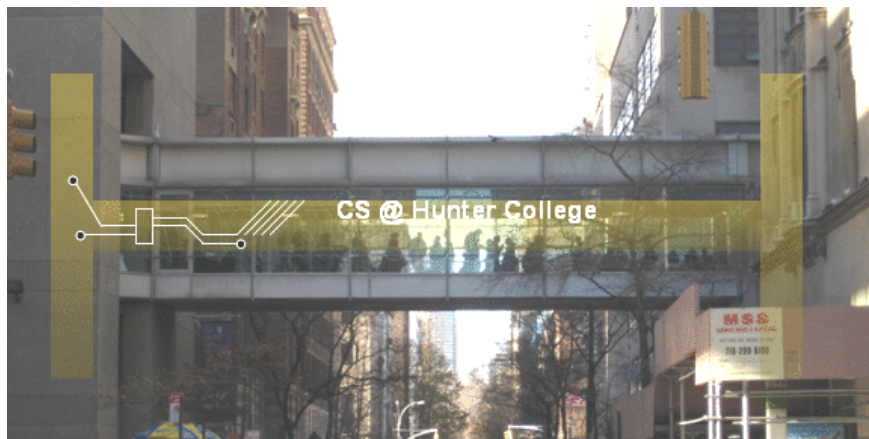


CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Where to find Course Content

- **Course Website:** <https://huntercsci127.github.io/f23.html>

Where to find Course Content

- **Course Website:** <https://huntercsci127.github.io/f23.html>
- **Blackboard:** Announcements and lecture previews

Where to find Course Content

- **Course Website:** <https://huntercsci127.github.io/f23.html>
- **Blackboard:** Announcements and lecture previews
- **Gradescope:** Programming assignment submission

Where to find Course Content

- **Course Website:** <https://huntercsci127.github.io/f23.html>
- **Blackboard:** Announcements and lecture previews
- **Gradescope:** Programming assignment submission
 - ▶ Entry code: 5JXD6K
- **Piazza:** Discussion board
 - ▶ [Sign up link](#)

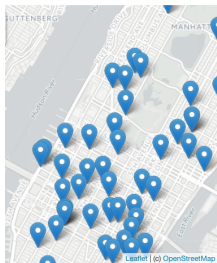
Syllabus

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.

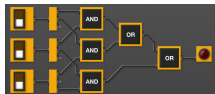
Syllabus: Topics



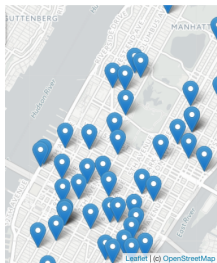
- **This course assumes no previous programming experience.**

pandas

$$X_i = \beta_0 + \beta_1 X_i + \mu_i + \epsilon_i$$



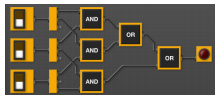
Syllabus: Topics



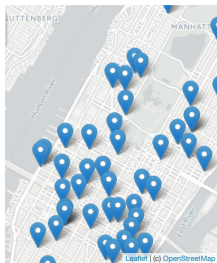
- **This course assumes no previous programming experience.**
- Overview:

pandas

$$X_i = \beta_0 + \beta_1 X_{i1} + \mu_i + \epsilon_{i1}$$



Syllabus: Topics



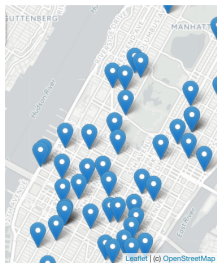
- **This course assumes no previous programming experience.**
- Overview:
 - ▶ Introduce coding constructs in Python,

pandas

$$X_i = \beta_0 + \beta_1 X_{i1} + \mu_i + \epsilon_{i1}$$



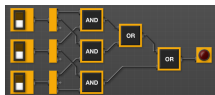
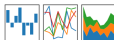
Syllabus: Topics



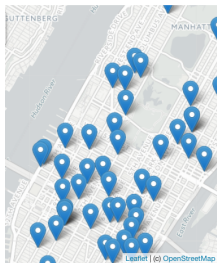
- **This course assumes no previous programming experience.**
- Overview:
 - ▶ Introduce coding constructs in Python,
 - ▶ Apply those ideas to different problems (e.g. analyzing & mapping data),

pandas

$$X_i = \beta_0 + \beta_1 X_{i1} + \mu_i + \epsilon_i$$

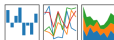


Syllabus: Topics



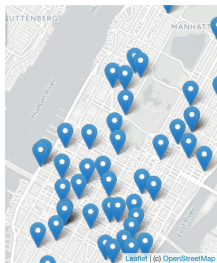
pandas

$$X_i = \beta_0 + \beta_1 X_{i1} + \mu_i + \epsilon_i$$



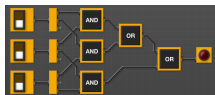
- **This course assumes no previous programming experience.**
- Overview:
 - ▶ Introduce coding constructs in Python,
 - ▶ Apply those ideas to different problems (e.g. analyzing & mapping data),
 - ▶ See constructs again:

Syllabus: Topics



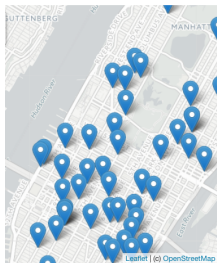
pandas

$X_t = \beta_0 + \beta_1 X_{t-1} + \epsilon_t$



- **This course assumes no previous programming experience.**
- Overview:
 - ▶ Introduce coding constructs in Python,
 - ▶ Apply those ideas to different problems (e.g. analyzing & mapping data),
 - ▶ See constructs again:
 - ★ for logical circuits,

Syllabus: Topics



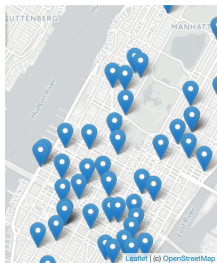
pandas

$M_t = \beta_0 + \beta_1 X_t + \epsilon_t$



- **This course assumes no previous programming experience.**
- Overview:
 - ▶ 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



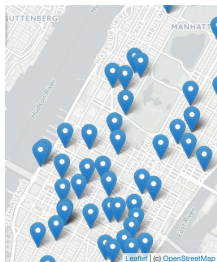
pandas

$M_t = \beta_0 + \beta_1 X_t + \epsilon_t$



- **This course assumes no previous programming experience.**
- Overview:
 - ▶ Introduce coding constructs in Python,
 - ▶ Apply those ideas to different problems (e.g. analyzing & mapping data),
 - ▶ See constructs again:
 - ★ for logical circuits,
 - ★ for Unix command line interface,
 - ★ for the markup language for GitHub,

Syllabus: Topics



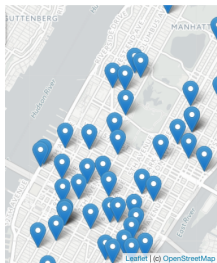
pandas

$M_i = \beta_0 + \beta_1 x_i + \epsilon_i$



- **This course assumes no previous programming experience.**
- Overview:
 - ▶ Introduce coding constructs in Python,
 - ▶ Apply those ideas to different problems (e.g. analyzing & mapping data),
 - ▶ See constructs again:
 - ★ for logical circuits,
 - ★ for Unix command line interface,
 - ★ for the markup language for GitHub,
 - ★ for the simplified machine language, &

Syllabus: Topics



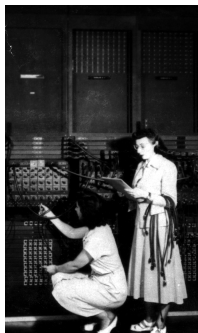
pandas

$M_1 = \sum_{i=1}^n x_i + \mu_i + \epsilon_i$



- **This course assumes no previous programming experience.**
- Overview:
 - ▶ 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++.

Lecture

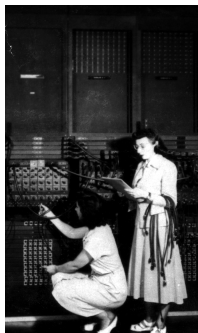


First “computers”

ENIAC, 1945.

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

Lecture

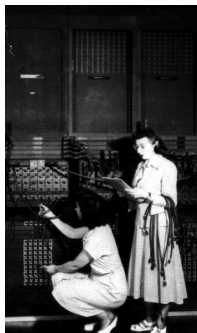


First “computers”

ENIAC, 1945.

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

Lecture

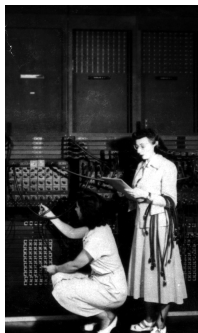


First “computers”

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

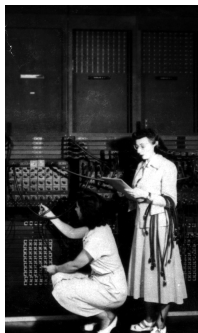


First “computers”

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.

Lecture

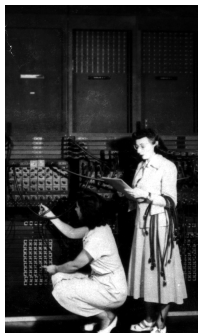


First “computers”

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.

Online Lab



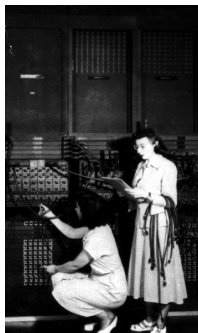
First “computers”

ENIAC, 1945.

Each Week:

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

Online Lab



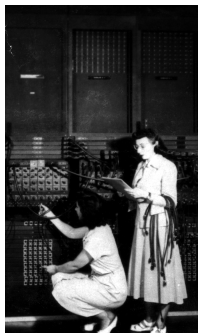
First “computers”

ENIAC, 1945.

Each Week:

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

Online Lab



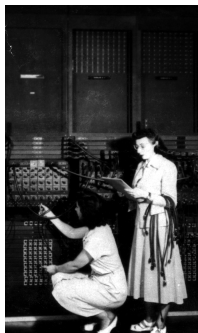
First “computers”

ENIAC, 1945.

Each Week:

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

Online Lab



First “computers”

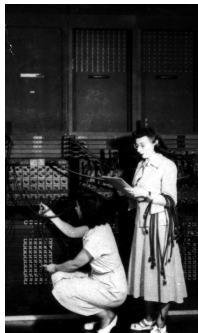
ENIAC, 1945.

Each Week:

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

In-person Quiz & Code Review

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

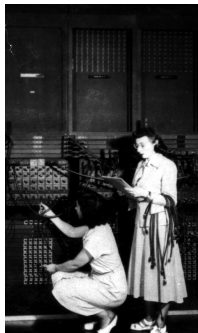


First “computers”

ENIAC, 1945.

In-person Quiz & Code Review

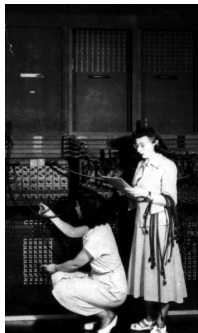
- **Every week you must take a paper quiz** in Lab 1001G Hunter North
- Quizzes are directly related to the current week's lab content



First "computers"

ENIAC, 1945.

In-person Quiz & Code Review

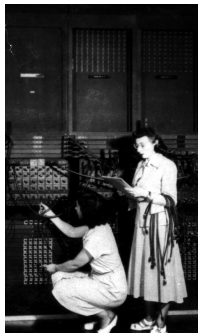


First “computers”

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

In-person Quiz & Code Review

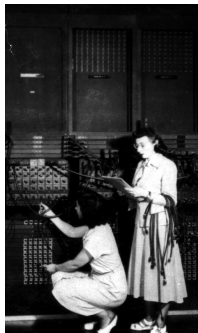


First "computers"

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)

In-person Quiz & Code Review

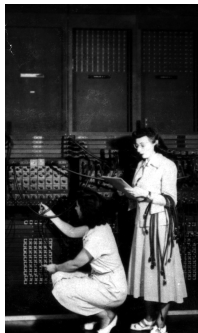


First “computers”

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!

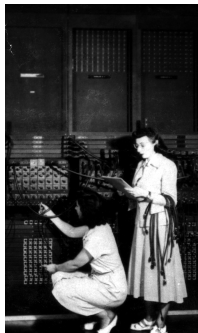
In-person Quiz & Code Review



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

In-person Quiz & Code Review



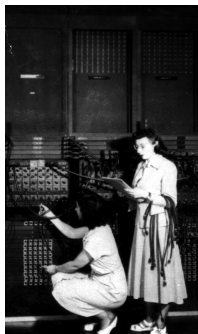
First "computers"
ENIAC, 1945.

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

Programming Assignments

Each Week:

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.



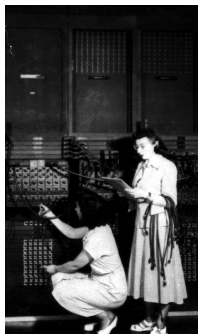
First "computers"

ENIAC, 1945.

Programming Assignments

Each Week:

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **5 Programming Assignments each week!**

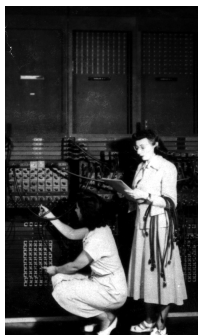


First "computers"

ENIAC, 1945.

Programming Assignments

Each Week:



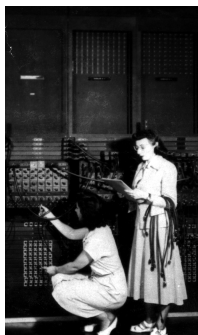
First "computers"

ENIAC, 1945.

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!

Programming Assignments

Each Week:



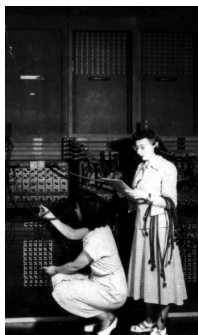
First "computers"

ENIAC, 1945.

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.

Programming Assignments

Each Week:



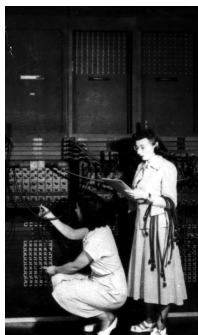
First "computers"

ENIAC, 1945.

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.

Programming Assignments

Each Week:



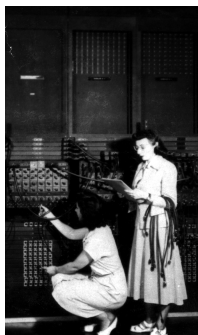
First "computers"

ENIAC, 1945.

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.

Programming Assignments

Each Week:

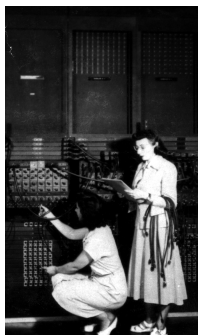


First "computers"

ENIAC, 1945.

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **5 Programming Assignments each week!**
- **Work ahead!!!** Students who work on programs on the due date often miss the deadline!
- Description on Course Webpage.
- Implement and test on your computer.
- Submit to Gradescope.
- Multiple submissions accepted.

Programming Assignments

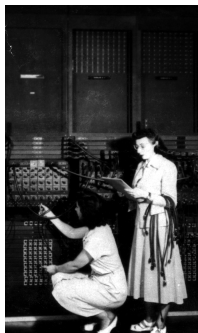


First "computers"
ENIAC, 1945.

Each Week:

- Starting Monday, Sept 11, there will be one program due each day at 6 PM.
- **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!

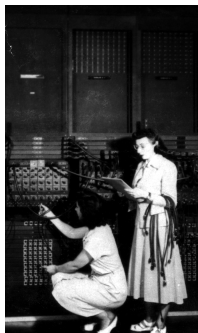


First “computers”

ENIAC, 1945.

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

Make Your Schedule!

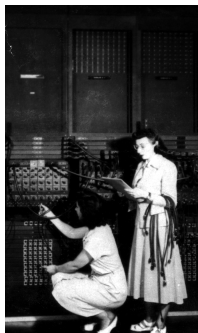


First “computers”

ENIAC, 1945.

- This is a hybrid course: there is work you must do independently outside of lecture.
- Make time every week for the **Online Lab**.

Make Your Schedule!

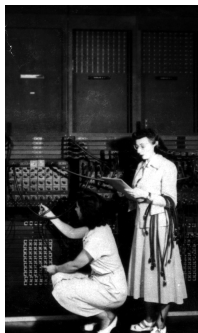


First “computers”

ENIAC, 1945.

- This is a hybrid course: there is work you must do independently outside of lecture.
- Make time every week for the **Online Lab**.
- Schedule an appointment for the **Quizzes and Code Reviews**, plan ahead!

Make Your Schedule!

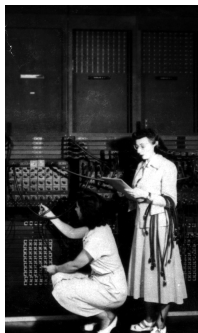


First “computers”

ENIAC, 1945.

- This is a hybrid course: there is work you must do independently outside of lecture.
- Make time every week for the **Online Lab**.
- Schedule an appointment for the **Quizzes and Code Reviews**, plan ahead!
- Work on **Programming Assignments** regularly, ideally every day.

Make Your Schedule!

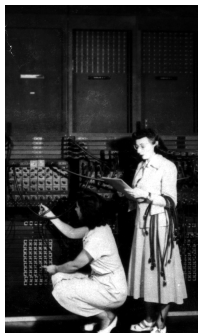


First “computers”

ENIAC, 1945.

- This is a hybrid course: there is work you must do independently outside of lecture.
- Make time every week for the **Online Lab**.
- Schedule an appointment for the **Quizzes and Code Reviews**, plan ahead!
- Work on **Programming Assignments** regularly, ideally every day.
- Remember to take the **Lecture Preview** every week.

Make Your Schedule!

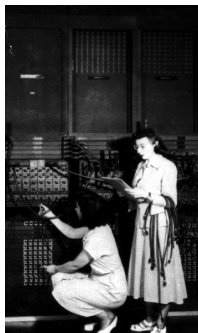


First “computers”

ENIAC, 1945.

- This is a hybrid course: there is work you must do independently outside of lecture.
- Make time every week for the **Online Lab**.
- Schedule an appointment for the **Quizzes and Code Reviews**, plan ahead!
- Work on **Programming Assignments** regularly, ideally every day.
- Remember to take the **Lecture Preview** every week.
- Put them in your calendar now!

Help and Support

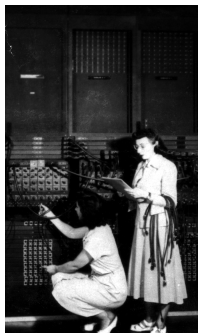


First “computers”

ENIAC, 1945.

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

Help and Support

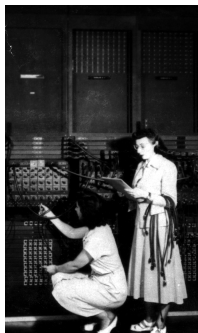


First “computers”

ENIAC, 1945.

- Peer-mentor Support (UTAs)
 - ▶ **Tutoring:** in-person tutoring and programming help in 1001G HN
 - ▶ Schedule an appointment for tutoring

Help and Support

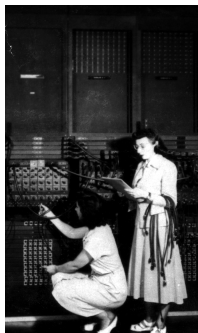


First “computers”

ENIAC, 1945.

- Peer-mentor Support (UTAs)
 - ▶ **Tutoring**: in-person tutoring and programming help in 1001G HN
 - ▶ Schedule an appointment for tutoring
 - ▶ **Discussion Board** on Piazza

Help and Support

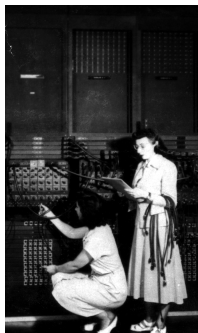


First “computers”

ENIAC, 1945.

- Peer-mentor Support (UTAs)
 - ▶ **Tutoring**: in-person tutoring and programming help in 1001G HN
 - ▶ Schedule an appointment for tutoring
 - ▶ **Discussion Board** on Piazza
 - ▶ Available **M-F 11:30am-5:30pm** when classes are in session

Help and Support



First “computers”

ENIAC, 1945.

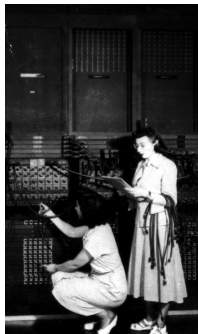
- Peer-mentor Support (UTAs)
 - ▶ **Tutoring**: in-person tutoring and programming help in 1001G HN
 - ▶ Schedule an appointment for tutoring
 - ▶ **Discussion Board** on Piazza
 - ▶ Available **M-F 11:30am-5:30pm** when classes are in session

Benefits of Tutoring and Code Review



Academic Dishonesty

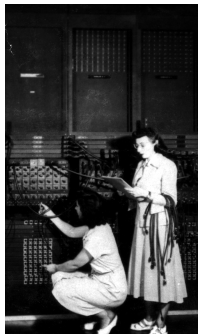
- *The person who does the work gets the benefit! Learning is personal!!!*



First "computers"

ENIAC, 1945.

Academic Dishonesty

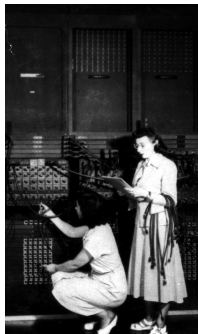


First "computers"

ENIAC, 1945.

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

Academic Dishonesty

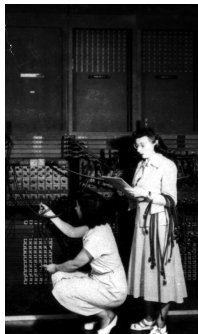


First "computers"

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

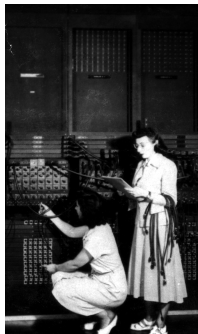


First "computers"

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

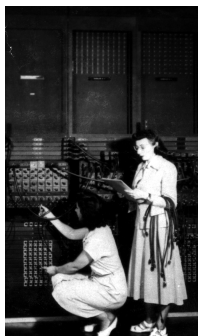


First "computers"

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

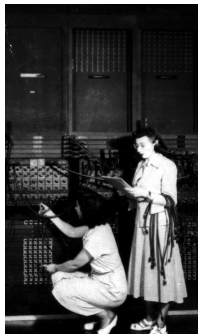


First "computers"

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**

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 extensibility, 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 extensibility, 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:  Melissa Lynch  
#Date:  Aug 29, 2023  
#This program prints:  Hello, World!  
  
print("Hello, World!")
```

First Program: Hello, World!

```
#Name:  Melissa Lynch
```

← *These lines are comments*

```
#Date:  August 29, 2023
```

← *(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:  Melissa Lynch
```

← *These lines are comments*

```
#Date:  August 29, 2023
```

← *(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:  Melissa Lynch
```

← *These lines are comments*

```
#Date:  August 29, 2023
```

← *(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: Melissa Lynch
#Date: August 29, 2023
#This program prints multiple strings

print("I like turtles")
print("Python is fun")
print("First day of class")
```

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.
- *History:* Turtle graphics are a key feature of Logo, an educational programming language designed in 1967.

Turtles Introduction



- A simple, whimsical graphics package for Python.
- *History:* Turtle graphics are a key feature of Logo, an educational programming language designed in 1967.
- (Hexagon)

Turtles Introduction



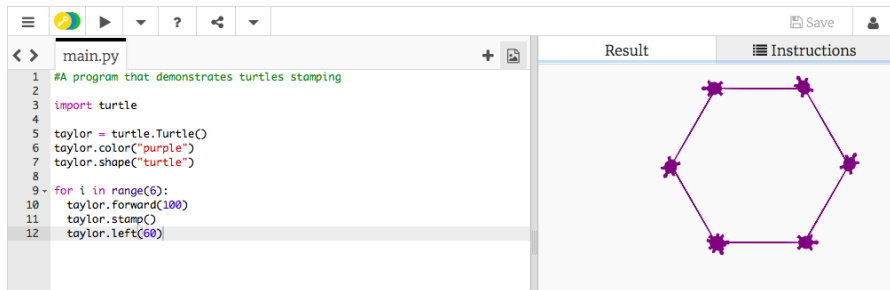
- A simple, whimsical graphics package for Python.
- *History:* Turtle graphics are a key feature of Logo, an educational programming language designed in 1967.
- (Hexagon)
- (Fancy hexagon)

Today's Topics



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

Turtles Introduction



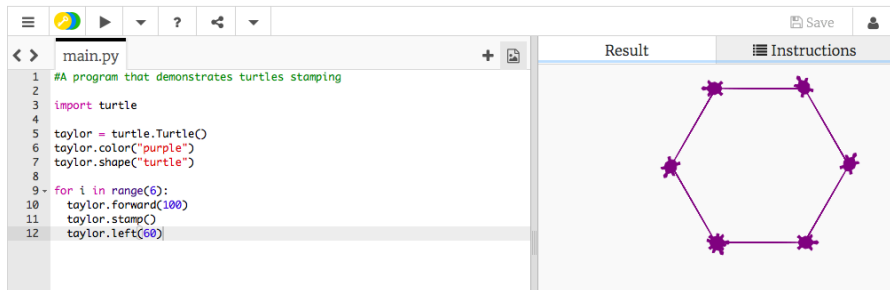
The screenshot shows a Python IDE with a code editor on the left and a preview window on the right. The code editor displays the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

The preview window on the right is divided into two tabs: "Result" and "Instructions". The "Result" tab is active and shows a purple hexagon with a turtle stamp at each of its six vertices.

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

Turtles Introduction



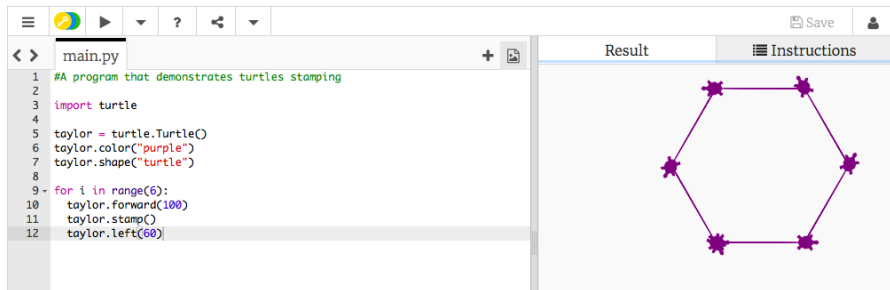
The screenshot shows a Python IDE interface. On the left, a code editor window titled 'main.py' contains the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

On the right, the 'Result' pane displays the output of the code: a purple hexagon with a turtle-shaped stamp at each of its six vertices. The 'Instructions' pane is currently empty.

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

Turtles Introduction



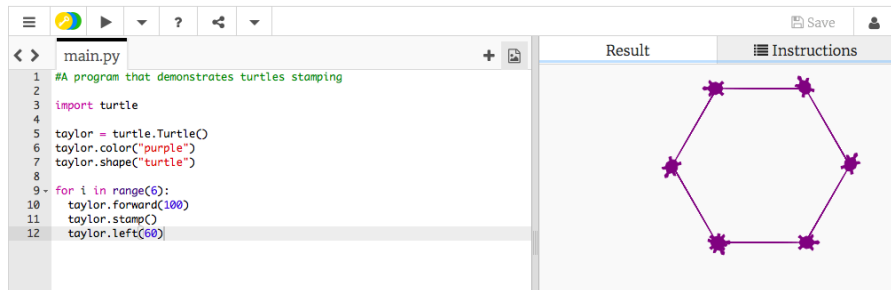
The screenshot shows a Python IDE with a code editor on the left and a preview window on the right. The code editor displays the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

The preview window on the right is divided into two tabs: "Result" and "Instructions". The "Result" tab is active and shows a purple hexagon with a turtle-shaped stamp at each of its six vertices.

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

Turtles Introduction



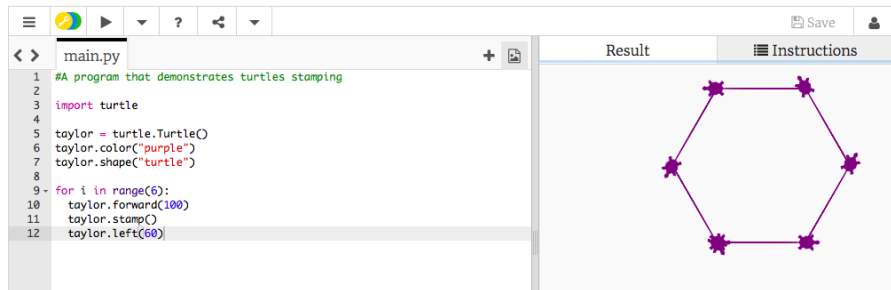
The screenshot shows a Python IDE with a code editor on the left and a result viewer on the right. The code editor contains the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

The result viewer on the right has two tabs: "Result" and "Instructions". The "Result" tab is active and displays a purple hexagon with a turtle-shaped stamp at each of its six vertices.

- 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



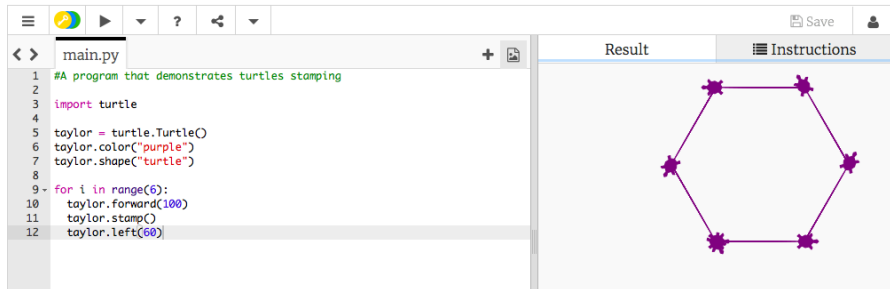
The screenshot shows a Python IDE with a code editor on the left and a result pane on the right. The code editor contains the following Python code:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

The result pane on the right is titled "Result" and "Instructions" and displays a purple hexagon with a turtle-shaped stamp at each of its six vertices.

- 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



The screenshot shows a code editor window with a file named 'main.py'. The code is as follows:

```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

On the right side of the editor, there are two tabs: 'Result' and 'Instructions'. The 'Result' tab is active and displays a purple hexagon with a turtle-shaped stamp at each of its six vertices.

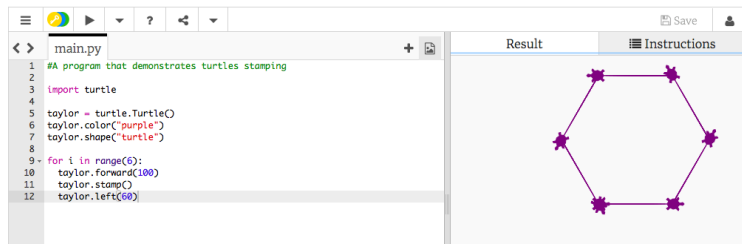
- Creates a turtle **variable**, called taylor.
- Changes the color (to purple) and shape (to turtle-shaped).
- Repeats 6 times:
 - ▶ Move forward; stamp; and turn left 60 degrees.
- Repeats any instructions **indented** in the “loop block”
- This is a **definite** loop because it repeats a fixed number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

Decagon Program



```
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

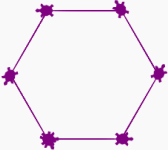
The screenshot shows a Python IDE with a code editor on the left and a result window on the right. The code editor contains a Python script named 'main.py' that uses the turtle module to draw a purple hexagon with star-shaped stamps at each vertex. The result window shows the output of the program, which is a purple hexagon with six purple star-shaped stamps at each vertex.

- Start with the hexagon program.

Decagon Program

```
main.py
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

Result



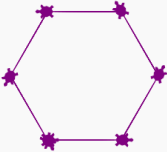
Instructions

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

Decagon Program

```
main.py
1 #A program that demonstrates turtles stamping
2
3 import turtle
4
5 taylor = turtle.Turtle()
6 taylor.color("purple")
7 taylor.shape("turtle")
8
9 for i in range(6):
10     taylor.forward(100)
11     taylor.stamp()
12     taylor.left(60)
```

Result



Instructions

- 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 like turtles`

three times.

Work Program

- ② Write a program that will repeat the line:

```
I like turtles
```

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 like turtles
```

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 like turtles
```

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 like turtles")
```

Today's Topics



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

What is an Algorithm?

From our textbook:

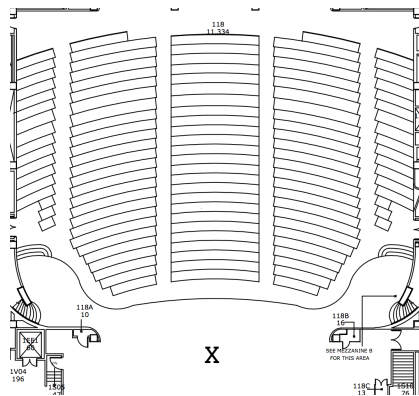
- An **algorithm** is a process or sequence of steps to be followed to solve a problem.

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.

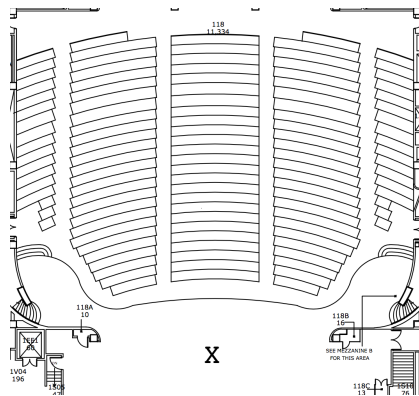
Group Work



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.

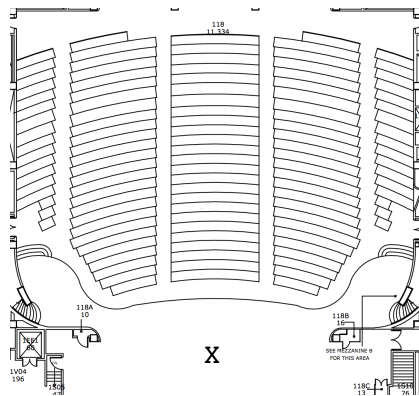
Group Work



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:

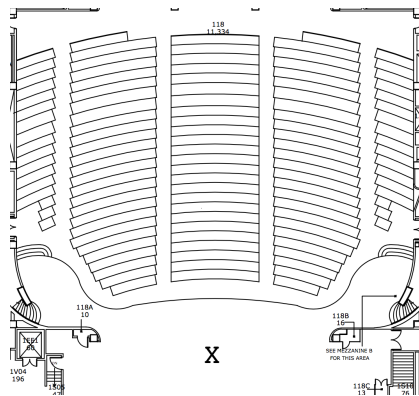
Group Work



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.

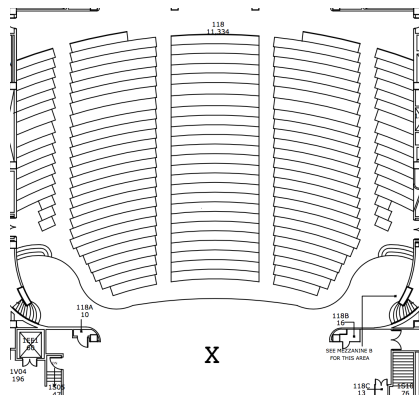
Group Work



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.

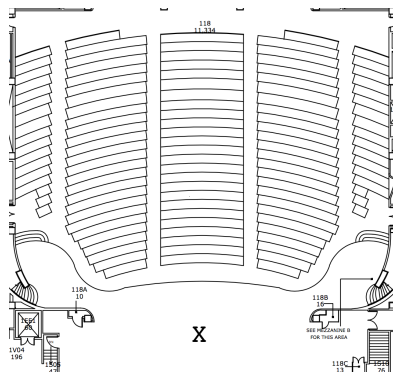
Group Work



Working in pairs or triples:

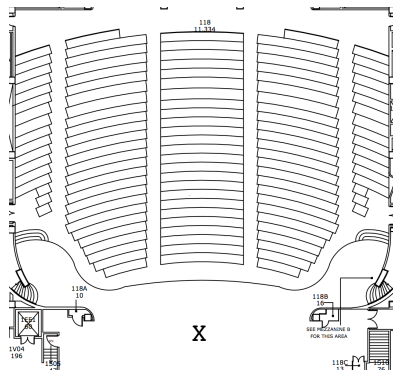
- 1 On the floorplan, mark your current location.
- 2 Write an algorithm (step-by-step directions) to get to X.
- 3 Basic Rules:
 - ▶ Use turtle commands.
 - ▶ Do not run turtles into walls, chairs, obstacles, etc.
 - ▶ Turtles cannot climb walls, must use stairs.

Group Work



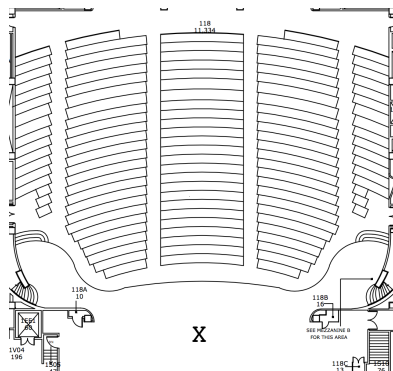
- Have one person in your group be the “turtle.”

Group Work



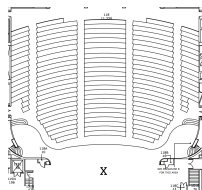
- Have one person in your group be the “turtle.”
- Follow the directions to get to X.

Group Work



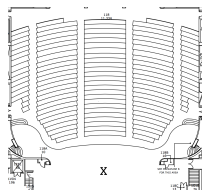
- Have one person in your group be the “turtle.”
- Follow the directions to get to X.
- Annotate any changes needed to the directions (i.e. debug your work).

Recap



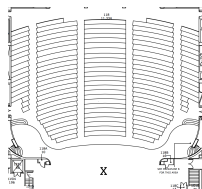
- Writing precise algorithms is difficult.

Recap



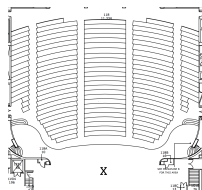
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



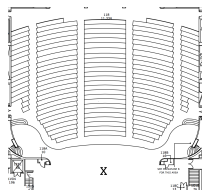
- Writing precise algorithms is difficult.
- In Python, we introduced:
 - ▶ **strings**, or sequences of characters,

Recap



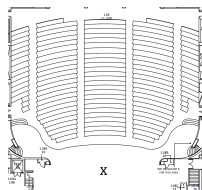
- Writing precise algorithms is difficult.
- In Python, we introduced:
 - ▶ `strings`, or sequences of characters,
 - ▶ `print()` statements,

Recap



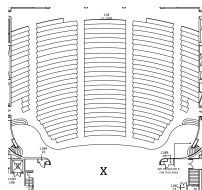
- Writing precise algorithms is difficult.
- In Python, we introduced:
 - ▶ `strings`, or sequences of characters,
 - ▶ `print()` statements,
 - ▶ `for`-loops with `range()` statements, &

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.