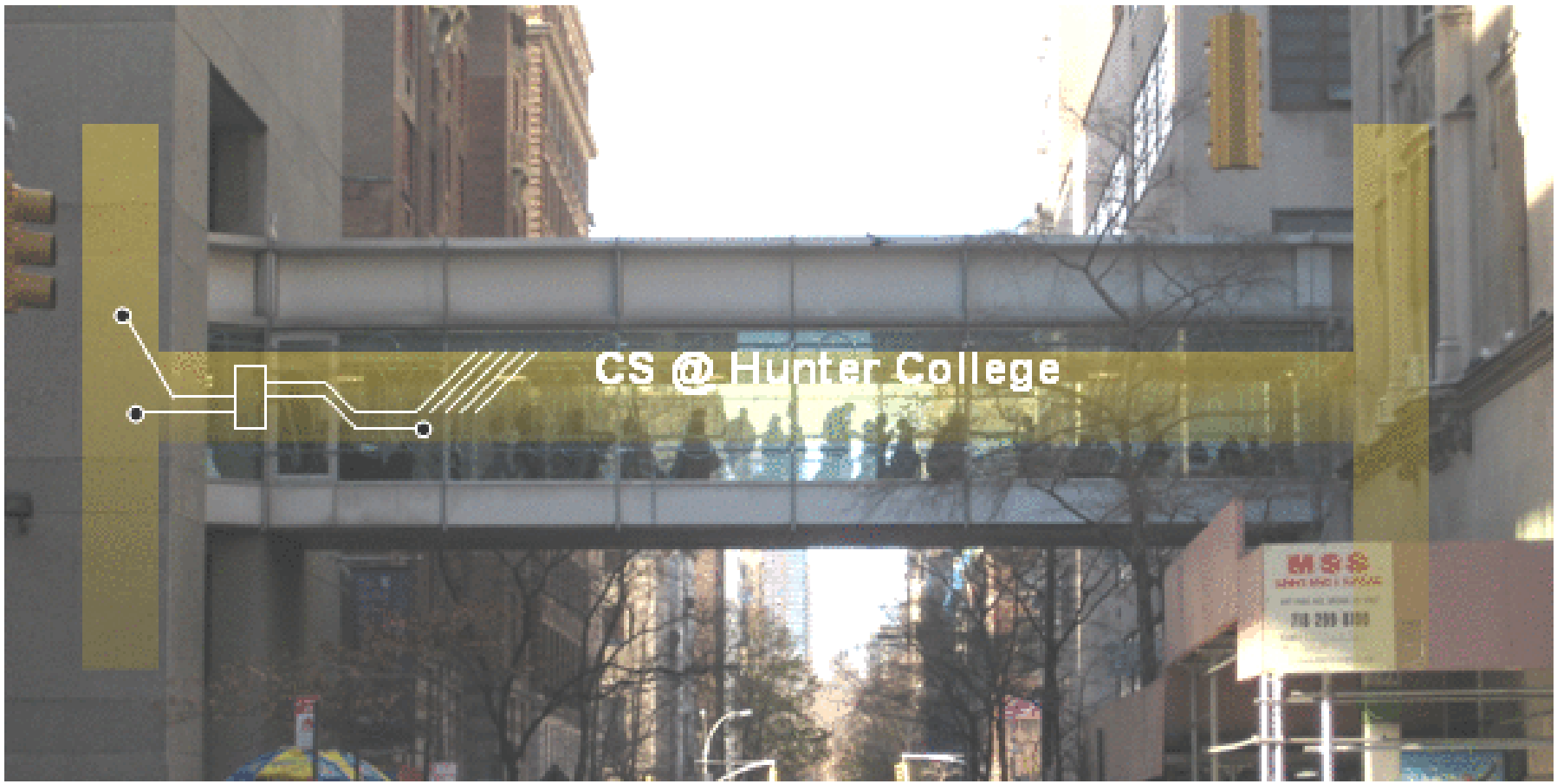
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- *Postponed until next week:*
  CS Survey: Anna Whitney
  Google Storage Infrastructure Team

Announcements

- *Postponed until next week:*
  CS Survey: Anna Whitney
  Google Storage Infrastructure Team
- Instead: Final Exam Review
  Extra Handout: fall exam
  (similar, but due to typos, not identical to exam given)

Today's Topics

- Recap: Folium
- Indefinite loops
- Design Patterns: Max (Min)
- Final Exam Overview

Today's Topics

- **Recap: Folium**
- Indefinite loops
- Design Patterns: Max (Min)
- Final Exam Overview

In Pairs or Triples:

What does this code do?

```
import folium
import pandas as pd

cuny = pd.read_csv('cunyLocations.csv')
mapCUNY = folium.Map(location=[40.75, -74.125])

for index,row in cuny.iterrows():
    lat = row["Latitude"]
    lon = row["Longitude"]
    name = row["Campus"]
    if row["College or Institution Type"] == "Senior Colleges":
        collegeIcon = folium.Icon(color="purple")
    else:
        collegeIcon = folium.Icon(color="blue")
    newMarker = folium.Marker([lat, lon], popup=name, icon=collegeIcon)
    newMarker.add_to(mapCUNY)

mapCUNY.save(outfile='cunyLocationsSenior.html')
```

Folium example

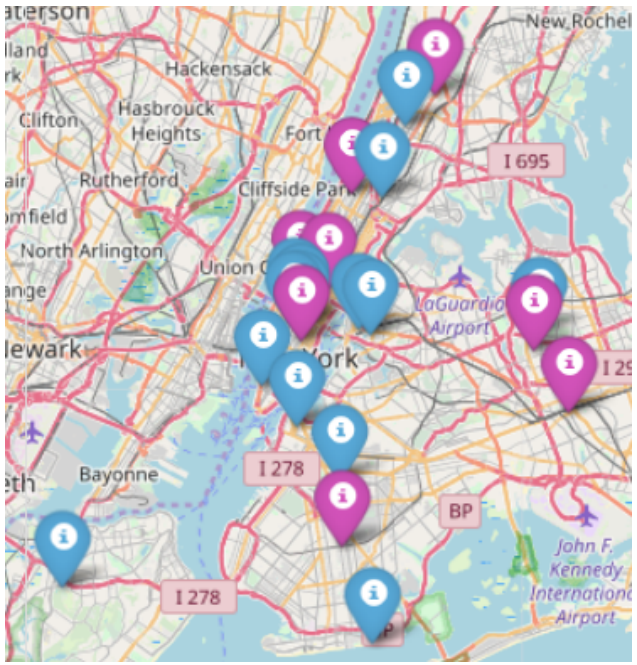
What does this code do?

```
import folium
import pandas as pd

cuny = pd.read_csv('cunyLocations.csv')
mapCUNY = folium.Map(location=[40.75, -74.125])

for index,row in cuny.iterrows():
    lat = row["Latitude"]
    lon = row["Longitude"]
    name = row["Campus"]
    if row["College or Institution Type"] == "Senior Colleges":
         collegeIcon = folium.Icon(color="purple")
    else:
         collegeIcon = folium.Icon(color="blue")
    newMarker = folium.Marker([lat, lon], popup=name, icon=collegeIcon)
    newMarker.add_to(mapCUNY)

mapCUNY.save(outfile='cunyLocationsSenior.html')
```

Folium example

What does this code do?

```
import folium
import pandas as pd

cuny = pd.read_csv('cunyLocations.csv')
mapCUNY = folium.Map(location=[40.75, -74.125])

for index,row in cuny.iterrows():
    lat = row["Latitude"]
    lon = row["Longitude"]
    name = row["Campus"]
    if row["College or Institution Type"] == "Senior Colleges":
         collegeIcon = folium.Icon(color="purple")
    else:
         collegeIcon = folium.Icon(color="blue")
    newMarker = folium.Marker([lat, lon], popup=name, icon=collegeIcon)
    newMarker.add_to(mapCUNY)

mapCUNY.save(outfile='cunyLocationsSenior.html')
```



Folium

- A module for making HTML maps.

Folium

- A module for making HTML maps.
- It's a Python interface to the popular `leaflet.js`.

Folium

- A module for making HTML maps.
- It's a Python interface to the popular `leaflet.js`.
- Outputs `.html` files which you can open in a browser.

Folium

- A module for making HTML maps.
- It's a Python interface to the popular `leaflet.js`.
- Outputs `.html` files which you can open in a browser.
- An extra step:

Folium

- A module for making HTML maps.
- It's a Python interface to the popular `leaflet.js`.
- Outputs `.html` files which you can open in a browser.
- An extra step:

Write code. → *Run program.* → *Open .html in browser.*

Today's Topics

- Recap: Folium
- **Indefinite loops**
- Design Patterns: Max (Min)
- Python Recap

In Pairs or Triples:

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

Coding

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number..

Coding

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

```
def getYear():
```

Coding

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

```
def getYear():




    return(num)
```

Coding

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

```
def getYear():
    num = 0



    return(num)
```

Coding

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

```
def getYear():
    num = 0
    while num <= 2000 or num >= 2018:



    return(num)
```

Coding

- Write a function that asks a user for number after 2000 but before 2018. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

```
def getYear():
    num = 0
    while num <= 2000 or num >= 2018:
        num = int(input('Enter a number > 2000 & < 2018'))

    return(num)
```

Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

- Indefinite loops repeat as long as the condition is true.

Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

- Indefinite loops repeat as long as the condition is true.
- Could execute the body of the loop zero times, 10 times, infinite number of times.

Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

- Indefinite loops repeat as long as the condition is true.
- Could execute the body of the loop zero times, 10 times, infinite number of times.
- The condition determines how many times.

Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

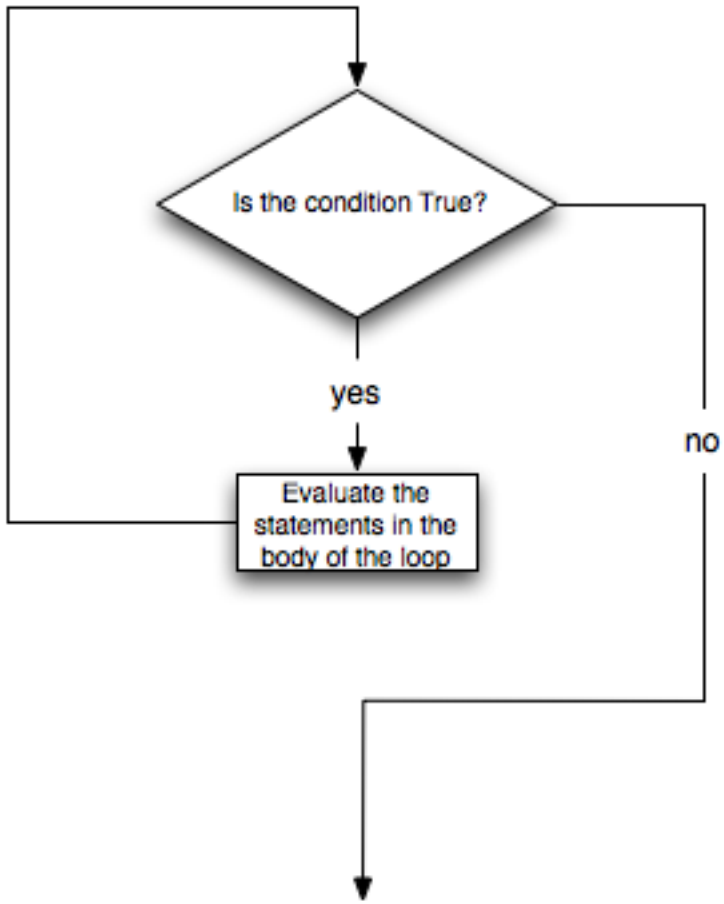
- Indefinite loops repeat as long as the condition is true.
- Could execute the body of the loop zero times, 10 times, infinite number of times.
- The condition determines how many times.
- Very useful for checking input, simulations, and games.

Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

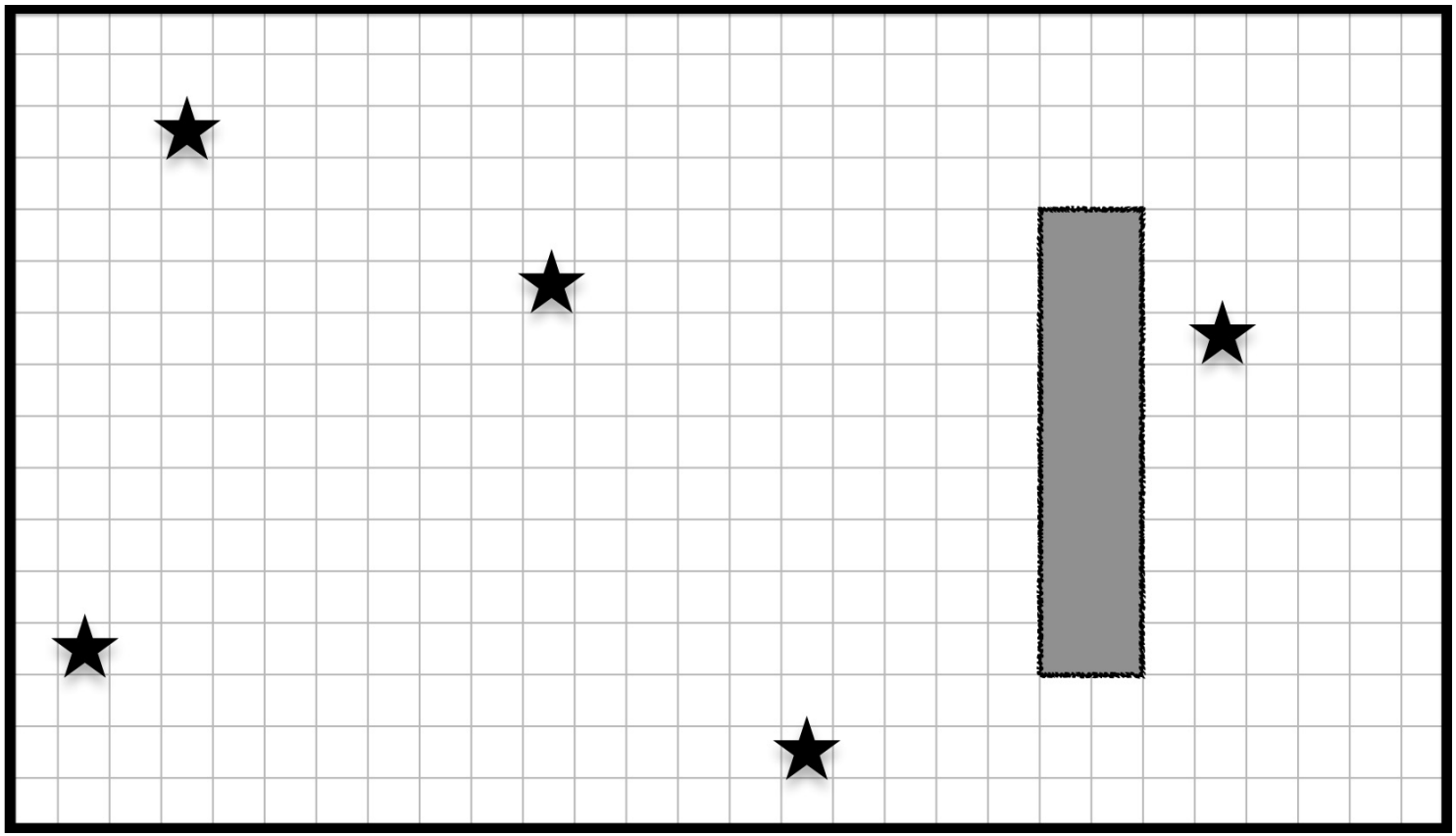
Indefinite Loops

```
import turtle
import random

trey = turtle.Turtle()
trey.speed(10)

for i in range(100):
  trey.forward(10)
  a = random.randrange(0,360,90)
  trey.right(a)
```

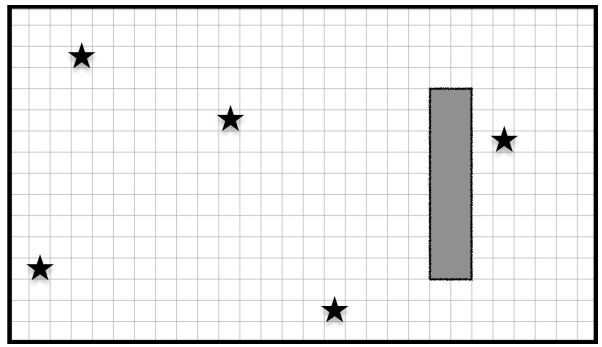


From Last Time: Design Challenge

Collect all five stars (locations randomly generated):

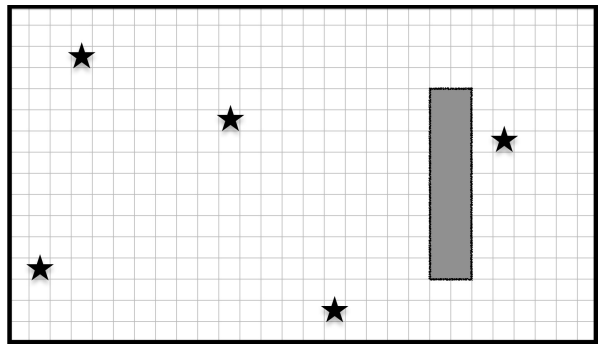


From Last Time: Design Challenge



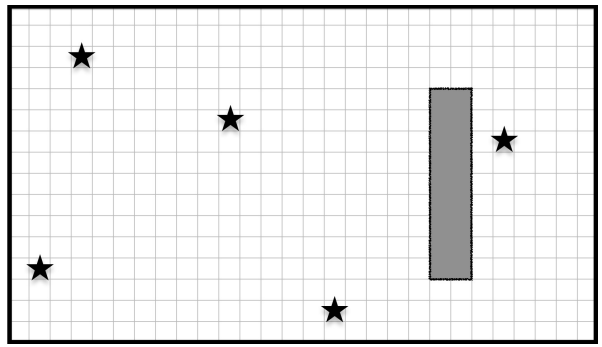
- Possible approaches:

From Last Time: Design Challenge



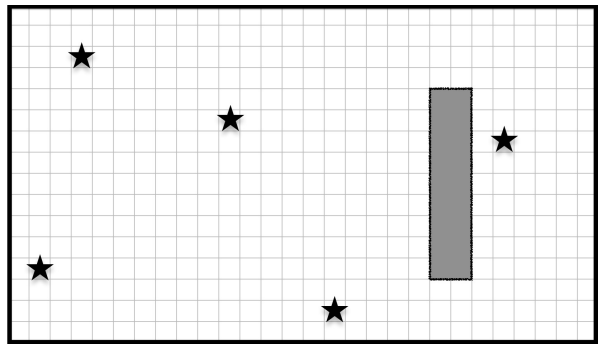
- Possible approaches:
  - Randomly wander until all 5 collected, or

From Last Time: Design Challenge



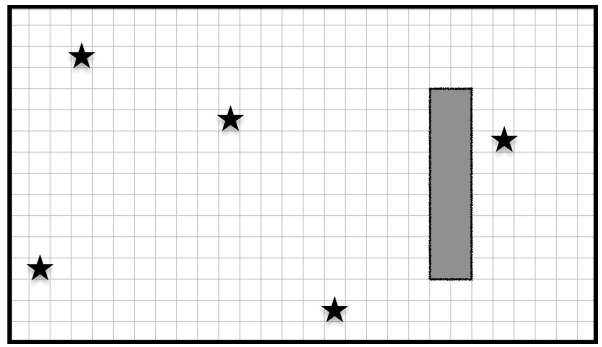
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.

From Last Time: Design Challenge



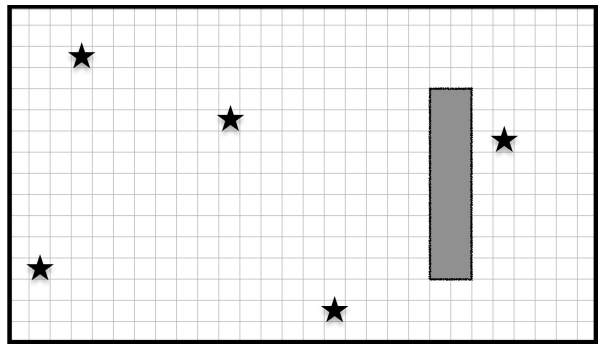
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'

From Last Time: Design Challenge



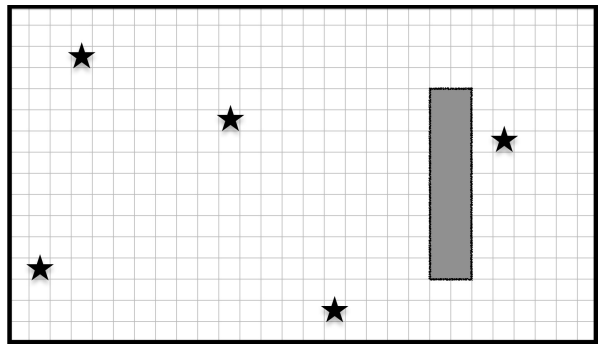
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.

From Last Time: Design Challenge



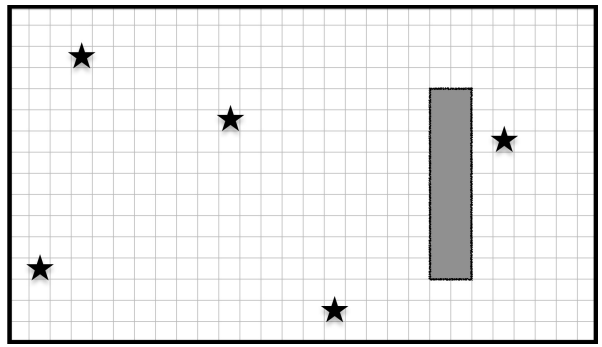
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.

From Last Time: Design Challenge



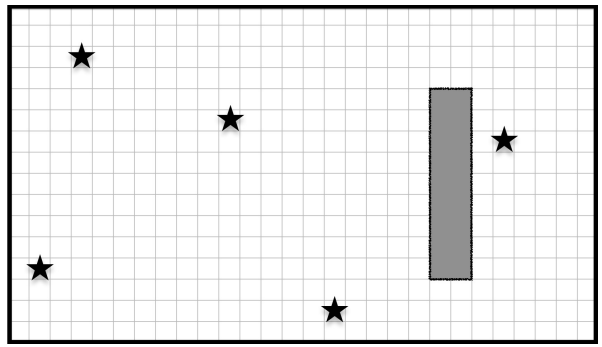
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible algorithms: while numStars < 5:

From Last Time: Design Challenge



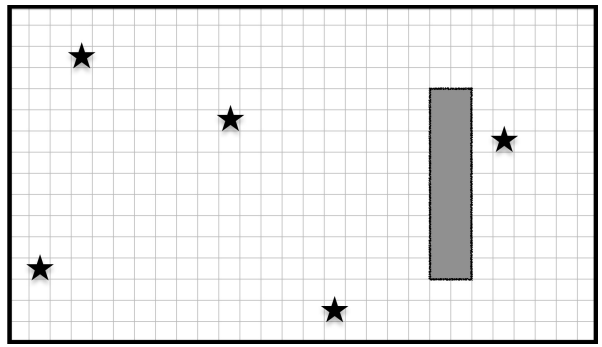
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible algorithms: while numStars < 5:
  - Move forward.

From Last Time: Design Challenge



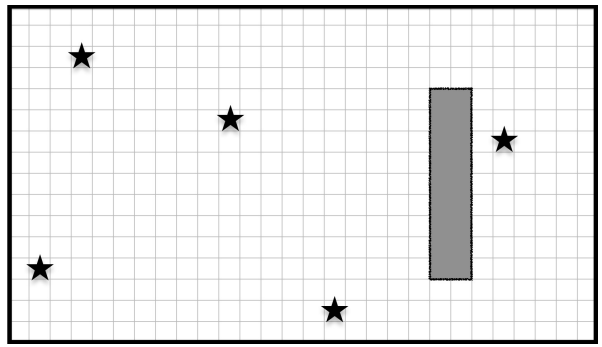
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible algorithms: while numStars < 5:
  - Move forward.
  - If wall, mark 0 in map, randomly turn left or right.

From Last Time: Design Challenge



- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible algorithms: while numStars < 5:
  - Move forward.
  - If wall, mark 0 in map, randomly turn left or right.
  - If star, mark 1 in map and add 1 to numStars.

From Last Time: Design Challenge



- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible algorithms: while numStars < 5:
  - Move forward.
  - If wall, mark 0 in map, randomly turn left or right.
  - If star, mark 1 in map and add 1 to numStars.
  - Otherwise, mark 2 in map that it's an empty square.

In Pairs or Triples

Predict what this code does:

```
#Random search
import turtle
import random
tess = turtle.Turtle()
tess.color('steelBlue')
tess.shape('turtle')
tess.penup()
#Start off screen:
tess.goto(-250,-250)
#Remember:  abs(x) < 25 means absolute value: -25 < x < 25
while abs(tess.xcor()) > 25 or abs(tess.ycor()) > 25:
  x = random.randrange(-200,200)
  y = random.randrange(-200,200)
  tess.goto(x,y)
  tess.stamp()
  print(tess.xcor(), tess.ycor())
print('Found the center!')
```

Trinket Demo

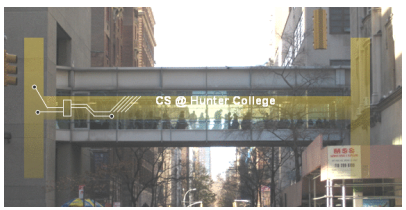
```
#Random search
import turtle
import random
tess = turtle.Turtle()
tess.color('steelBlue')
tess.shape('turtle')
tess.penup()
#Start off screen:
tess.goto(-250,-250)
#Remember:  abs(x) < 25 means absolute value: -25 < x < 25
while abs(tess.xcor()) > 25 or abs(tess.ycor()) > 25:
  x = random.randrange(-200,200)
  y = random.randrange(-200,200)
  tess.goto(x,y)
  tess.stamp()
  print(tess.xcor(), tess.ycor())
print('Found the center!')
```

(Demo with `trinket`)

Today's Topics

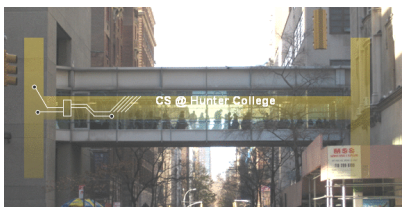
- Recap: Folium
- Indefinite loops
- **Design Patterns: Max (Min)**
- Python Recap

Design Patterns



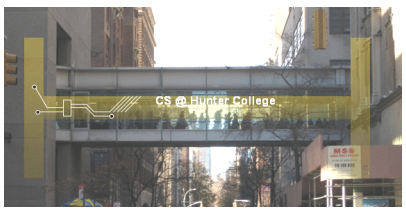
- A **design pattern** is a standard algorithm or approach for solving a common problem.

Design Patterns



- A **design pattern** is a standard algorithm or approach for solving a common problem.
- The pattern is independent of the programming language.

Design Patterns



- A **design pattern** is a standard algorithm or approach for solving a common problem.
- The pattern is independent of the programming language.
- Can think of as a master recipe, with variations for different situations.

In Pairs or Triples:

Predict what the code will do:

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

Python Tutor

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

(Demo with pythonTutor)

Max Design Pattern

- Set a variable to the smallest value.

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

Max Design Pattern

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

- Set a variable to the smallest value.
- Loop through the list,

Max Design Pattern

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

- Set a variable to the smallest value.
- Loop through the list,
- If the current number is larger, update your variable.

Max Design Pattern

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

- Set a variable to the smallest value.
- Loop through the list,
- If the current number is larger, update your variable.
- Print/return the largest number found.

Max Design Pattern

```
nums = [1,4,10,6,5,42,9,8,12]

maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print('The max is', maxNum)
```

- Set a variable to the smallest value.
- Loop through the list,
- If the current number is larger, update your variable.
- Print/return the largest number found.
- Similar idea works for finding the minimum value.

In Pairs or Triples:

Answer the following questions on your lecture slip:

Of the students in the room,

- Whose name comes first alphabetically?
- Whose name comes last alphabetically?
- Is there someone in the room with your initials?

In Pairs or Triples:

Design a program that takes a CSV file and a set of initials:

- Whose name comes first alphabetically?
- Whose name comes last alphabetically?
- Is there someone in the room with your initials?

Design Question: Find first alphabetically

- In Pandas, lovely built-in functions:

Design Question: Find first alphabetically

- In Pandas, lovely built-in functions:
  - `df.sort_values('First Name')` and
  - `df['First Name'].min()`

Design Question: Find first alphabetically

- In Pandas, lovely built-in functions:
  - `df.sort_values('First Name')` and
  - `df['First Name'].min()`
- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.

Design Question: Find first alphabetically

- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.
  - Print/return X.

Design Question: Find Matching Initials

- How do we stop, if we find a match?

Design Question: Find Matching Initials

- How do we stop, if we find a match?
- Change the loop to be indefinite (i.e. `while` loop):
  - Set a variable to `found = False`

Design Question: Find Matching Initials

- How do we stop, if we find a match?
- Change the loop to be indefinite (i.e. `while` loop):
  - Set a variable to `found = False`
  - while there are items in the list and not `found`

Design Question: Find Matching Initials

- How do we stop, if we find a match?
- Change the loop to be indefinite (i.e. `while` loop):
  - Set a variable to `found = False`
  - while there are items in the list and not `found`
    - ★ If item matches your value, set `found = True`

Design Question: Find Matching Initials

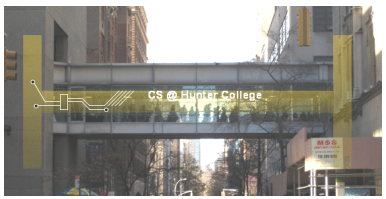
- How do we stop, if we find a match?
- Change the loop to be indefinite (i.e. `while` loop):
  - Set a variable to `found = False`
  - while there are items in the list and not `found`
    - ★ If item matches your value, set `found = True`
  - Print/return value.

Today's Topics

- Recap: Folium
- Indefinite loops
- Design Patterns: Max (Min)
- **Final Exam Overview**

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).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.

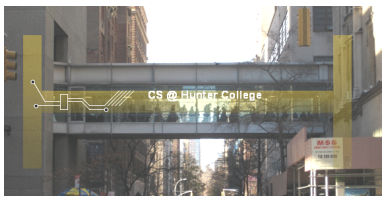
Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.
- Introduced the max design pattern.

Recap



- On lecture slip, write down a topic you wish we had spent more time (and why).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.
- Introduced the max design pattern.
- 10 minute overview of the first 10 weeks of class.

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.
- Introduced the max design pattern.
- 10 minute overview of the first 10 weeks of class.
- When possible, design so that your code is flexible to be reused ("code reuse").

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.
- Introduced the max design pattern.
- 10 minute overview of the first 10 weeks of class.
- When possible, design so that your code is flexible to be reused ("code reuse").
- Pass your lecture slips to the aisles for the UTAs to collect.

Final Overview: Format

- The exam is 2 hours long.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - 10 questions, each worth 10 points.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - 10 questions, each worth 10 points.
  - Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
  - Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - 10 questions, each worth 10 points.
  - Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
  - Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
  - More on logistics after spring break.

Final Overview: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - 10 questions, each worth 10 points.
  - Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
  - Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
  - More on logistics after spring break.
- Past exams available on webpage (includes answer keys).

Exam from Last Semester

FINAL EXAM, VERSION 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

19 December 2018

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, smart watch, or other electronic device.
- Do not open this exam until instructed to do so.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmpID:
Email:
Signature: