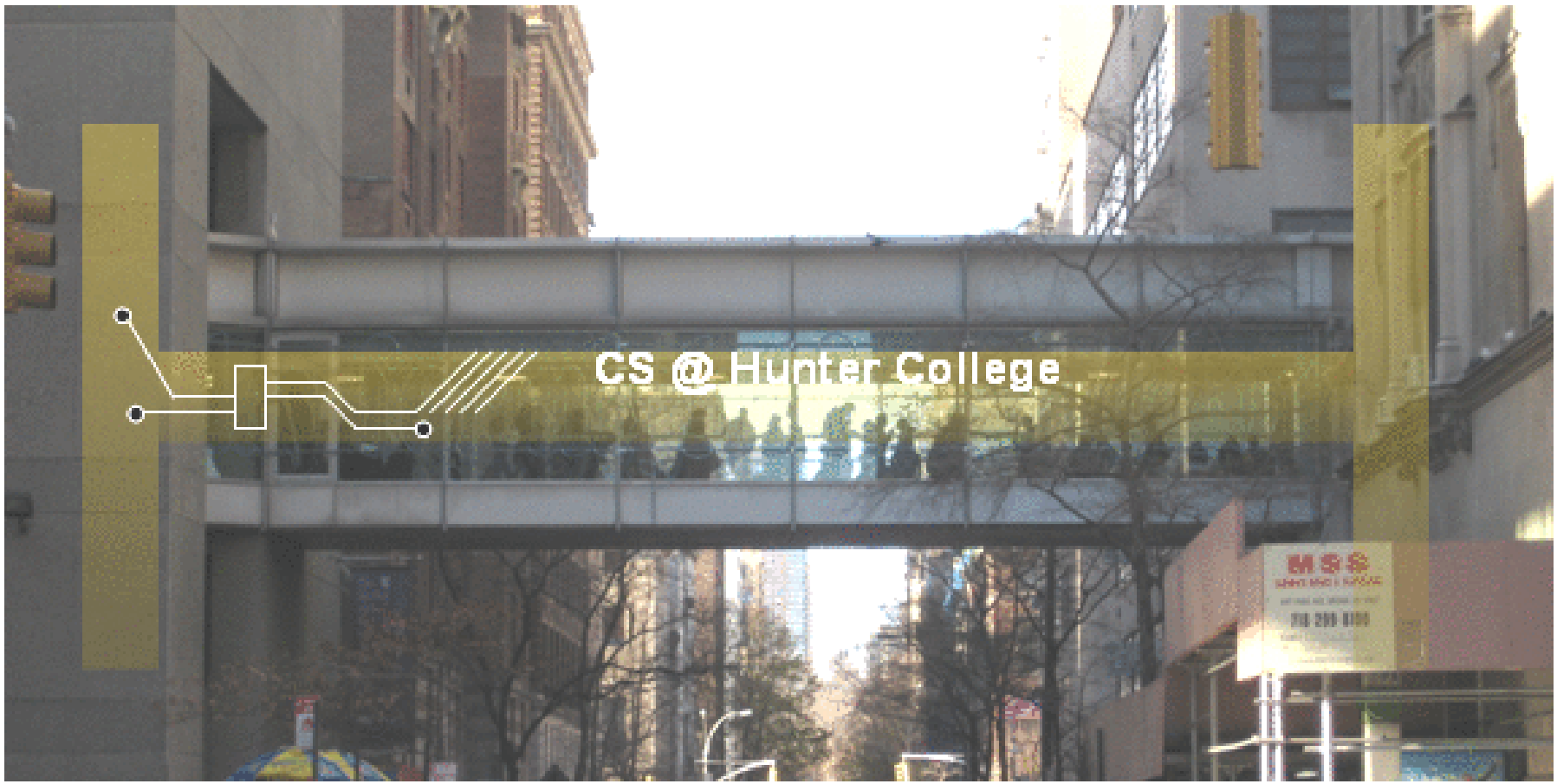
# CSci 127: Introduction to Computer Science




hunter.cuny.edu/csci

# Announcements

- Please always read all Blackboard announcements

# Announcements

- Please always read all Blackboard announcements
- Online help is available in multiple forms:

# Announcements

- Please always read all Blackboard announcements
- Online help is available in multiple forms:
  - **Email**

# Announcements

- Please always read all Blackboard announcements
- Online help is available in multiple forms:
  - **Email**
  - **Discussion Board**: on Blackboard, link on purple menu bar

# Announcements

- Please always read all Blackboard announcements
- Online help is available in multiple forms:
  - **Email**
  - **Discussion Board**: on Blackboard, link on purple menu bar
  - **Drop-in tutoring (Tu/W 1p - 2p)**

# Today's Topics

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

# Today's Topics

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

# Challenge:

*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

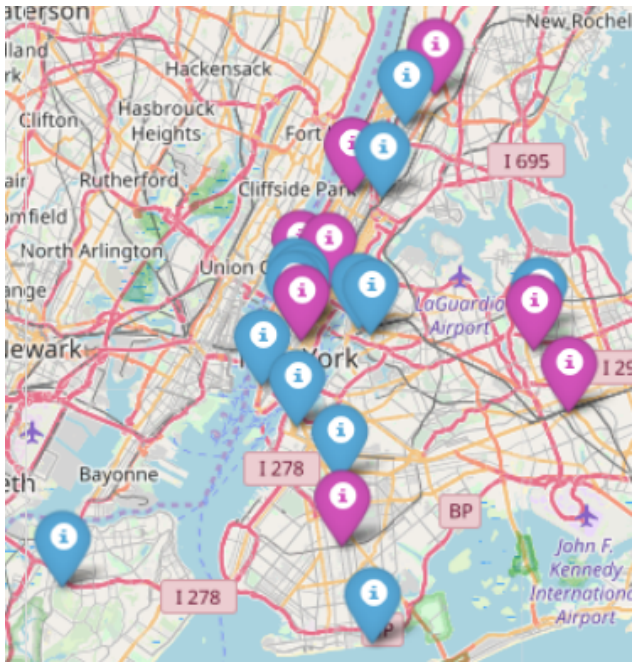
*What does this code do?*

```
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)
- Design Challenge

# Challenge:

- 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

```
#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

- 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)
```

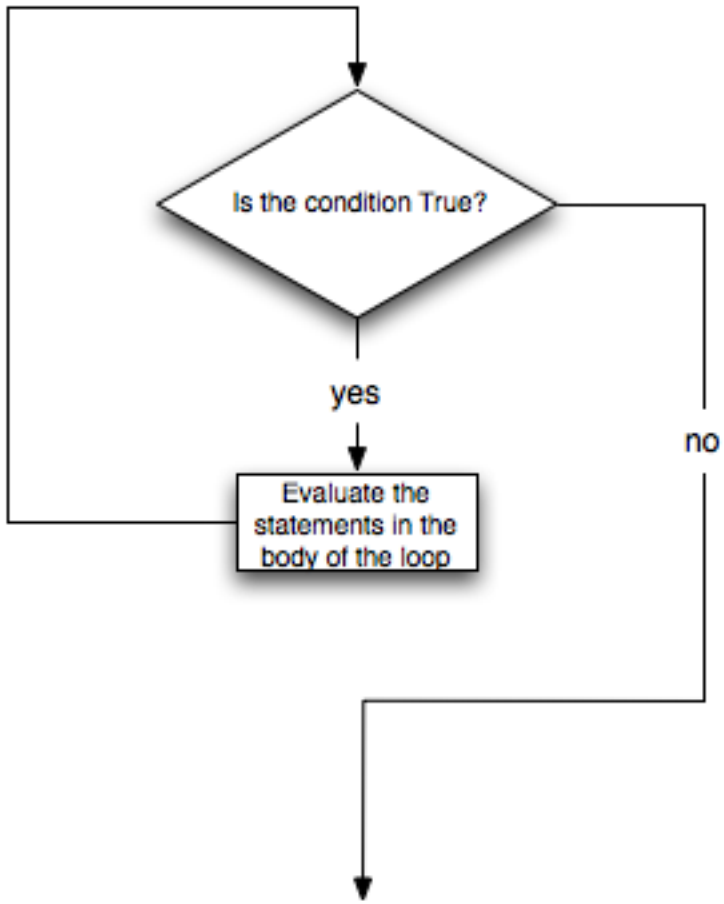
- Indefinite loops repeat as long as the condition is true.
- Could execute the body of the loop zero times, 10 times, infinite number of times.
- The condition determines how many times.
- Very useful for checking input, simulations, and games.

# 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

```
#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)
```



# Challenge

*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

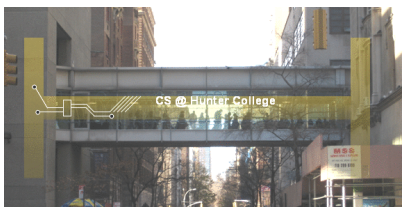
```
#Random search
import turtle
import random
tess = turtle.Turtle()
tess.color('steelBlue')
tess.shape('turtle')
tess.penup()
#Start off screen:
tess.goto(-250,-250)
#Remember:  abs(x) < 25 means absolute value: -25 < x < 25
while abs(tess.xcor()) > 25 or abs(tess.ycor()) > 25:
  x = random.randrange(-200,200)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

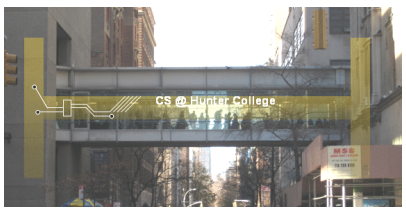
- Recap: Folium
- Indefinite loops
- **Design Patterns: Max (Min)**
- Design Challenge

# Design Patterns



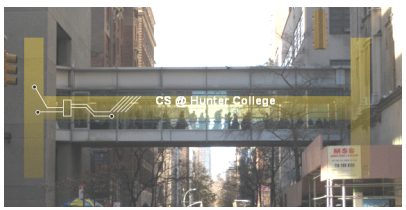

- A **design pattern** is a standard algorithm or approach for solving a common problem.

# 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.

# 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:
        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

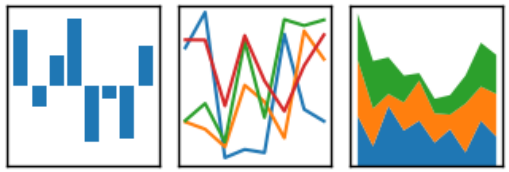
```
nums = [1,4,10,6,5,42,9,8,12]

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.

# Pandas: Minimum Values

pandas

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

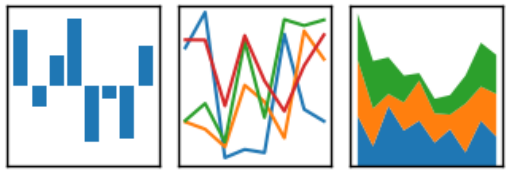


- In Pandas, lovely built-in functions:

# Pandas: Minimum Values

pandas

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

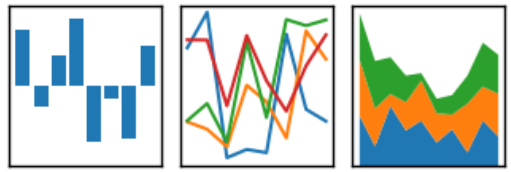


- In Pandas, lovely built-in functions:
  - `df.sort_values('First Name')` and
  - `df['First Name'].min()`

# Pandas: Minimum Values

pandas

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

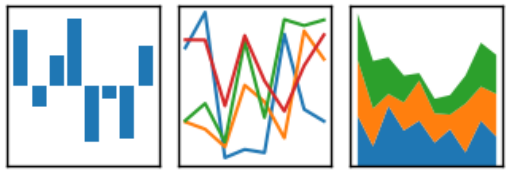


- In Pandas, lovely built-in functions:
  - `df.sort_values('First Name')` and
  - `df['First Name'].min()`
- What if you don't have a CSV and DataFrame, or data not ordered?

# Design Question: Find first alphabetically

pandas

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

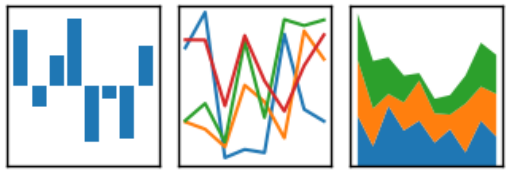


- What if you don't have a CSV and DataFrame, or data not ordered?

# Design Question: Find first alphabetically

pandas

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

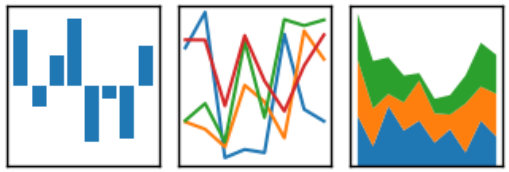


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max

# Design Question: Find first alphabetically

pandas

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

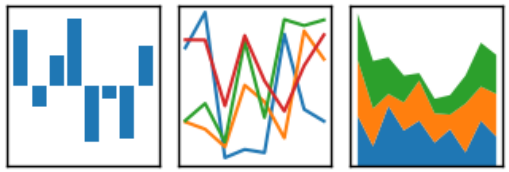


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).

# Design Question: Find first alphabetically

pandas

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

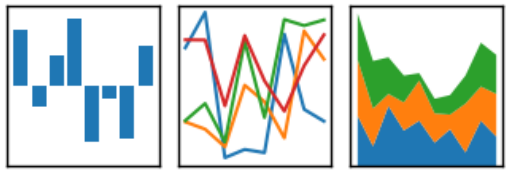


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:

# Design Question: Find first alphabetically

pandas

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

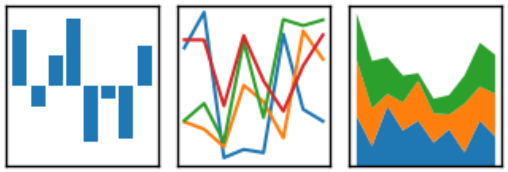


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.

# Design Question: Find first alphabetically

pandas

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

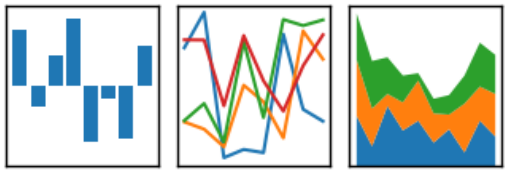


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.

# Design Question: Find first alphabetically

pandas

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



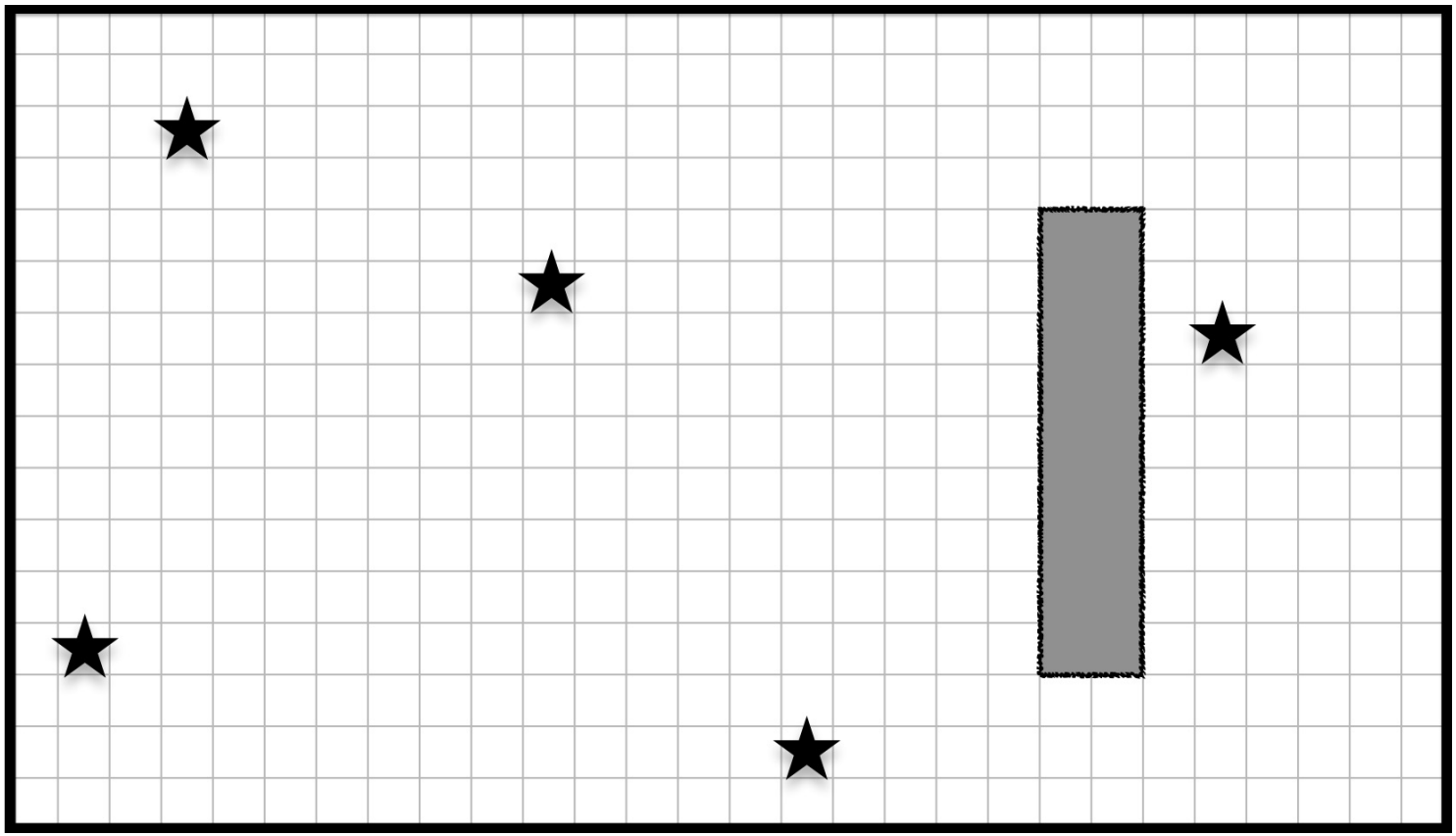
- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.
  - Print/return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

# Today's Topics

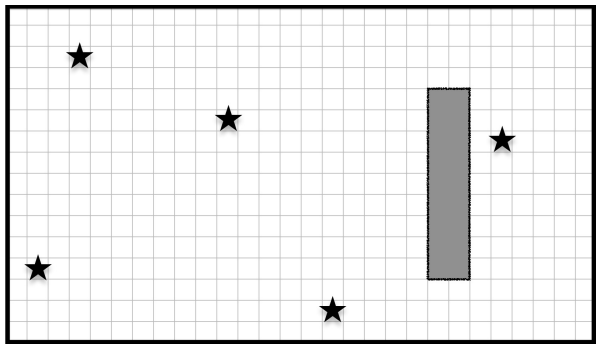
- Recap: Folium
- Indefinite loops
- Design Patterns: Max (Min)
- **Design Challenge**

# Design Challenge

Collect all five stars (locations randomly generated):

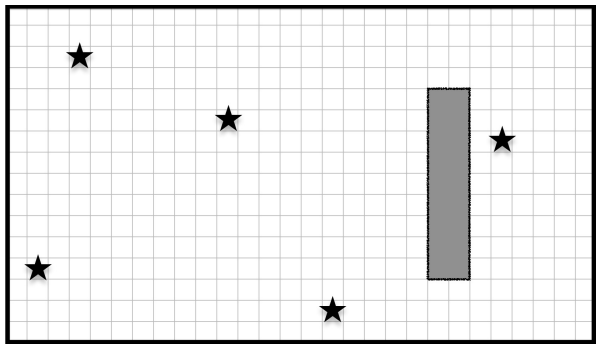


# Design Challenge



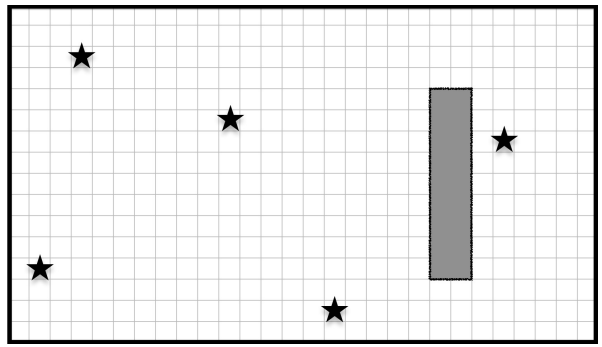
- Possible approaches:

# Design Challenge



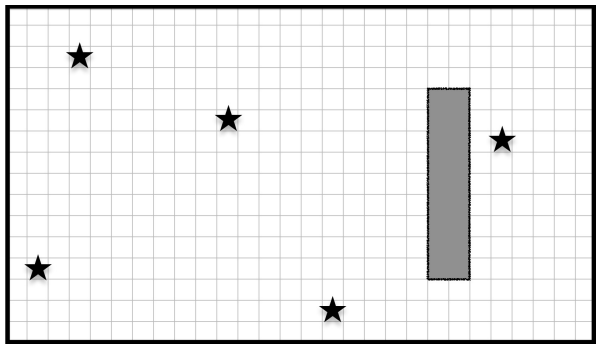
- Possible approaches:
  - Randomly wander until all 5 collected, or

# Design Challenge



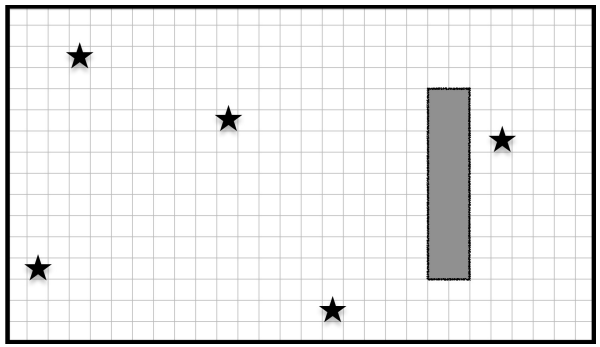
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.

# Design Challenge



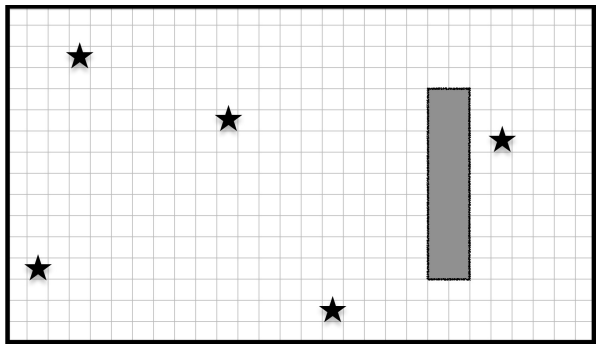
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'

# Design Challenge



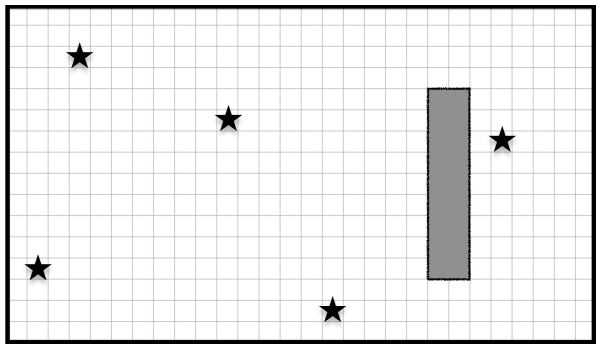
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.

# Design Challenge



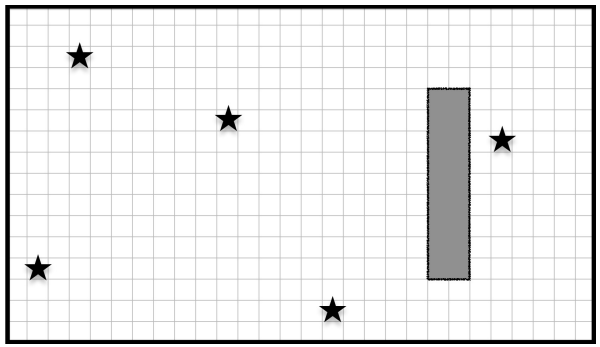
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.

# Design Challenge



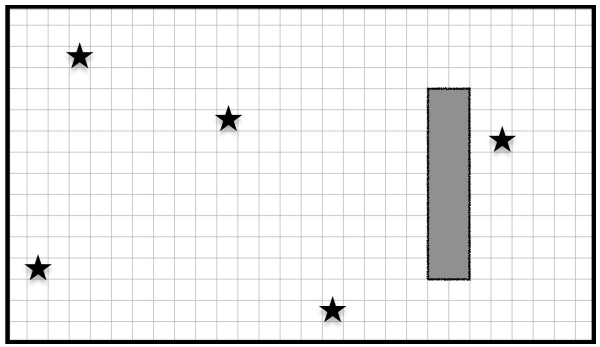
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

# Design Challenge



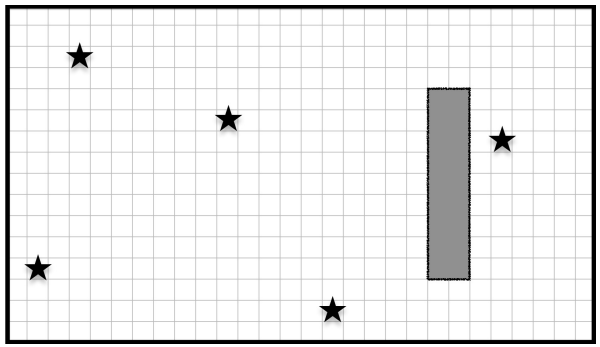
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

# Design Challenge



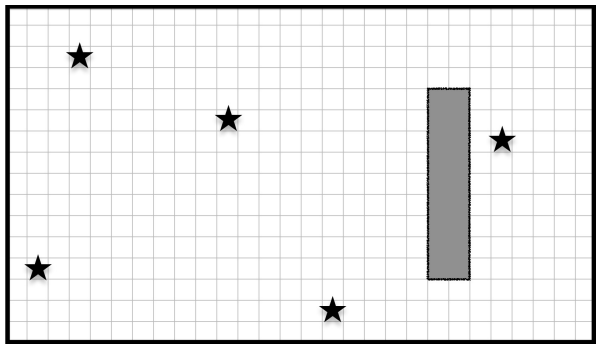
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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

# Design Challenge



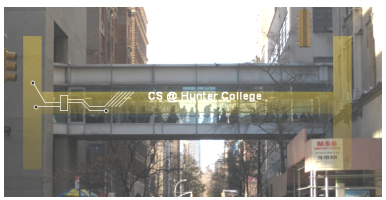
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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



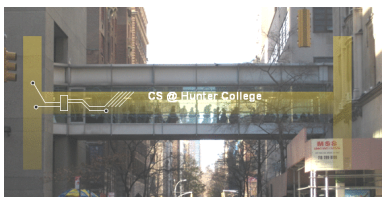
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point.
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy` array with -1 everywhere.
- Possible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



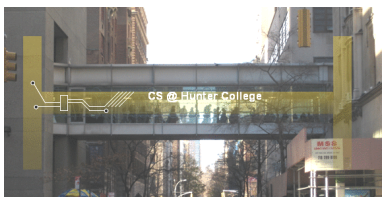

- Quick recap of a Python library, `Folium` for creating interactive HTML maps.

# Recap



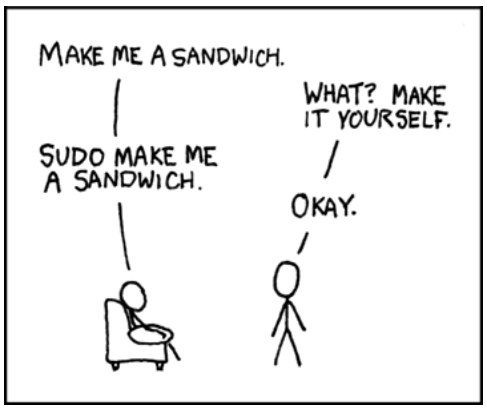

- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.

# 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 design pattern.

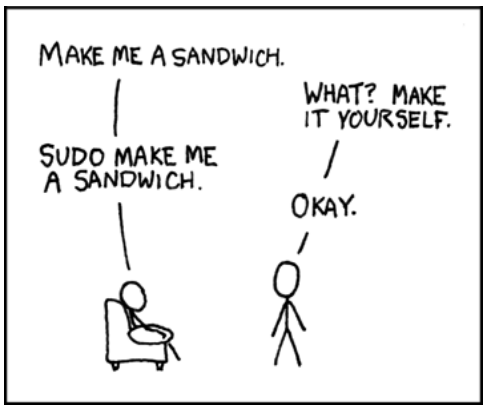
# Final Exam Prep: UNIX




xkcd 149

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

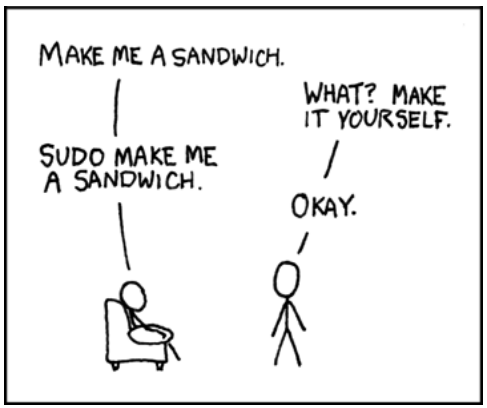
# Final Exam Prep: UNIX




xkcd 149

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

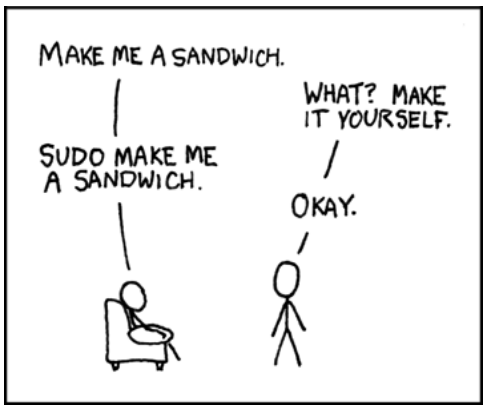
# Final Exam Prep: UNIX




xkcd 149

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

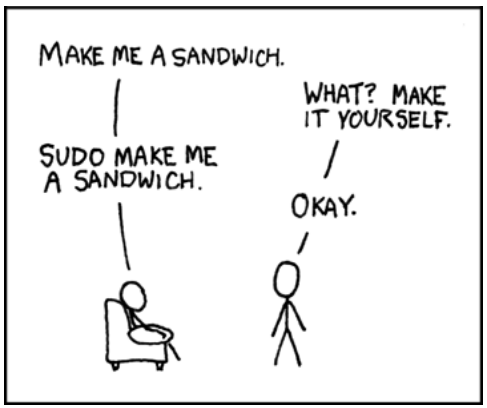
# Final Exam Prep: UNIX




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

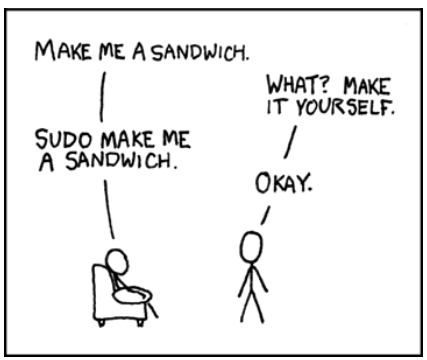



xkcd 149

- 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 on-line labs

# Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

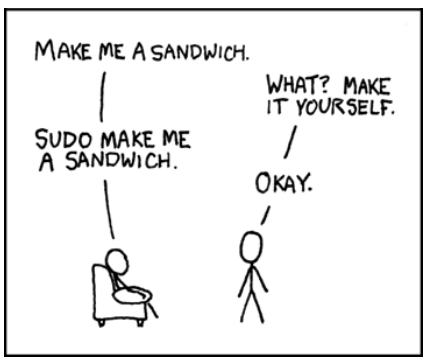



xkcd 149

# Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

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

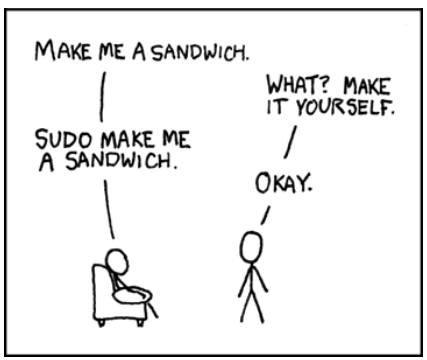



`xkcd 149`

# Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

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

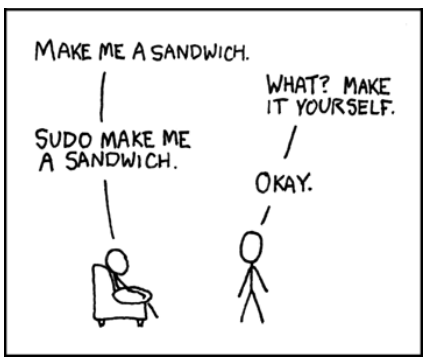



`xkcd 149`

# Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

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

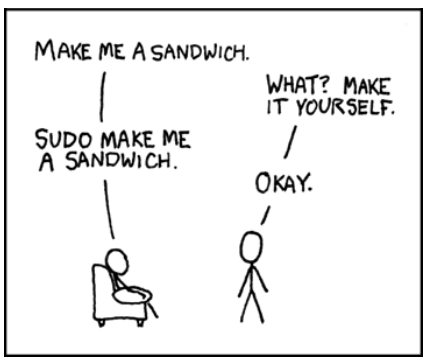



`xkcd 149`

# Final Exam Prep: UNIX

Unix commands in the weekly on-line 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 ~`

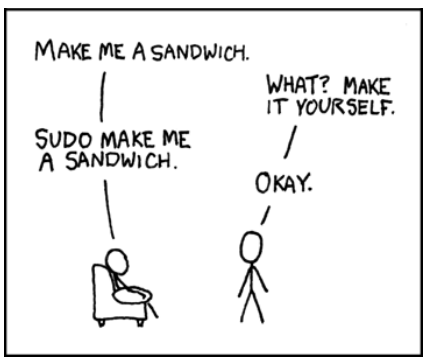



xkcd 149

# Final Exam Prep: UNIX

Unix commands in the weekly on-line 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`

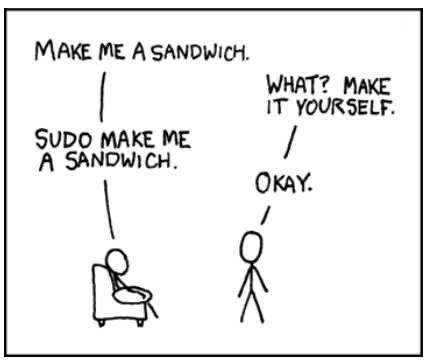



`xkcd 149`

# Final Exam Prep: UNIX

Unix commands in the weekly on-line 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

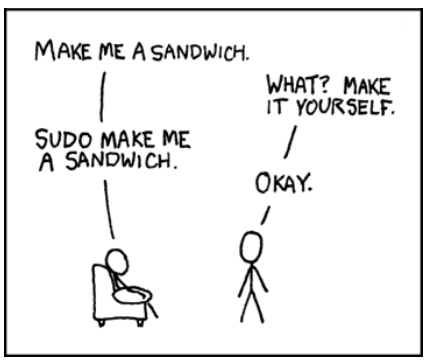



xkcd 149

# Final Exam Prep: UNIX

Unix commands in the weekly on-line 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

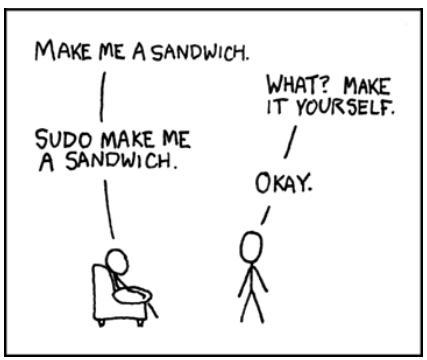



xkcd 149

# Final Exam Prep: UNIX

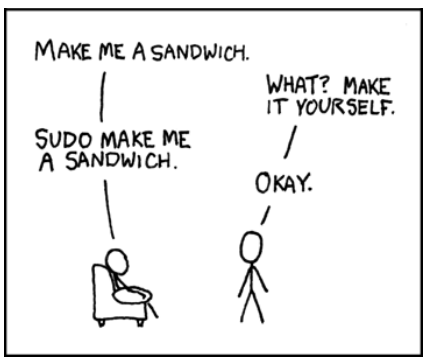
Unix commands in the weekly on-line labs:

- *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

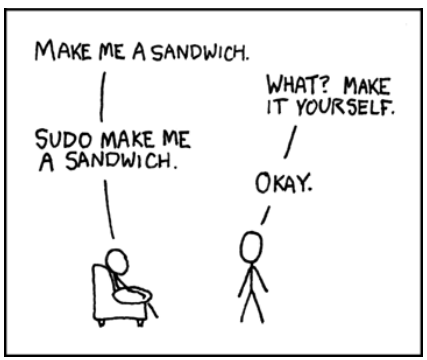



xkcd 149

Unix commands in the weekly on-line 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

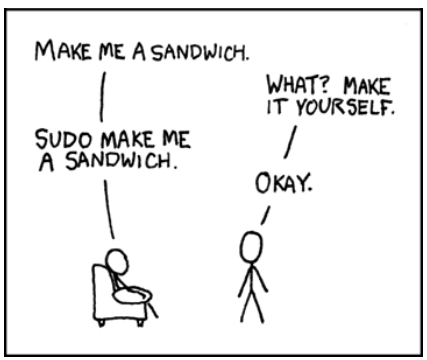



xkcd 149

Unix commands in the weekly on-line 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

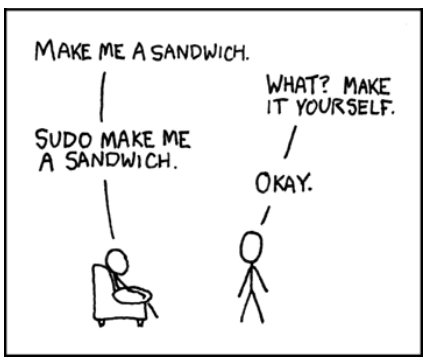



xkcd 149

Unix commands in the weekly on-line 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 on-line 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`