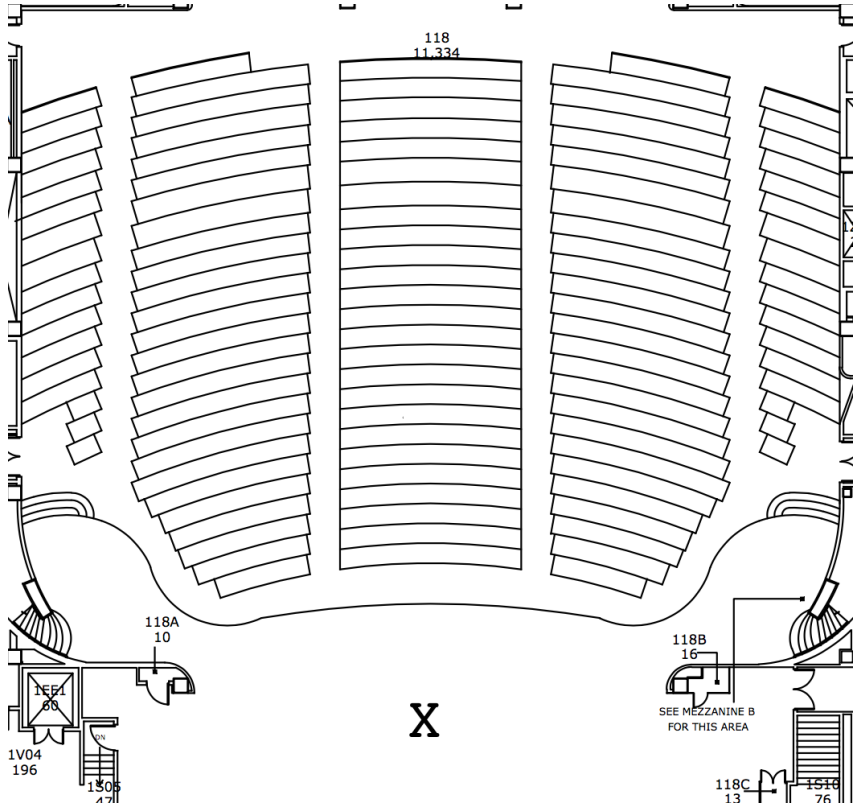
Group Work



Working in pairs or triples:

1. On the floorplan, mark your current location.
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.

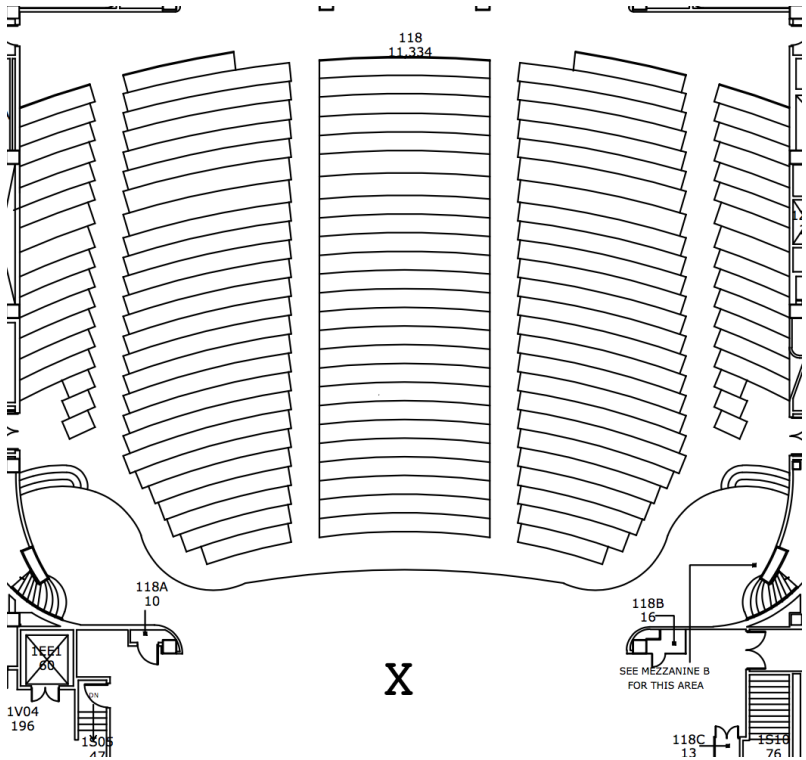
Group Work



Working in pairs or triples:

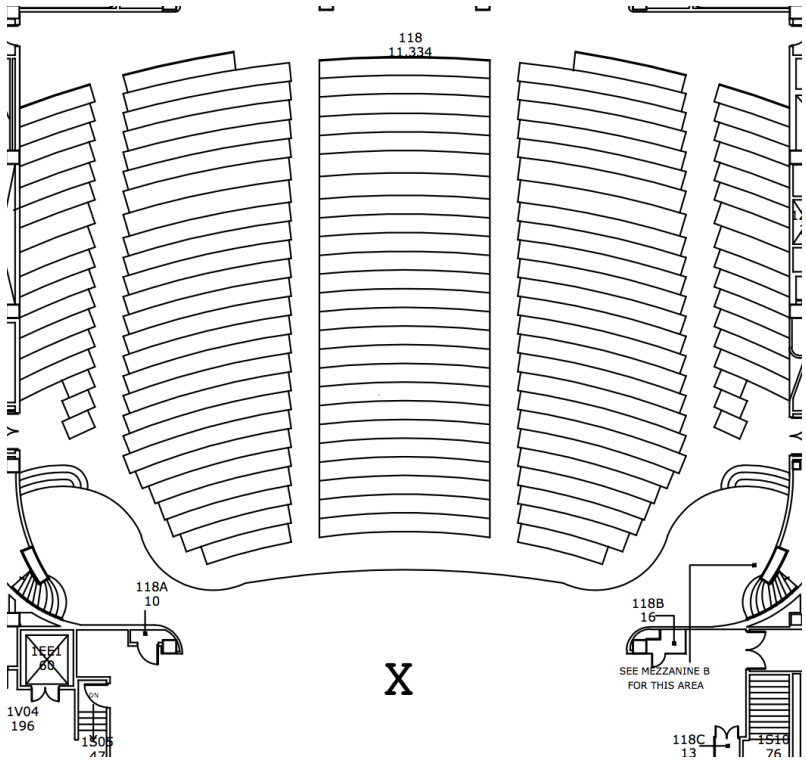
1. On the floorplan, mark your current location.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:
   - Use `turtle` commands.
   - Do not run turtles into walls, chairs, obstacles, etc.
   - Turtles cannot climb walls, must use stairs.

Group Work



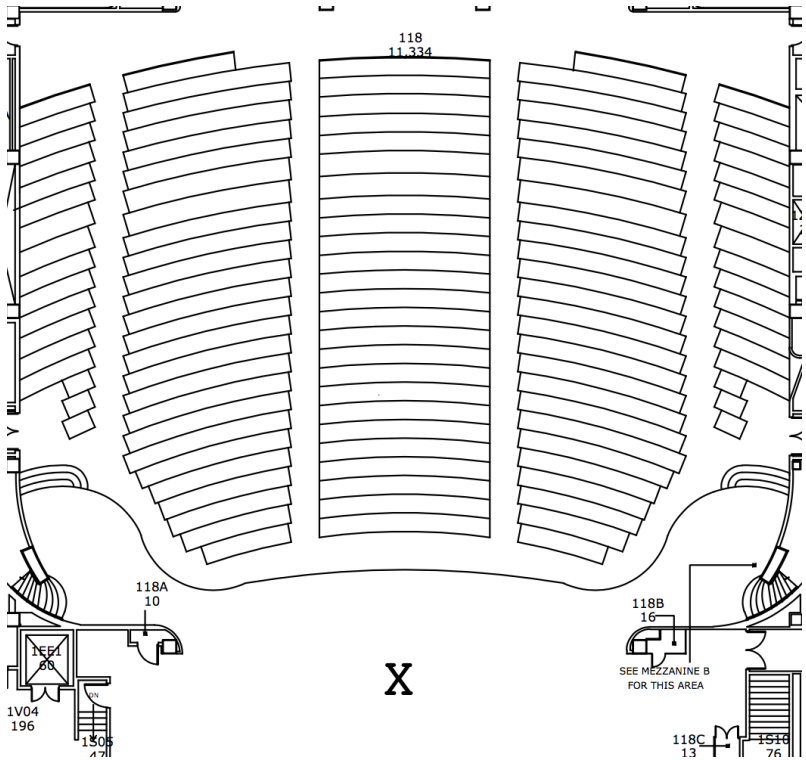
- Have one person in your group be the "turtle."

Group Work



- Have one person in your group be the "turtle."
- Follow the directions to get to `X`.

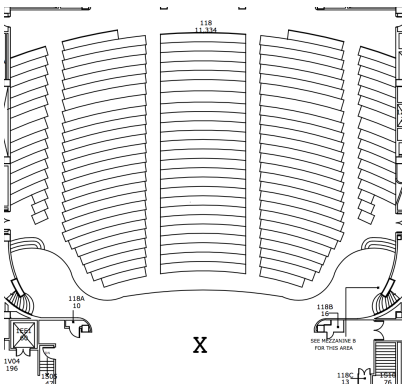
Group Work



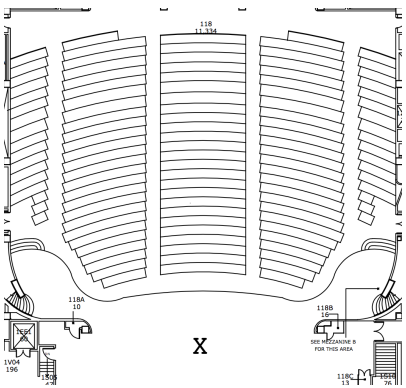
- Have one person in your group be the "turtle."
- Follow the directions to get to `X`.
- Annotate any changes needed to the directions (i.e. debug your work).

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).

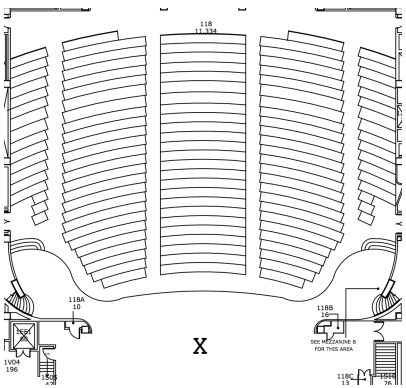


Recap



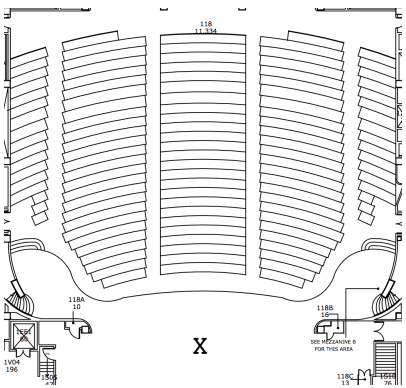
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.

Recap



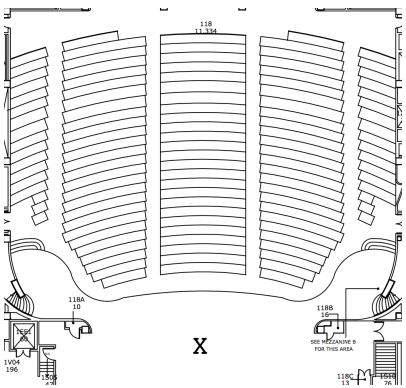
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



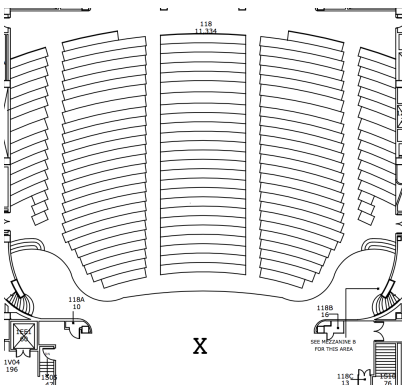
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,

Recap



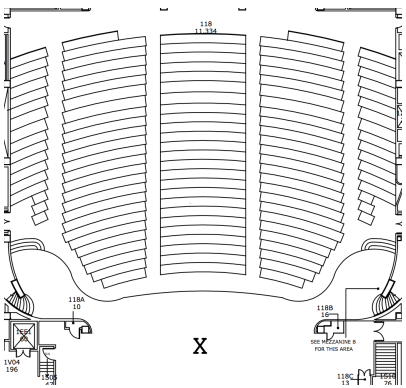
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,

Recap



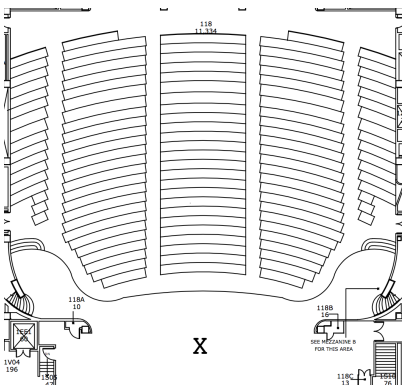
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &

Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- 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



- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &
  - variables containing turtles.
- Pass your lecture slips to the aisle for the UTA's to collect.

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
- Schedule an appointment to take the Quiz in lab 1001G Hunter North

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- 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
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (programs 1-5)
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30pm

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments (programs 1-5)
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)

Lecture Slips & Writing Boards

- Hand your lecture slip to a UTA
- Return writing boards as you leave.