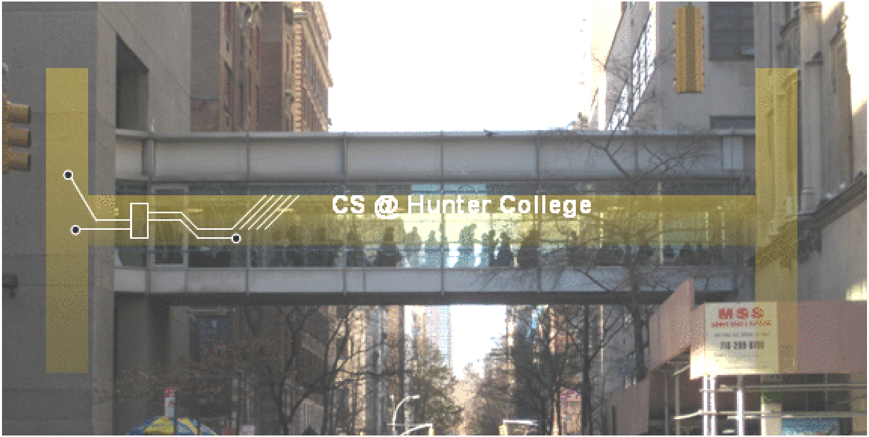
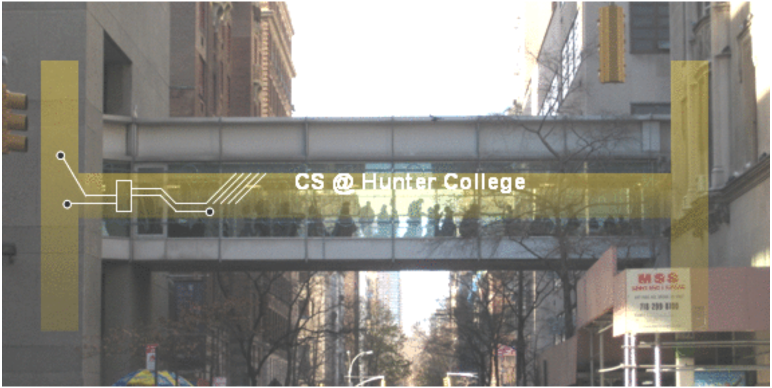
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Welcome



- This lecture will be recorded

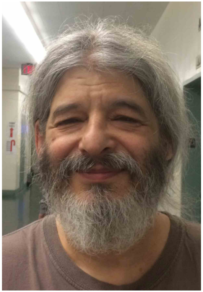
Introductions: Course Designers

Dr. Katherine St. John

Professor,
Interim Chair

Dr. William Sakas

Associate Professor,
Chair



Prof. Eric Schweitzer

Undergraduate Program
Coordinator

Introductions: Instructors

Lola Samigjonova

Early College
Initiative

Dr. Tiziana Ligorio

Large Lecture
Course Coordinator

Introductions: Undergraduate Teaching Assistants

Aida Jevric

Andrew Robinson

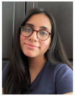
Arterio Rodrigues

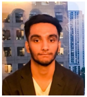
Bahtija Durakovic

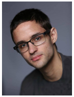
Christopher Asma

David Lin

Destiny Barbery


Diana Luna

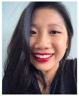

Ghazanfar Shahbaz


ilya Baburashvili

Jessie Lin

Leonardo Matone

Mandy Yu


Nancy Ng

Omer Skaljic

Roziena Badree

Sadab Hafiz

Seth Spiegel

Sheikh Fuad

Stephanie Yung

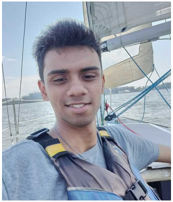
Syeda Nahar

Tyler Robinson

Yash Mahtani

Yoomin Song

Introductions: Autograder Programmers

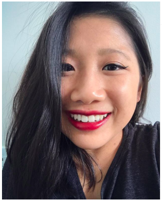


Ifte Ahmed

Leonardo Matone

Lola Samigjonova

Mandy Yu

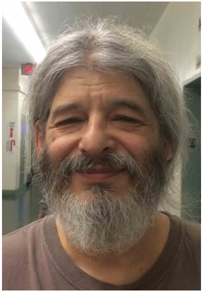


Nancy Ng

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/f21.html

Where to find Course Content

- Course Website: https://huntercsci127.github.io/f21.html
- Blackboard

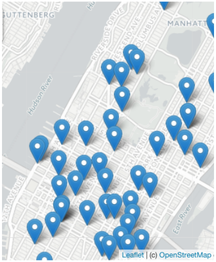
Where to find Course Content

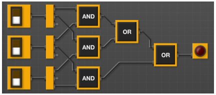
- Course Website: https://huntercsci127.github.io/f21.html
- Blackboard
- Gradescope (program submission)

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.

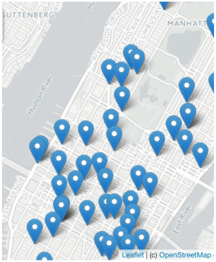
Syllabus: Topics

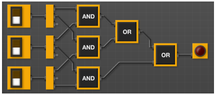




- **This course assumes no previous programming experience.**

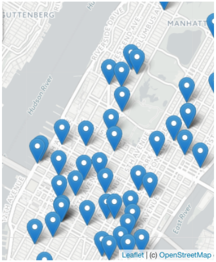
Syllabus: Topics

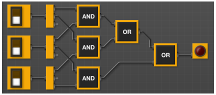




- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:

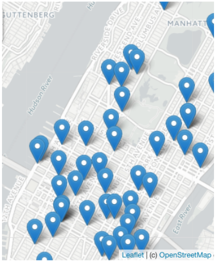
Syllabus: Topics

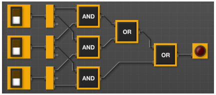




- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,

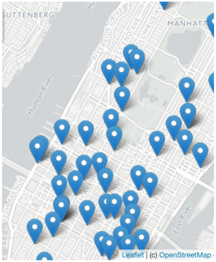
Syllabus: Topics

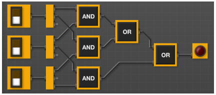




- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),

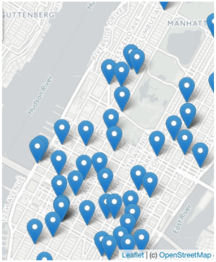
Syllabus: Topics



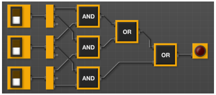


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:

Syllabus: Topics

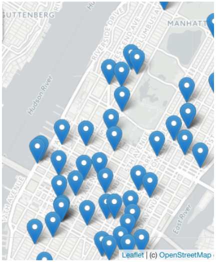


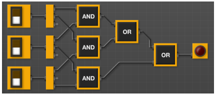
Leaflet | (c) OpenStreetMap



- **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,

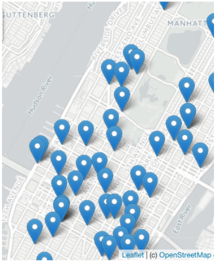
Syllabus: Topics

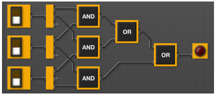




- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - for logical circuits,
    - for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

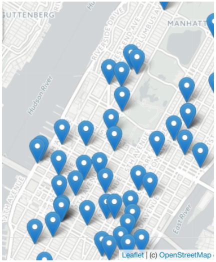
Syllabus: Topics

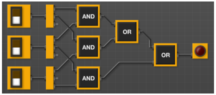




- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - for logical circuits,
    - for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

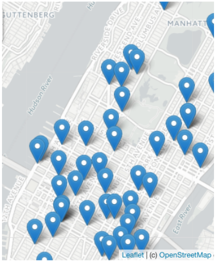
Syllabus: Topics

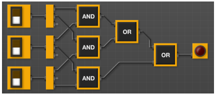




- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

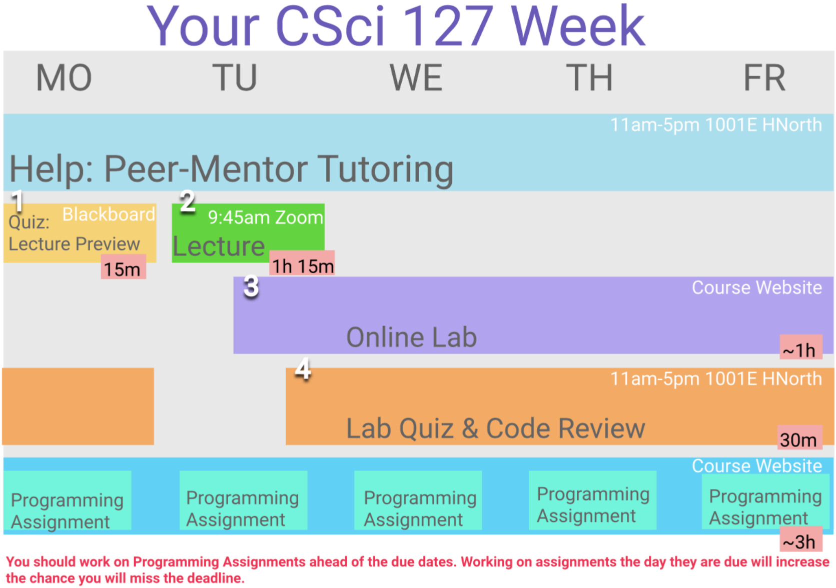
Syllabus: Topics





- **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



Your CSci 127 Week

MO TU WE TH FR

11am-5pm 1001E HNorth

Help: Peer-Mentor Tutoring

1 Quiz: Blackboard Lecture Preview 15m

2 9:45am Zoom Lecture 1h 15m

3 Course Website Online Lab ~1h

4 11am-5pm 1001E HNorth Lab Quiz & Code Review 30m

Course Website

Programming Assignment

Programming Assignment

Programming Assignment

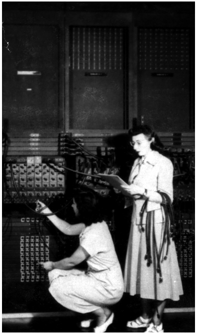
Programming Assignment

Programming Assignment

~3h

You should work on Programming Assignments ahead of the due dates. Working on assignments the day they are due will increase the chance you will miss the deadline.

1&2 - Lecture

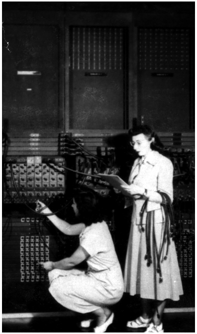


First "computers"

ENIAC, 1945.

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

1&2 - Lecture

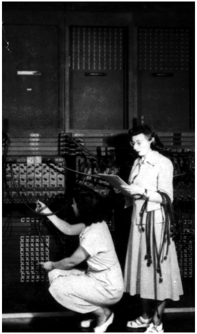


First "computers"

ENIAC, 1945.

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

1&2 - Lecture

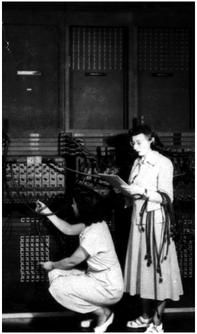


First "computers"

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).

1&2 - Lecture

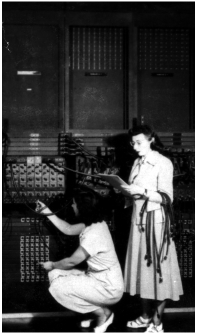


First "computers"

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.

1&2 - Lecture

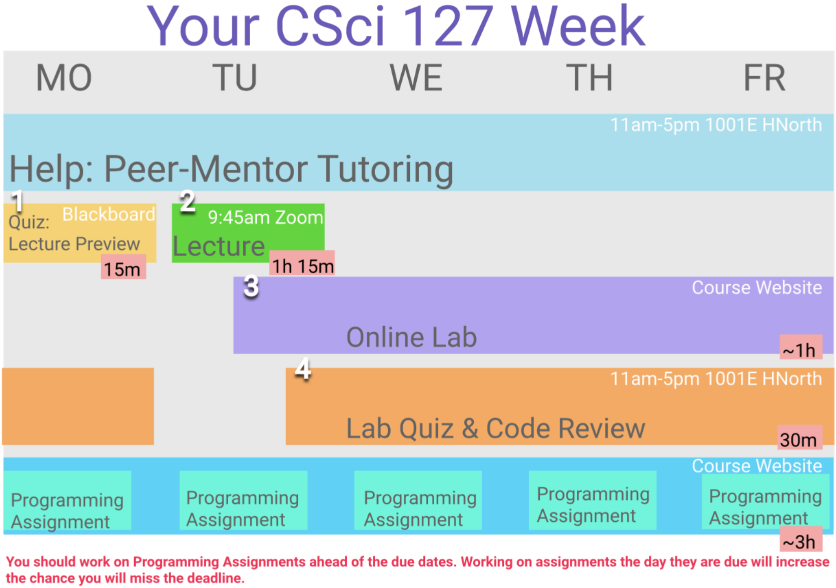


First "computers"

ENIAC, 1945.

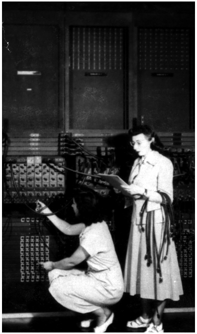
- Tuesdays, 9:45-11:00am, on Zoom.
- Mix of explanation, challenges & active concept application.
- Lecture Preview: 15 minutes Quiz on Blackboard **prior** to each lecture (opens on Mondays).
- Lecture Quiz: on Gradescope during lecture.
- Ask questions in Q&A.

Course Structure



You should work on Programming Assignments ahead of the due dates. Working on assignments the day they are due will increase the chance you will miss the deadline.

3 - Online Lab



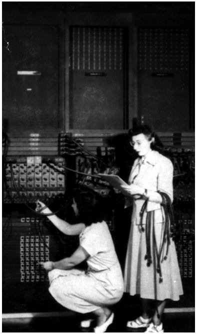
First "computers"

ENIAC, 1945.

Each Week:

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

3 - Online Lab



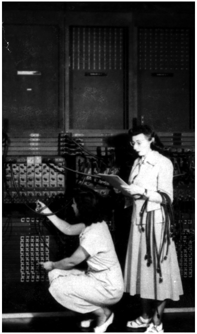
First "computers"

ENIAC, 1945.

Each Week:

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

3 - Online Lab



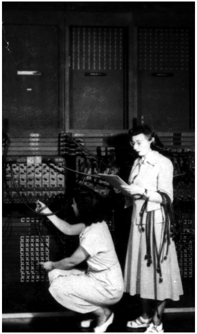
First "computers"

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.

3 - Online Lab



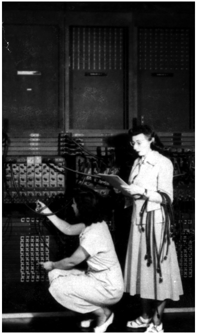
First "computers"

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.
- Lab content directly supports weekly programming assignments.

3 - Online Lab



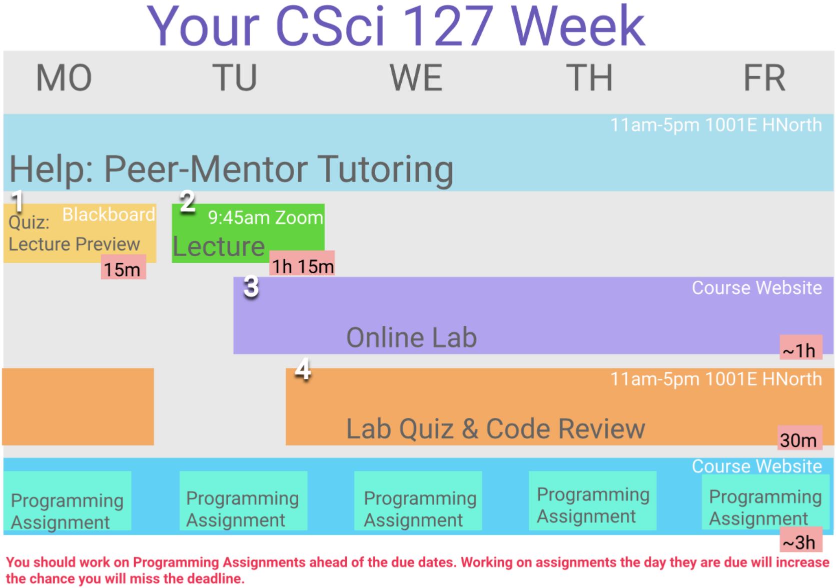
First "computers"

ENIAC, 1945.

Each Week:

- **You must independently read through the weekly online Lab.**
- Replaces scheduled recitation meeting.
- 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 (Handouts column in Course Outline)

Course Structure



Your CSci 127 Week

MO TU WE TH FR

Help: Peer-Mentor Tutoring — 11am-5pm 1001E HNorth

1 Quiz: Lecture Preview — Blackboard — 15m

2 Lecture — 9:45am Zoom — 1h 15m

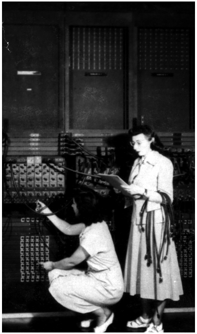
3 Online Lab — Course Website — ~1h

4 Lab Quiz & Code Review — 11am-5pm 1001E HNorth — 30m

Programming Assignment · Programming Assignment · Programming Assignment · Programming Assignment · Programming Assignment — Course Website — ~3h

You should work on Programming Assignments ahead of the due dates. Working on assignments the day they are due will increase the chance you will miss the deadline.

4 -In-person Quiz & Code Review

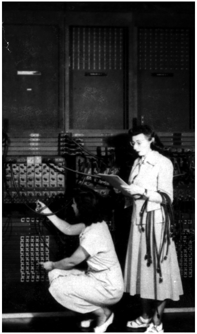


First "computers"

ENIAC, 1945.

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

4 -In-person Quiz & Code Review

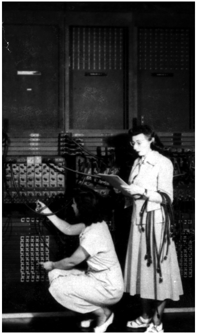


First "computers"

ENIAC, 1945.

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

4 -In-person Quiz & Code Review

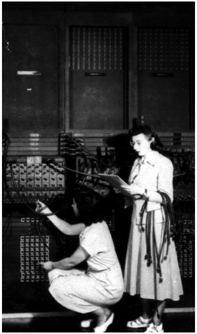


First "computers"

ENIAC, 1945.

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

4 -In-person Quiz & Code Review

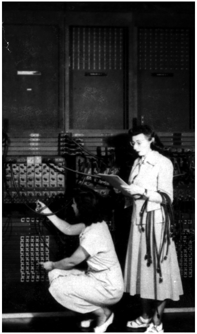


First "computers"

ENIAC, 1945.

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

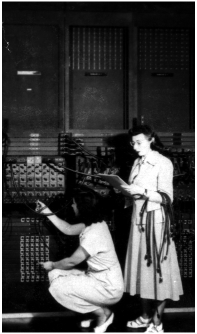


First "computers"

ENIAC, 1945.

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

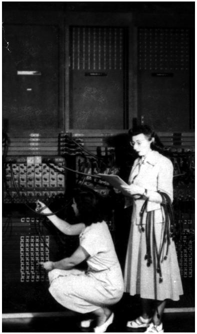


First "computers"

ENIAC, 1945.

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

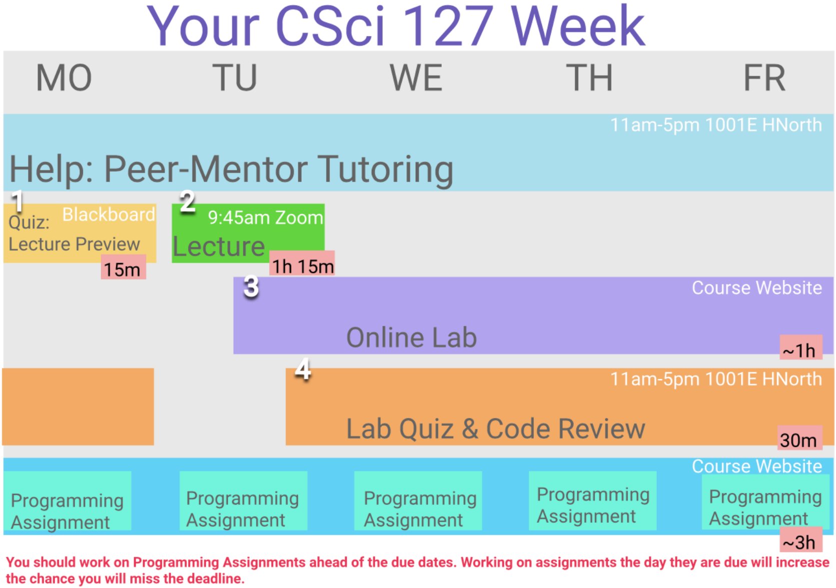


First "computers"

ENIAC, 1945.

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

Course Structure



Your CSci 127 Week

MO TU WE TH FR

11am-5pm 1001E HNorth

Help: Peer-Mentor Tutoring

1 Quiz: Blackboard
Lecture Preview
15m

2 9:45am Zoom
Lecture
1h 15m

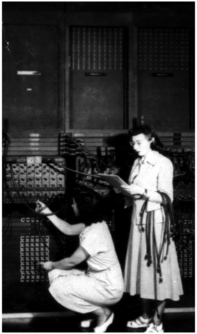
3 Course Website
Online Lab
~1h

4 11am-5pm 1001E HNorth
Lab Quiz & Code Review
30m

Course Website
Programming Assignment
Programming Assignment
Programming Assignment
Programming Assignment
Programming Assignment
~3h

You should work on Programming Assignments ahead of the due dates. Working on assignments the day they are due will increase the chance you will miss the deadline.

Homework



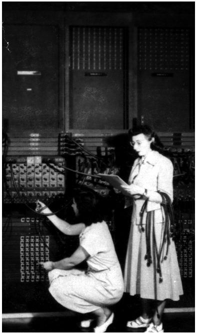
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!

Homework



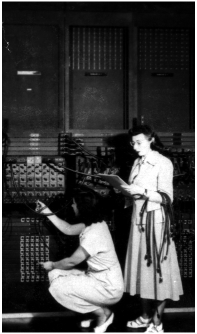
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**

Homework



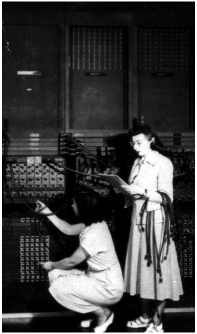
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**
- Work ahead!!! Students who work on programs on the due date often miss the deadline!

Homework



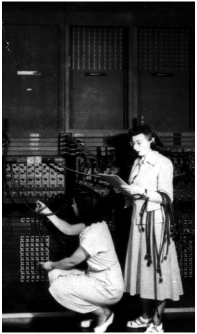
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**
- Work ahead!!! Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.

Homework



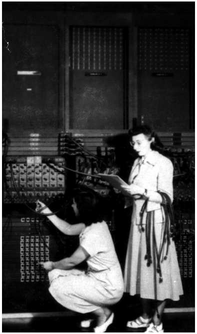
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**
- 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.

Homework



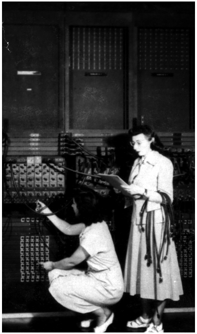
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**
- 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.

Homework



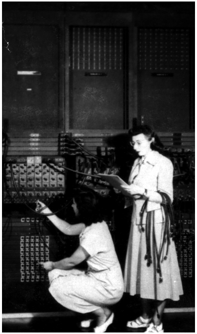
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**
- 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.

Homework



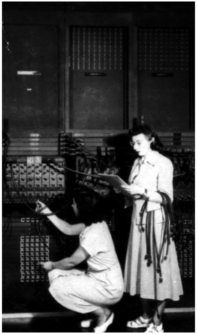
First "computers"

ENIAC, 1945.

Each Week:

- Starting September 13, there will be one program due each day!
- **5 Programming Assignments.**
- 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 1001E lab.

Make Your Schedule!

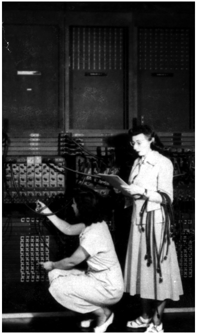


First "computers"

ENIAC, 1945.

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

Make Your Schedule!

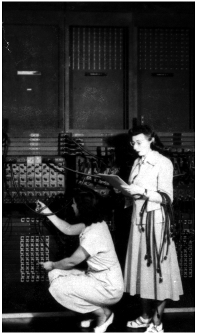


First "computers"

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**.

Make Your Schedule!

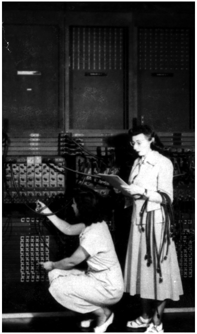


First "computers"

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**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!

Make Your Schedule!

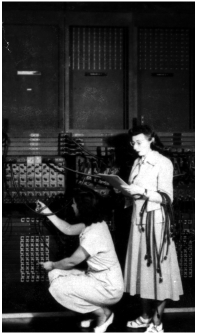


First "computers"

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**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!
- Schedule a regular time for working on **programming assignments**.

Make Your Schedule!

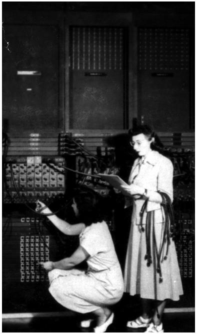


First "computers"

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**.
- 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!

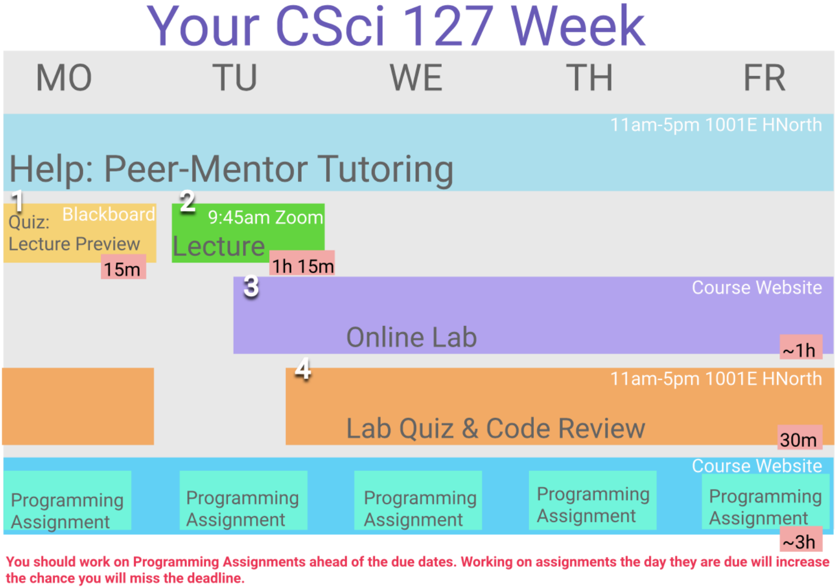


First "computers"

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**.
- 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.

Course Structure



Your CSci 127 Week

MO TU WE TH FR

Help: Peer-Mentor Tutoring — 11am-5pm 1001E HNorth

1 Quiz: Lecture Preview — Blackboard — 15m

2 Lecture — 9:45am Zoom — 1h 15m

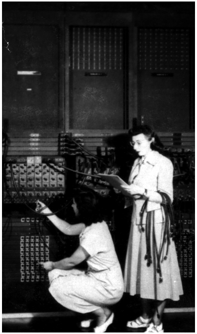
3 Online Lab — Course Website — ~1h

4 Lab Quiz & Code Review — 11am-5pm 1001E HNorth — 30m

Programming Assignment — Programming Assignment — Programming Assignment — Programming Assignment — Programming Assignment — Course Website — ~3h

You should work on Programming Assignments ahead of the due dates. Working on assignments the day they are due will increase the chance you will miss the deadline.

Help and Support

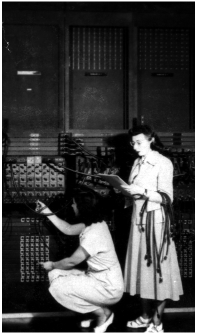


First "computers"

ENIAC, 1945.

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

Help and Support

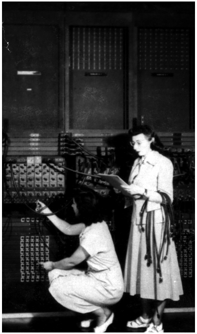


First "computers"

ENIAC, 1945.

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

Help and Support

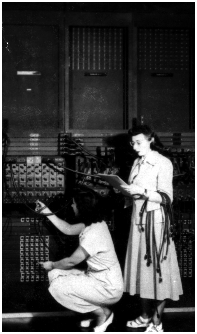


First "computers"

ENIAC, 1945.

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

Help and Support

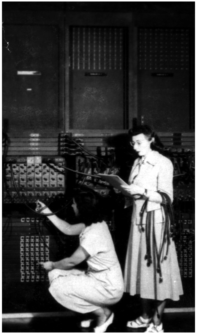


First "computers"

ENIAC, 1945.

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

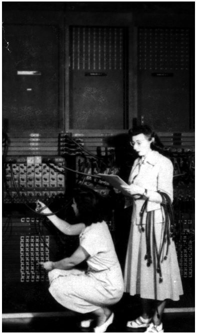


First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001E 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 11am-5pm** when classes are in session

Help and Support



First "computers"

ENIAC, 1945.

- Peer-mentor Support (UTAs)
  - **Tutoring**: in-person tutoring and programming help in 1001E 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 11am-5pm** when classes are in session
- Office Hours with Prof. Ligorio
  - Drop-in Hours: **Tuesday 11am-1pm**
  - Zoom link on Blackboard under *L*ecture & Recordings

Benefits of Tutoring and Code Review

Explain Technical Concepts

Develop your Technical Vocabulary

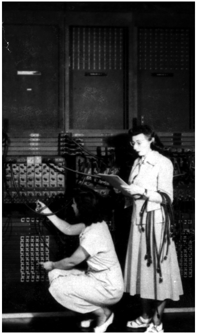
Expert Help

Procedural Thought Organization

Learn Debugging

Build Community

Academic Dishonesty

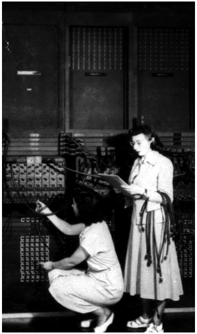


First "computers"

ENIAC, 1945.

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

Academic Dishonesty

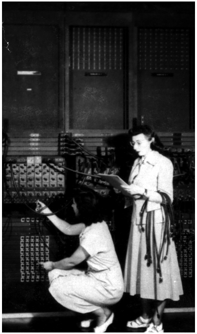


First "computers"

ENIAC, 1945.

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

Academic Dishonesty

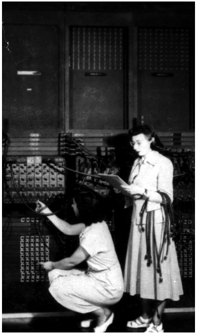


First "computers"

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

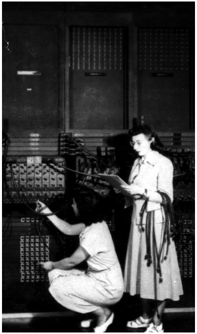


First "computers"

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

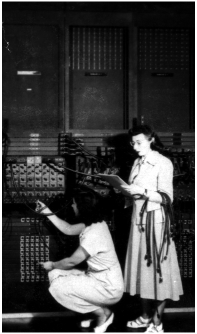


First "computers"

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

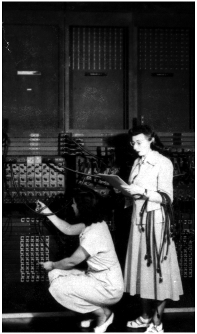


First "computers"

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

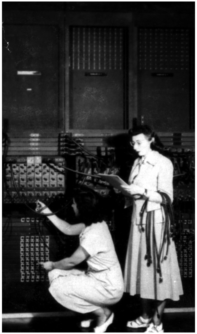


First "computers"

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

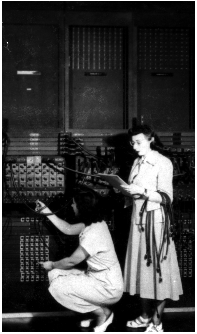


First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard

Communication

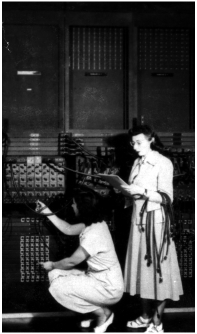


First "computers"

ENIAC, 1945.

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

Communication

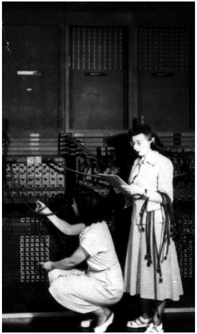


First "computers"

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:  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
#Date:  September 1, 2017
#This program prints:  Hello, World!

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

← These lines are comments

← (for us, not computer to read)

← (this one also)

← 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!`

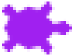
```
#Name:  L-M Miranda
#Date:  Hunter College HS ‘98
#This program prints intro lyrics

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

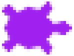
- Introduction to Python
- **Turtle Graphics**
- Definite Loops (`for`-loops)
- Algorithms

Turtles Introduction



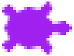
- A simple, whimsical graphics package for Python.

Turtles Introduction



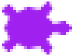
- A simple, whimsical graphics package for Python.
- Dates back to Logo Turtles in the 1960s.

Turtles Introduction



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

Turtles Introduction

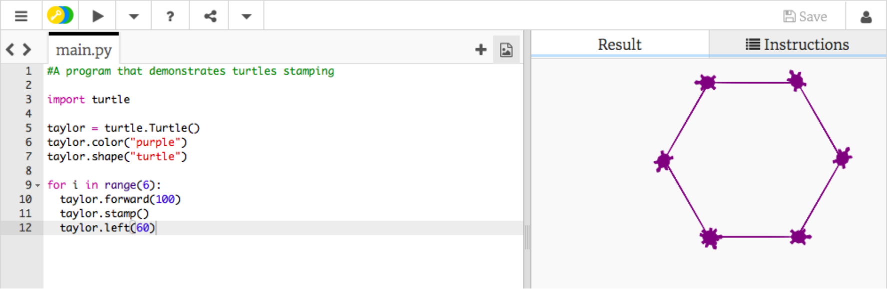


- A simple, whimsical graphics package for Python.
- Dates back to Logo Turtles in the 1960s.
- (Demo from webpage)
- (Fancier turtle demo)

Today's Topics

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

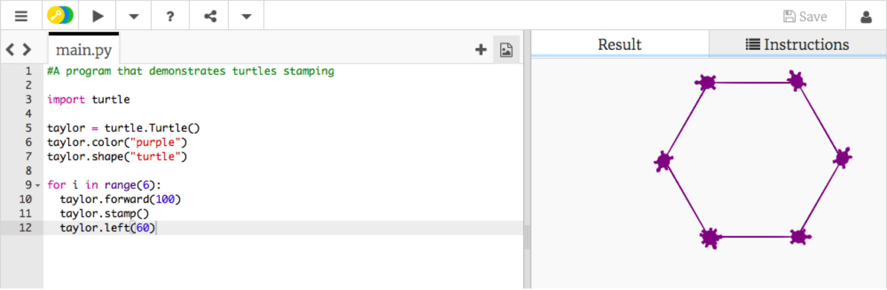
Turtles Introduction



```
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

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

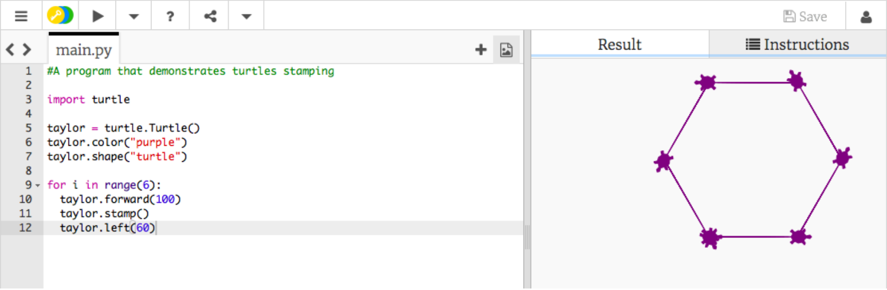
Turtles Introduction



```
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

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

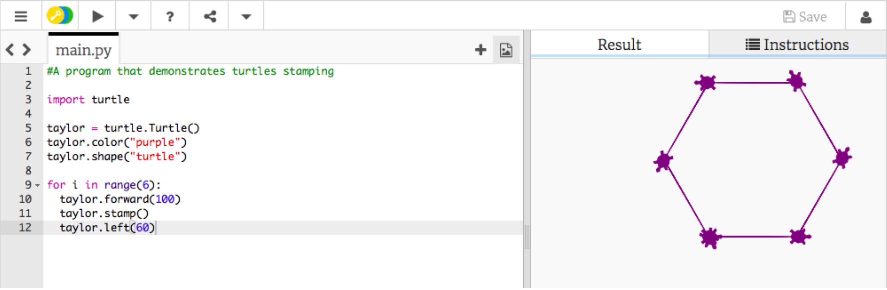
Turtles Introduction



```python
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

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

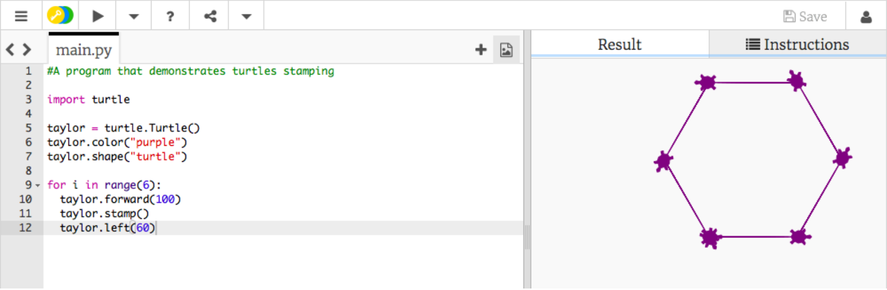
Turtles Introduction



```
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

- 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.

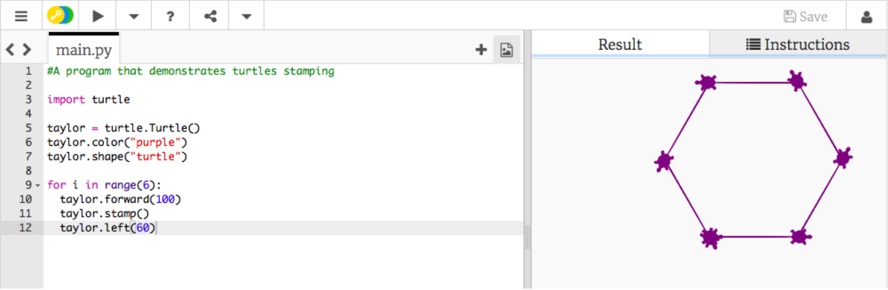
Turtles Introduction



```python
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

- 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



```python
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

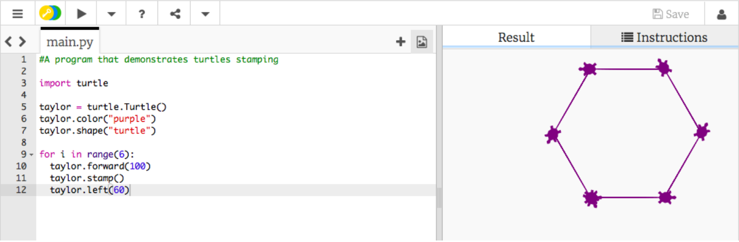
- Creates a turtle **variable**, called `taylor`.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
  - Move forward; stamp; and turn left 60 degrees.
- Repeats any instructions **indented** in the "loop block"
- This is a **definite** loop because it repeats a fixed number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

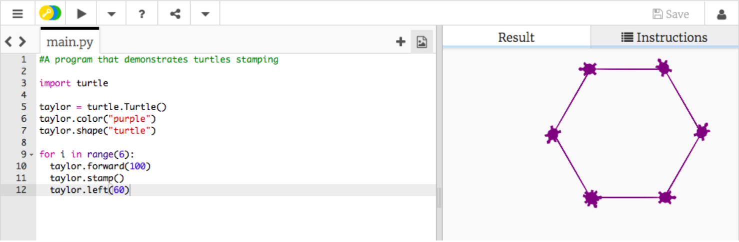
Decagon Program



```
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

- Start with the hexagon program.

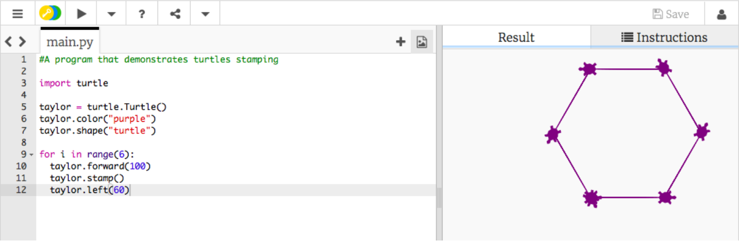
Decagon Program



```
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

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

Decagon Program



```
#A program that demonstrates turtles stamping

import turtle

taylor = turtle.Turtle()
taylor.color("purple")
taylor.shape("turtle")

for i in range(6):
  taylor.forward(100)
  taylor.stamp()
  taylor.left(60)
```

- 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:

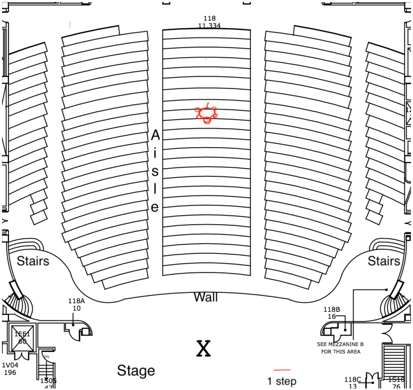
- An algorithm is a process or sequence of steps to be followed to solve a problem.

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.

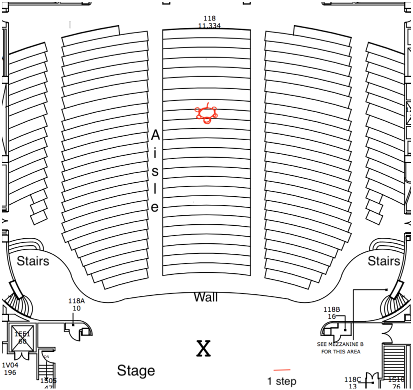
Your Turn!!!



Try to solve this challenge:

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

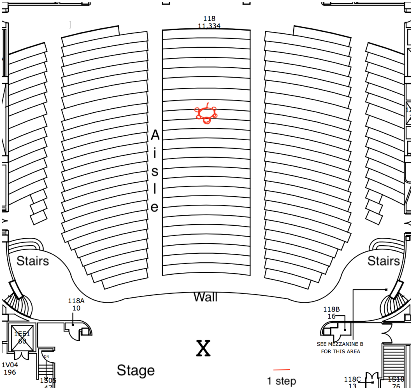
Your Turn!!!



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.

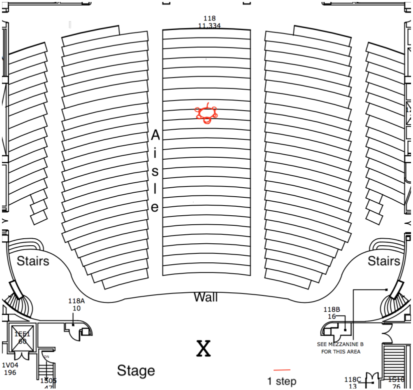
Your Turn!!!



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:

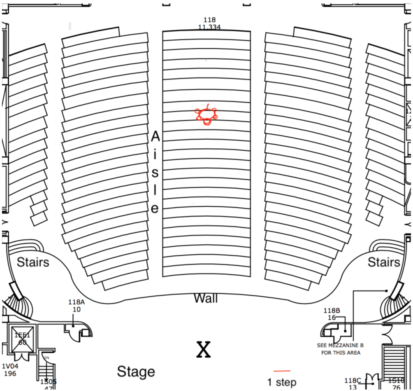
Your Turn!!!



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.

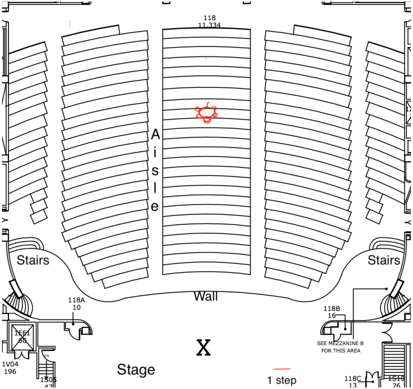
Your Turn!!!



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.

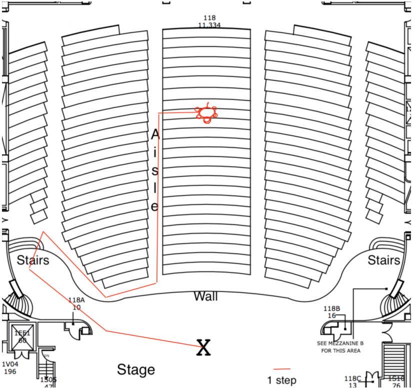
Your Turn!!!



Try to solve this challenge:

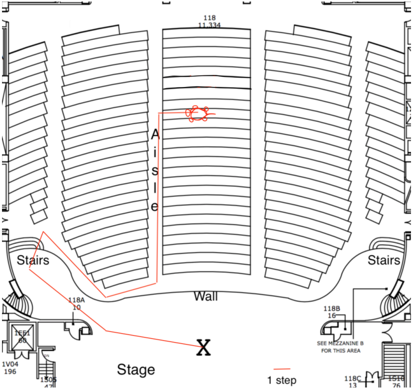
1. This is the floor plan of Assembly Hall at Hunter College.
2. Write an algorithm (step-by-step directions) to the red turtle to the `X` on Stage.
3. Basic Rules:
    - Use `turtle` commands.
    - Do not run turtles into walls, chairs, obstacles, etc.
    - Turtles cannot climb walls, must use stairs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

Your Turn!!!



One possible solution:

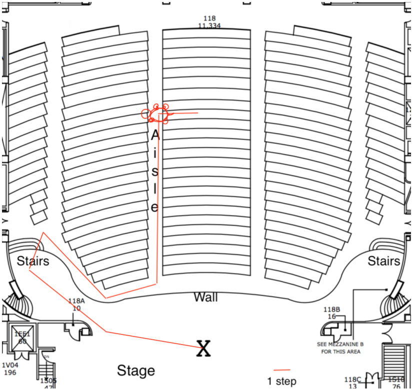
Your Turn!!!



One possible solution:

- Turn right 90 degrees.

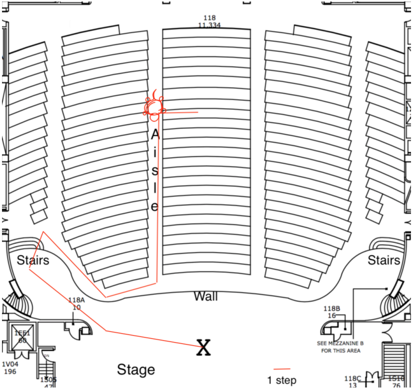
Your Turn!!!



One possible solution:

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

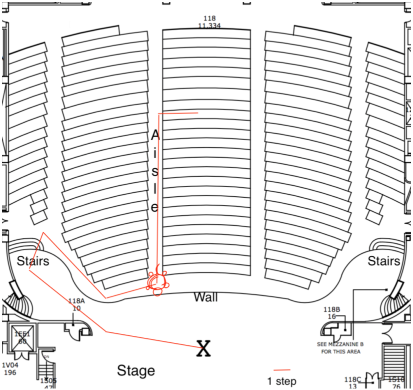
Your Turn!!!



One possible solution:

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

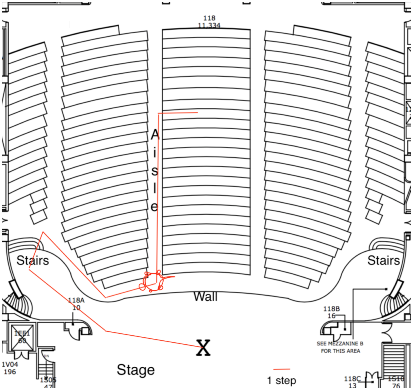
Your Turn!!!



One possible solution:

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

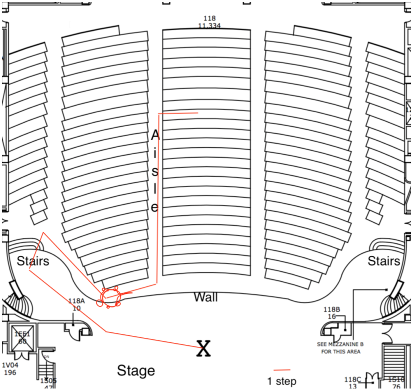
Your Turn!!!



One possible solution:

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

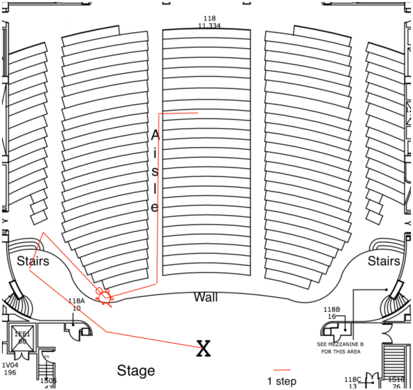
Your Turn!!!



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.

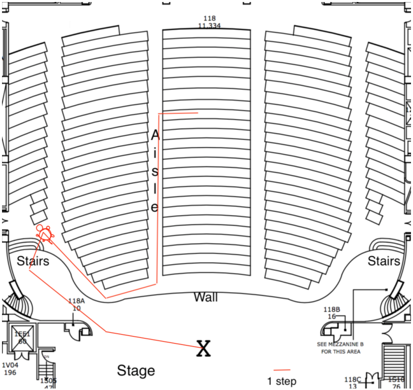
Your Turn!!!



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.

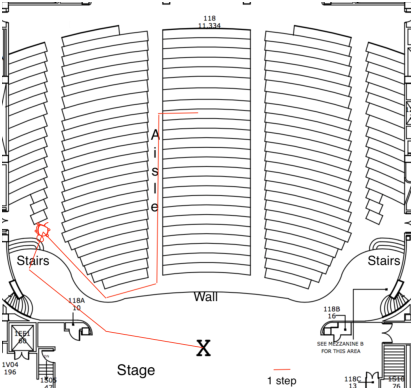
Your Turn!!!



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.

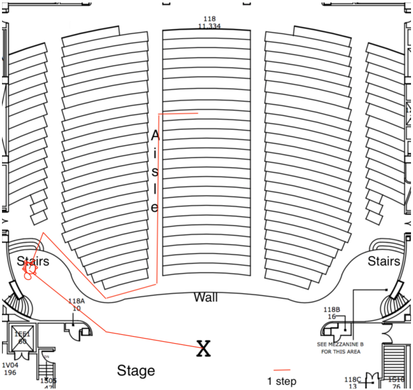
Your Turn!!!



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.

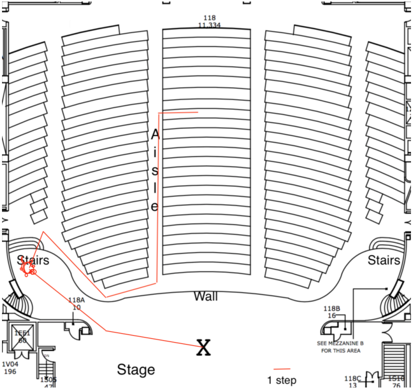
Your Turn!!!



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.

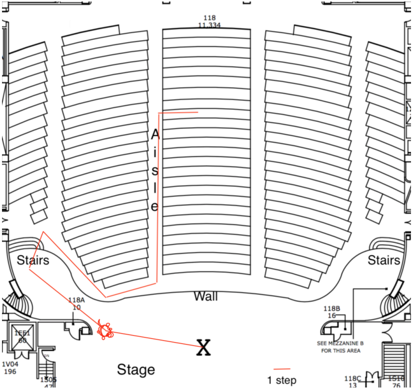
Your Turn!!!



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.

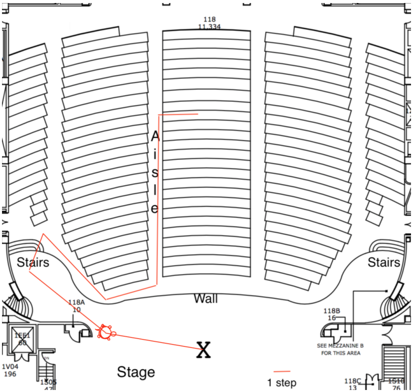
Your Turn!!!



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.

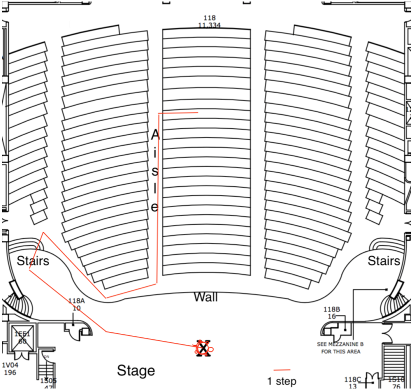
Your Turn!!!



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.

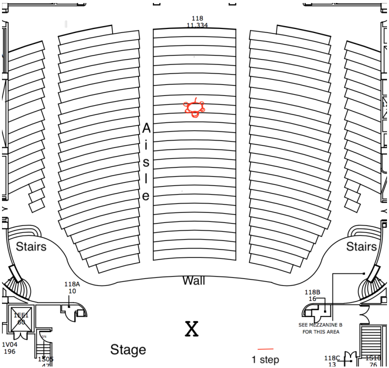
Your Turn!!!



One possible solution:

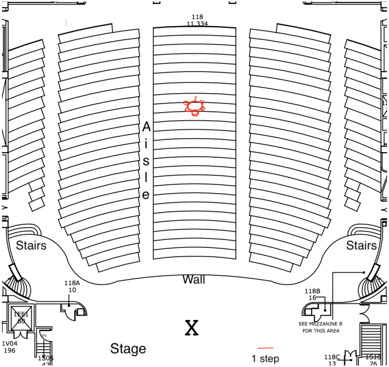
- Turn right 90 degrees.
- Walk forward 3 steps.
- Turn left 90 degrees.
- Walk forward 10 steps.
- Turn right 65 degrees.
- Walk forward 4 steps.
- Turn right 45 degrees.
- Walk forward 6 steps.
- Turn left 110 degrees.
- Walk forward 3 steps.
- Turn left 80 degrees.
- Walk forward 5 steps.
- Turn left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

Your Turn!!!



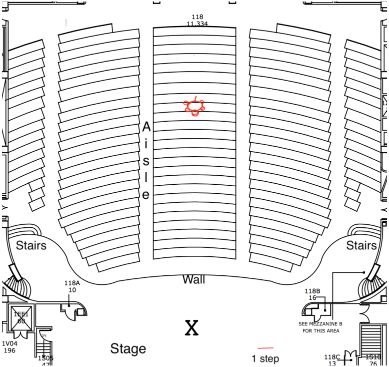
- For fun, post your algorithm on the "Turtle on Stage" forum in the Discussion Board on Blackboard

Your Turn!!!



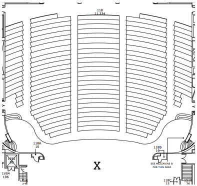
- For fun, post your algorithm on the "Turtle on Stage" forum in the Discussion Board on Blackboard
- "Test and Debug" other students' posted solutions and reply to their posts if you find a bug!

Your Turn!!!



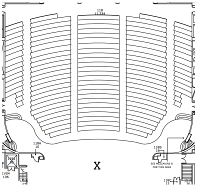
- For fun, post your algorithm on the "Turtle on Stage" forum in the Discussion Board on Blackboard
- "Test and Debug" other students' posted solutions and reply to their posts if you find a bug!
- Degrees the turtle turns are approximate, any good approximation is considered correct.

Recap



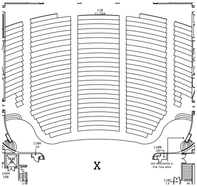
- Writing precise algorithms is difficult.

Recap



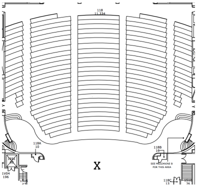
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



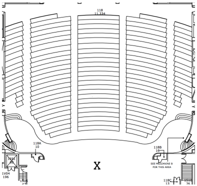
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,

Recap



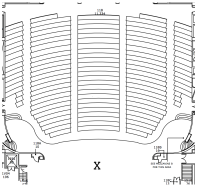
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,

Recap



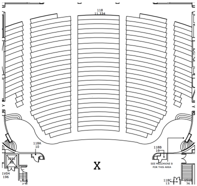
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &

Recap



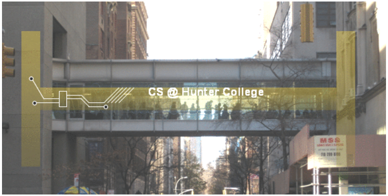
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &
  - variables containing turtles.

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.

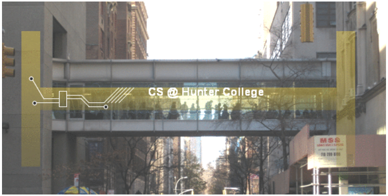
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

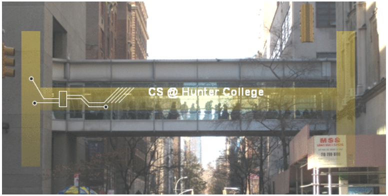
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North

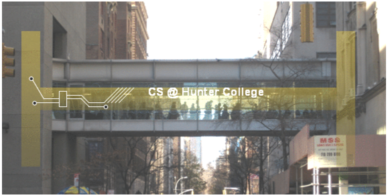
Weekly Reminders!



Before next lecture, don't forget to:

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

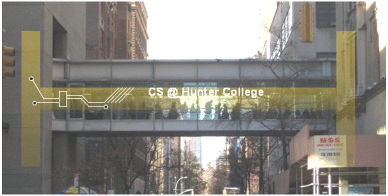
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001E Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**one every two weeks**) in lab 1001E Hunter North
- Submit this week's 5 programming assignments (programs 1-5)

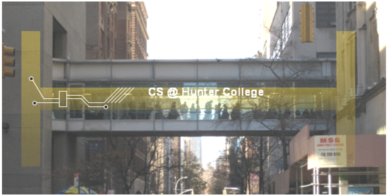
Weekly Reminders!



Before next lecture, don't forget to:

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

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