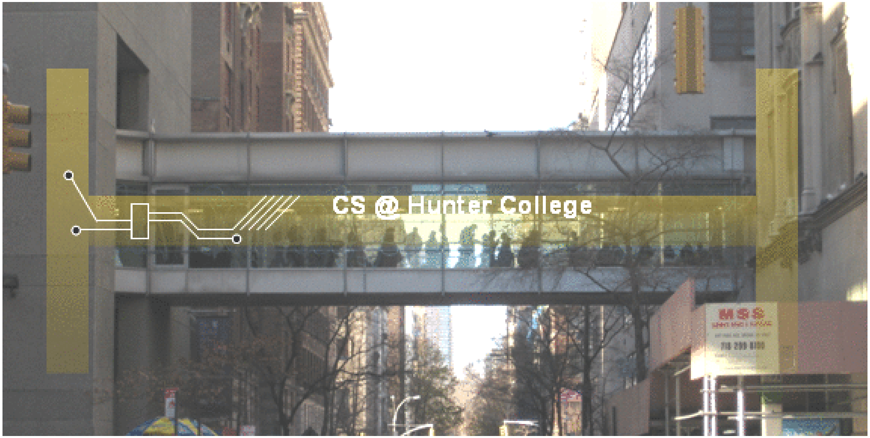
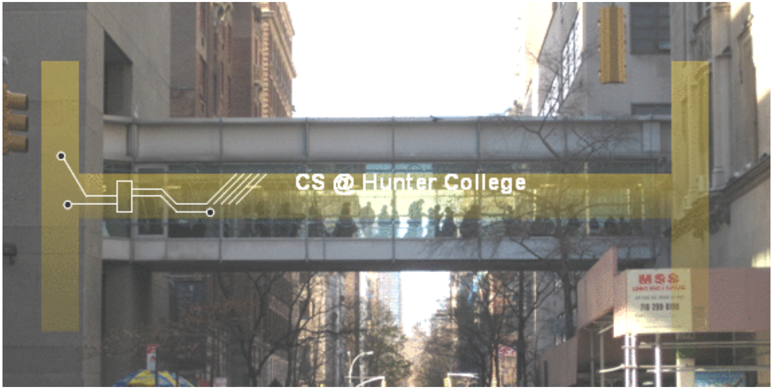
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Welcome



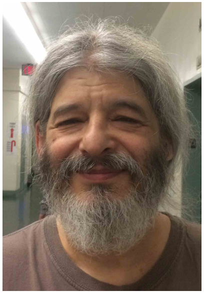
Introductions: Course Designers

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

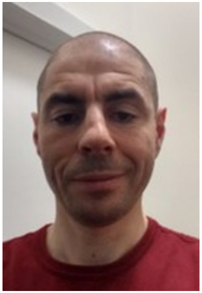
Introductions: Undergraduate Teaching Assistants

Alvin Wu	Anika Sujana	Arsen Tumanian	Arterio Rodrigues	Bode Chiu
Brendan South	Christopher Asma	Diana Luna	Filip Trzcinka	Georgina Woo
Ghazanfar Shahbaz	Hanz De Guzman	Hnin Lwin	Jeffrey Waters	Kazi Mansha
Michelle Thaung	Moududur "Moody" Rahman	Omer Skaljic	Rita Chen	Roy Delgado
Ryan Vaz	Sheikh Fuad	Tyler Robinson	Yoomin Song	Adrian Mysliwiec
Amy Ng	Andy Li	Arshadul Monir	Farhin Bhuiyan	Jessica Flores Olmos
Kevin Perez	Maliha Tasnim	Manuel Reyes	Mashiyat Mahdi	Zeeshan Gondal Ahmed

Introductions: Advisors

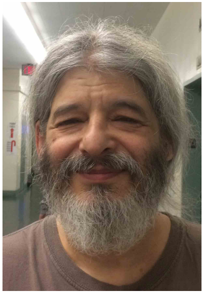
Justing Tojeira
CS Advisor

jtojeira@hunter.cuny.edu



Pavel Shostak
CS Advisor

ps57@hunter.cuny.edu



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

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

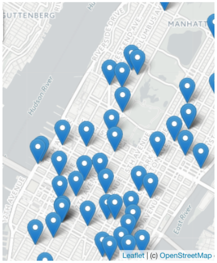
Where to find Course Content

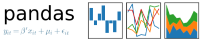
- Course Website: https://huntercsci127.github.io/s23.html
- Blackboard
- Gradescope (program submission)

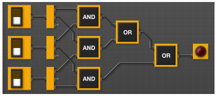
Syllabus

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

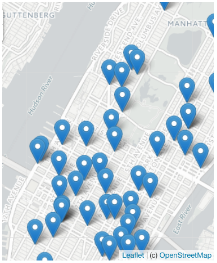


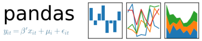


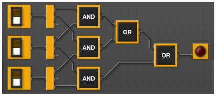


- **This course assumes no previous programming experience.**

Syllabus: Topics

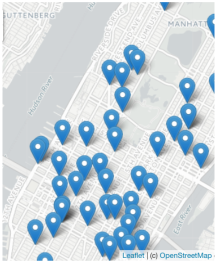


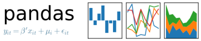


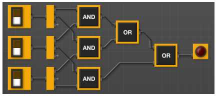


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:

Syllabus: Topics

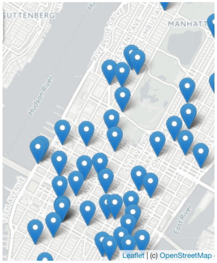


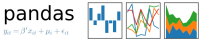


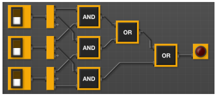


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,

Syllabus: Topics

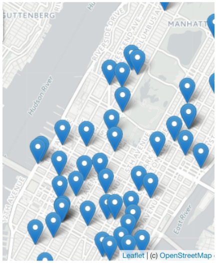


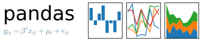


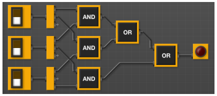


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),

Syllabus: Topics

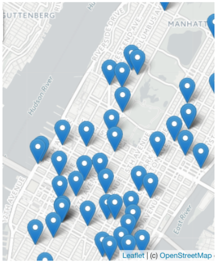


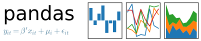


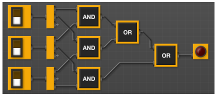


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:

Syllabus: Topics

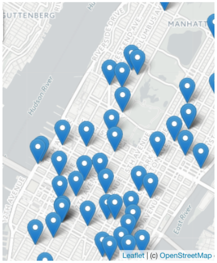


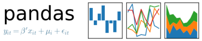


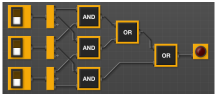


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,

Syllabus: Topics

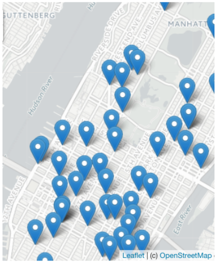


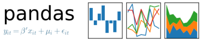


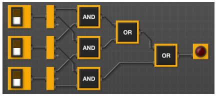


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

Syllabus: Topics

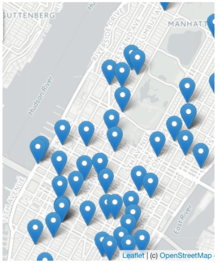


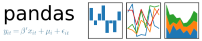


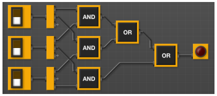


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - for logical circuits,
    - for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

Syllabus: Topics

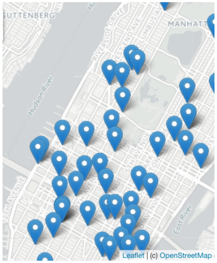


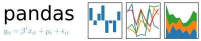


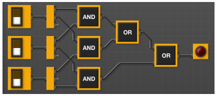


- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - ★ for logical circuits,
    -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

Syllabus: Topics

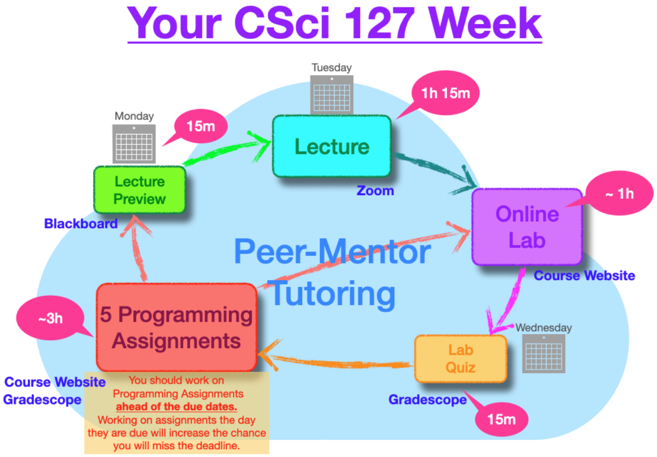






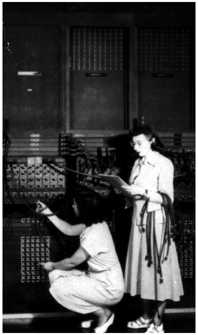
- **This course assumes no previous programming experience.**
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:
    - for logical circuits,
    - for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

Course Structure



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

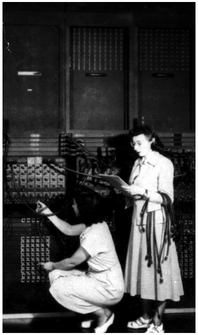


First "computers"

ENIAC, 1945.

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

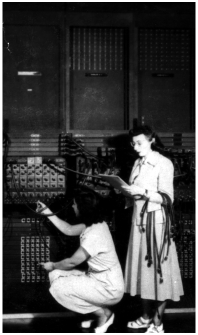
1&2 - Lecture


First "computers"

ENIAC, 1945.

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

1&2 - Lecture

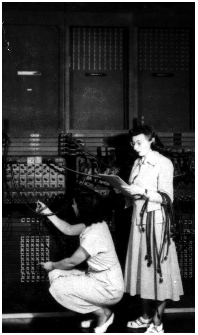


First "computers"

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

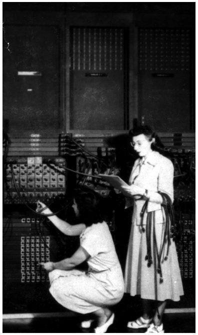


First "computers"

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

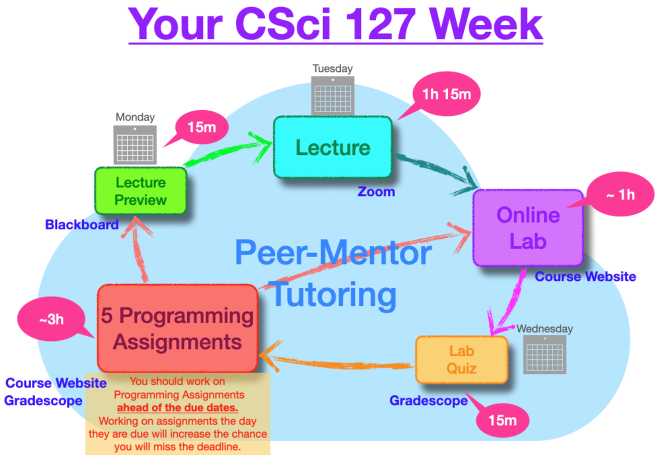


First "computers"

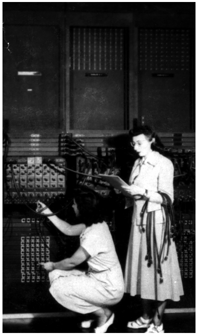
ENIAC, 1945.

- 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.

Course Structure


Your CSci 127 Week
Tuesday
1h 15m
Monday
15m
Lecture
Lecture Preview
Zoom
Blackboard
Online Lab
~ 1h
Peer-Mentor Tutoring
Course Website
~3h
5 Programming Assignments
Wednesday
Lab Quiz
Course Website
Gradescope
Gradescope
15m
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

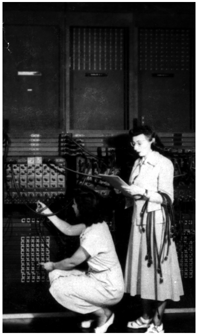

First "computers"

ENIAC, 1945.

Each Week:

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

3 - Online Lab

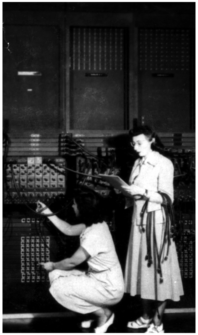

First "computers"

ENIAC, 1945.

Each Week:

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

3 - Online Lab



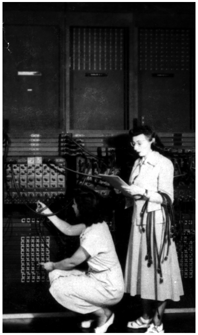
First "computers"

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

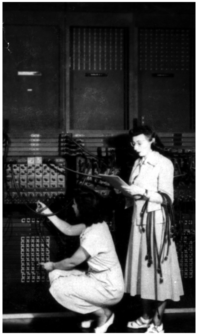

First "computers"

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

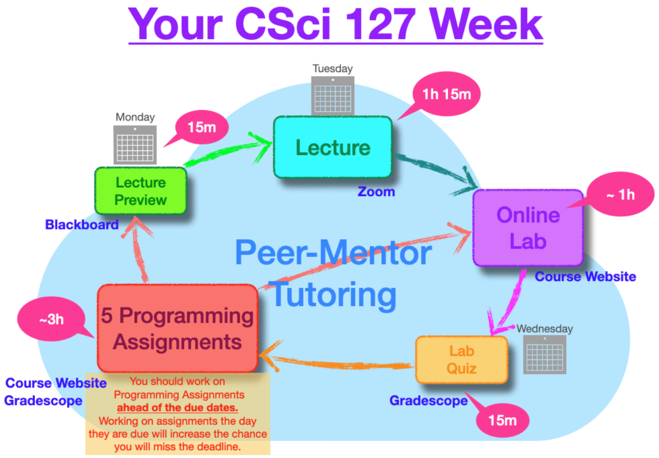

First "computers"

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



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

- **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

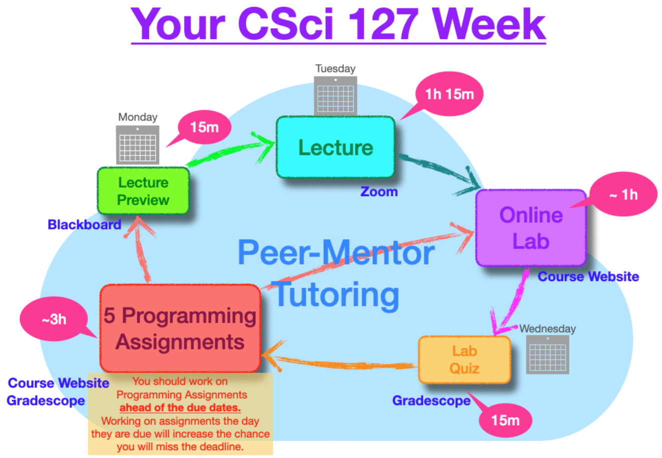
4 -In-person Quiz & Code Review

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

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.

Homework

First "computers"

ENIAC, 1945.

Each Week:

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

Homework

First "computers"

ENIAC, 1945.

Each Week:

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

Homework

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!

Homework

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.

First "computers"

ENIAC, 1945.

Homework

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.

Homework

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.

Homework

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.

Homework

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**.

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**

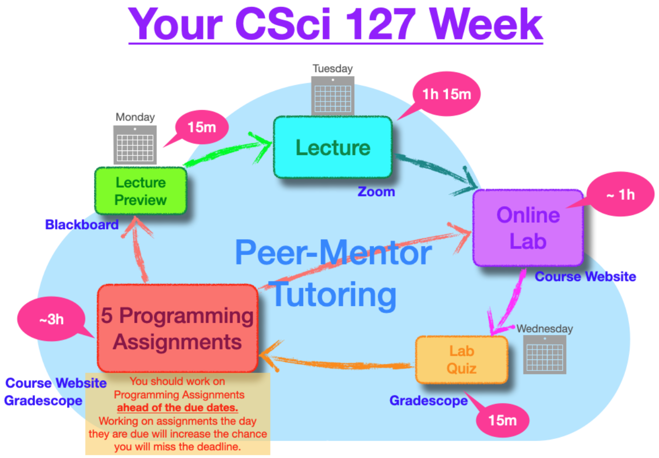
Make Your Schedule!

First "computers"

ENIAC, 1945.

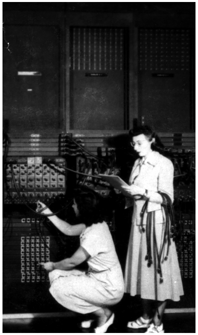
- This is a hybrid course: there is some work you must do independently outside of class meetings.
- Schedule a regular time for the **Online lab**.
- Schedule a regular time for the **Quizzes and Code Review**, plan ahead!
- Schedule a regular time for working on **programming assignments**.
- Schedule a regular time for taking the **Lecture Preview**
- Put them in your calendar now and then adjust if necessary.

Course Structure



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.

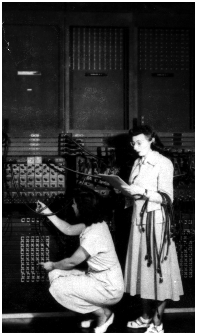
Help and Support


First "computers"

ENIAC, 1945.

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

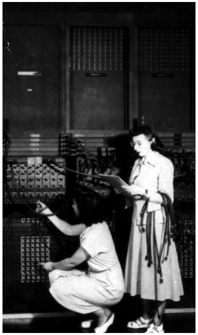
Help and Support


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

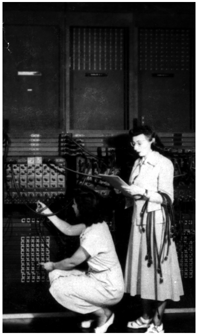
Help and Support


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

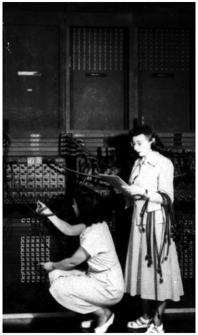
Help and Support


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

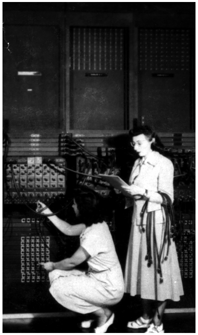


First "computers"

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

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!

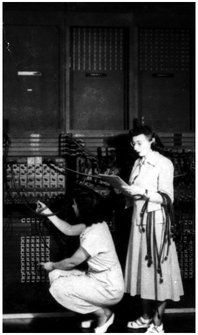
Academic Dishonesty

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**

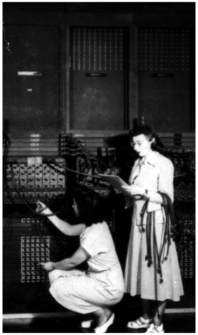
Communication


First "computers"

ENIAC, 1945.

- Important weekly communication sent via Blackboard

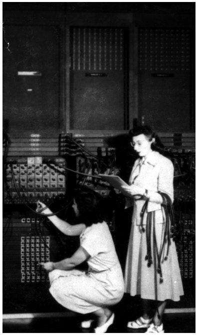
Communication


First "computers"

ENIAC, 1945.

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

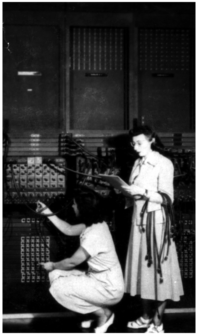
Communication


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:  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
#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

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)

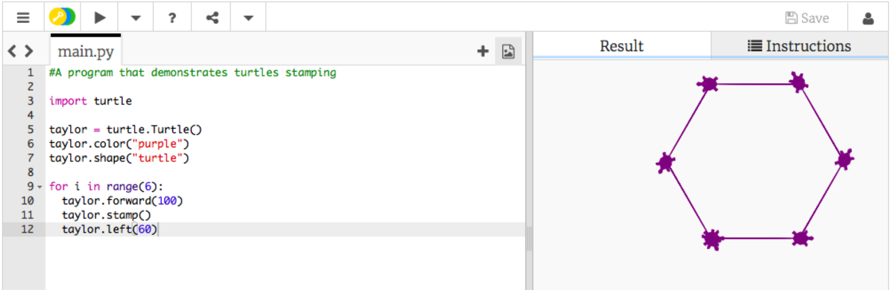
Turtles Introduction

- 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

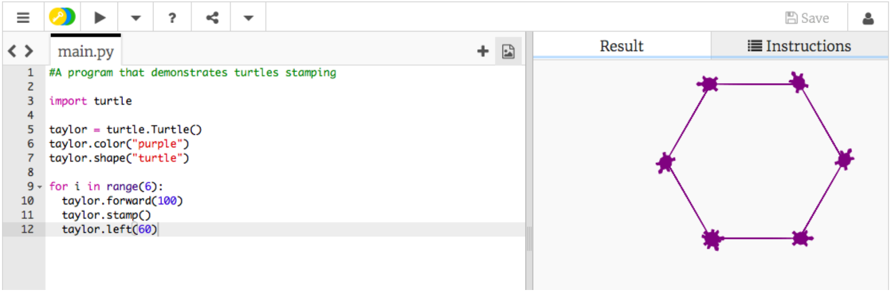
Turtles Introduction



```
#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`.

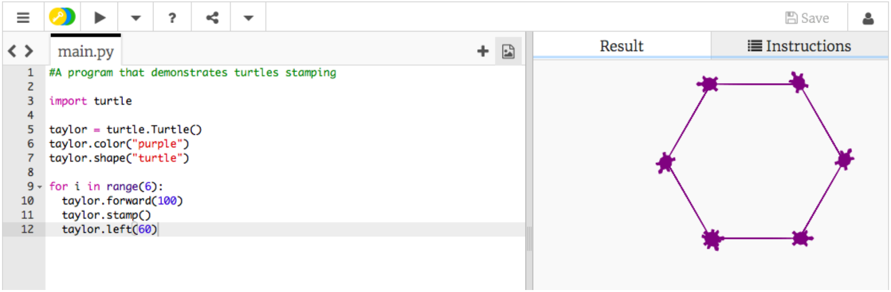
Turtles Introduction



```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).

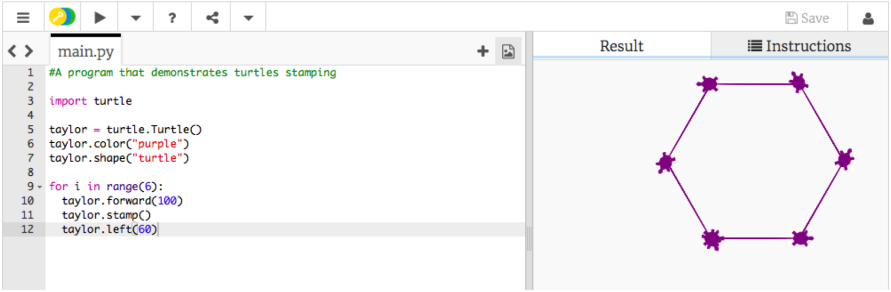
Turtles Introduction



```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:

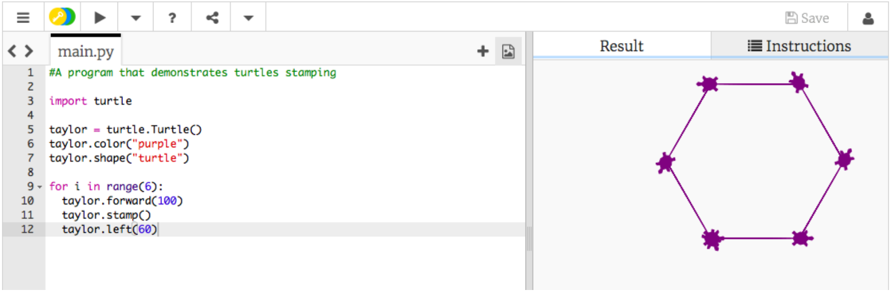
Turtles Introduction



```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.

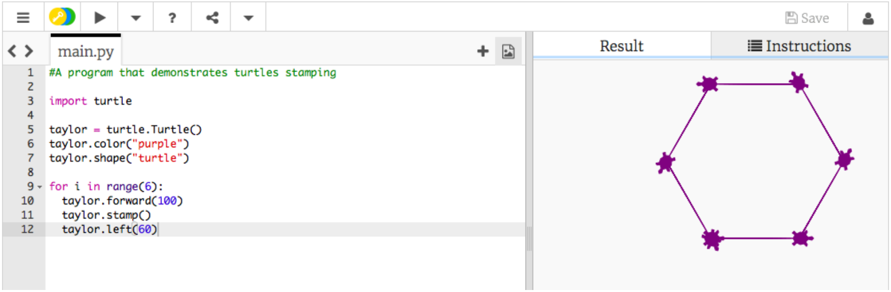
Turtles Introduction



```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



```
#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

Group Work

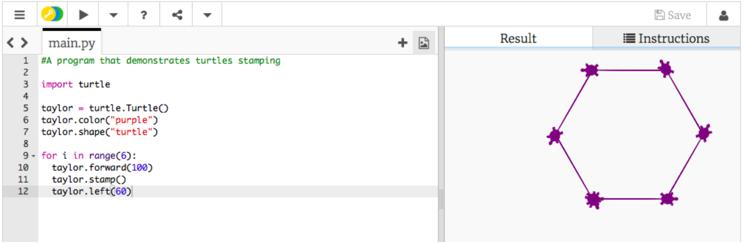
Working in pairs or triples:

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.

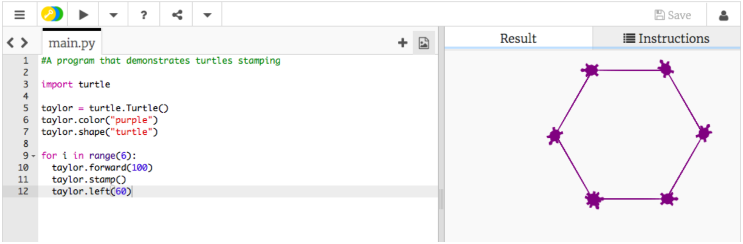
Decagon Program



```
#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.

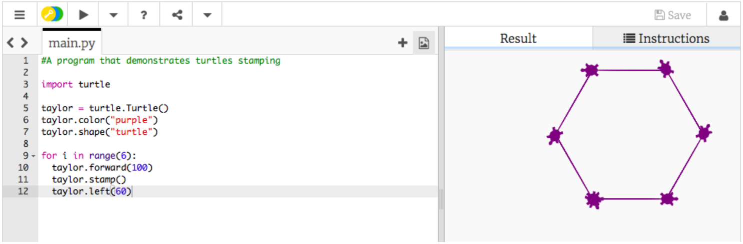
Decagon Program



```
#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!")
  ```

Today's Topics

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

What is an Algorithm?

From our textbook:

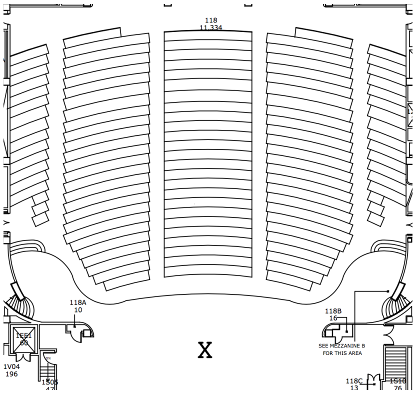
- An algorithm is a process or sequence of steps to be followed to solve a problem.

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.

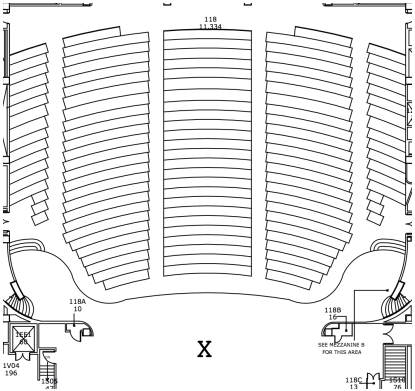
Group Work



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`.

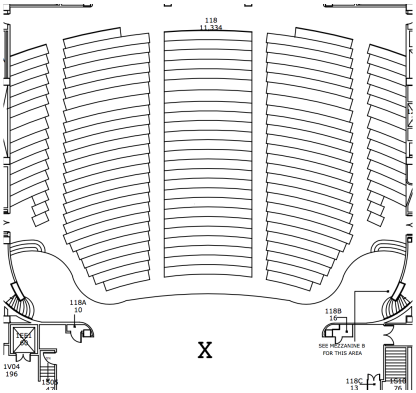
Group Work



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:

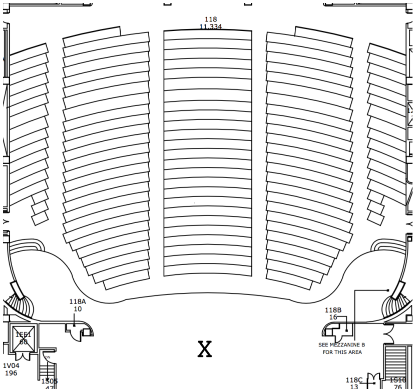
Group Work



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.

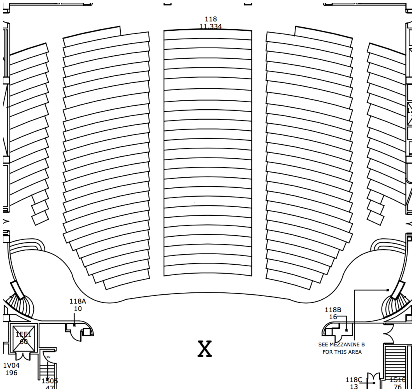
Group Work



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.

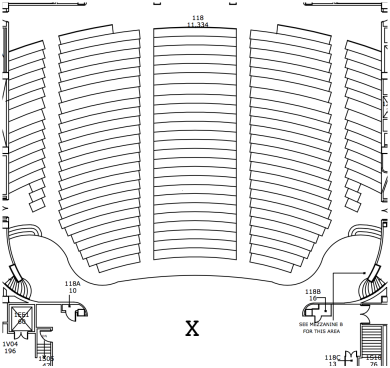
Group Work



Working in pairs or triples:

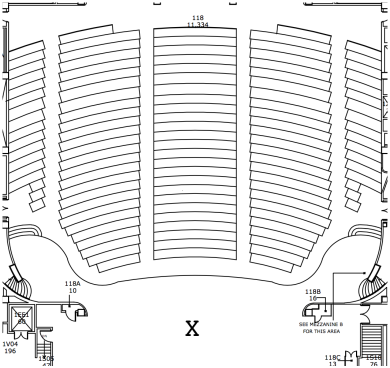
1. On the floorplan, mark your current location.
2. Write an algorithm (step-by-step directions) to get to `X`.
3. Basic Rules:
    - Use `turtle` commands.
    - Do not run turtles into walls, chairs, obstacles, etc.
    - Turtles cannot climb walls, must use stairs.

Group Work



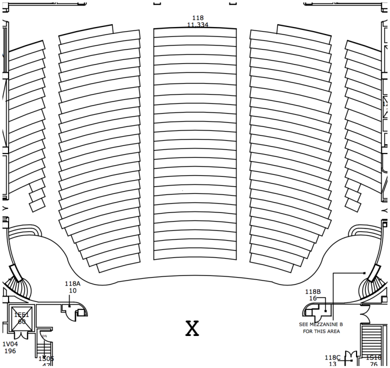
- Have one person in your group be the “turtle.”

Group Work



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

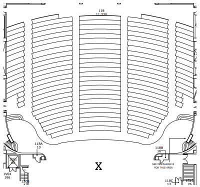
Group Work



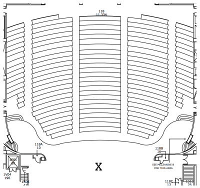
- Have one person in your group be the "turtle."
- Follow the directions to get to X.
- Annotate any changes needed to the directions (i.e. debug your work).

Recap

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

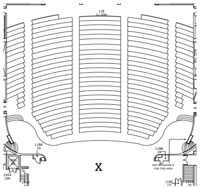


Recap



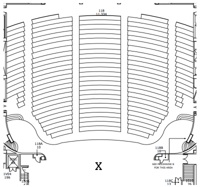
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.

Recap



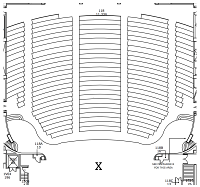
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



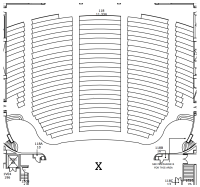
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,

Recap



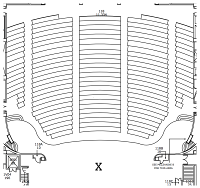
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,

Recap



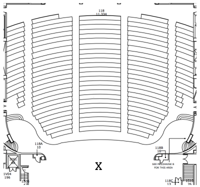
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - `strings`, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &

Recap



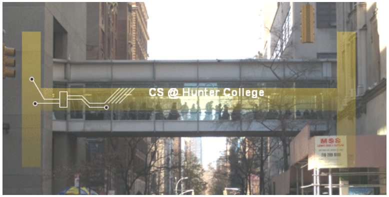
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &
  - variables containing turtles.

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.

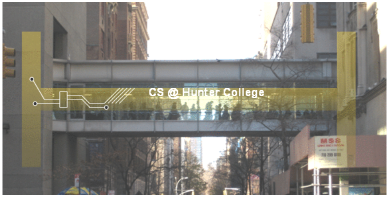
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

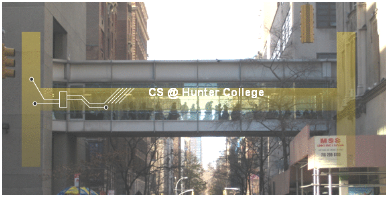
Weekly Reminders!



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

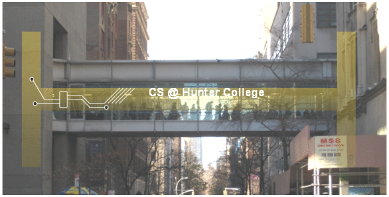
Weekly Reminders!



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

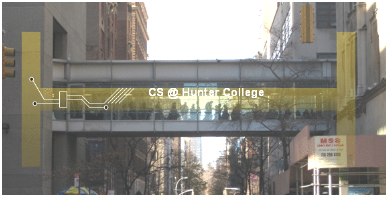
Weekly Reminders!



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)

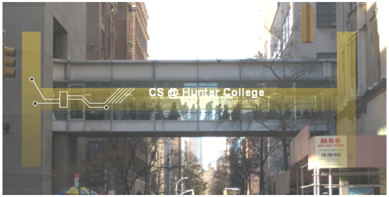
Weekly Reminders!



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

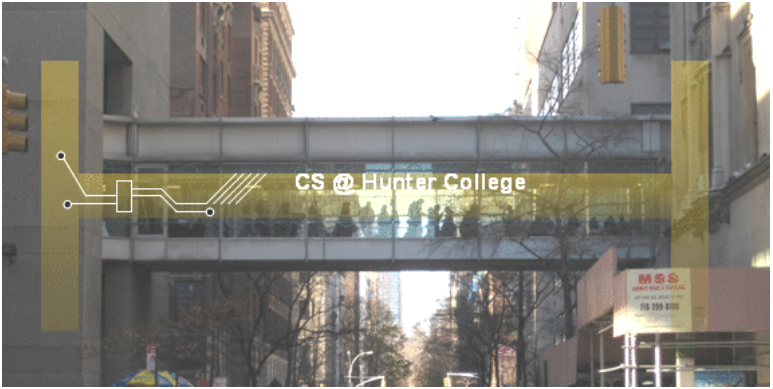
Weekly Reminders!



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.