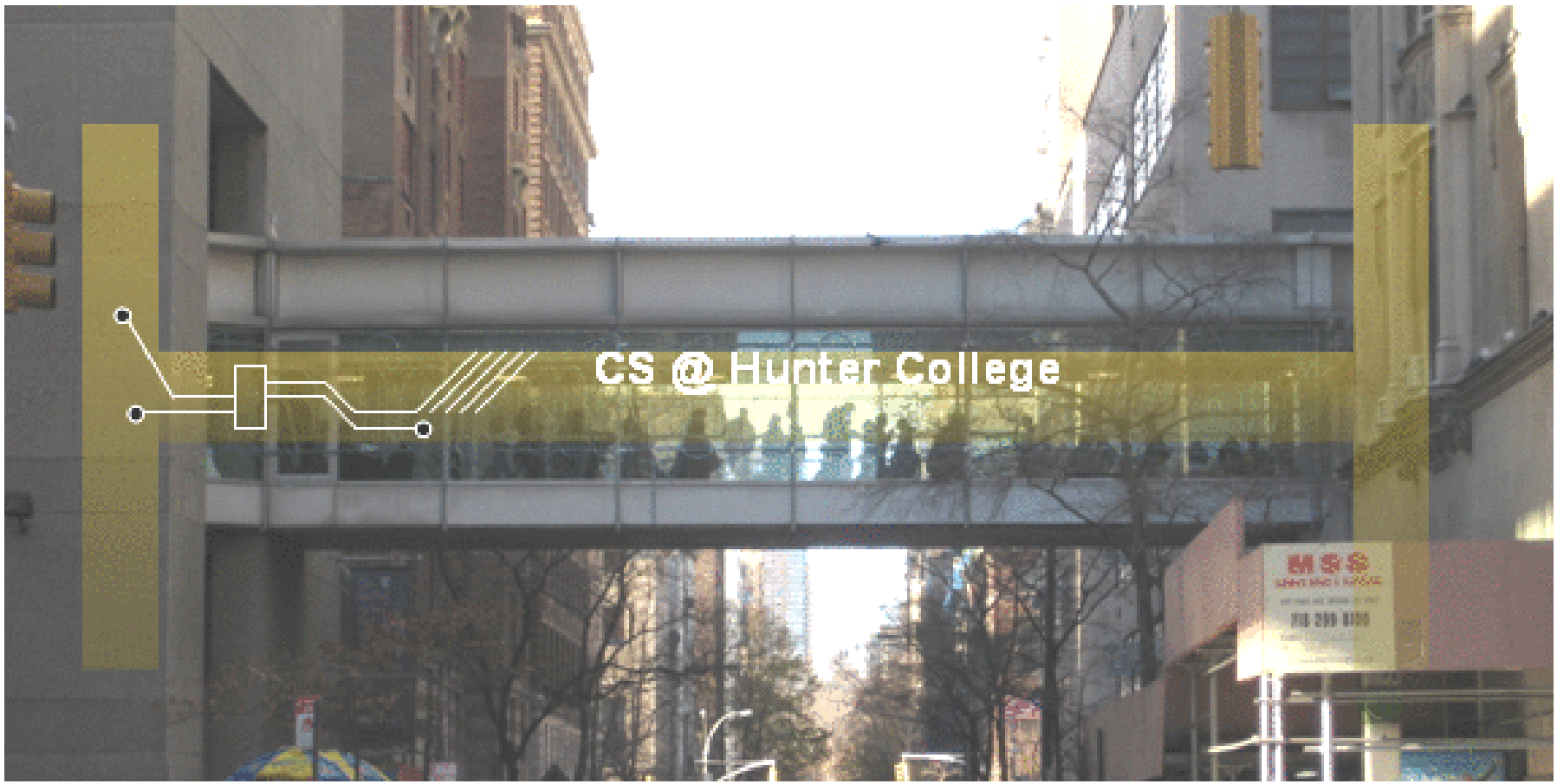
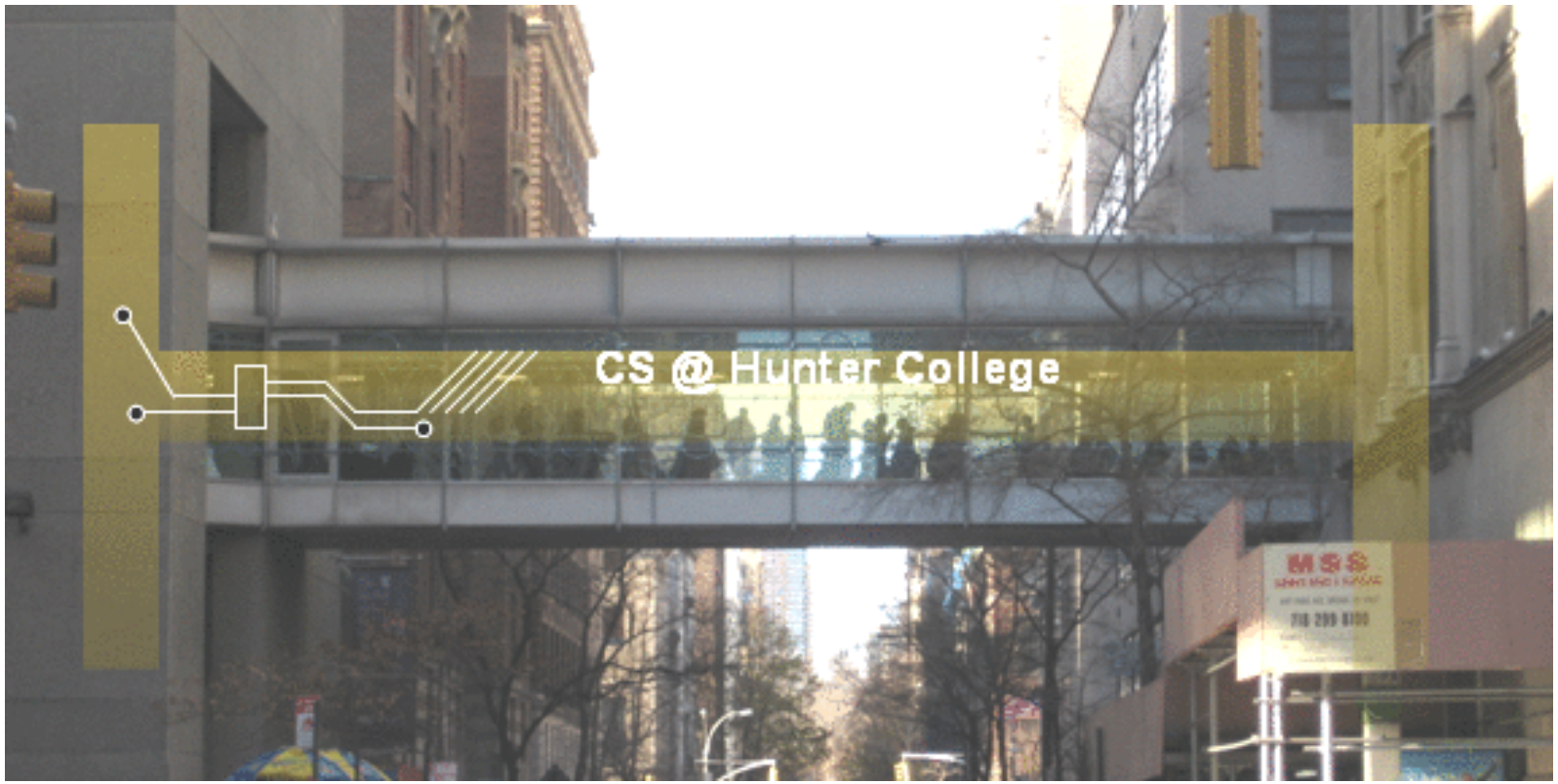
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Welcome



- Prof. Sakas, Department Chair and Course Coordinator

Introductions: Course Coordinators

Dr. Katherine St. John

Instructor

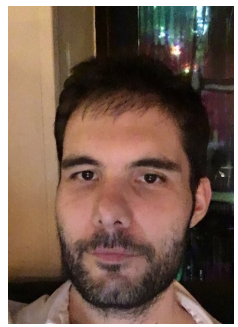
Dr. William Sakas

Chair

Introductions: Recitation Instructors


Basak Taylan


Ekaterina Kistanova


Gwenael Gatto

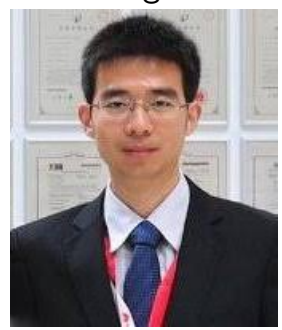

Jiaxing Tan


Katherine Howitt


Minh Nguyen


Subhadarshi Panda

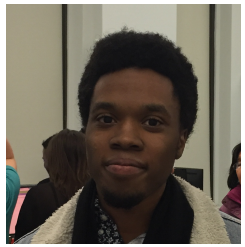

Xiaojie Zhang


Xiaoke (Jimmy) Shen

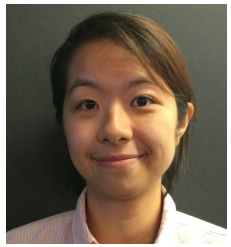
Introductions: Undergraduate Teaching Assistants


Antonio Bountouvas


Alvin Lam


Brian Campbell

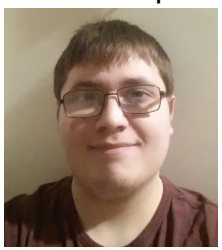

Calvin Quach


Carol Chau


Dandan Lin


Jaime Canizales


Jack Chen


Jakub Taraska

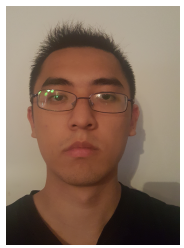

Jesse Goodspeed


Karen Medlin


Karoline Dubin

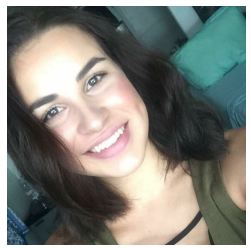

Lily Caplan


Munem Rastgir


Nicky Cen


Olga Kent


Qiuqun Wang


Savannah Nester


Silvena Chan


Usmaan Sahak


Vincent Zheng


Yasmeen Hassan

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. Other topics include: organization of hardware, software, and how information is structured on contemporary computing devices. 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.

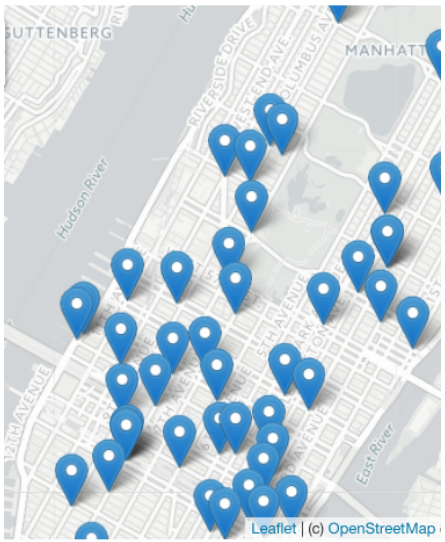
Syllabus

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. Other topics include: organization of hardware, software, and how information is structured on contemporary computing devices. 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.

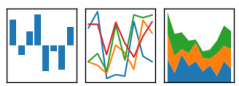
(Show syllabus webpage)

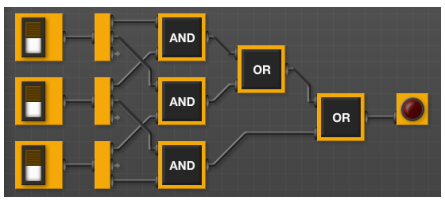
Syllabus: Topics



pandas

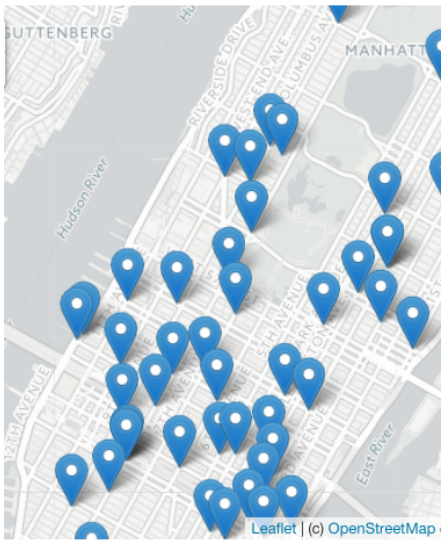
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

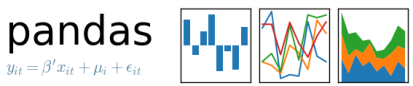


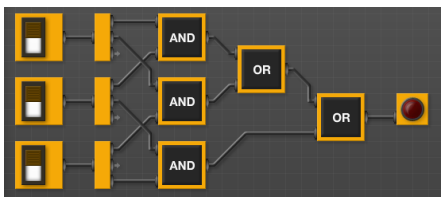


- **This course assumes no previous programming experience.**

Syllabus: Topics

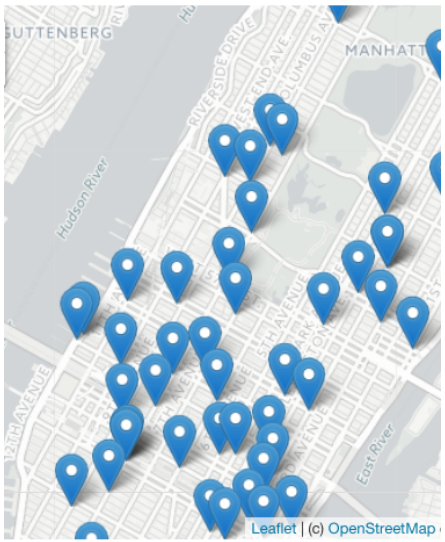


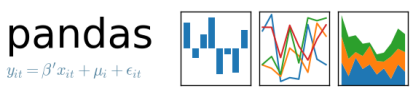


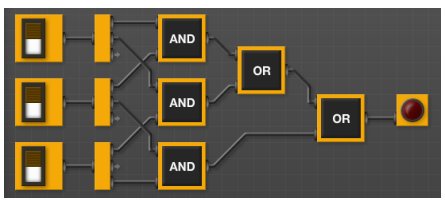


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."

Syllabus: Topics

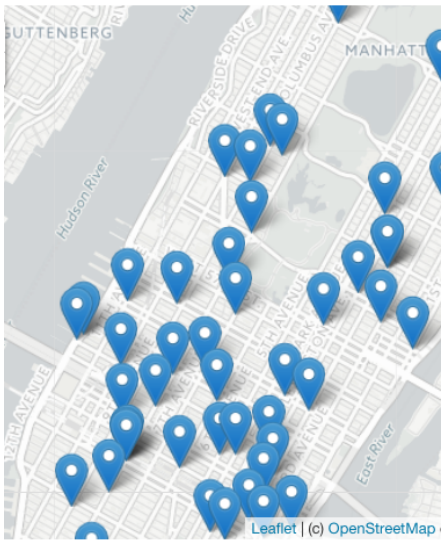


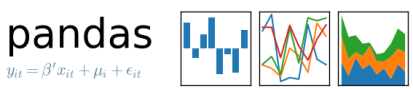


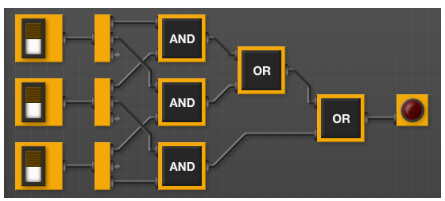


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:

Syllabus: Topics

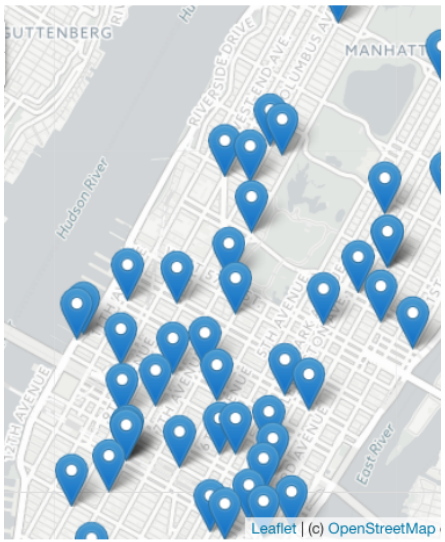


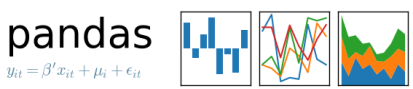


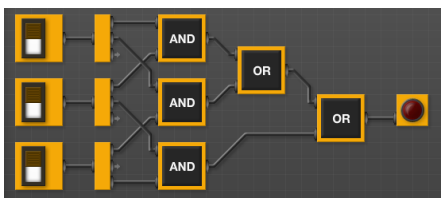


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,

Syllabus: Topics

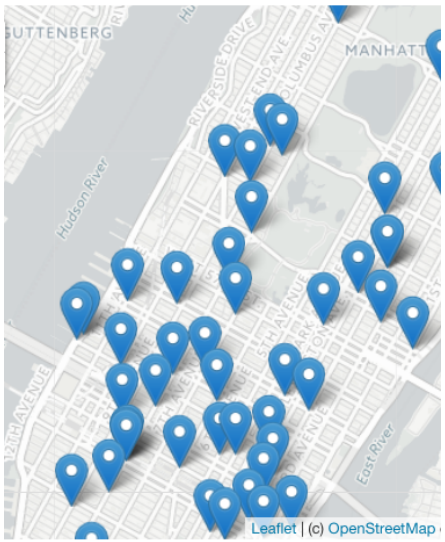






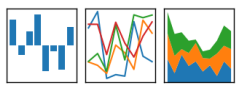
- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),

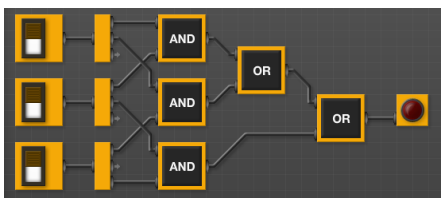
Syllabus: Topics



pandas

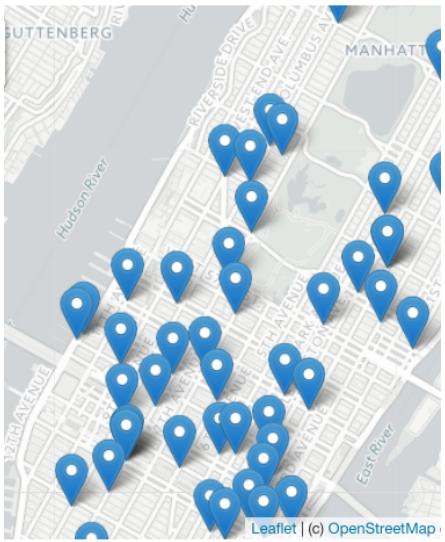
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

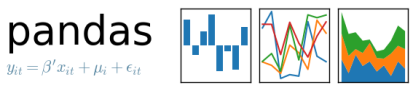


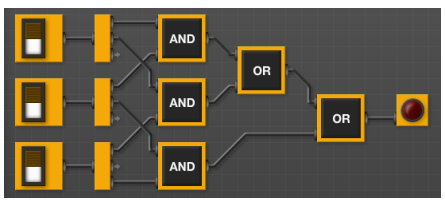


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas to different problems (e.g. analyzing & mapping data),
  - See constructs again:

Syllabus: Topics

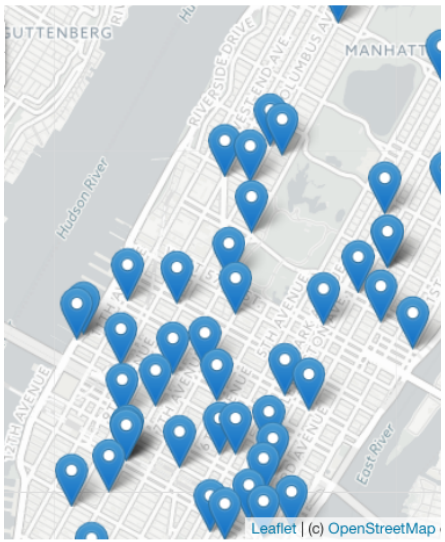






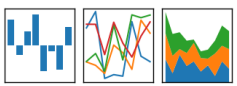
- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas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

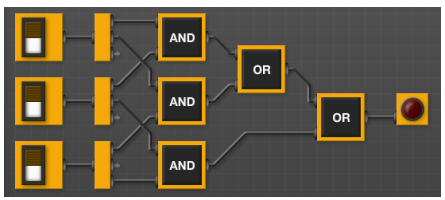
Syllabus: Topics



pandas

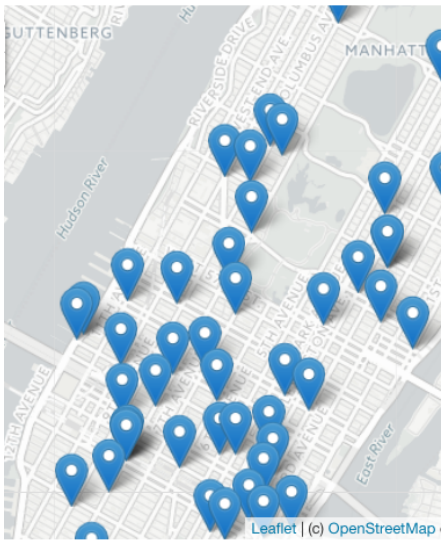
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$





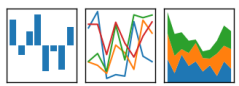
- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas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

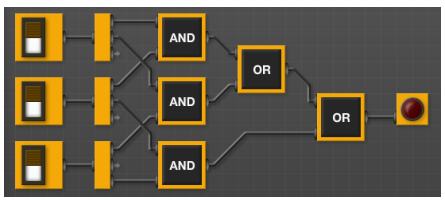
Syllabus: Topics



pandas

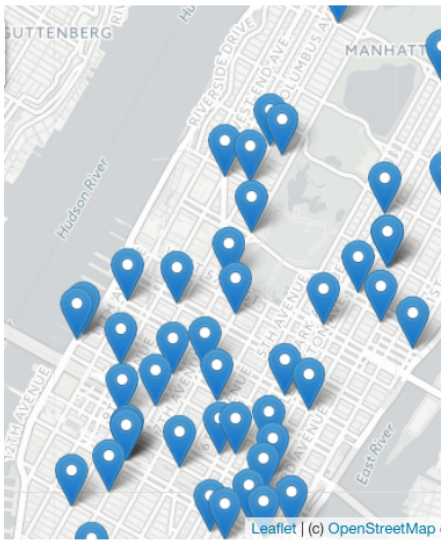
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

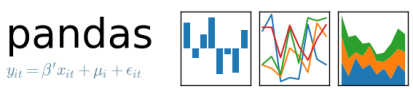


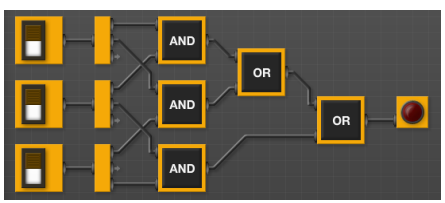


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas 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

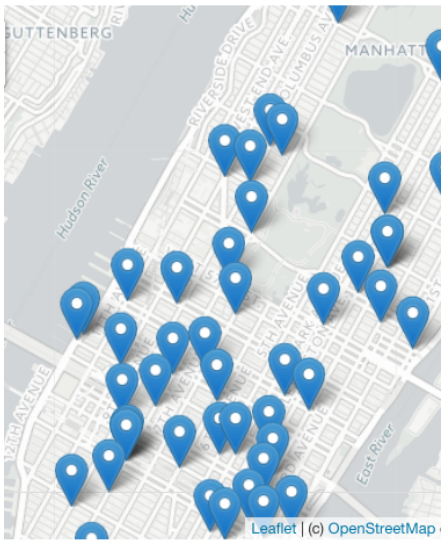


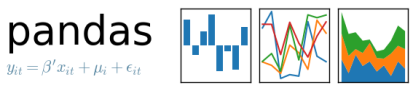


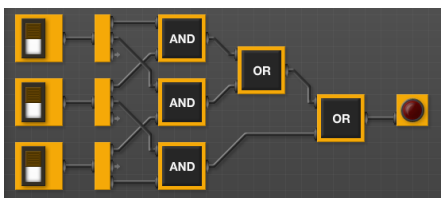


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas 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





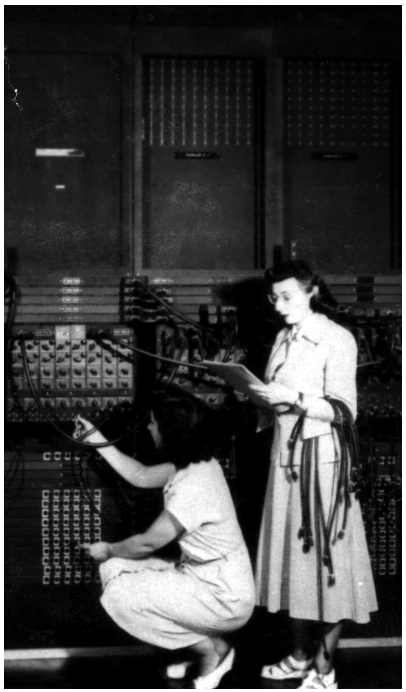


- **This course assumes no previous programming experience.**
- "... Emphasis on problem-solving and computational thinking through 'coding': computer programming for beginners..."
- Organized like a fugue, with variations on this theme:
  - Introduce coding constructs in Python,
  - Apply those ideas 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++.

Class Structure

Lecture:

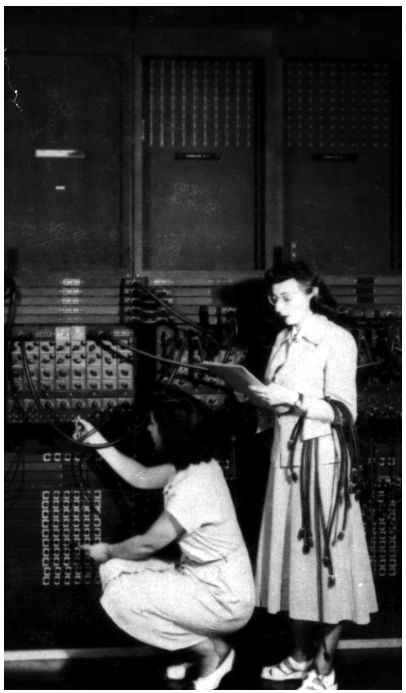
- Tuesdays, 11:10am-12:25pm, 118 North



First "computers"

ENIAC, 1945.

Class Structure



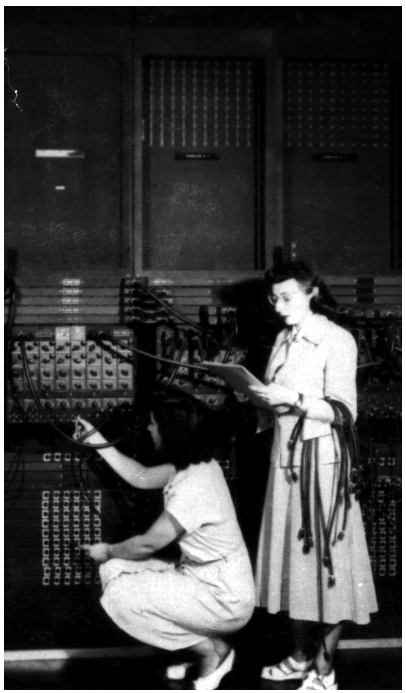
First "computers"

ENIAC, 1945.

Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips

Class Structure



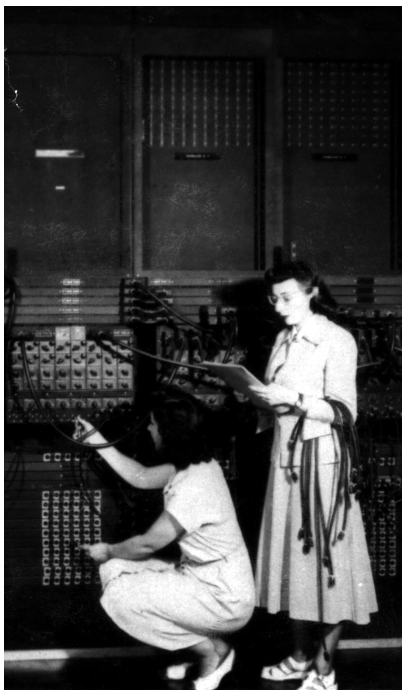
First "computers"

ENIAC, 1945.

Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.

Class Structure



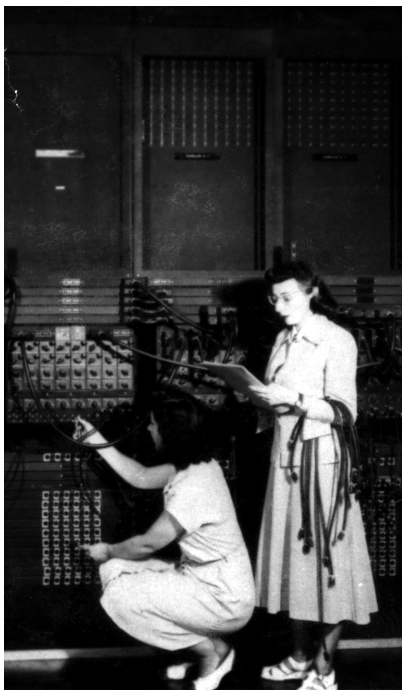
First "computers"

ENIAC, 1945.

Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Class Structure



First "computers"

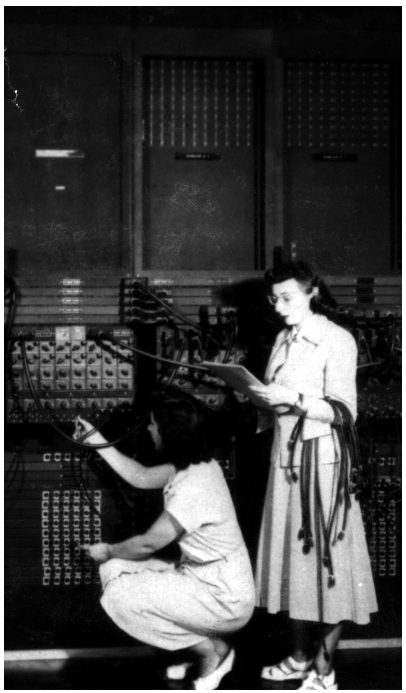
ENIAC, 1945.

Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

Class Structure



First "computers"

ENIAC, 1945.

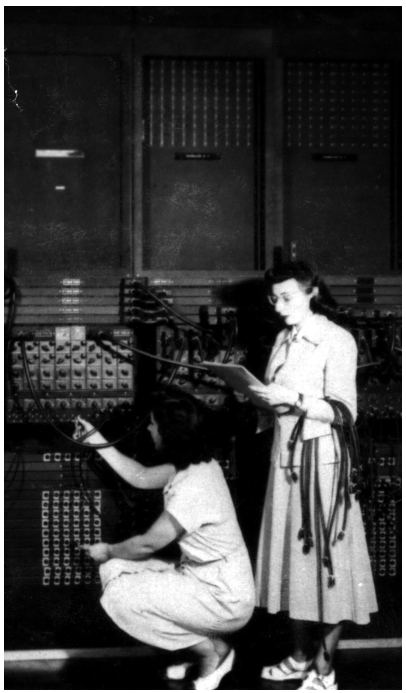
Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

1. Quiz: final exam replaces low/missing quizzes.

Class Structure



First "computers"

ENIAC, 1945.

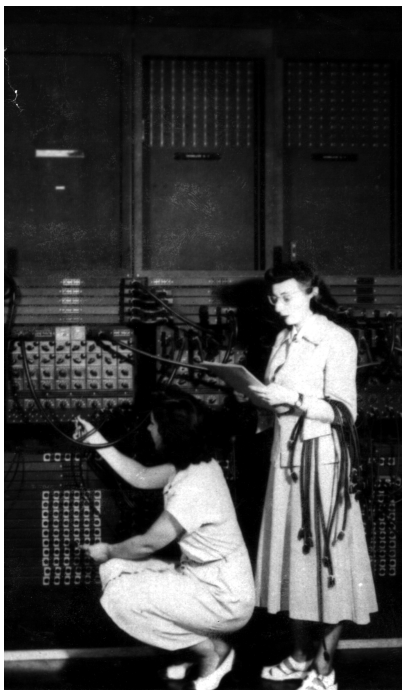
Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

1. Quiz: final exam replaces low/missing quizzes.
2. Brief overview of lab & programs for the week.

Class Structure



First "computers"

ENIAC, 1945.

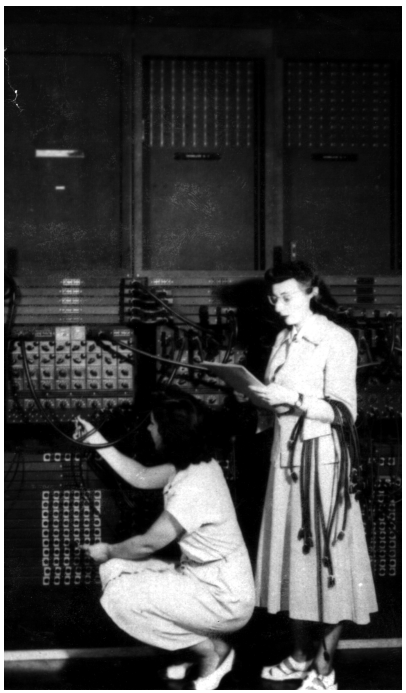
Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

1. Quiz: final exam replaces low/missing quizzes.
2. Brief overview of lab & programs for the week.
3. One-on-one help with instructors & UTAs.

Class Structure



First "computers"

ENIAC, 1945.

Lecture:

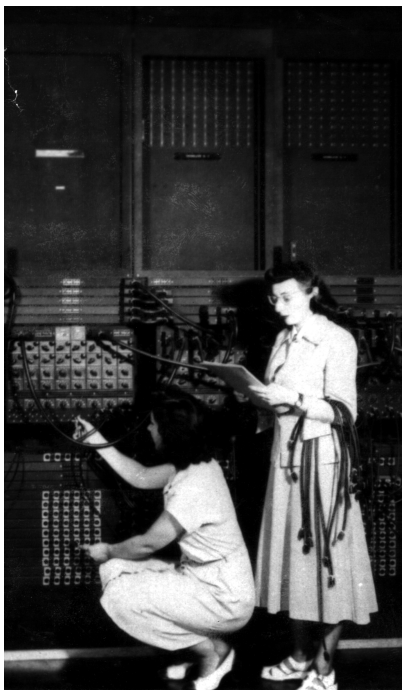
- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

1. Quiz: final exam replaces low/missing quizzes.
2. Brief overview of lab & programs for the week.
3. One-on-one help with instructors & UTAs.

Software Platforms:

Class Structure



First "computers"

ENIAC, 1945.

Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

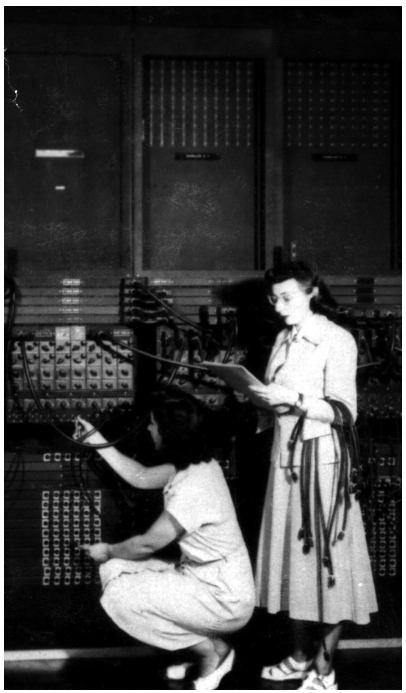
Recitation Section:

1. Quiz: final exam replaces low/missing quizzes.
2. Brief overview of lab & programs for the week.
3. One-on-one help with instructors & UTAs.

Software Platforms:

- Blackboard: visit ICIT for access issues.

Class Structure



First "computers"

ENIAC, 1945.

Lecture:

- Tuesdays, 11:10am-12:25pm, 118 North
- Lecture Slips: only help your grade, final exam replaces incomplete or missing slips
- Mix of explanation, challenges, & group work.
- Hard to ask questions in a large lecture, ask UTAs & instructors during group work.

Recitation Section:

1. Quiz: final exam replaces low/missing quizzes.
2. Brief overview of lab & programs for the week.
3. One-on-one help with instructors & UTAs.

Software Platforms:

- Blackboard: visit ICIT for access issues.
- Gradescope: email invite sent Sunday.

Introductions: Your Turn

- Introduce yourself to two classmates (that you have not met before).
- Write down names & interesting fact on lecture slip.

Today's Topics

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

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.
- 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.
- 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 in lab).

First Program: `Hello, World!`

Demo in `pythonTutor`

First Program: Hello, World!

```
#Name:  Thomas Hunter
#Date:  September 1, 2017
#This program prints:  Hello, World!

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

First Program: Hello, World!

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)
```

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
```

First Program: `Hello, World!`

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)

print("Hello, World!")                         ← Prints the string "Hello, World!" to the screen
```

- Output to the screen is: `Hello, World!`

First Program: `Hello, World!`

```
#Name:  Thomas Hunter                          ← These lines are comments
#Date:  September 1, 2017                      ← (for us, not computer to read)
#This program prints:  Hello, World!           ← (this one also)

print("Hello, World!")                         ← Prints the string "Hello, World!" to the screen
```

- Output to the screen is: `Hello, World!`
- 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,')
```

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")
```

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.")
```

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.

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.

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.

Turtles Introduction

- A simple, whimsical graphics package for Python

Turtles Introduction

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

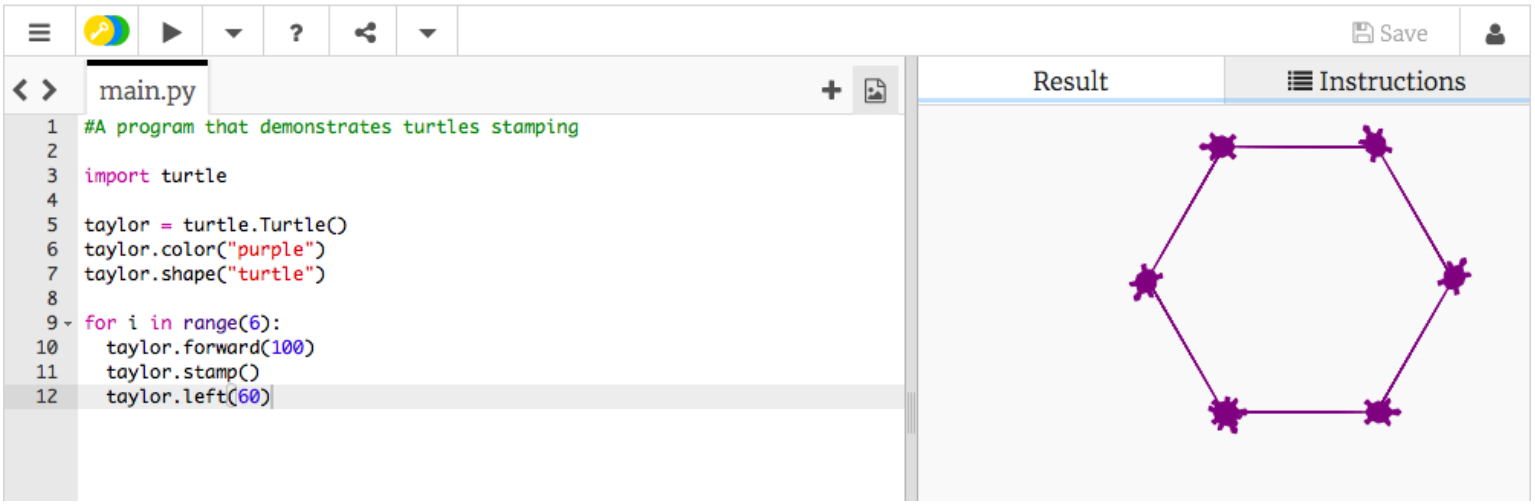
Turtles Introduction

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

Turtles Introduction

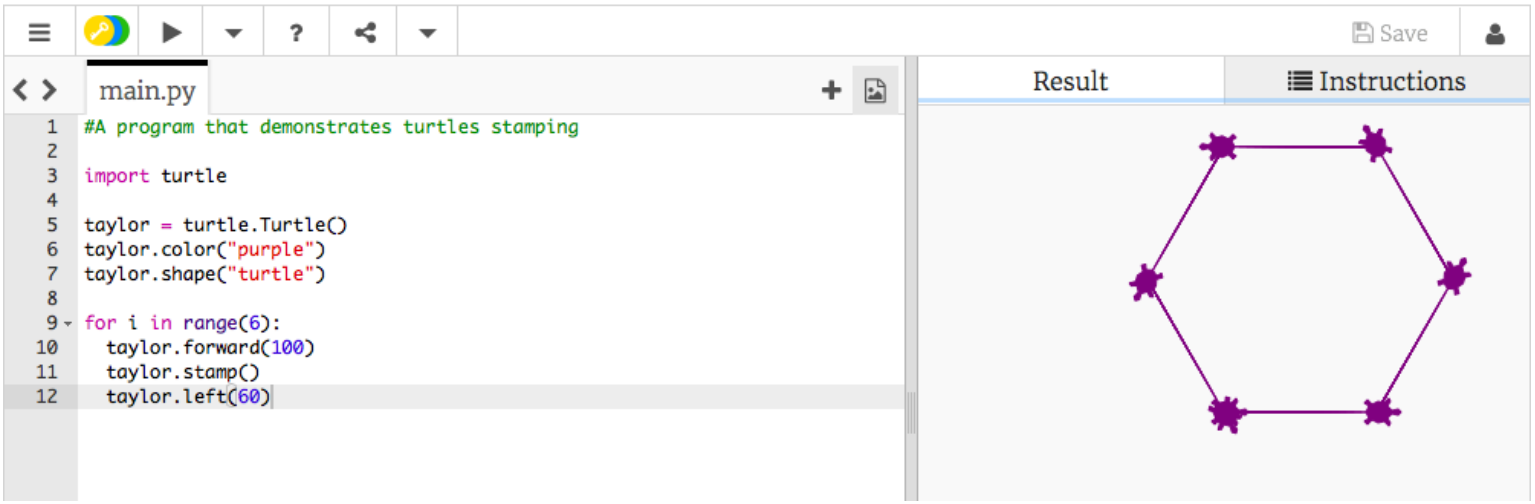
- A simple, whimsical graphics package for Python
- Dates back to Logos Turtles in the 1960s
- (Demo from webpage)
- (Fancier turtle demo)

Turtles Introduction



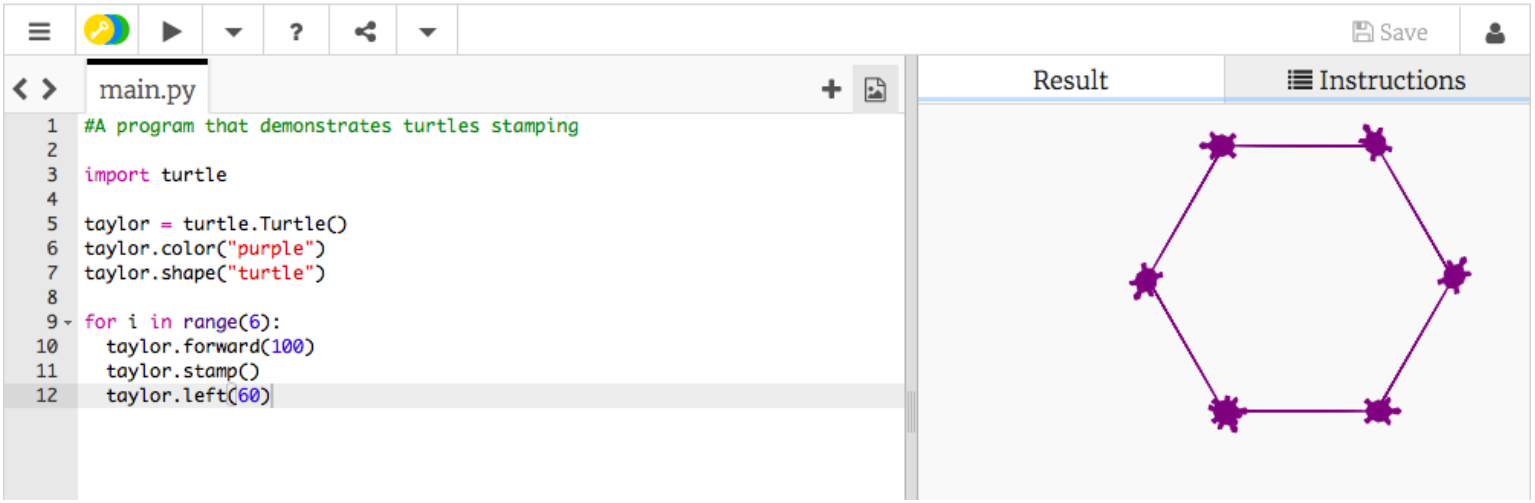
- Creates a turtle, called `taylor`

Turtles Introduction



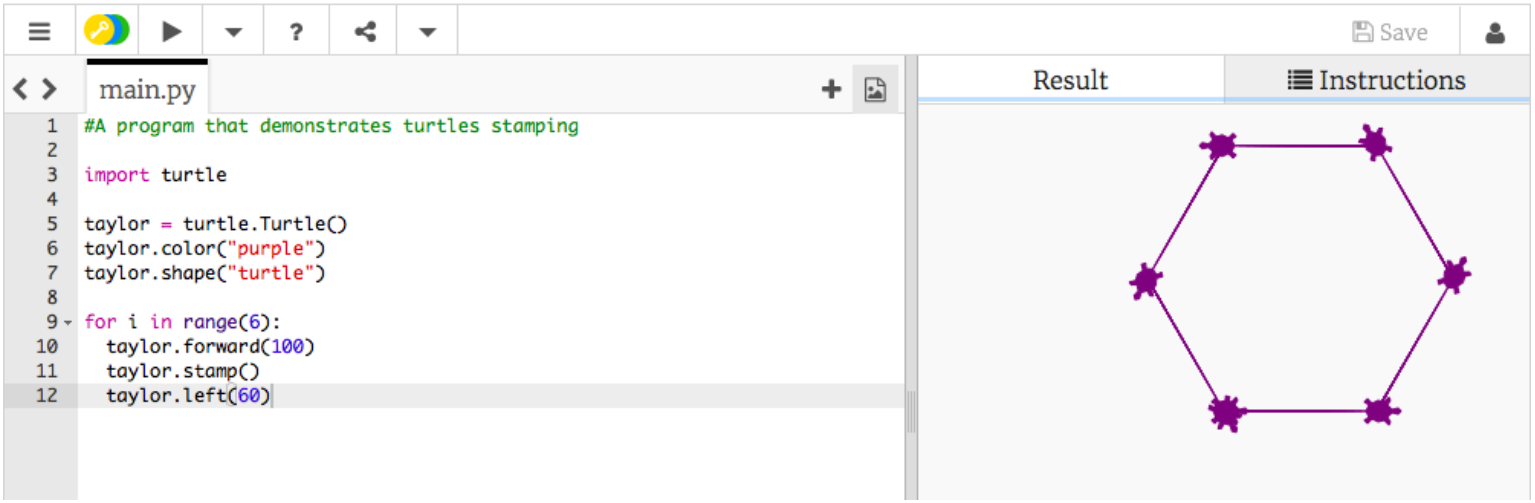
- Creates a turtle, called `taylor`
- Changes the color (to purple) and shape (to turtle-shaped)

Turtles Introduction



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

Turtles Introduction



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

Group Work

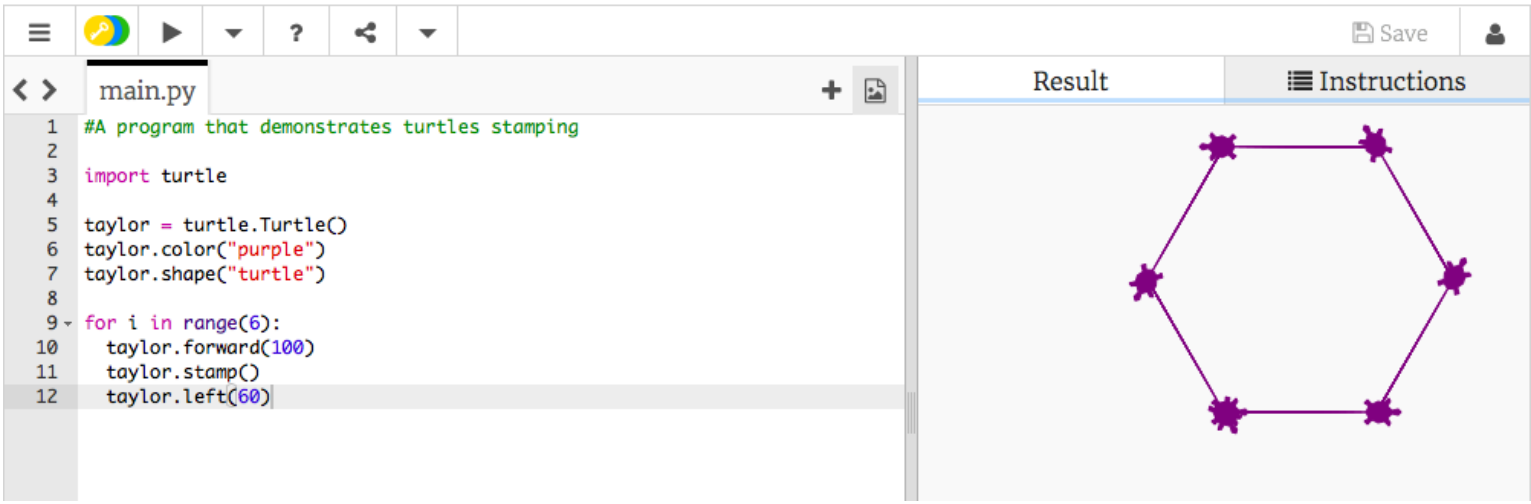
Working in pairs or triples:

1. Write a program that will draw a 10-sided polygon.
2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

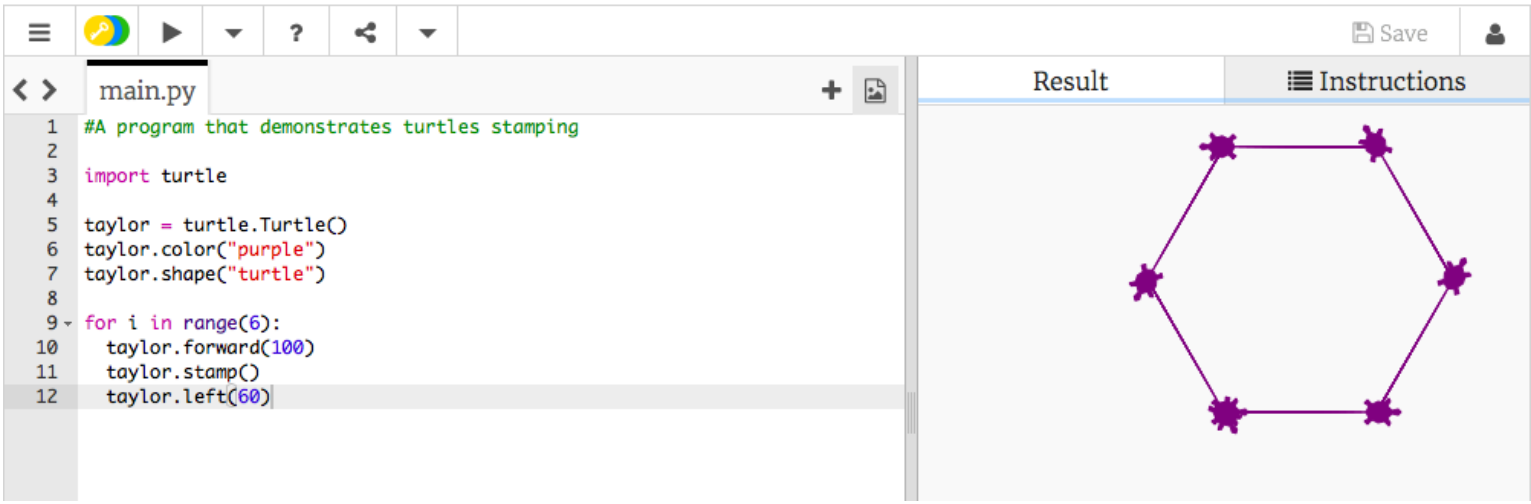
   three times.

Decagon Program



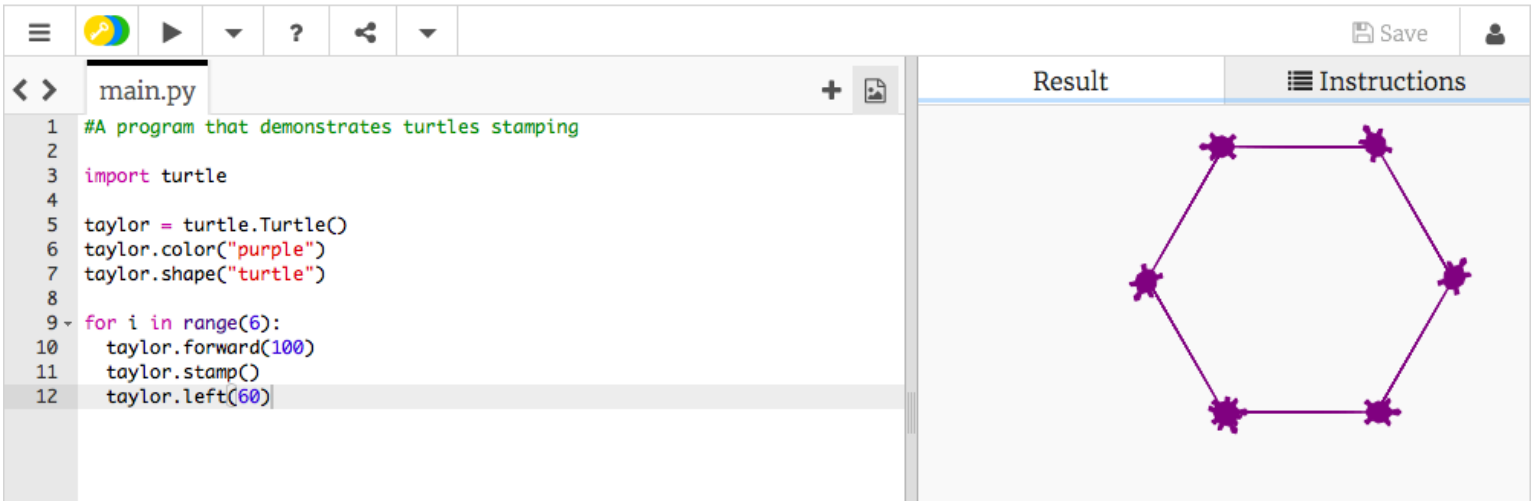
- Start with the hexagon program.

Decagon Program



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

Decagon Program



- Start with the hexagon program.
- Has 10 sides (instead of 6), so change the `range(6)` to `range(10)`.
- Makes 10 turns (instead of 6),
  so change the `taylor.left(60)` to `taylor.left(360/10)`.

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

- Repeats three times, so, use `range(3)`:

  ```
  for i in range(3):
  ```

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

- Repeats three times, so, use `range(3)`:

  ```
  for i in range(3):
  ```

- Instead of turtle commands, repeating a print statement.

Work Program

2. Write a program that will repeat the line:

   ```
   I'm lookin' for a mind at work!
   ```

   three times.

- Repeats three times, so, use `range(3)`:

  ```
  for i in range(3):
  ```

- Instead of turtle commands, repeating a print statement.
- Completed program:

  ```
  # Your name here!
  for i in range(3):
      print("I'm lookin' for a mind at work!")
  ```

What is an Algorithm?

From our textbook:

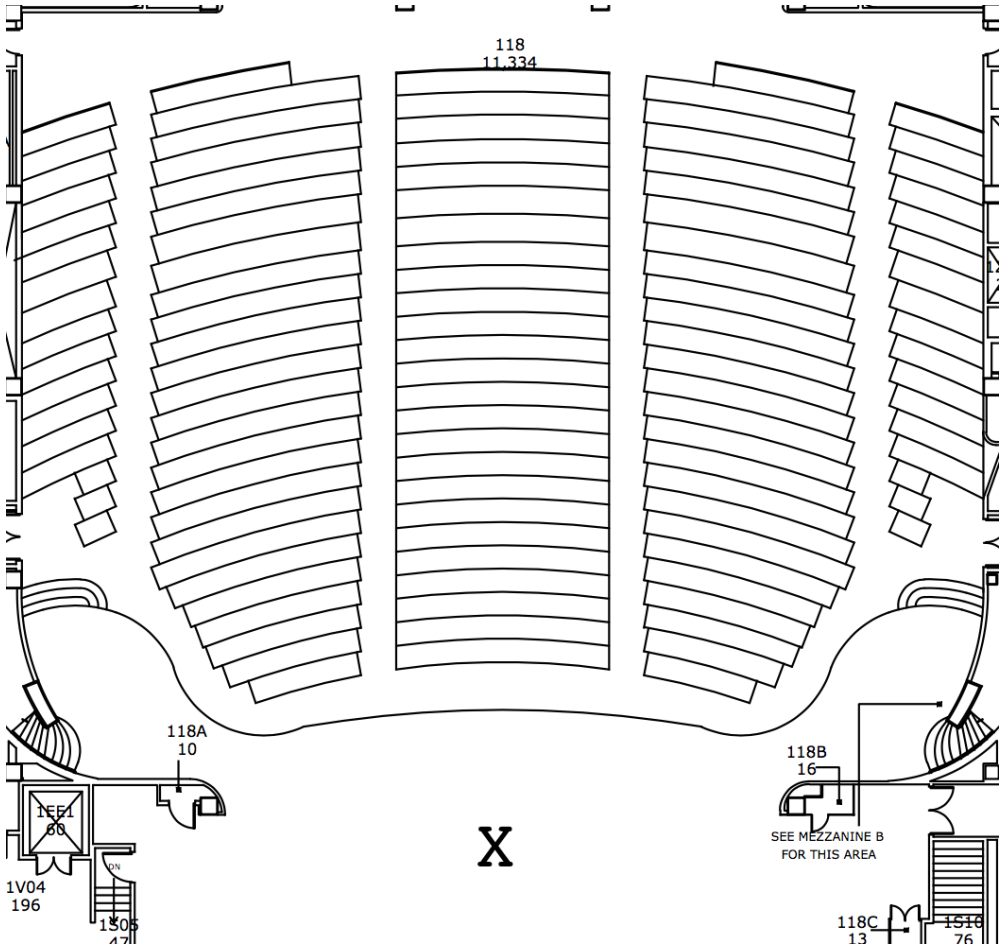
- An algorithm is a process or set of rules to be followed to solve a problem.

What is an Algorithm?

From our textbook:

- An algorithm is a process or set of rules 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 followed by a computer.

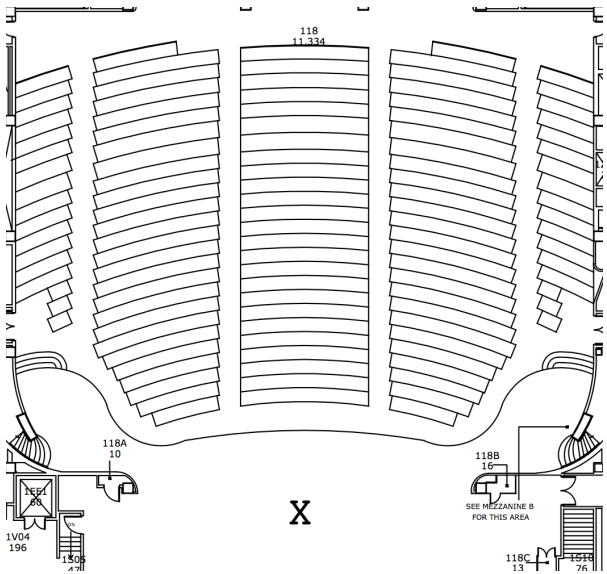
Group Work



Working in pairs or triples:

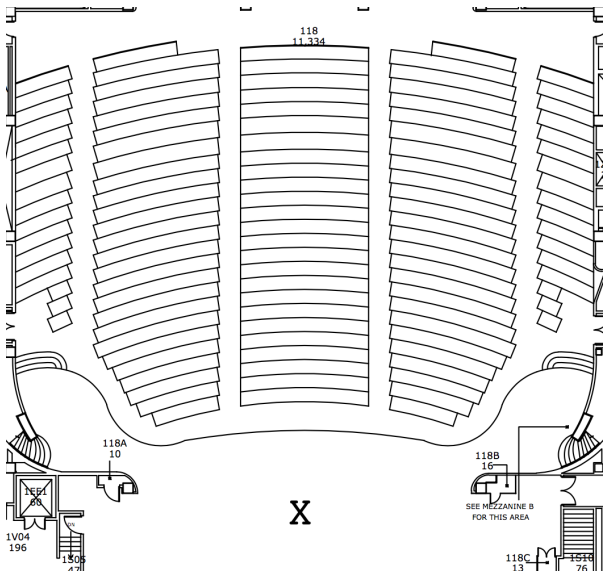
1. On the floorplan, mark your current location.
2. Write step-by-step directions to get to/from `X`.

Group Work



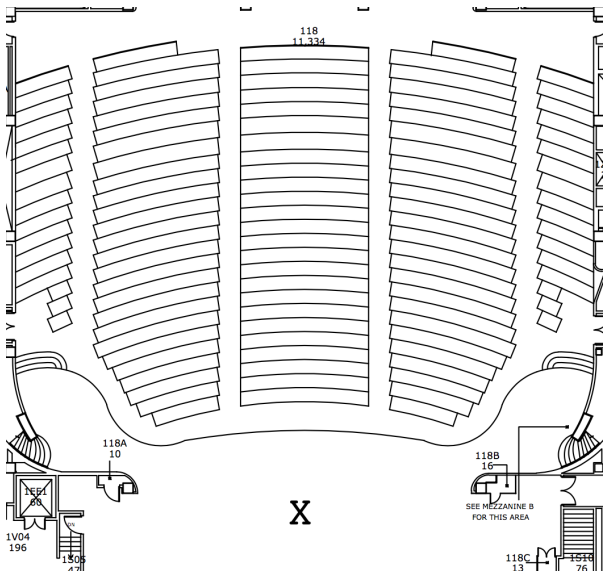
- Find another group, near you, that's going in the "opposite" way.

Group Work



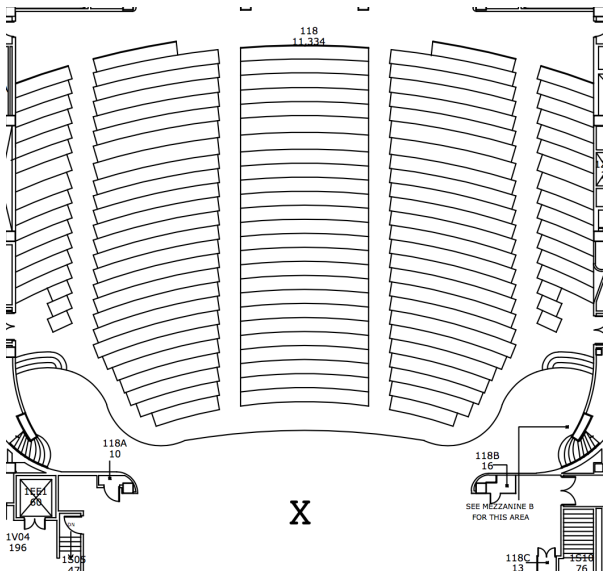
- Find another group, near you, that's going in the "opposite" way.
- Follow the directions to get to `X`.

Group Work



- Find another group, near you, that's going in the "opposite" way.
- Follow the directions to get to `X`.
- Follow the other set of directions form `X` back to your seat.

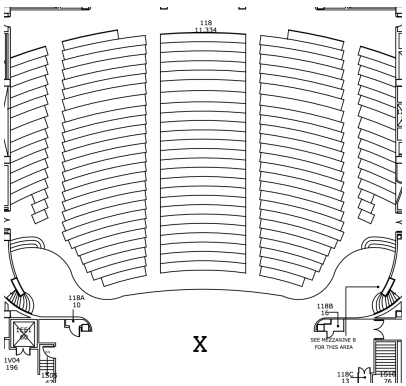
Group Work



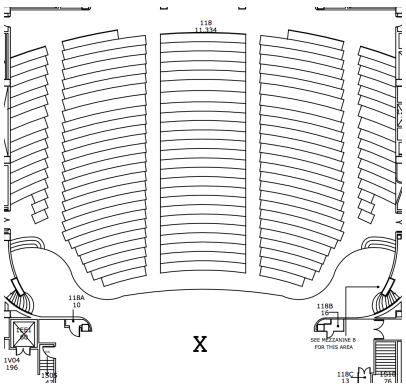
- Find another group, near you, that's going in the "opposite" way.
- Follow the directions to get to `X`.
- Follow the other set of directions form `X` back to your seat.
- Annotate any changes needed to the directions.

Recap

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

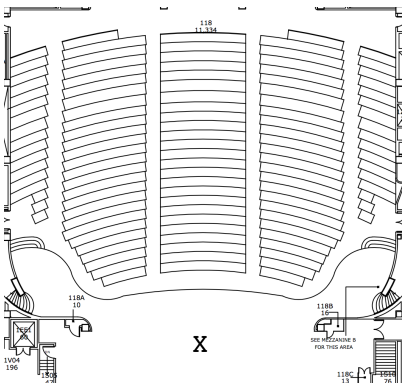


Recap



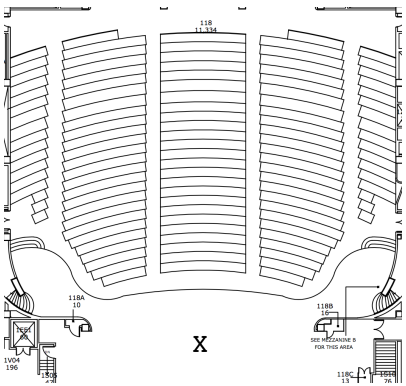
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.

Recap



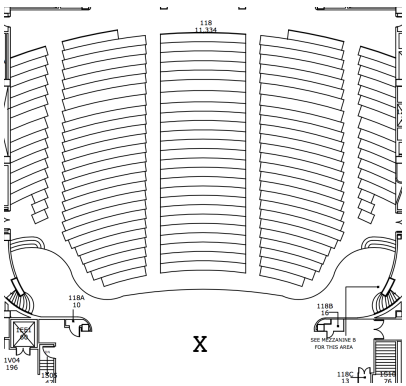
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:

Recap



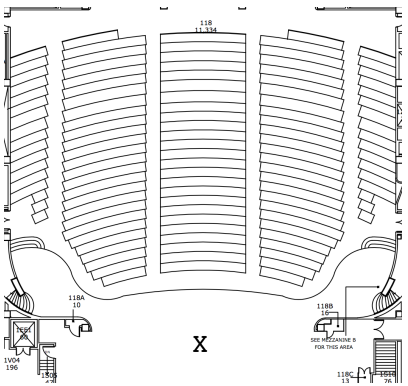
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,

Recap



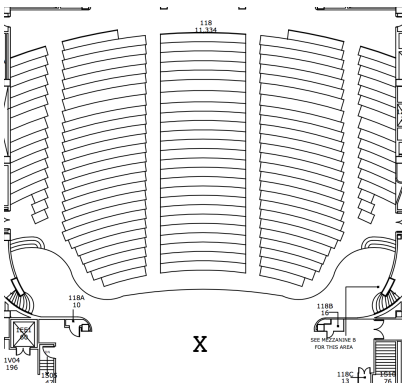
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,

Recap



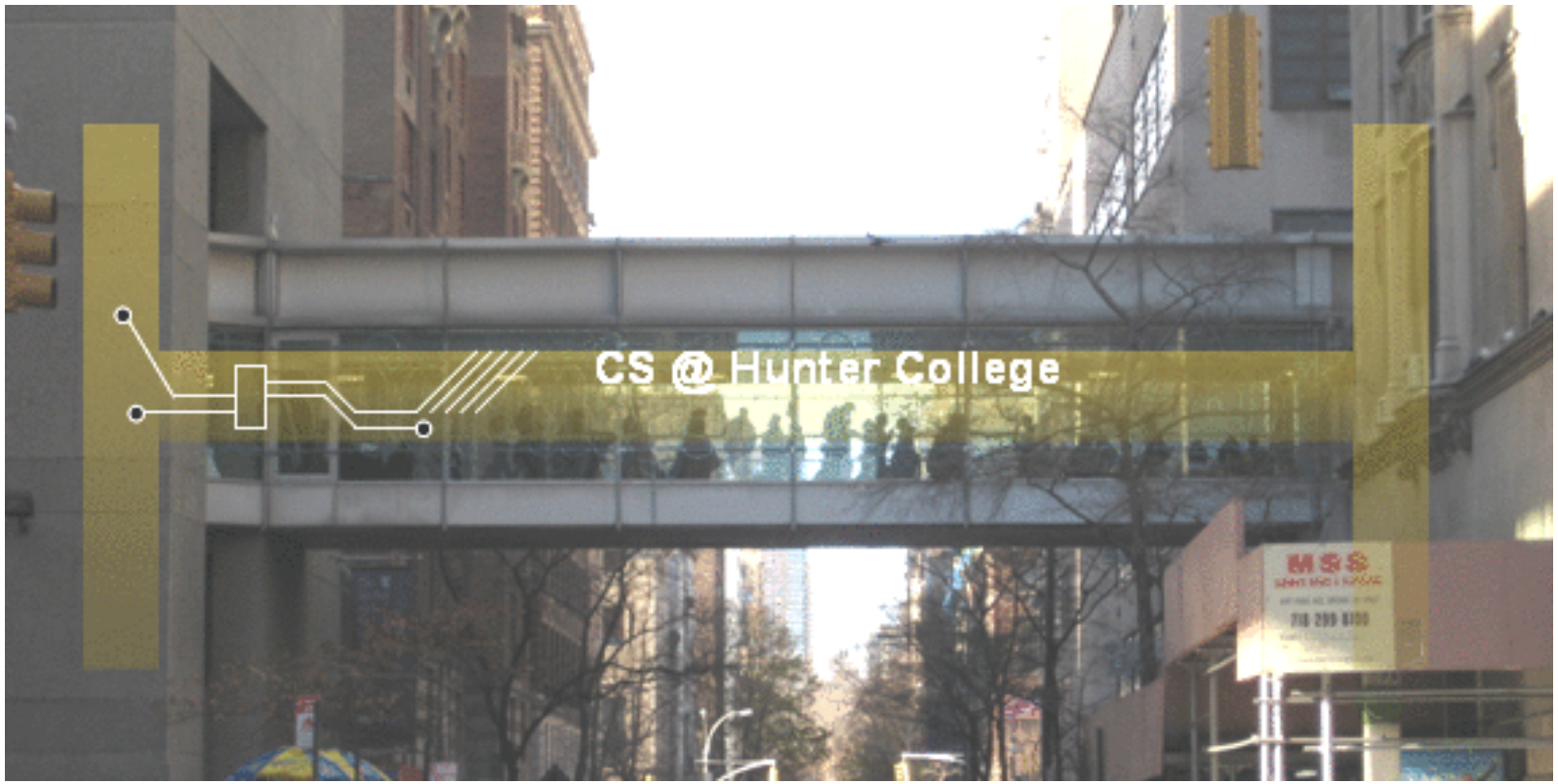
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Writing precise algorithms is difficult.
- In Python, we introduced:
  - strings, or sequences of characters,
  - `print()` statements,
  - `for`-loops with `range()` statements, &

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.

Lecture Slips & Writing Boards



- Turn in lecture slips & writing boards as you leave...