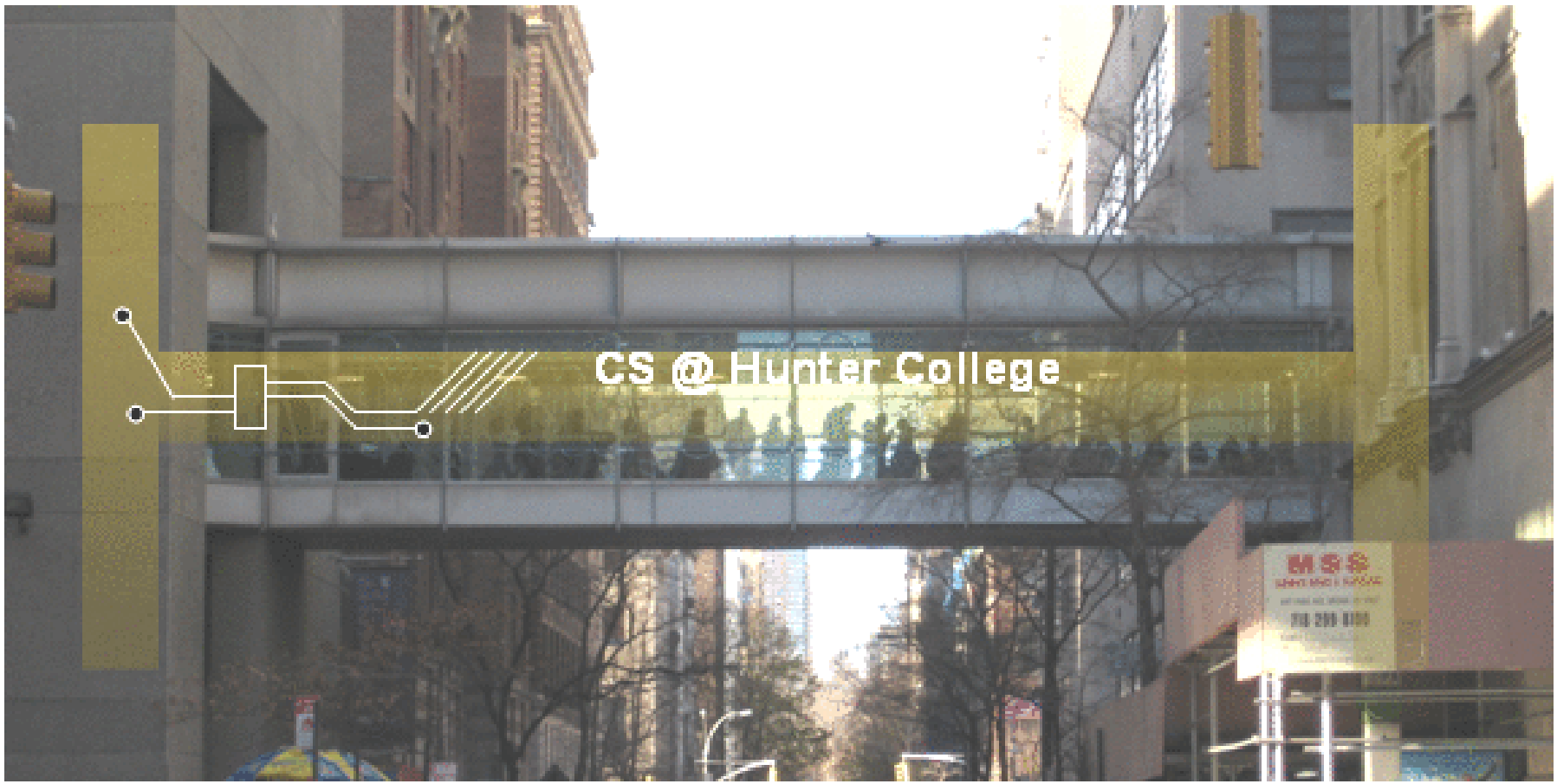
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- CS Survey: Prof. Susan Epstein
  Machine Learning

Announcements

- CS Survey: Prof. Susan Epstein
  Machine Learning
- Popular request from wrap-ups: Unix

Announcements

- CS Survey: Prof. Susan Epstein
  Machine Learning
- Popular request from wrap-ups: Unix
  End of lecture: focus on Unix

Today's Topics

- Recap: Folium
- Indefinite loops
- Design Patterns: Max (Min)
- CS Survey

Today's Topics

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

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

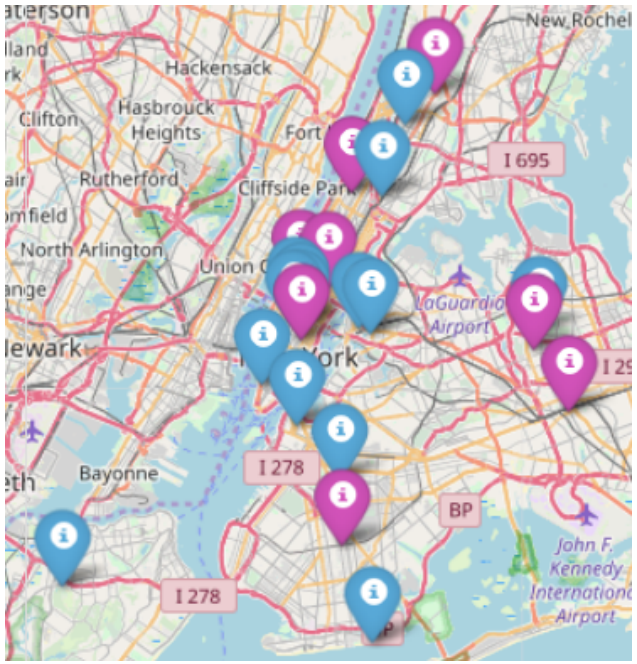
What does this code do?

```
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. $\rightarrow$ *Run program.* $\rightarrow$ *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)
```

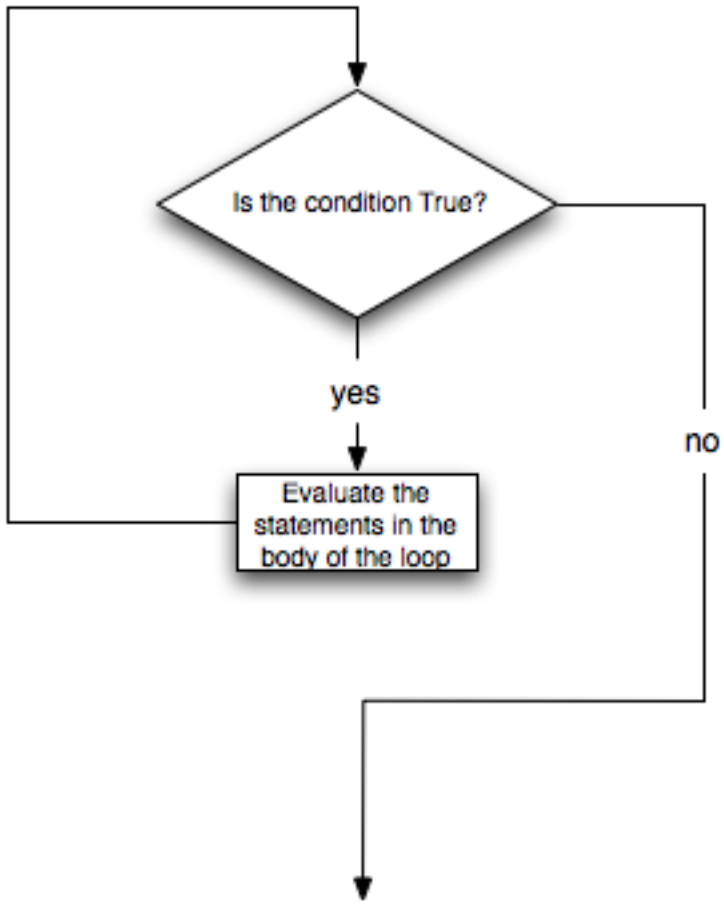
- Indefinite loops repeat as long as the condition is true.
- Could execute the body of the loop zero times, 10 times, infinite number of times.
- The condition determines how many times.
- Very useful for checking input, simulations, and games.

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)
```



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

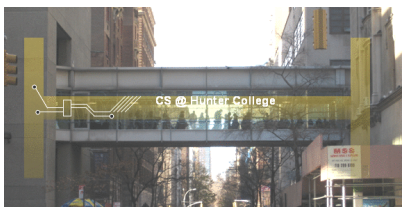
```
#Random search
import turtle
import random
tess = turtle.Turtle()
tess.color('steelBlue')
tess.shape('turtle')
tess.penup()
#Start off screen:
tess.goto(-250,-250)
#Remember:  abs(x) < 25 means absolute value: -25 < x < 25
while abs(tess.xcor()) > 25 or abs(tess.ycor()) > 25:
  x = random.randrange(-200,200)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

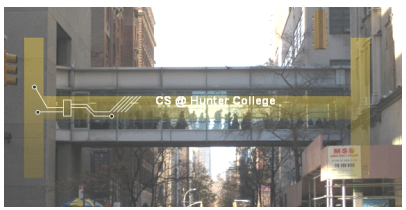
- Recap: Folium
- Indefinite loops
- **Design Patterns: Max (Min)**
- Python Recap

Design Patterns



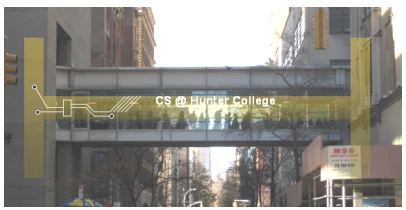
- A **design pattern** is a standard algorithm or approach for solving a common problem.

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

```
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.

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)
```

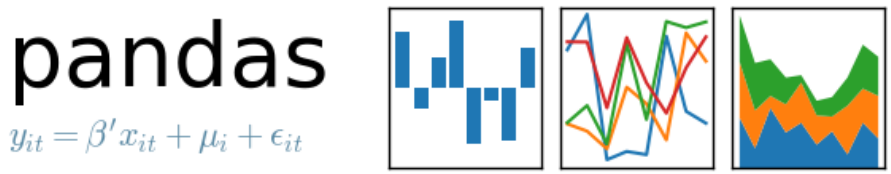
- Set a variable to the smallest value.
- Loop through the list,
- If the current number is larger, update your variable.
- Print/return the largest number found.

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)
```

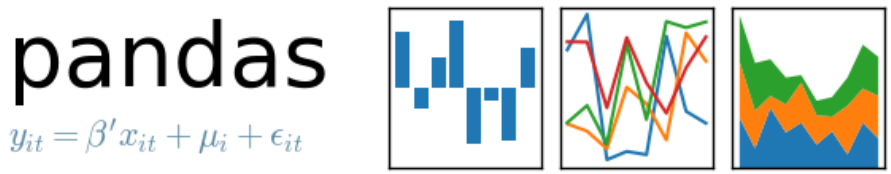
- Set a variable to the smallest value.
- Loop through the list,
- If the current number is larger, update your variable.
- Print/return the largest number found.
- Similar idea works for finding the minimum value.

Pandas: Minimum Values



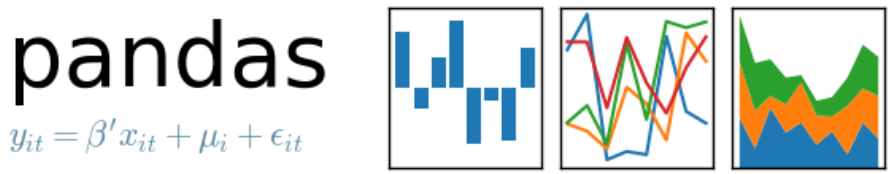
- In Pandas, lovely built-in functions:

Pandas: Minimum Values



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

Pandas: Minimum Values

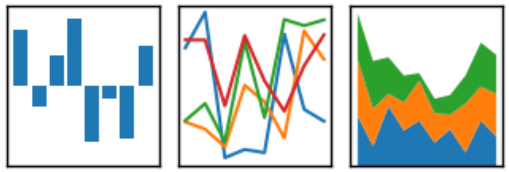


- In Pandas, lovely built-in functions:
  - `df.sort_values('First Name')` and
  - `df['First Name'].min()`
- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

pandas

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

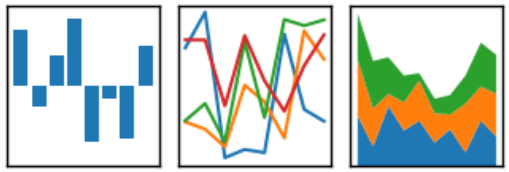


- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

pandas

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

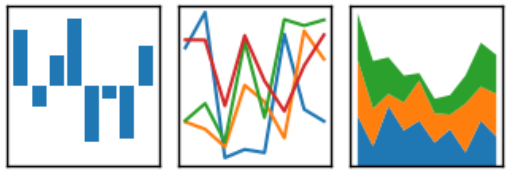


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max

Design Question: Find first alphabetically

pandas

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

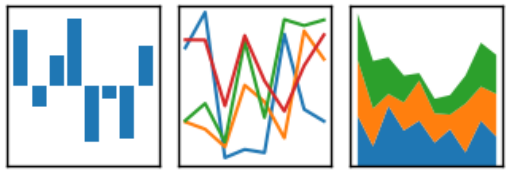


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).

Design Question: Find first alphabetically

pandas

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

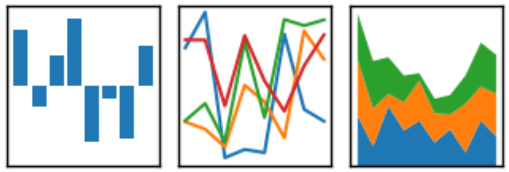


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:

Design Question: Find first alphabetically

pandas

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

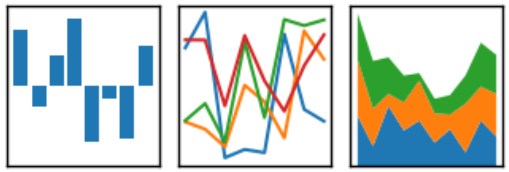


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.

Design Question: Find first alphabetically

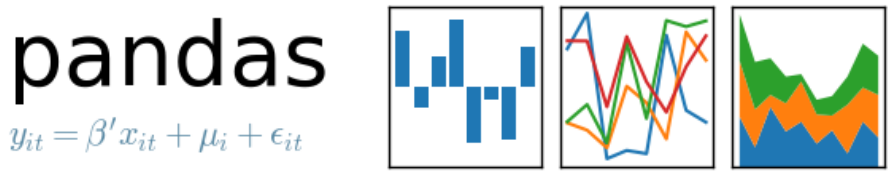
pandas

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



- 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.

Today's Topics

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

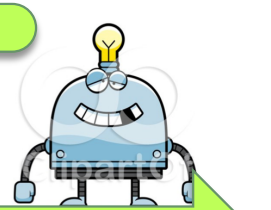
CS Survey Talk

Prof. Susan Epstein
(Machine Learning)

Computational agents

- Computational system implements decisions and actions on a physical device
- A computational agent executes a perpetual sense-decide-act loop

```
Do forever
    Sense the world
    Select an action
    Execute that action
```

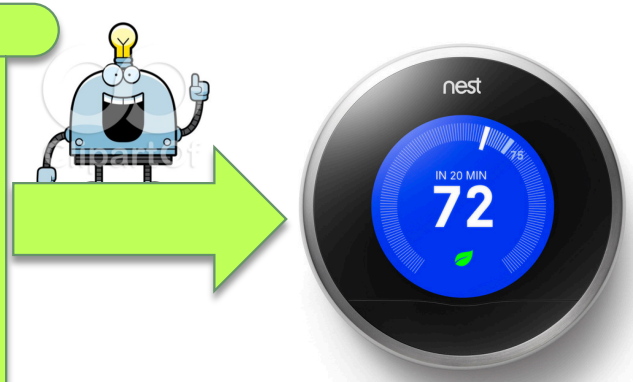


- How to sense the world: infrared sonar radar Kinect microphone camera
- Given a set of possible actions, an agent decides by selecting one

Artificial intelligence (AI)

- An AI agent doesn't have to be a robot (embodied in the world)
- An AI agent doesn't have to be autonomous (make decisions entirely on its own)
- But it does have to be smart…
- That means it has to make smart decisions
- Artificial intelligence = simulation of intelligent (human) behavior by a computational agent

```
Do forever
    Sense the world
    Select an action
    Execute that action
```



Nest

- Controlled by Wi-Fi from a smartphone
- Reprograms itself based on human behavior

What AI does

- Tackles hard, interesting problems
  - Does this image show cancer?
  - Should I move this car through the intersection?
  - How do I get to that concert?
- Builds models of perception, thinking, and action
- Uses these models to build smarter programs

Apollo and ROSie

Our autonomous robot navigators

- Despite uncertainty, noise, and constant changes in the world
- Learn models of their environment
- Make smart decisions with those models

How our robots navigate

- We built SemaFORR, a robot controller that makes decisions autonomously
- First the robots learn to travel by building a model of the world we put them in
- Then they prove they can find both hard and easy targets there

- Apollo has already done this on a small part of the 10th floor here
- And in simulation ROSie has traveled
  - Through much of Hunter, The Graduate Center, and MOMA
  - Through moving crowds of people
  - Without collision and without coming too close to people
  - And explained her behavior in natural language

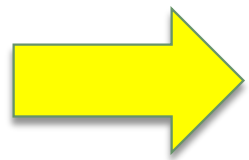
How to build an intelligent agent

- Find good problems
- Start simple
- Run lots of experiments
- Analyze the results carefully
- …and repeat

Fun problems

Good reasons

Learning algorithms



Want to know more?

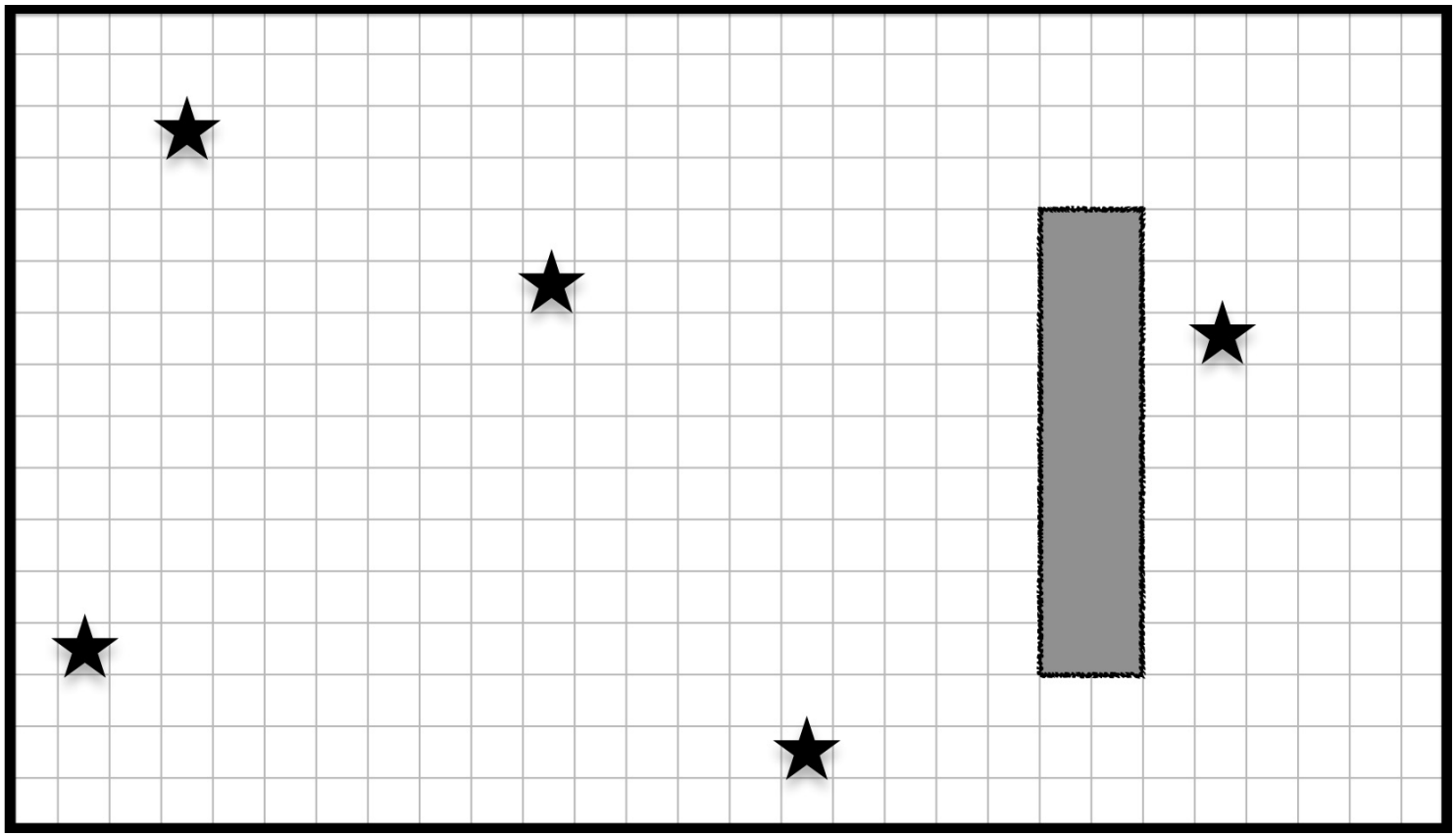
- Fall 2019: SCI 111 Brains, Minds, and Machines = cognitive neuroscience + cognitive psychology + AI
- Fall 2019: CSCI 353 Machine Learning
- 2021: CSCI 350 Artificial Intelligence

- …and then there's my lab, where workstations run 24/7, learning to be intelligent agents

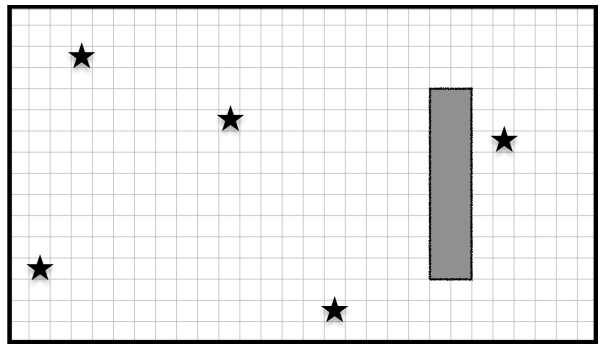
Susan Epstein, Professor of Computer Science
1090C Hunter North
susan.epstein@hunter.cuny.edu
http://www.cs.hunter.cuny.edu/~epstein/

Design Challenge

Collect all five stars (locations randomly generated):

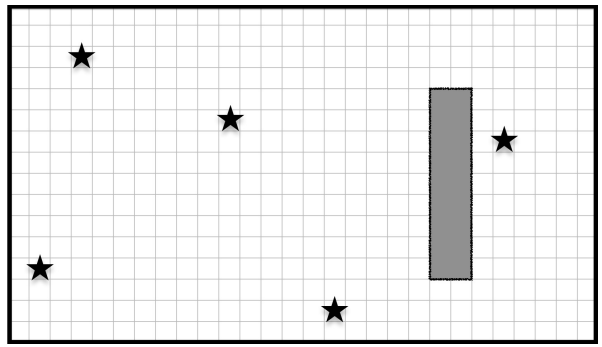


Design Challenge



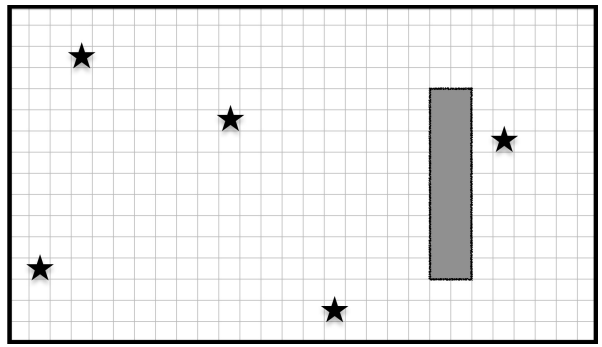
- Possible approaches:

Design Challenge



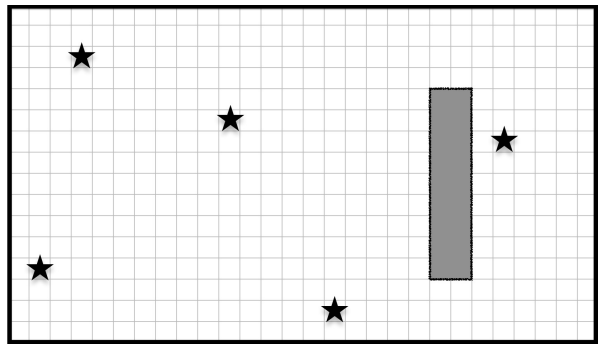
- Possible approaches:
  - Randomly wander until all 5 collected, or

Design Challenge



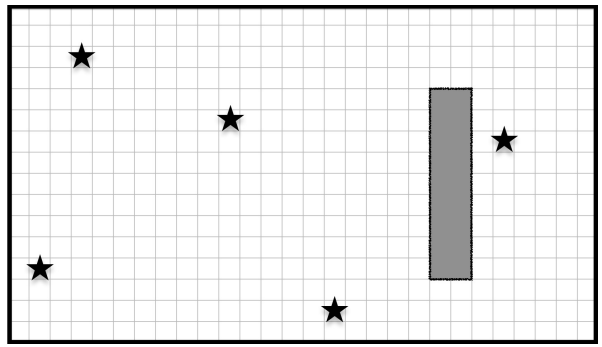
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.

Design Challenge



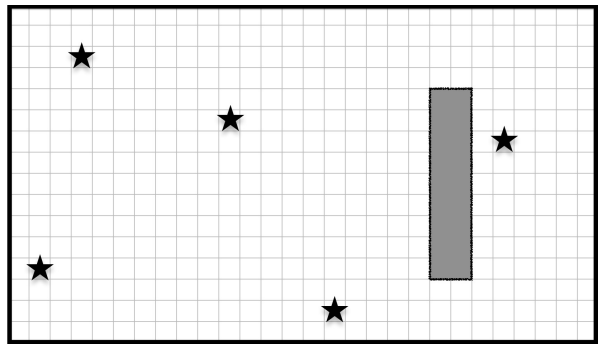
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'

Design Challenge



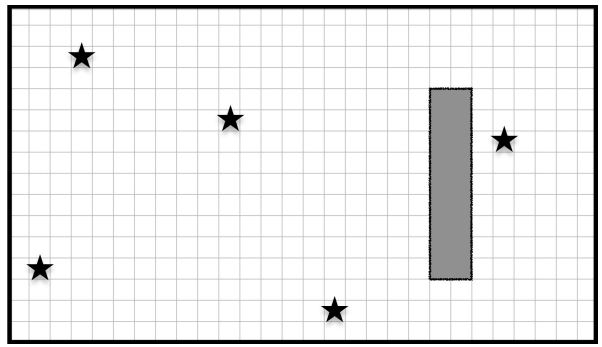
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.

Design Challenge



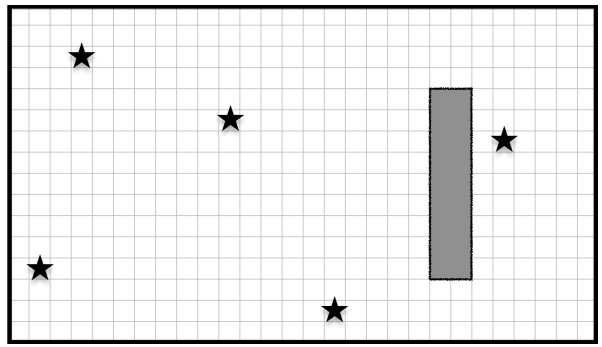
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.

Design Challenge



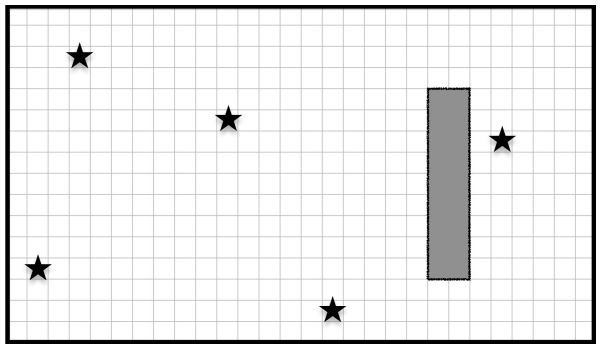
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

Design Challenge



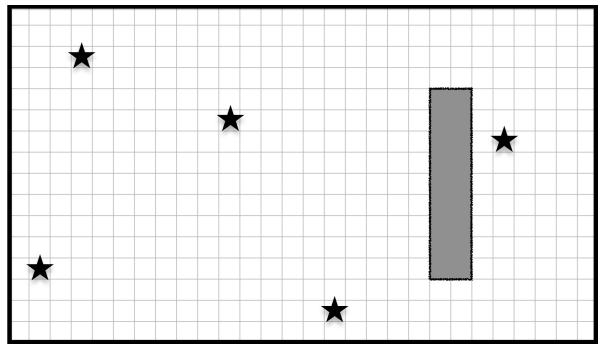
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

Design Challenge



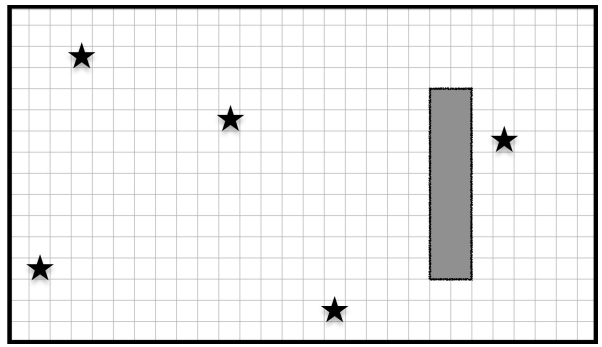
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

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.

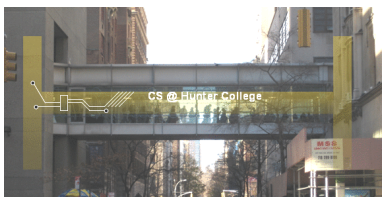
Design Challenge



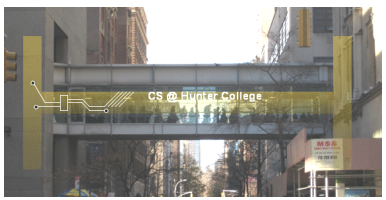
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

Recap

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

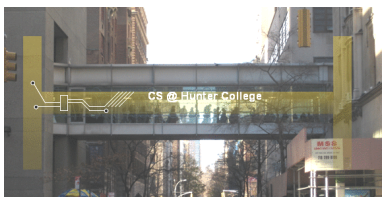


Recap



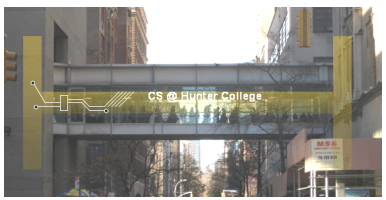
- On lecture slip, write down a topic you wish we had spent more time (and why).
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.

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.

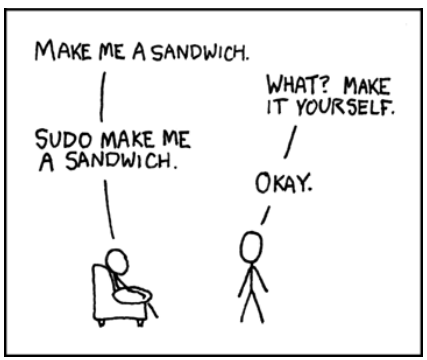
Recap



- 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.
- Pass your lecture slips to the aisles for the UTAs to collect.

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

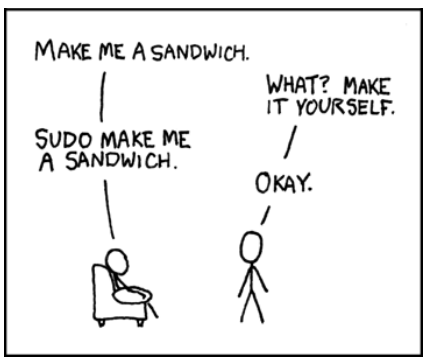


xkcd 149

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`

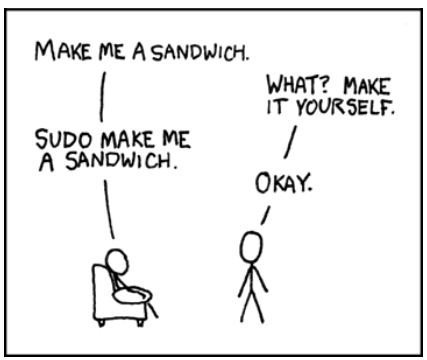


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`

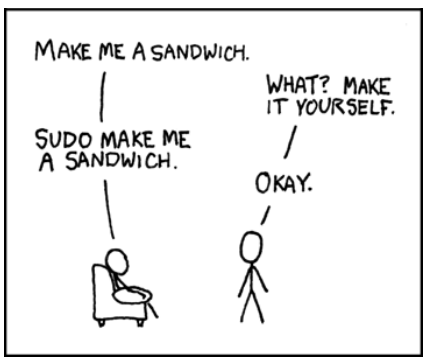


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)

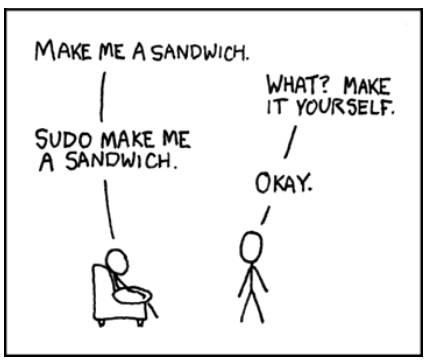


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`

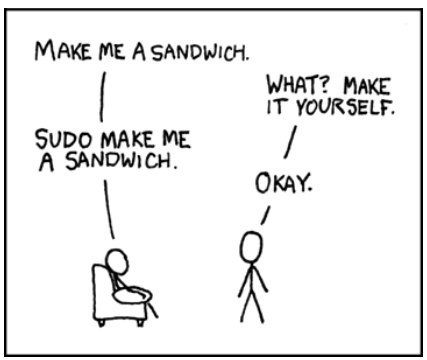


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`

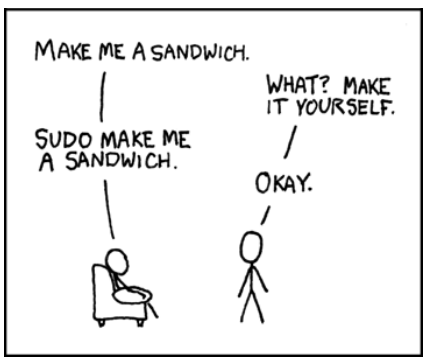


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line

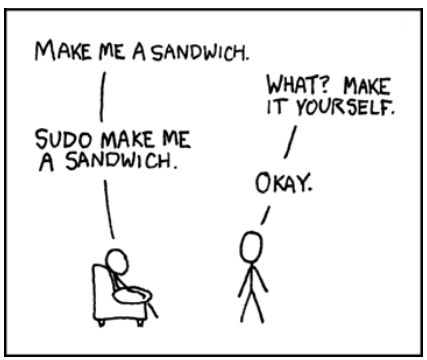


xkcd 149

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line
- *Lab 7:* git from the command line

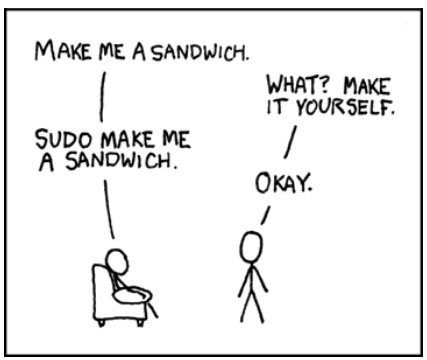


xkcd 149

Final Exam Prep: UNIX

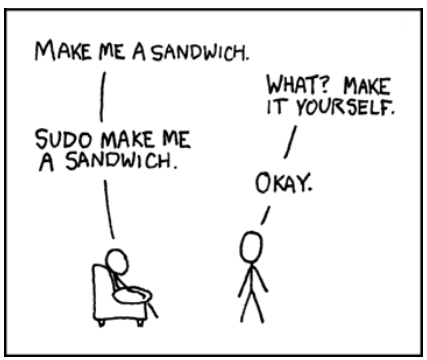
Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line
- *Lab 7:* git from the command line
- *Lab 8:* ls *.py (wildcards)



xkcd 149

Final Exam Prep: UNIX

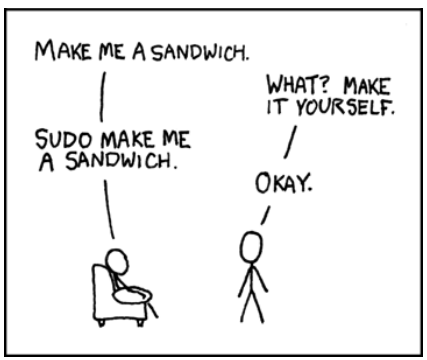


xkcd 149

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line
- *Lab 7:* git from the command line
- *Lab 8:* ls *.py (wildcards)
- *Lab 9:* More on scripts, `vim`

Final Exam Prep: UNIX

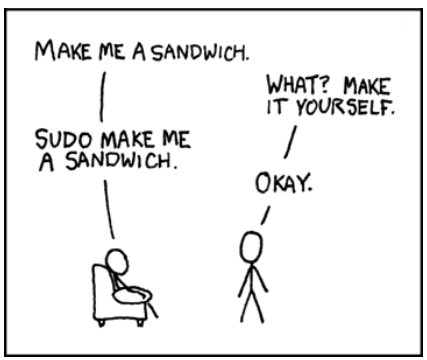


xkcd 149

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line
- *Lab 7:* git from the command line
- *Lab 8:* ls *.py (wildcards)
- *Lab 9:* More on scripts, `vim`
- *Lab 10:* `ls | wc -c` (pipes), `grep`, `wc`

Final Exam Prep: UNIX

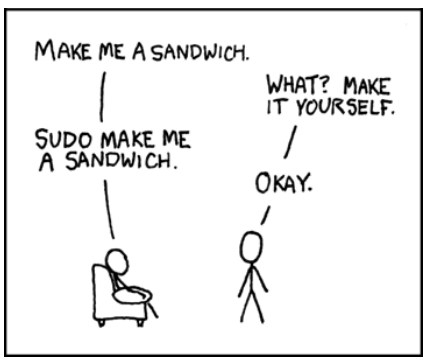


xkcd 149

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line
- *Lab 7:* git from the command line
- *Lab 8:* ls *.py (wildcards)
- *Lab 9:* More on scripts, `vim`
- *Lab 10:* `ls | wc -c` (pipes), `grep`, `wc`
- *Lab 11:* `file`, `which`

Final Exam Prep: UNIX



xkcd 149

Unix commands in the weekly on-line labs:

- *Lab 1:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 2:* `ls -l`, `cp`, `mv`
- *Lab 3:* `cd ../` (relative paths)
- *Lab 4:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 5:* Scripts, `chmod`
- *Lab 6:* Running Python from the command line
- *Lab 7:* git from the command line
- *Lab 8:* ls *.py (wildcards)
- *Lab 9:* More on scripts, `vim`
- *Lab 10:* `ls | wc -c` (pipes), `grep`, `wc`
- *Lab 11:* `file`, `which`
- *Lab 12:* `man`, `more`, `w`