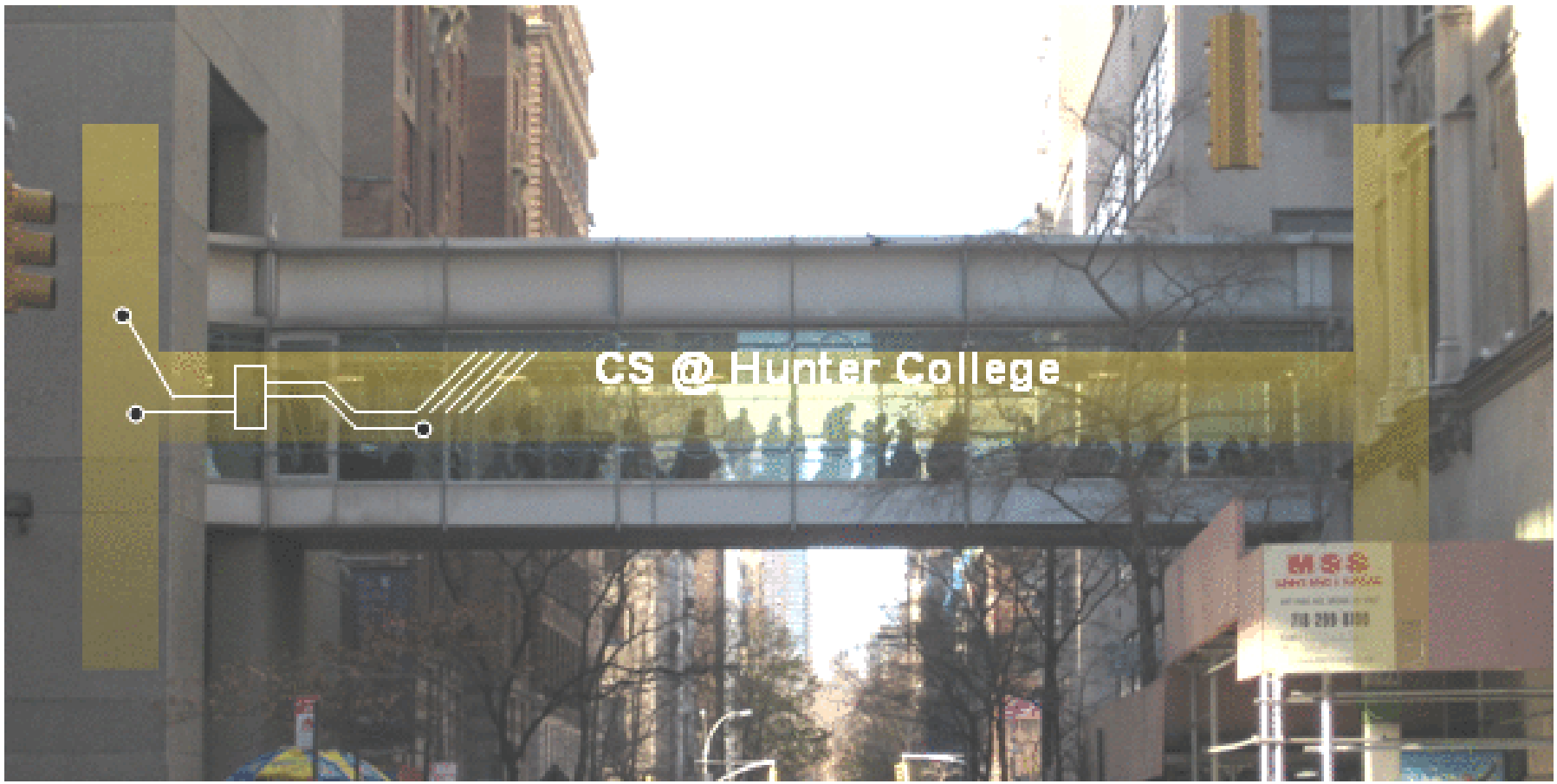
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

- **When is the final?**

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.
- **I have another final then. What do I do?**

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.
- **I have another final then. What do I do?**
  We are arranging an alternative date: Wednesday, May 17. Time and location are TBD.

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.
- **I have another final then. What do I do?**
  We are arranging an alternative date: Wednesday, May 17. Time and location are TBD.
- **Do I have to take the final?**

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.
- **I have another final then. What do I do?**
  We are arranging an alternative date: Wednesday, May 17. Time and location are TBD.
- **Do I have to take the final?**
  Yes, you must pass the final (60 out of 100 points) to pass the class.

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.
- **I have another final then. What do I do?**
  We are arranging an alternative date: Wednesday, May 17. Time and location are TBD.
- **Do I have to take the final?**
  Yes, you must pass the final (60 out of 100 points) to pass the class.
- **I'd like to take more courses in computer science. What's next?**

Frequently Asked Questions

- **When is the final?**
  Monday, May 22 from 9 am to 11 am in Assembly Hall (118 Hunter North)
- **What is the format?**
  Content and format will be similar to past paper exams.
- **I have another final then. What do I do?**
  We are arranging an alternative date: Wednesday, May 17. Time and location are TBD.
- **Do I have to take the final?**
  Yes, you must pass the final (60 out of 100 points) to pass the class.
- **I'd like to take more courses in computer science. What's next?**
  - CSCI 135: Software Analysis and Design I
  - CSCI 150: Discrete Structures

Today's Topics

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

Today's Topics

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

Folium

Contents of `cunyLocations.csv`.

```
College or Institution Type,Campus,...,Latitude,Longitude,...
Senior Colleges,Baruch College,...,40.740977,-73.984252,...
Senior Colleges,Brooklyn College,...,40.630276,-73.955545,...
Community Colleges,Borough of Manhattan Community College,...,40.717367,-74.012178,
...
```

Folium

What does this code do? (1/3)

```
import folium
import pandas as pd

#Use pandas (alias pd) to read a csv file,
#save the returned dataframe object in variable cuny.
cuny = pd.read_csv("cunyLocations.csv")

#Create a map object centered at 40.75, -74.125,
#save in variable mapCUNY.
mapCUNY = folium.Map(location=[40.75, -74.125])
```

Folium

What does this code do? (2/3)

```
#Go through each row in the dataframe
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")

    #Create a marker, specify its latitude, longitude,
    #pop up name, and icon, save in variable newMarker.
    newMarker = folium.Marker([lat, lon], popup=name,
        icon=collegeIcon)
    newMarker.add_to(mapCUNY)
```

Folium

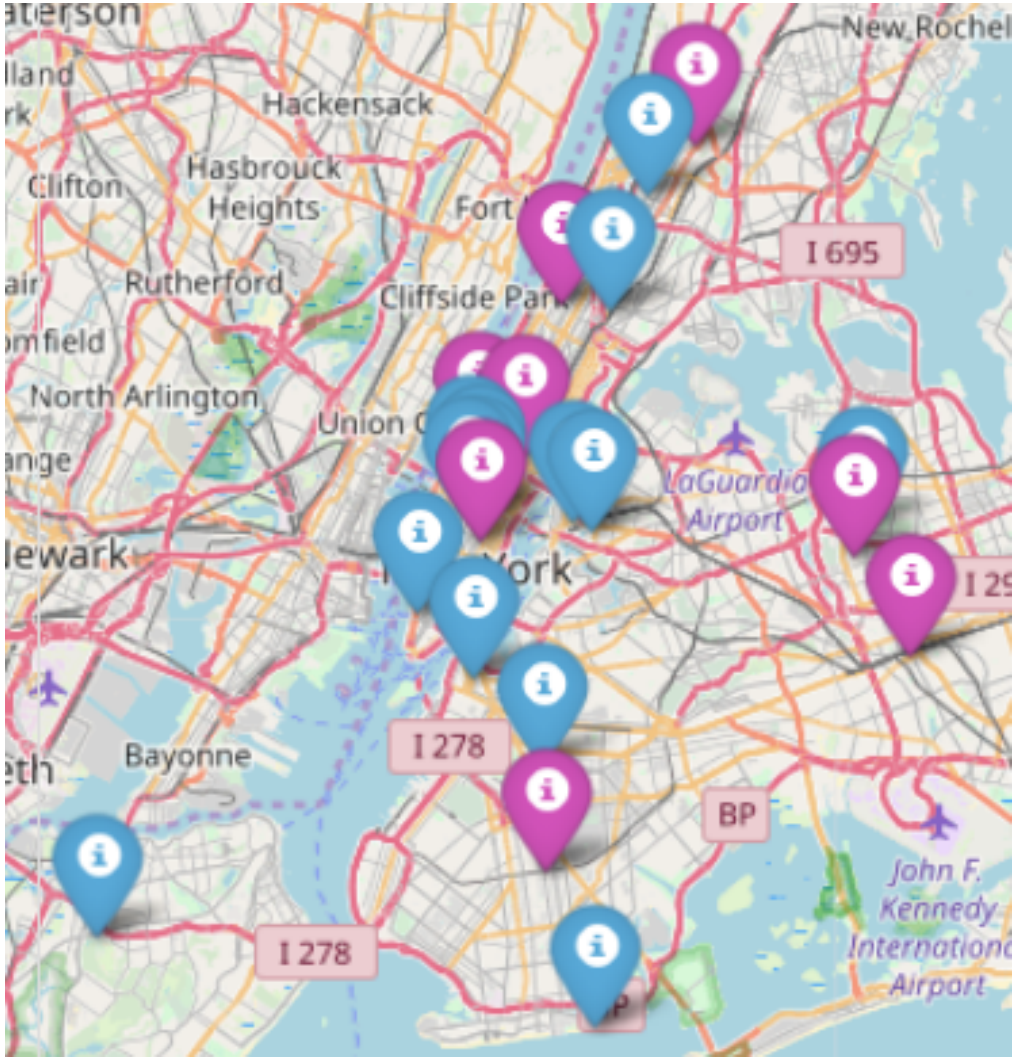
What does this code do? (3/3)

```
filename = "cunyLocationsSenior.html"

#save mapCUNY to filename
mapCUNY.save(outfile=filename)
```

Recap: Folium

What does this code do?



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.

Today's Topics

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

Challenge:

- Write a function that asks a user for number after 2000 but before 2021. 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 2021. The function should repeatedly ask the user for a number until they enter one within the range and return the number.

Questions:

- Is 2000 a valid input?
- Is 2021 a valid input?
- Is 2001 a valid input?

Define function header.

```
def getYear():
```

Coding

- Write a function that asks a user for number after 2000 but before 2021. The function should repeatedly ask the user for a number until they enter one within the range and **return** the number.

```
def getYear():
    #TODO: initialize num

    return num
```

Coding

- Write a function that asks a user for a number after 2000 but before 2021. The function should continue to prompt the user for a number if the one entered is not within the range. Once a valid number is entered then that number should be returned.

```
def getYear():
    num = 0 #initialize num


    return num
```

Coding

- Write a function that asks a user for a number after 2000 but before 2021. The function should continue to prompt the user for a number if the one entered is not within the range. Once a valid number is entered then that number should be returned.

```
def getYear():
    num = 0 #initialize num
    while num <= 2000 or num >= 2021:
        num = int(input("Enter a number after 2000 and before
            2021: "))
    return num #outside loop
```

Define and Call function getYear

```
def getYear():
    #set an initially invalid value for num so that
    #the while loop executes at least once
    num = 0
    while num <= 2000 or num >= 2021:
        num = int(input("Enter a number after 2000 and before
            2021: "))
    return num

def main():
    #num in main is independent of num in getYear
    num = getYear()
    print("The year is", num)

if __name__ == "__main__":
   main()
```

Can you spot an error?

```
num = 0
def getYear():
    while num <= 2000 or num >= 2021:
        num = int(input("Enter a number after 2000 and before
            2021: "))
    return num
```

Indefinite Loops

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

```
#Spring 2012 Final Exam, #8

nums = [1,4,0,6,5,2,9,8,12]
print(nums)
i=0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i=i+1

print(nums)
```

Indefinite Loops

```
#Spring 2012 Final Exam, #8

nums = [1,4,0,6,5,2,9,8,12]
print(nums)
i=0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i=i+1

print(nums)
```

- 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

```
#Spring 2012 Final Exam, #8

nums = [1,4,0,6,5,2,9,8,12]
print(nums)
i=0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i=i+1

print(nums)
```

- 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

```
#Spring 2012 Final Exam, #8

nums = [1,4,0,6,5,2,9,8,12]
print(nums)
i=0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i=i+1

print(nums)
```

- 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.

Switch adjacent elements if the left element is smaller

```
nums = [1, 4, 0, 6, 5, 2]
print(nums)
i = 0
while i < len(nums)-1:
    if nums[i] < nums[i+1]:
        nums[i], nums[i+1] = nums[i+1], nums[i]
    i = i+1
print(nums)
```

Challenge

Predict what the code does: (1/2)

```
def move():
    tess = turtle.Turtle()
    tess.color("steelBlue")
    tess.shape("turtle")
    tess.penup()
    #Start off-screen:
    tess.goto(-250,-250)
```

Challenge

Predict what the code does: (2/2)

```
#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!")

turtle.done()
```

Trinket Demo

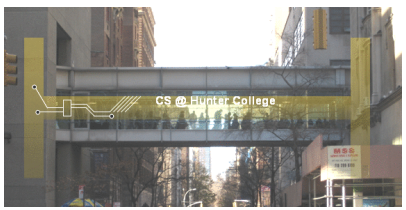
```
#Random search
import turtle
import random
tess = turtle.Turtle()
tess.color('steelBlue')
tess.shape('turtle')
tess.penup()
#Start off screen:
tess.goto(-250,-250)
#Remember:  abs(x) < 25 means absolute value: -25 < x < 25
while abs(tess.xcor()) > 25 or abs(tess.ycor()) > 25:
  x = random.randrange(-200,200)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

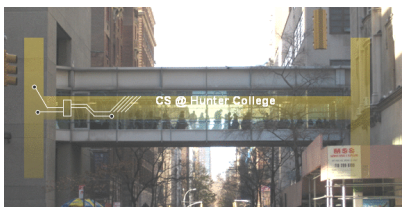
- Recap: Folium
- Indefinite loops
- **Design Patterns: Max (Min)**

Design Patterns



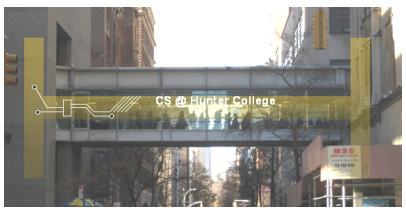
- A **design pattern** is a standard algorithm or approach for solving a common problem.

Design Patterns



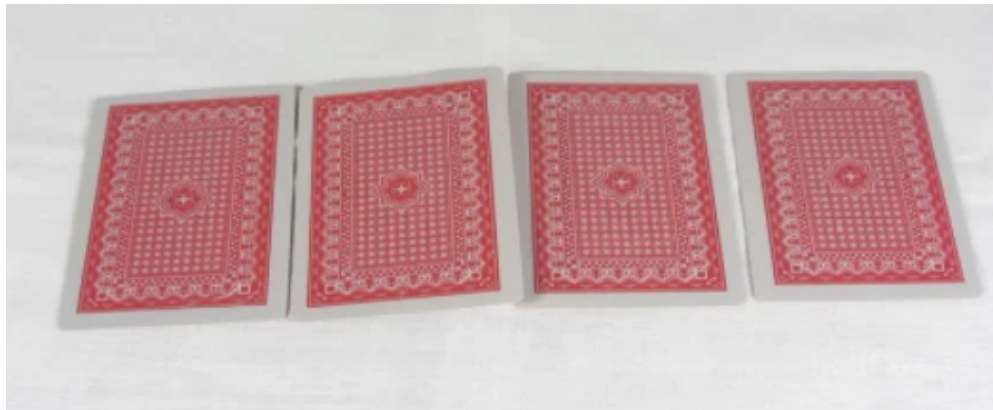
- A **design pattern** is a standard algorithm or approach for solving a common problem.
- The pattern is independent of the programming language.

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.

Design Question:



You can uncover one card at a time.
How would you go about finding the highest card?

Challenge:

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:
        #TODO: update maxNum to be n

print(maxNum)
```

Challenge:

Getting the largest value from a list:

```
nums = [1, 4, 10, 6, 5, 42, 9, 8, 12]
maxNum = 0
for n in nums:
    if n > maxNum:
        maxNum = n
print(maxNum)
```

Improvement

Initialize maxNum to be the smallest number in system, so that any actual number is no smaller than it (guaranteed to be larger or equal to it).

```
nums = [-1, -5, -4]

maxNum = float("-inf")
for n in nums:
    if n > maxNum:
        maxNum = n

print(maxNum)
```

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

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

```
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,
- 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.
- Must look at entire list to determine max is found

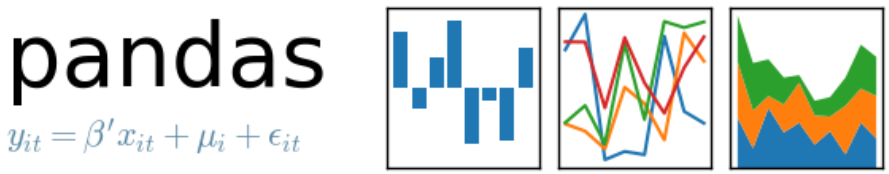
Max Design Pattern

```
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.
- Must look at entire list to determine max is found
- Similar idea works for finding the minimum value.

Pandas: Minimum Values

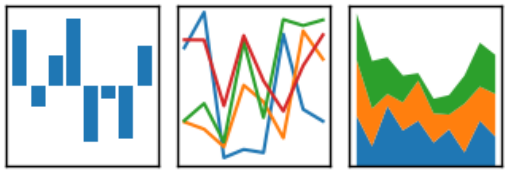


- In Pandas, lovely built-in functions:

Pandas: Minimum Values

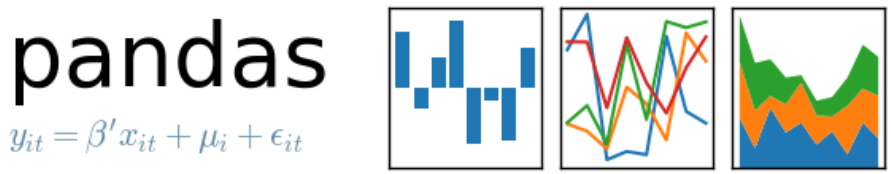
pandas

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



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

Pandas: Minimum Values

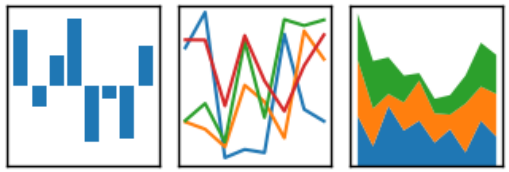


- In Pandas, lovely built-in functions:
  - `df.sort_values("First Name")` and
  - `df["First Name"].min()`
- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

pandas

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

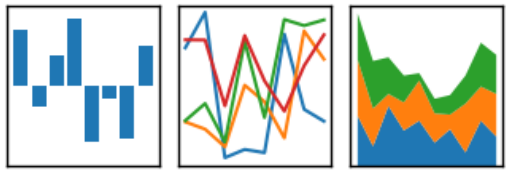


- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

pandas

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

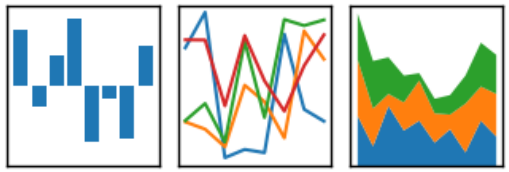


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max

Design Question: Find first alphabetically

pandas

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

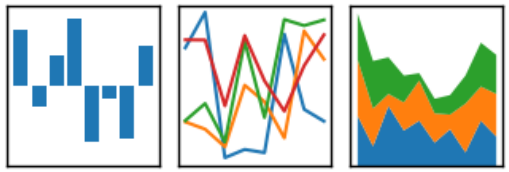


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (e.g. `maxNum = 0` or `first = "ZZ"`).

Design Question: Find first alphabetically

pandas

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

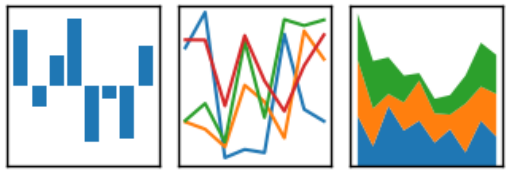


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (e.g. `maxNum = 0` or `first = "ZZ"`).
  - For each item, X, in the list:

Design Question: Find first alphabetically

pandas

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

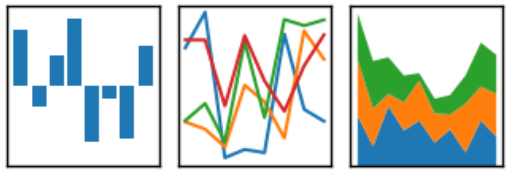


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (e.g. `maxNum = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.

Design Question: Find first alphabetically

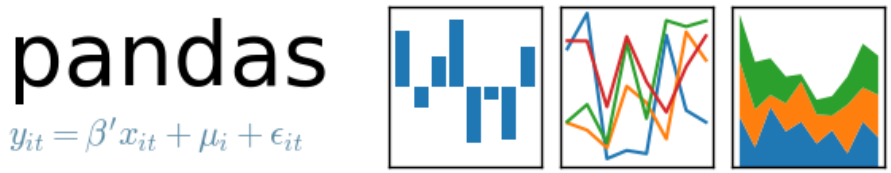
pandas

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



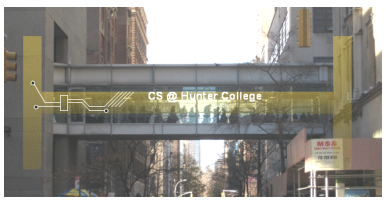
- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (e.g. `maxNum = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.

Design Question: Find first alphabetically



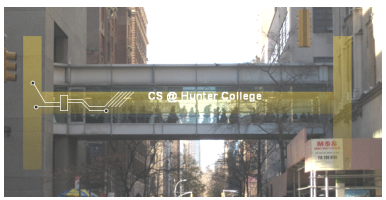
- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (e.g. `maxNum = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.
  - Print/return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

Recap



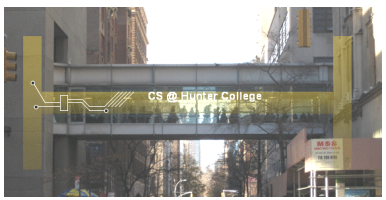
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.

Recap



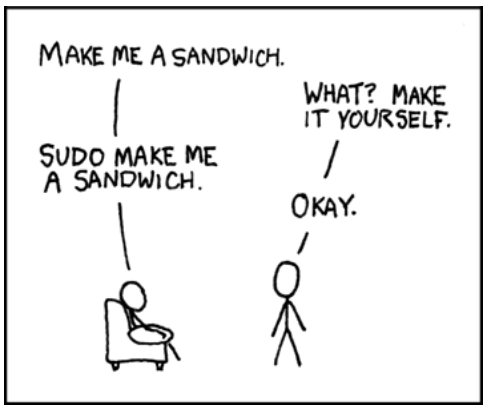
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.

Recap



- 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/min design pattern.

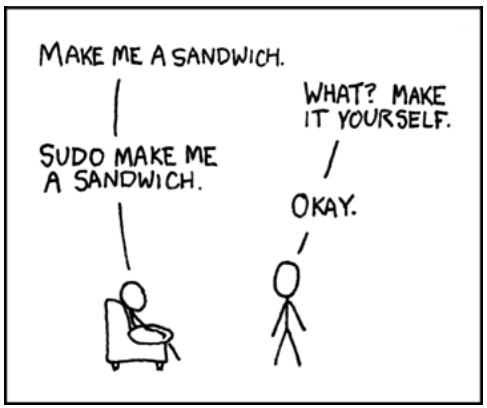
Final Exam Prep: UNIX



xkcd 149

- This course has three main themes:
  - Programming & Problem Solving

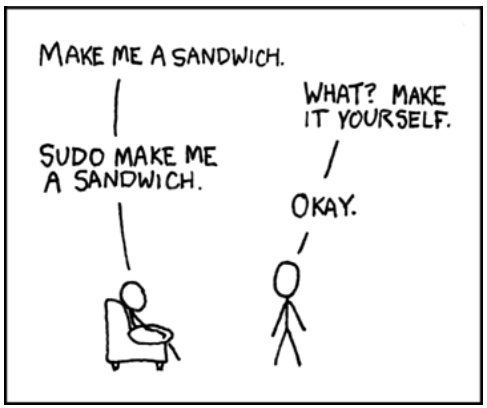
Final Exam Prep: UNIX



xkcd 149

- This course has three main themes:
  - Programming & Problem Solving
  - Organization of Hardware & Data

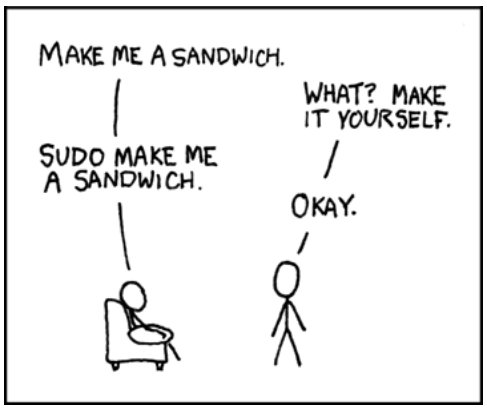
Final Exam Prep: UNIX



xkcd 149

- This course has three main themes:
  - Programming & Problem Solving
  - Organization of Hardware & Data
  - Design

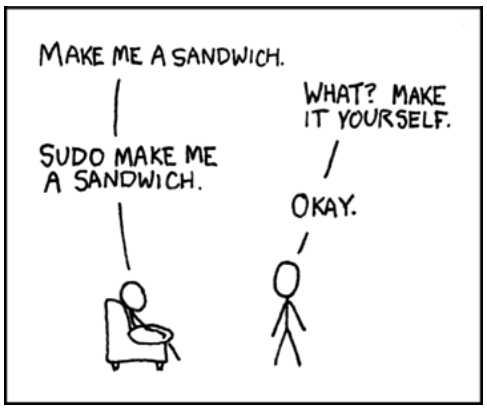
Final Exam Prep: UNIX



xkcd 149

- This course has three main themes:
  - Programming & Problem Solving
  - Organization of Hardware & Data
  - Design
- The operating system, Unix, is part of the second theme.

Final Exam Prep: UNIX

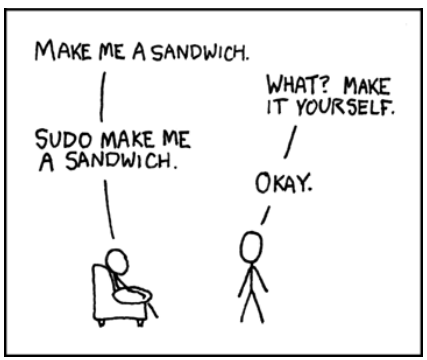


xkcd 149

- This course has three main themes:
  - Programming & Problem Solving
  - Organization of Hardware & Data
  - Design
- The operating system, Unix, is part of the second theme.
- Unix commands in the weekly online labs

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

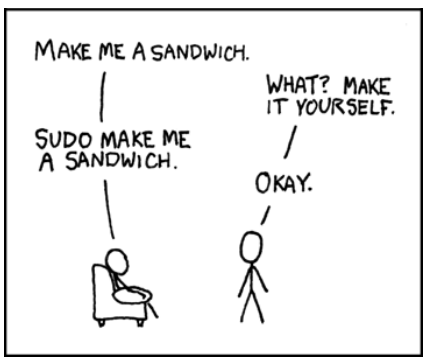


xkcd 149

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

- *Lab 2:* `pwd`, `ls`, `mkdir`, `cd`

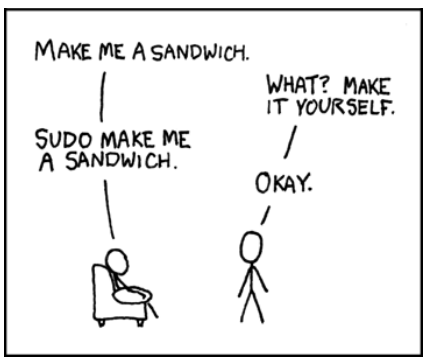


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

- *Lab 2:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 3:* `ls -l`, `cp`, `mv`

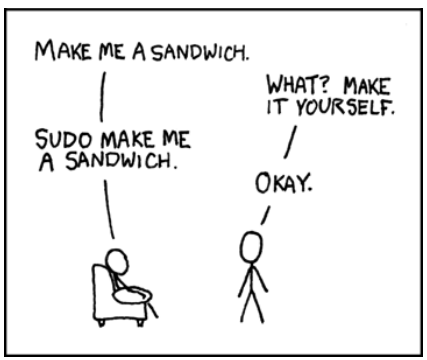


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

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

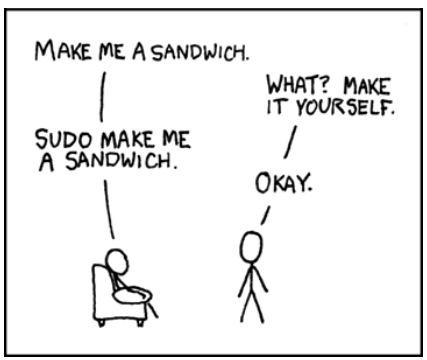


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

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

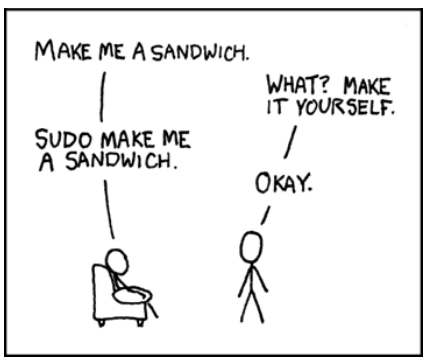


xkcd 149

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

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

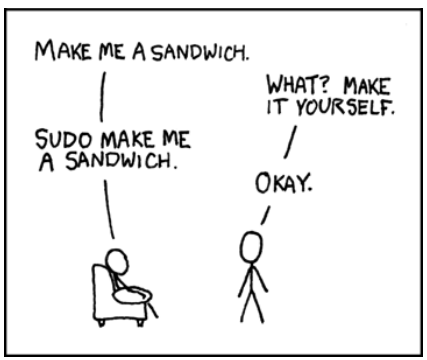


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

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

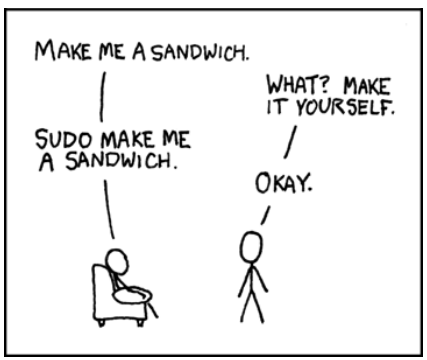


xkcd 149

Final Exam Prep: UNIX

Unix commands in the weekly online labs:

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

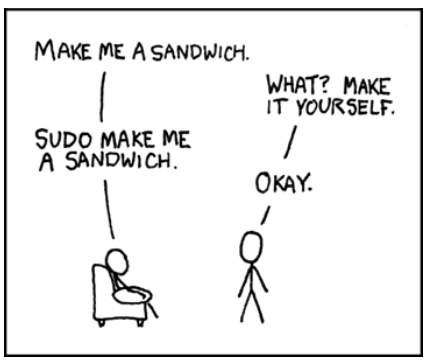


xkcd 149

Final Exam Prep: UNIX

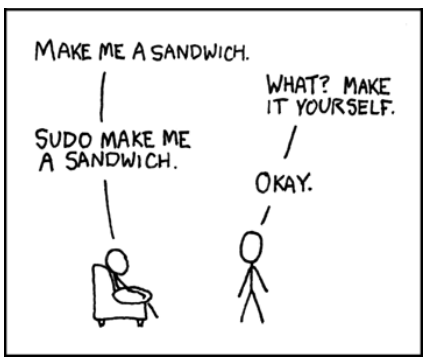
Unix commands in the weekly online labs:

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



xkcd 149

Final Exam Prep: UNIX

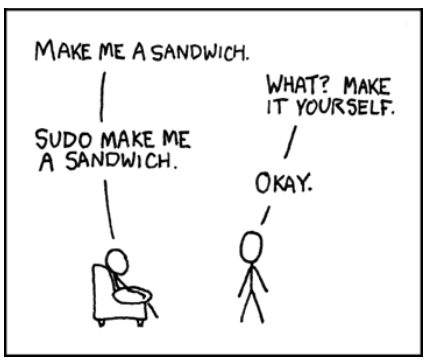


xkcd 149

Unix commands in the weekly online labs:

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

Final Exam Prep: UNIX

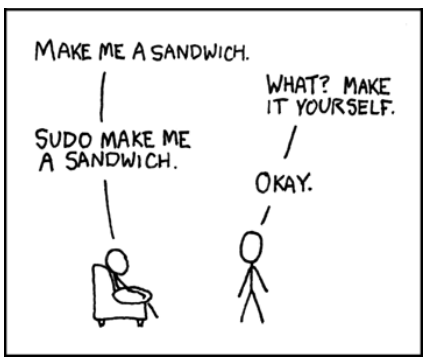


xkcd 149

Unix commands in the weekly online labs:

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

Final Exam Prep: UNIX

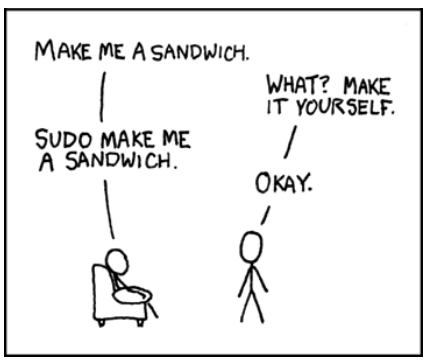


xkcd 149

Unix commands in the weekly online labs:

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

Final Exam Prep: UNIX

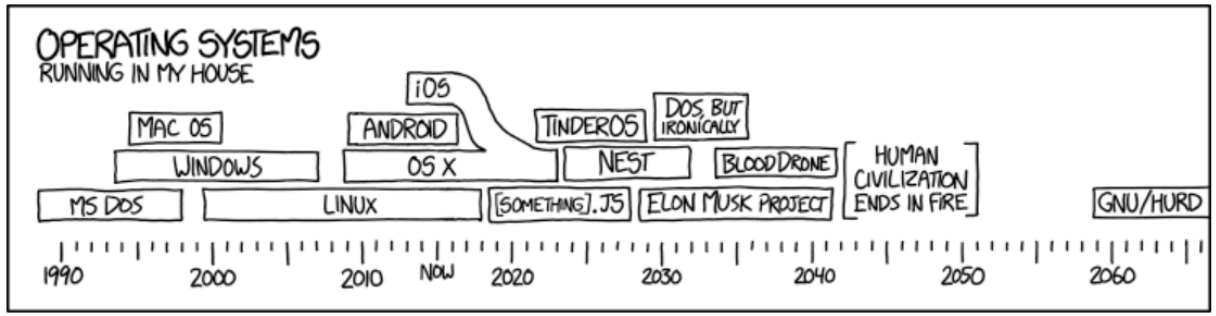


xkcd 149

Unix commands in the weekly online labs:

- *Lab 2:* `pwd`, `ls`, `mkdir`, `cd`
- *Lab 3:* `ls -l`, `cp`, `mv`
- *Lab 4:* `cd ../` (relative paths)
- *Lab 5:* `cd /usr/bin` (absolute paths), `cd ~`
- *Lab 6:* Scripts, `chmod`
- *Lab 7:* Running Python from the command line
- *Lab 8:* git from the command line
- *Lab 9:* ls *.py (wildcards)
- *Lab 10:* More on scripts, `vim`
- *Lab 11:* `ls | wc -c` (pipes), `grep`, `wc`
- *Lab 12:* `file`, `which`
- *Lab 13:* `man`, `more`, `w`

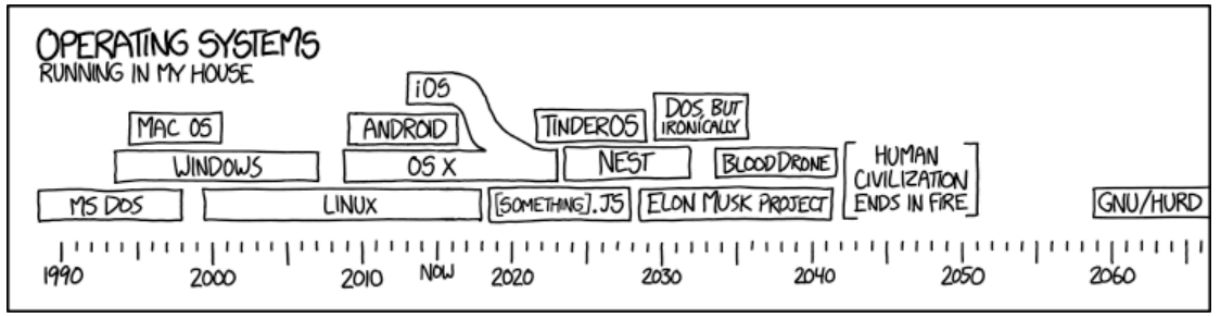
Practice Quiz & Final Questions



xkcd #1508

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).

Practice Quiz & Final Questions



xkcd #1508

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).
- Theme: Unix commands! (Spring 19 Version 3, #1.b)

Weekly Reminders!

Before the next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!

Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take this week's Quiz

Weekly Reminders!

Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take this week's Quiz
- Schedule an appointment to take this week's Code Review

Weekly Reminders!

Before the next lecture, don't forget to:

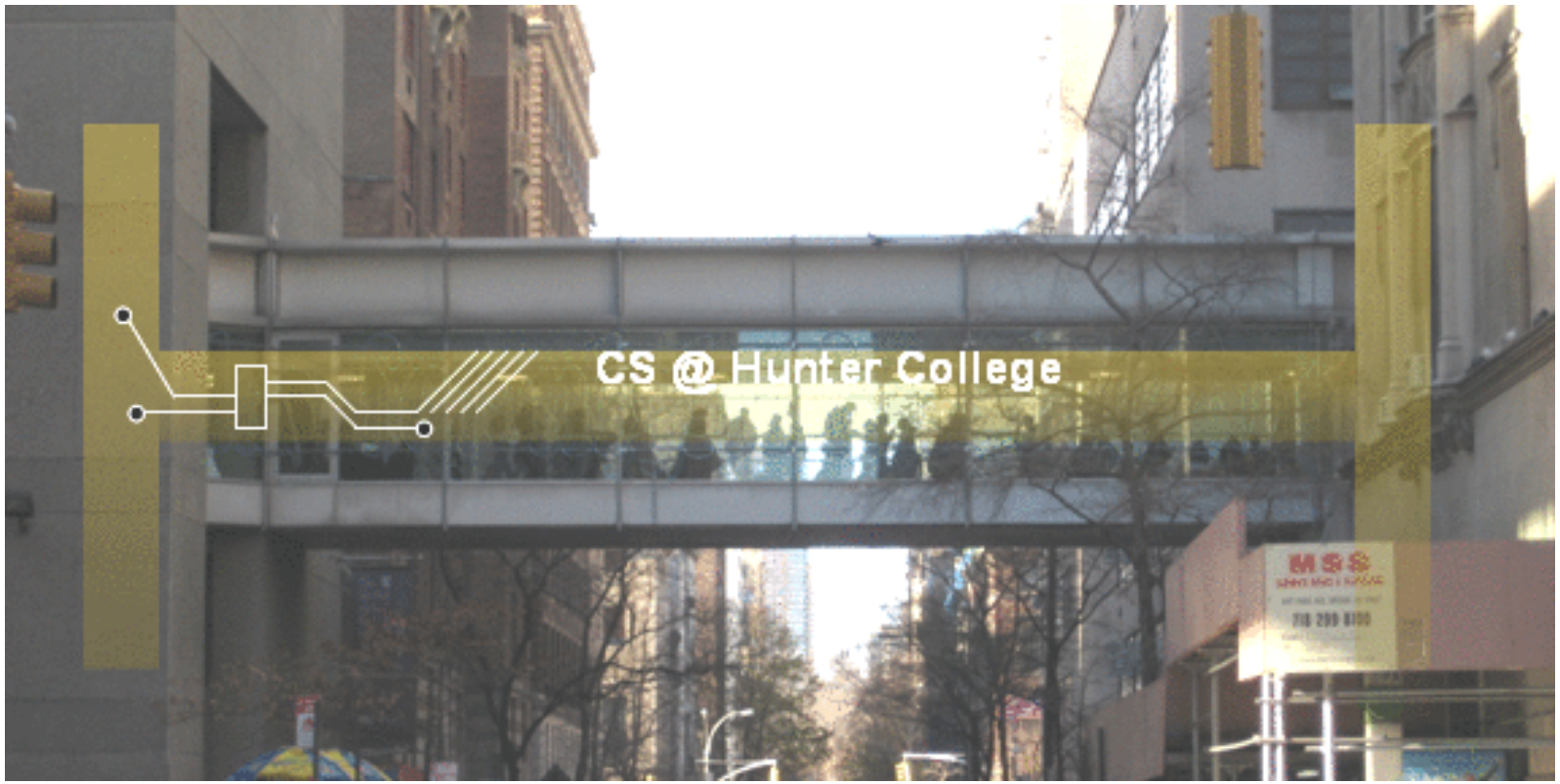
- Work on this week's Online Lab
- Schedule an appointment to take this week's Quiz
- Schedule an appointment to take this week's Code Review
- Take the Lecture Preview on Blackboard

Weekly Reminders!

Before the next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take this week's Quiz
- Schedule an appointment to take this week's Code Review
- Take the Lecture Preview on Blackboard
- If you need help, schedule an appointment for tutoring

Lecture Slips & Writing Boards



- Hand your lecture slip to a UTA.
- Return writing boards as you leave.