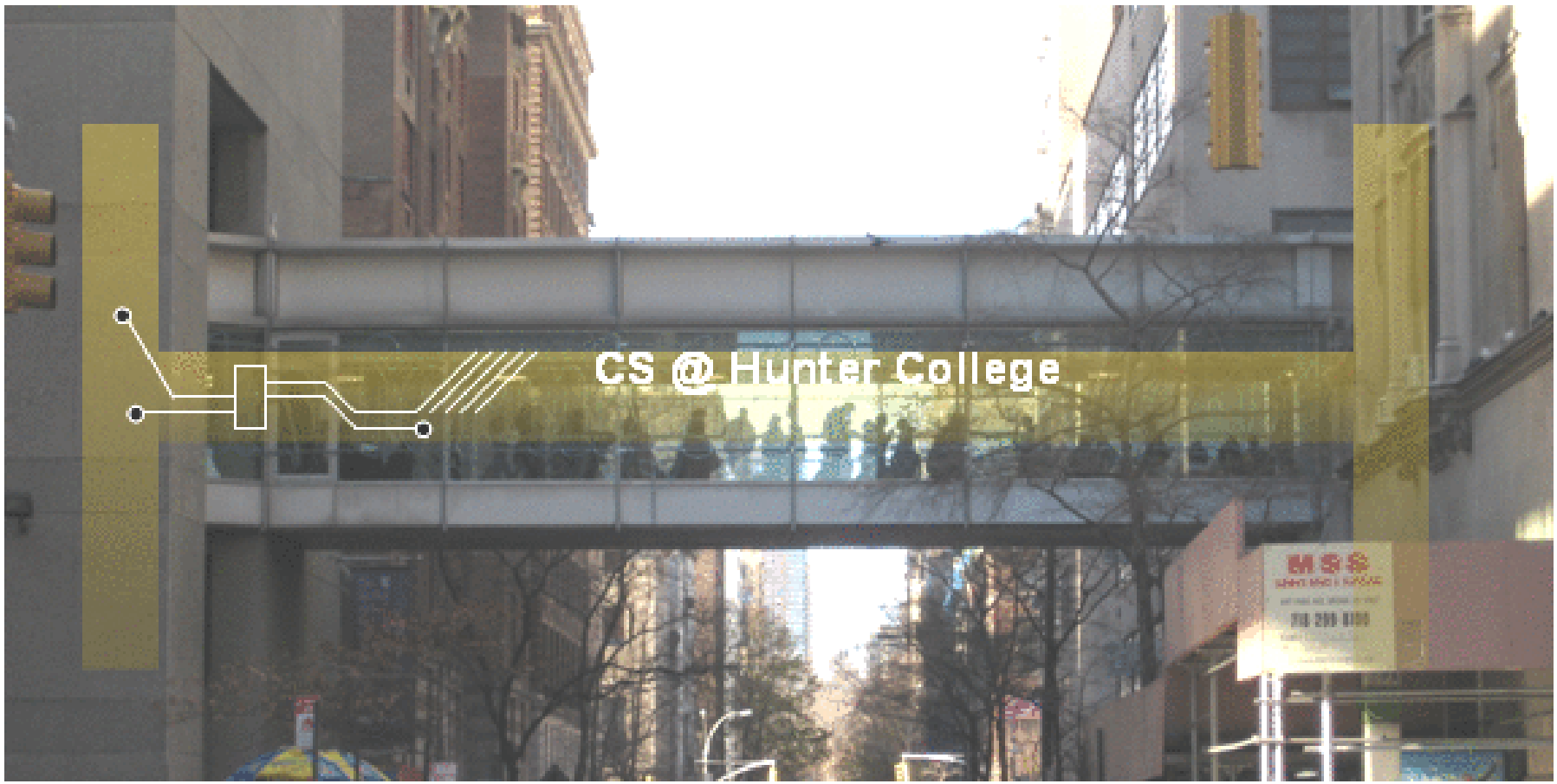
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, Assembly Hall: 118 Hunter North

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, Assembly Hall: 118 Hunter North
- **What is the format?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, Assembly Hall: 118 Hunter North
- **What is the format?**
  Content and format will be similar to past paper exams.

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, 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

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, 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 time: Friday December 16, 8-10 AM and room TBD.

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, 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 time: Friday December 16, 8-10 AM and room TBD.
- **Do I have to take the final?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, 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 time: Friday December 16, 8-10 AM and room TBD.
- **Do I have to take the final?**
  Yes, you must pass the final (60 out of 100 points) to the pass the class.

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, 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 time: Friday December 16, 8-10 AM and room TBD.
- **Do I have to take the final?**
  Yes, you must pass the final (60 out of 100 points) to the pass the class.
- I'd like to take more computer science. What's next?

Frequently Asked Questions

From lecture slips & recitation sections.

- **When is the final?**
  Monday Decemeber 19, 9am-11am, 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 time: Friday December 16, 8-10 AM and room TBD.
- **Do I have to take the final?**
  Yes, you must pass the final (60 out of 100 points) to the pass the class.
- I'd like to take more computer science. What's next?
  Fabulous! The next courses are:
  - *CSci 135: Programming in C++.*
    Lecture: **TBA**; *Sections: see schedule.*
  - *CSci 150: Discrete structures (math for computing).*
    Lecture: **TBA**; *Sections: see schedule.*

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?

```
1 import folium
2 import pandas as pd
3 import webbrowser #display html file
4 import os #use to find directory
```

What does this code do?: II

```
1 #Use pandas (alias pd) to read a csv file,
2 #save the returned data frame object in
3 #variable cuny.
4 cuny = pd.read_csv('cunyLocations.csv')
```

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,
...
```

What does this code do?: II

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

What does this code do? II

```
8  for index, row in cuny.iterrows():
9      #iterrow method of dataframe object cuny returns two
           values:
10     #the first is index, the second is row.
11     lat = row["Latitude"]
12     lon = row["Longitude"]
13     name = row["Campus"]
14     if row["College or Institution Type"] == "Senior
           Colleges":
15        collegeIcon = folium.Icon(color="purple")
16     else: collegeIcon = folium.Icon(color="blue")
17
18     #create a marker, sepcify its latitude, longitude,
19     #pop up name, and icon, save in variable newMarker.
20     newMarker = folium.Marker([lat, lon], popup=name,
           icon=collegeIcon)
21     newMarker.add_to(mapCUNY)
```

What does this code do? III

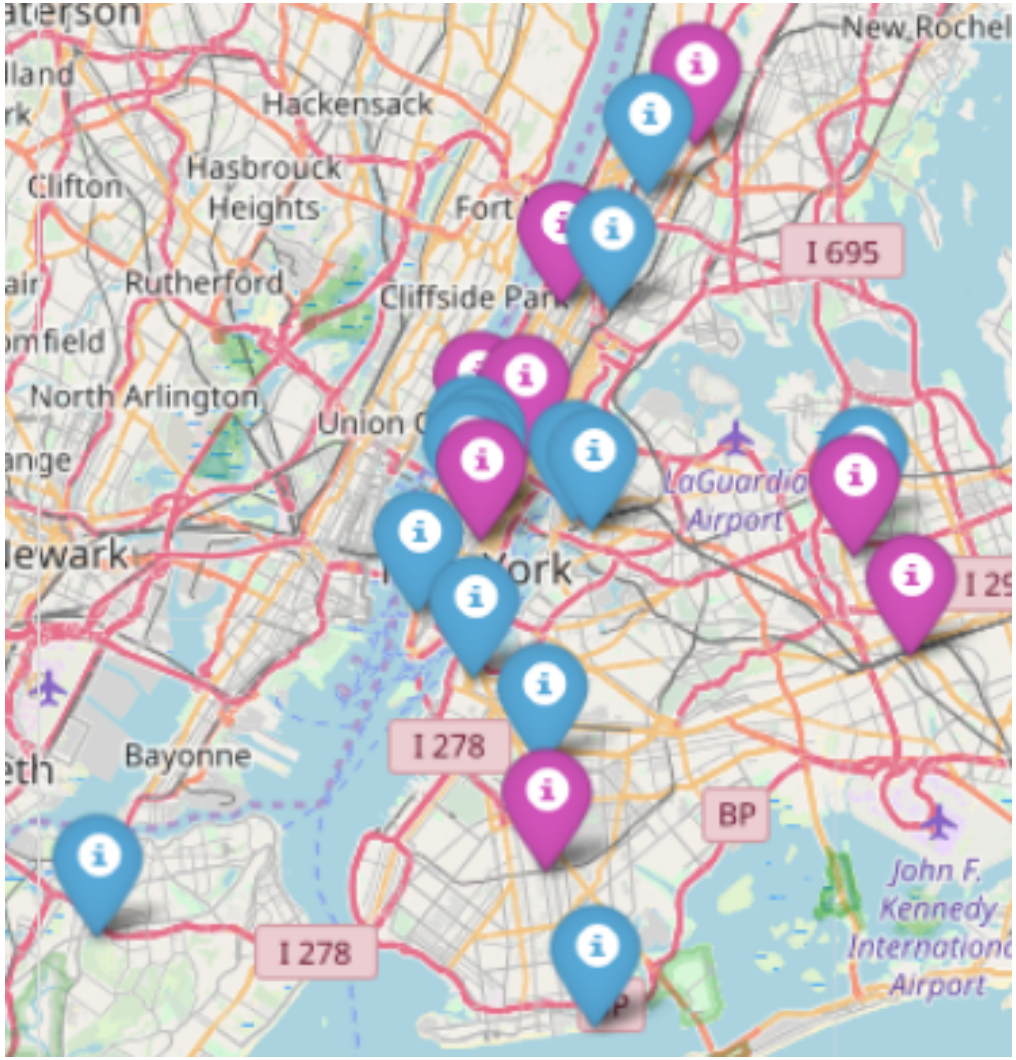
```
24 filename = 'cunyLocationsSenior.html'
25
26 #save mapCUNY to filename
27 mapCUNY.save(outfile = filename)
28
29 #display html using open method of
     webbrowser class.
30 webbrowser.open('file://' + os.path.
     realpath(filename))
```

What does the code do?: V

```
1  import folium
2  import pandas as pd
3  import webbrowser #display html file
4  import os #use to find directory
5
6  #Use pandas (alias pd) to read a csv file,
7  #save the return data frame object in variable cuny.
8  cuny = pd.read_csv('cunyLocations.csv')
9
10 #Create a map object centered at 40.75, -74.125,
11 #save in variable mapCUNY.
12 mapCUNY = folium.Map(location = [40.75, -74.125])
13
14 for index, row in cuny.iterrows():
15     lat = row["Latitude"]
16     lon = row["Longitude"]
17     name = row["Campus"]
18     if row["College or Institution Type"] == "Senior Colleges":
19        collegeIcon = folium.Icon(color="purple")
20     else:
21        collegeIcon = folium.Icon(color="blue")
22
23     #create a marker, sepcify its latitude, longitude,
24     #pop up name, and icon, save in variable newMarker.
25     newMarker = folium.Marker([lat, lon], popup=name, icon=collegeIcon)
26     newMarker.add_to(mapCUNY)
27
28 filename = 'cunyLocationsSenior.html'
29 #save mapCUNY to filename
30 mapCUNY.save(outfile = filename)
31
32 #display html using open method of webbrowser class
33 webbrowser.open('file://' + os.path.realpath(filename))
```

Folium example

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.

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.

Write code. $\rightarrow$ *Run program.*

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 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.

```
1 def getYear():
2     #TODO: initialize num
3
4     return num
```

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.

```
1 def getYear():
2     num = 0 #initialize num
3
4
5     return num
```

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.

```
1  def getYear():
2      num = 0 #initialize num
3      #Repeat entering num until it is in
4      #(2000, 2021), neither end is included.
5      #...invalid...(---valid---)...invalid...
6      #             2000        2021
7      while num <= 2000 or num >= 2021:
8          num = int(input("Enter a number after
               2000 and before 2021: "))
9      return num #outside loop
```

Define and Call function getYear

```
1  def getYear():
2      num = 0 #initialize num
3      #Repeat entering num until it is in (2000, 2021).
4      #...invalid...(--valid input--)...invalid...
5      #            2000          2021
6      while num <= 2000 or num >= 2021:
7          num = int(input("Enter num after 2000 and before
               2021: "))
8      return num
9
10 def main():
11     #num in main has nothing to do with num in getYear
12     num = getYear()
13     print("The year is", num)
14
15 if __name__ == '__main__':
16     main()
```

Can you spot an error?

```
1  num = 0
2
3  def getYear():
4      while num <= 2000 or num >= 2021:
5          num = int(input("Enter a
               number after 2000 and
               before 2021: "))
6
7      return num
```

Hints for Programming Assignment 44

Shades of red in Lab 2 looks like a petal.

```
1  import turtle
2
3  turtle.colormode(255) #Allows colors to be given as
       0...255
4  t = turtle.Turtle() #Create a turtle
5
6  #For 0, 10, 20, ..., 250
7  for i in range(0, 255, 10):
8      t.forward(10) #Move forward
9      t.pensize(i) #Set the drawing size to be i (larger
           each time)
10     t.color(i, 0, 0) #Set the red channel to be i (
           brighter each time)
```

Hints for Programming Assignment 44: II

How to change the color of shades? For example, how to generate shades of yellow?

```
1  import turtle
2
3  turtle.colormode(255) #Allows colors to be given as
       0...255
4  t = turtle.Turtle() #Create a turtle
5
6  #For 0, 10, 20, ..., 250
7  for i in range(0, 255, 10):
8      t.forward(10) #Move forward
9      t.pensize(i) #Set the drawing size to be i (larger
           each time)
10     t.color( i, i , 0) #Set the red channel to be i (
           brighter each time)
```

Hints for Programming Assignment 44: III

Define function petal, draw a petal in given color and tilted a given angle in the beginning. **Is there a return?**

Philosophy of function: someone needs to do the work!

```
1  import turtle
2  def petal(color, angle):
3      turtle.colormode(255)
4      t = turtle.Turtle()
5      #TODO: t turns (ie, tilts) left angle degree
6      for i in range(0, 255, 10):
7          t.forward(10)
8          t.pensize(i)
9          if color == 'red':
10             t.color( i , 0, 0)
11         elif color == 'green':
12             t.color(0, i , 0)
13         #omit the rest of color
```

Can you spot an error?

In Programming Assignment 44, which defines function petal(color, angle) and flower(color, numPetals).

```
1 import turtle
2
3 turtle.colormode(255)
4
5 def petal(color,angle):
6     t = turtle.Turtle()
7     #omit the rest codes
```

Draw a flower

For a given color and numPetals, draw a flower.

```
1 def flower(color, numPetals):
2     #numPetals: number of petals of a flower
3     angle = ? #initialize angle, what is the
          angle of the first petal?
4     Do the following for numPetals times:
5        (1) draw a petal for the given color
            and angle
6        (2) Update angle as the tilting angle
             of the petal right to the current
            petal. What is the angle between a
            petal and its right neighbor for a
            flower with numPetals petals?
```

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)
```

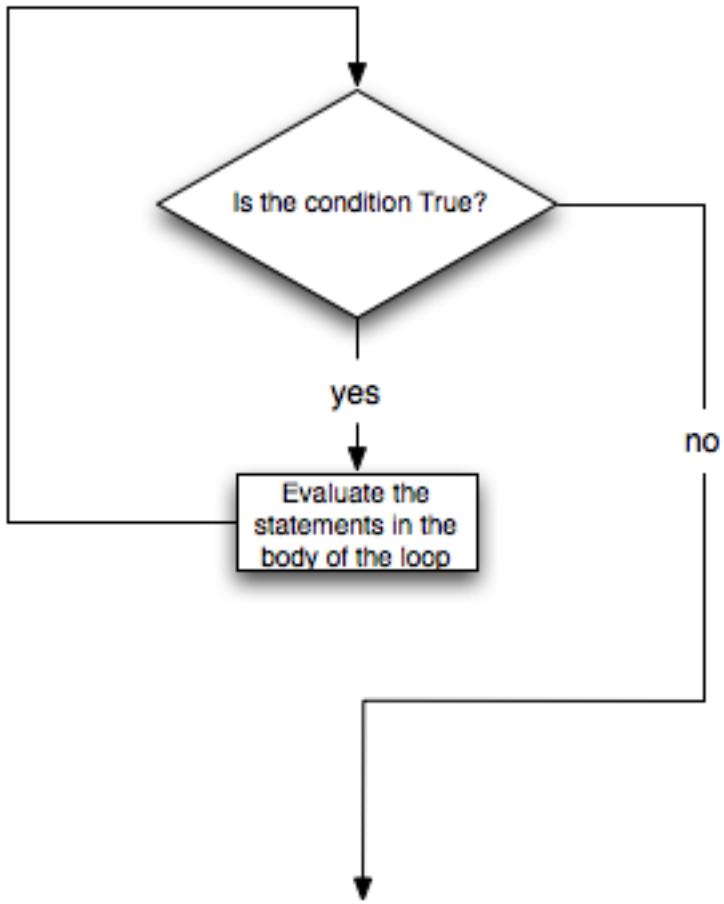
- Indefinite loops repeat as long as the condition is true.
- Could execute the body of the loop zero times, 10 times, infinite number of times.
- The condition determines how many times.
- Very useful for checking input, simulations, and games.

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)
```



Switch adjacent elements if left element is smaller

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

Switch adjacent elements if left element is smaller

```
1 nums = [1, 4, 0, 6, 5] #simplify with fewer numbers
2 print(nums)
3 i = 0
4 while i < len(nums)-1:
5     if nums[i] < nums[i+1]:
6         nums[i], nums[i+1] = nums[i+1], nums[i]
7     i = i+1
8 print(nums)
```

lens(nums) is 5.

i	i < len(nums)-1	if nums[i] < nums[i+1], swap nums[i] and nums[i+1]
0	yes	nums[0]=1 and nums[1]=4, swap, nums=[4,1,...]
1	yes	nums[1]=4 and nums[2]=0, no swap, nums[4,1,0,...]
2	yes	nums[2]=0 and nums[3]=6, swap. nums[4,1,6,0,...]
3	yes	nums[3]=0 and nums[4]=5, swap. nums[4,1,6,5,0]
4	no, exit loop	

Now nums is [4, 1, 6, 5, 0]

Challenge

Predict what this code does:

```
1 def move(): #move tess
2     tess = turtle.Turtle()
3     tess.color('steelBlue')
4     tess.shape('turtle')
5     tess.penup()
6     #Start off screen:
7     tess.goto(-250,-250)
```

Predict what the code do: II

```
8   #Remember: abs(x) < 25 means absolute
        value: -25 < x < 25
9   while abs(tess.xcor()) > 25 or abs(tess.
        ycor()) > 25:
10    x = random.randrange(-200,200)
11    y = random.randrange(-200,200)
12    tess.goto(x,y)
13    tess.stamp()
14    print(tess.xcor(), tess.ycor())
15  print('Found the center!')
16
17  turtle.done()
```

Trinket Demo

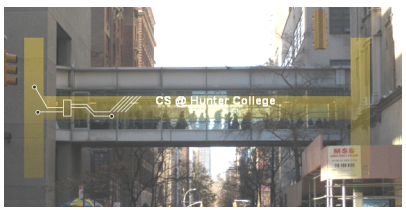
```
#Random search
import turtle
import random
tess = turtle.Turtle()
tess.color('steelBlue')
tess.shape('turtle')
tess.penup()
#Start off screen:
tess.goto(-250,-250)
#Remember:  abs(x) < 25 means absolute value: -25 < x < 25
while abs(tess.xcor()) > 25 or abs(tess.ycor()) > 25:
  x = random.randrange(-200,200)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

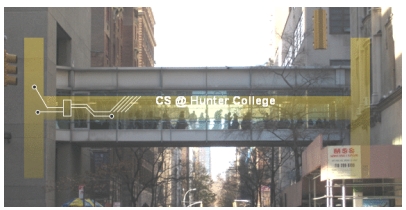
- Recap: Folium
- Indefinite loops
- **Design Patterns: Max (Min)**
- Design Challenge

Design Patterns



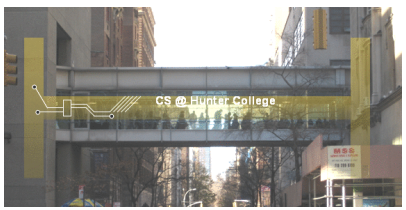
- A **design pattern** is a standard algorithm or approach for solving a common problem.

Design Patterns



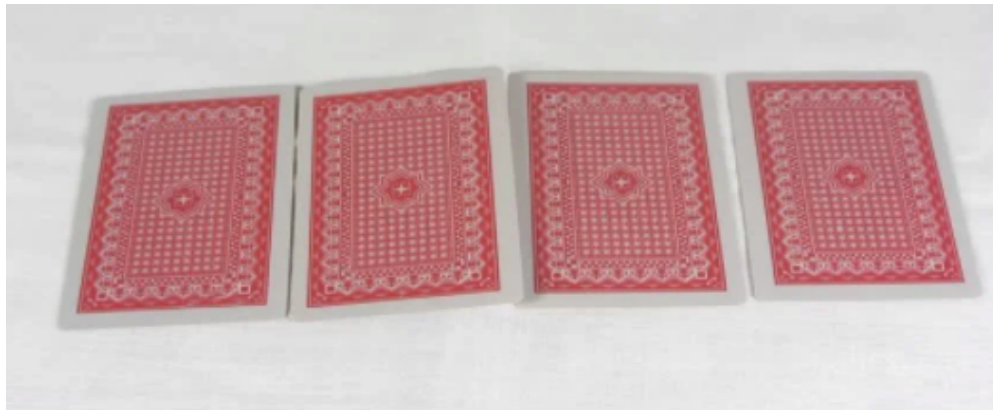
- A **design pattern** is a standard algorithm or approach for solving a common problem.
- The pattern is independent of the programming language.

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.

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:

```
1 nums = [1, 4, 10, 6, 5, 42, 9, 8, 12]
2 maxNum = 0
3 for n in nums:
4     if n > maxNum:
5         #TODO: update maxNum to be n
6
7 print(maxNum)
```

Fill in Code

```
1 nums = [1, 4, 10, 6, 5, 42, 9, 8, 12]
2 maxNum = 0
3 for n in nums:
4     if n > maxNum:
5         #TODO: update maxNum to be n
6         maxNum = n
7
8 print(maxNum)
```

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1
4	n > maxNum, set maxNum to 4	4

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1
4	n > maxNum, set maxNum to 4	4
10	n > maxNum, set maxNum to 10	10

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1
4	n > maxNum, set maxNum to 4	4
10	n > maxNum, set maxNum to 10	10
6	n is not > maxNum, no update on maxNum	10

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1
4	n > maxNum, set maxNum to 4	4
10	n > maxNum, set maxNum to 10	10
6	n is not > maxNum, no update on maxNum	10
5	n is not > maxNum, no update on maxNum	10

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1
4	n > maxNum, set maxNum to 4	4
10	n > maxNum, set maxNum to 10	10
6	n is not > maxNum, no update on maxNum	10
5	n is not > maxNum, no update on maxNum	10
42	n > maxNum, set maxNum to 42	42

Step by Step Explanation

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

n	if n > maxNum: maxNum = n	maxNum initialized to be 0
1	n > maxNum, set maxNum to 1	1
4	n > maxNum, set maxNum to 4	4
10	n > maxNum, set maxNum to 10	10
6	n is not > maxNum, no update on maxNum	10
5	n is not > maxNum, no update on maxNum	10
42	n > maxNum, set maxNum to 42	42
...	...	...

Improvement

```
1 nums = [-1, -5, -4]
2 maxNum = 0
3 for n in nums:
4     if n > maxNum:
5         maxNum = n
6
7 print(maxNum)
```

Improvement: II

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

```
1 import sys #use sys.maxsize
2
3 nums = [-1, -5, -4]
4 maxNum = -sys.maxsize-1
5 for n in nums:
6     if n > maxNum:
7         maxNum = n
8
9 print(maxNum)
```

Improvement: III

Initialize with the first element in the list, one of its own.

```
1  nums = [1, 4, 10, 6, 5, 42, 9, 8, 12]
2  size = len(nums)
3  if size == 0:
4      print("The list is empty and does not
           have a maximum")
5      exit(0) #cannot use return since this is
           not an function
6  maxNum = nums[0]
7  for i in range(1, size):
8      if nums[i] > maxNum:
9          maxNum = nums[i]
10 print(maxNum)
```

Improvement: IV

Define a function to return a maximum element in a given list.

Input: a list of numbers Return: maximum element of this list

```
1 def maximum(nums):
2     size = len(nums)
3     if size == 0:
4         #print("The list is empty and does
              not have a maximum") #do not print
              intermediate result unless the duty
               of the function is input/output.
5         return float('nan') #nan means Not a
              Number
6         #int('nan') does not work, since int
              in python is not bounded.
```

Improvement: IV

Define a function to return the maximum element of a list.

```
8       maxNum = nums[0]
9       for i in range(1, size):
10          if nums[i] > maxNum:
11              maxNum = nums[i]
12
13      return maxNum
```

Analog: Cute baby Competition

Babies may join competition at any time.

If the incoming baby is cuter than the current winner model, choose the new baby as model; otherwise, keep the current model.

Cute baby Competition: II

In January, Alex came and became the winner model.

Oh yea, no competitors yet!

Cute baby Competition: III

After several months, Bob comes.

Bob competes with winner model .

If Bob is cuter, then he becomes winner model .

winner model

Cute baby Competition: IV

After several months, Chris comes.

Chris competes with winner model .

If Chris is cuter, then he becomes winner model .

winner model

Pseudo code to find a winner model

1. The first comer is awarded as the winner model automatically.
2. Every time a new comer comes, he/she challenges the current winner model. If the new comer wins, he/she becomes the new winner model (what if the challenger does not win?)
3. Repeat (2) for every new comer.

General idea to find max or min

```
1  Initialize the min to be the first element
2
3  for the rest of elements:
4      if the incoming element is smaller than
         the current minimum, update the
         current minimum to be the current
         element
5    #smaller than the current smallest
6    #bigger than the current biggest
7    #stronger than the current strongest
8    #weaker than the current weakest
```

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)
```

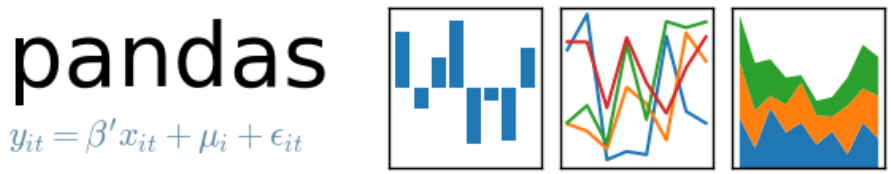
- Set a variable to the smallest value.
- Loop through the list,
- If the current number is larger, update your variable.
- Print/return the largest number found.
- Must look at entire list to determine max is found
- Similar idea works for finding the minimum value.

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.
- Different from **Linear Search**: can stop when value you are looking for is found.

Pandas: Minimum Values

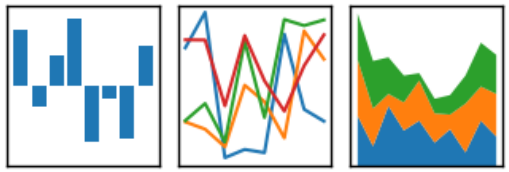


- In Pandas, lovely built-in functions:

Pandas: Minimum Values

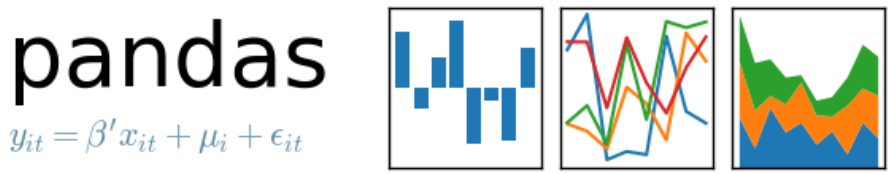
pandas

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



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

Pandas: Minimum Values

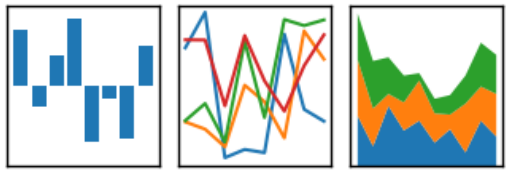


- In Pandas, lovely built-in functions:
  - `df.sort_values('First Name')` and
  - `df['First Name'].min()`
- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

pandas

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

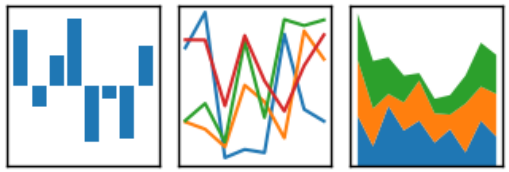


- What if you don't have a CSV and DataFrame, or data not ordered?

Design Question: Find first alphabetically

pandas

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

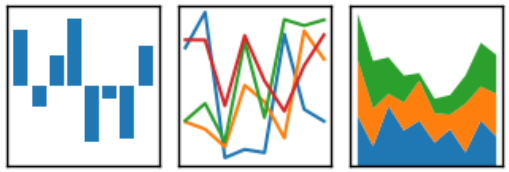


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max

Design Question: Find first alphabetically

pandas

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

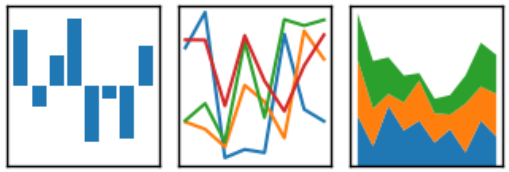


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).

Design Question: Find first alphabetically

pandas

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

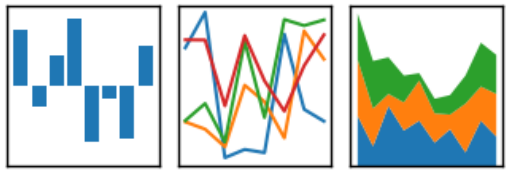


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:

Design Question: Find first alphabetically

pandas

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

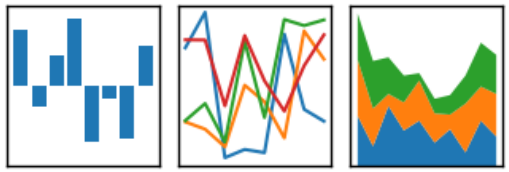


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.

Design Question: Find first alphabetically

pandas

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

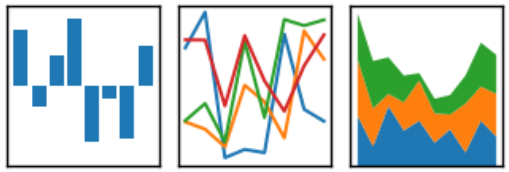


- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.

Design Question: Find first alphabetically

pandas

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



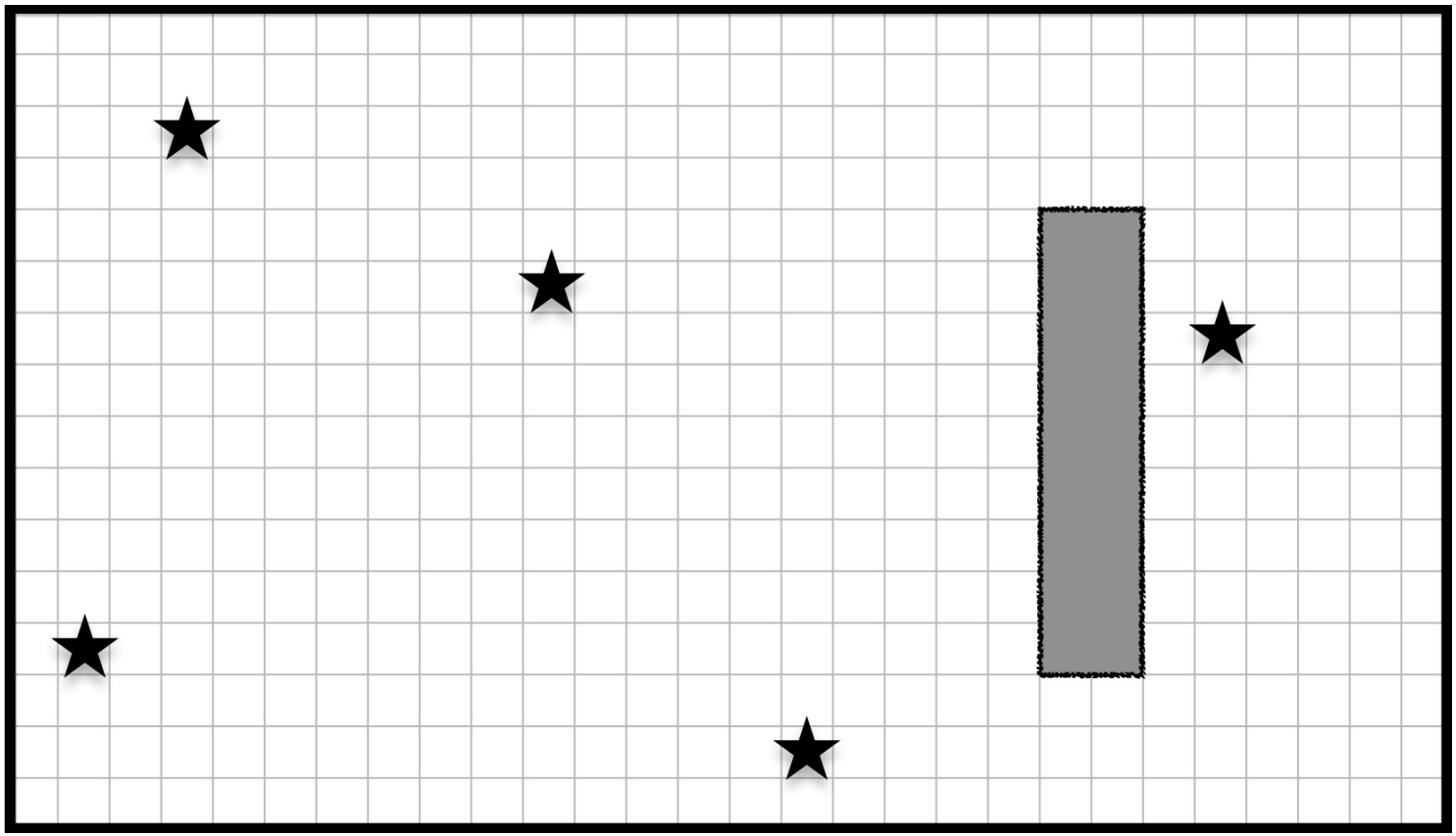
- What if you don't have a CSV and DataFrame, or data not ordered?
- Useful *Design Pattern*: min/max
  - Set a variable to worst value (i.e. `maxN = 0` or `first = "ZZ"`).
  - For each item, X, in the list:
    - ★ Compare X to your variable.
    - ★ If better, update your variable to be X.
  - Print/return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

Today's Topics

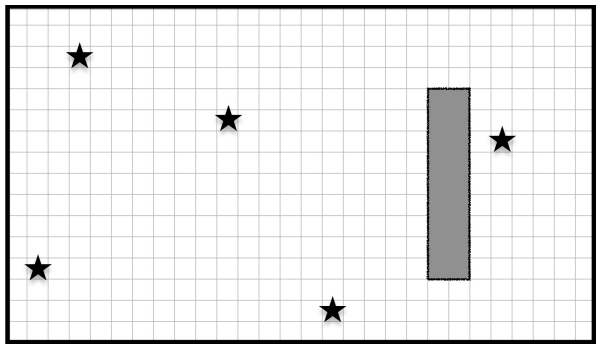
- Recap: Folium
- Indefinite loops
- Design Patterns: Max (Min)
- **Design Challenge**

Design Challenge

On your Lecture Slip: collect all five stars (locations randomly generated):

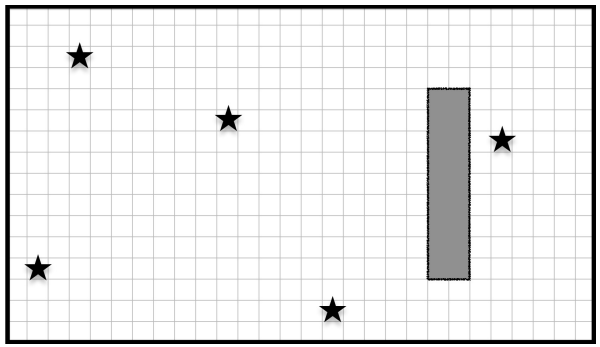


Design Challenge



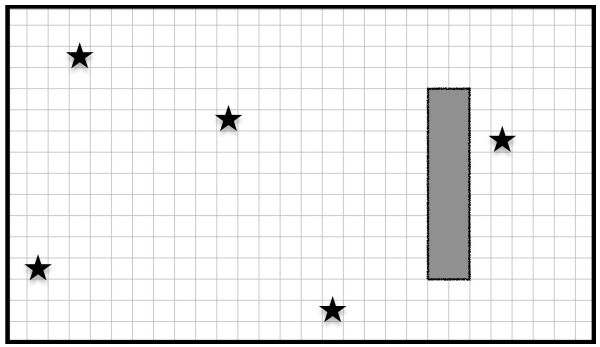
- Possible approaches:

Design Challenge



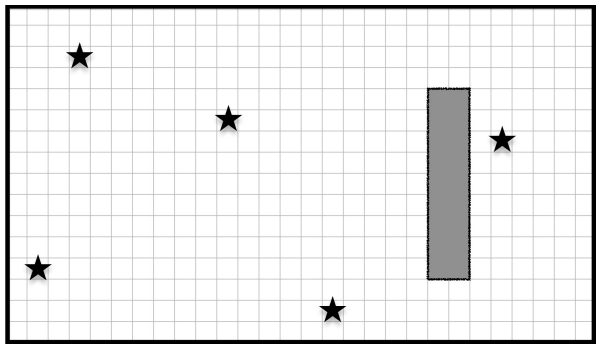
- Possible approaches:
  - Randomly wander until all 5 collected, or

Design Challenge



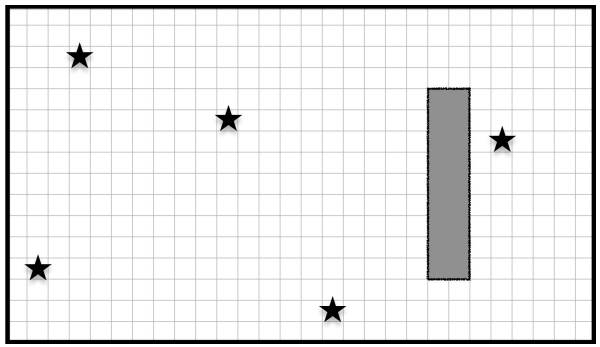
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).

Design Challenge



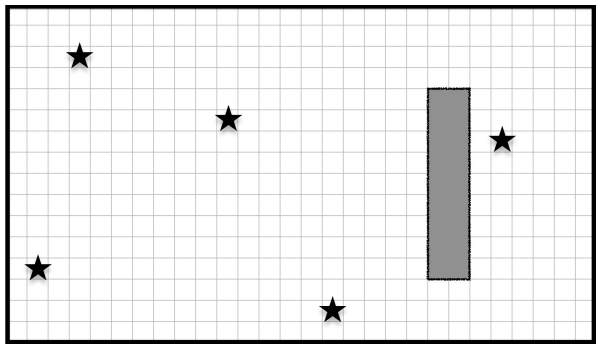
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'

Design Challenge



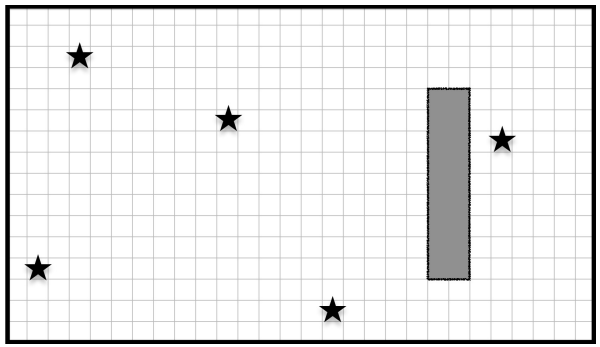
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.

Design Challenge



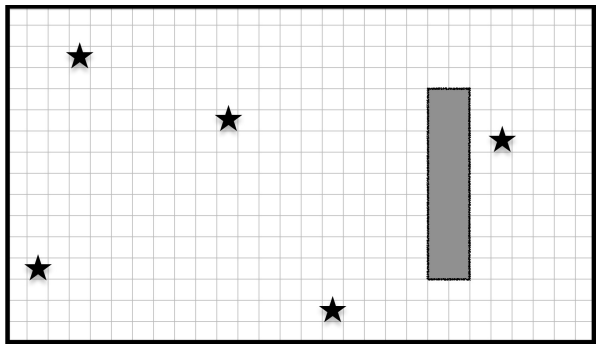
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy`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

Design Challenge



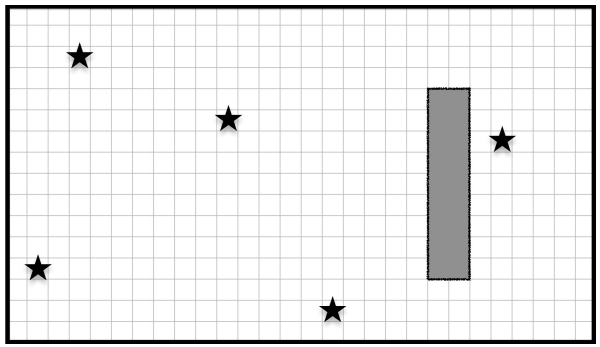
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy`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

Design Challenge



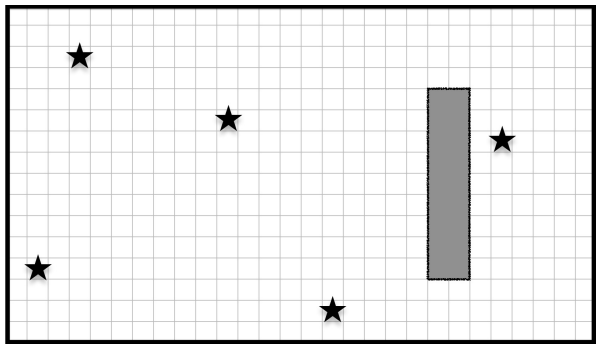
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy`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

Design Challenge



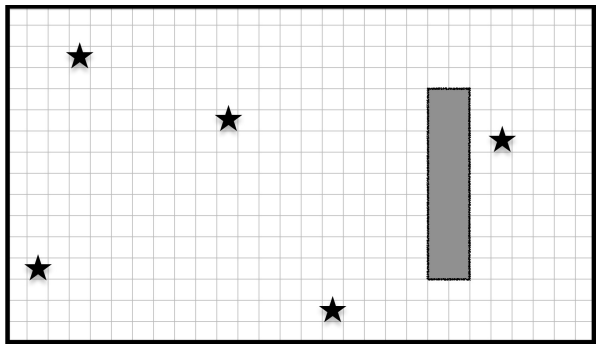
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy`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

Design Challenge



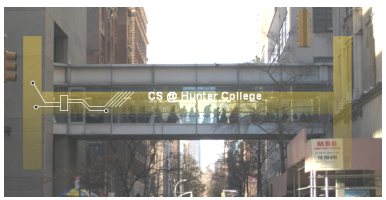
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy`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



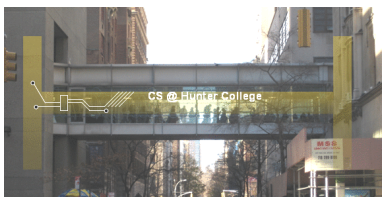
- Possible approaches:
  - Randomly wander until all 5 collected, or
  - Start in one corner, and systematically visit every point until 5 stars found (**Linear Search**).
- **Input:** The map of the 'world.'
- **Output:** Time taken and/or locations of the 5 stars.
- How to store locations? Use `numpy`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



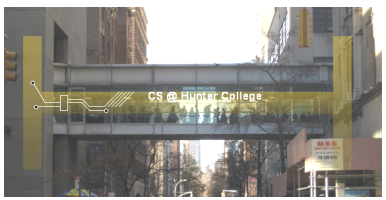
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.

Recap



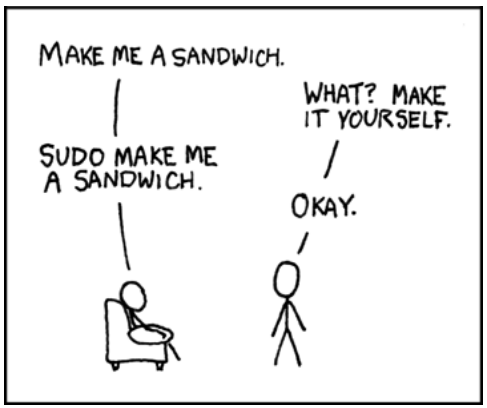
- Quick recap of a Python library, `Folium` for creating interactive HTML maps.
- More details on `while` loops for repeating commands for an indefinite number of times.

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/min and linear-search design pattern.

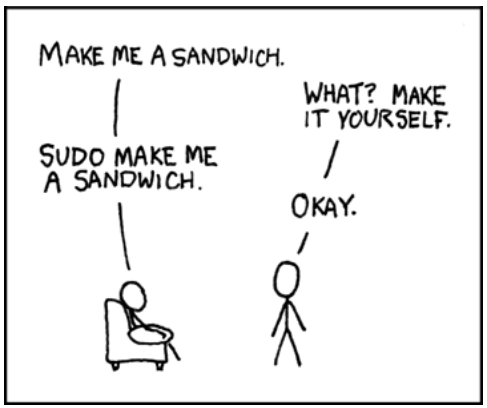
Final Exam Prep: UNIX



xkcd 149

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

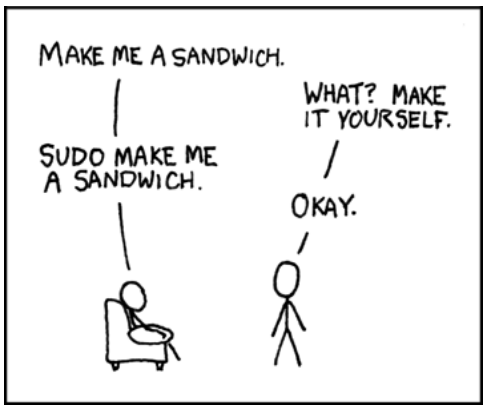
Final Exam Prep: UNIX



xkcd 149

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

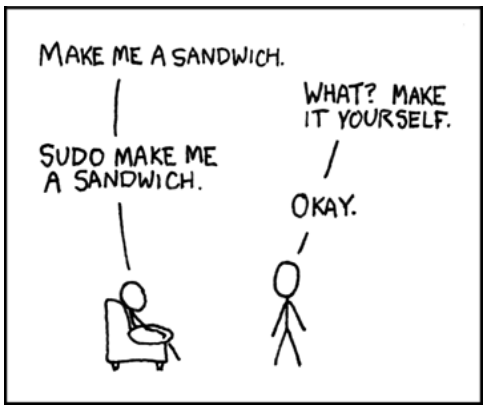
Final Exam Prep: UNIX



xkcd 149

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

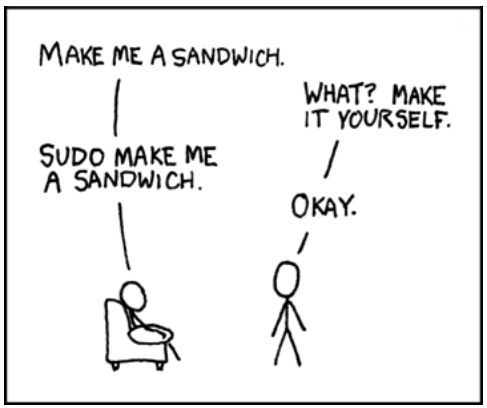
Final Exam Prep: UNIX



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

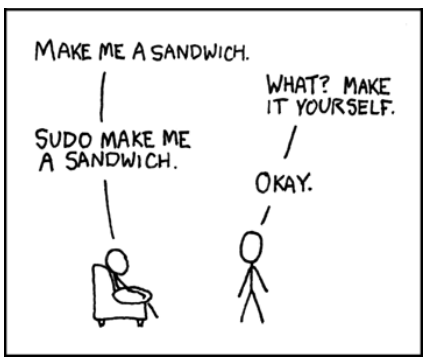


xkcd 149

- 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:

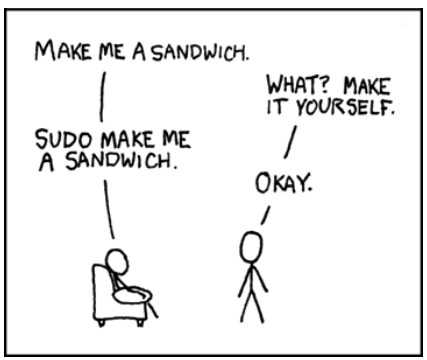


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

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

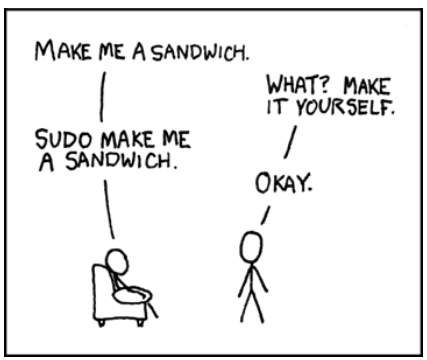


`xkcd 149`

Final Exam Prep: UNIX

Unix commands in the weekly on-line labs:

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

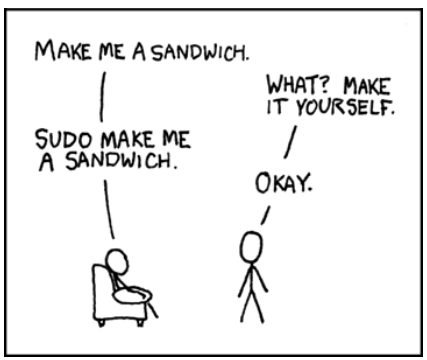


`xkcd 149`

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)

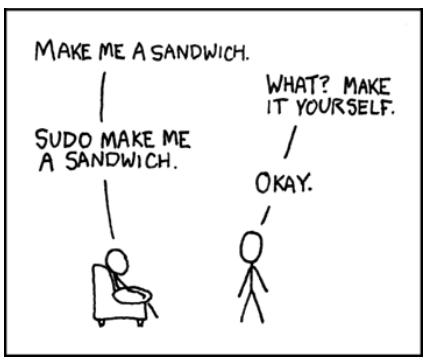


`xkcd 149`

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 ~`

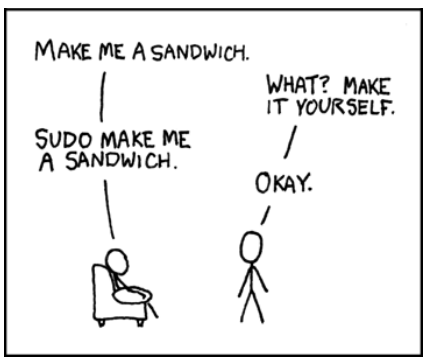


xkcd 149

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`

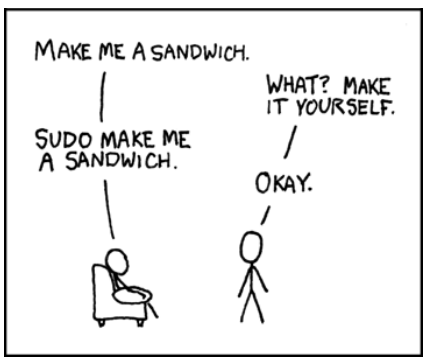


`xkcd 149`

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

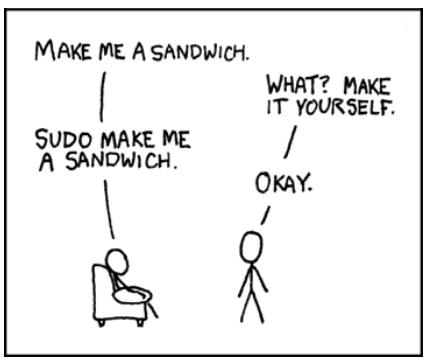


xkcd 149

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

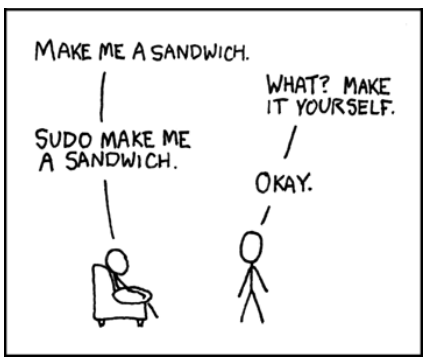


xkcd 149

Final Exam Prep: UNIX

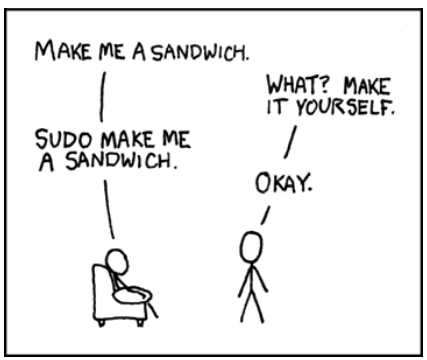
Unix commands in the weekly on-line labs:

- *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

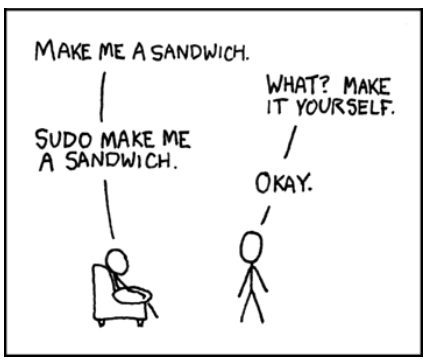


xkcd 149

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

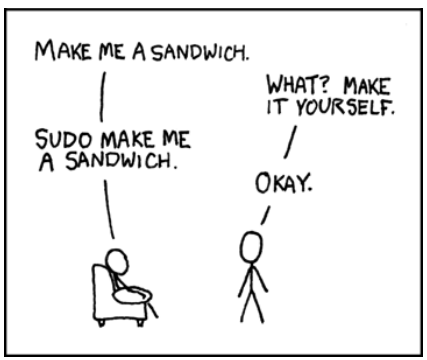


xkcd 149

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

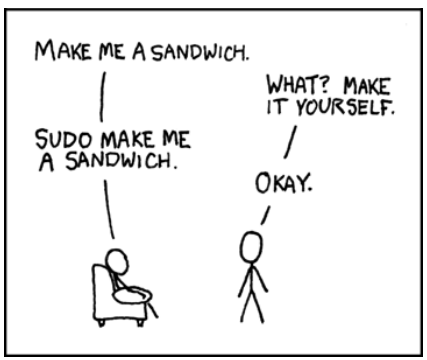


xkcd 149

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

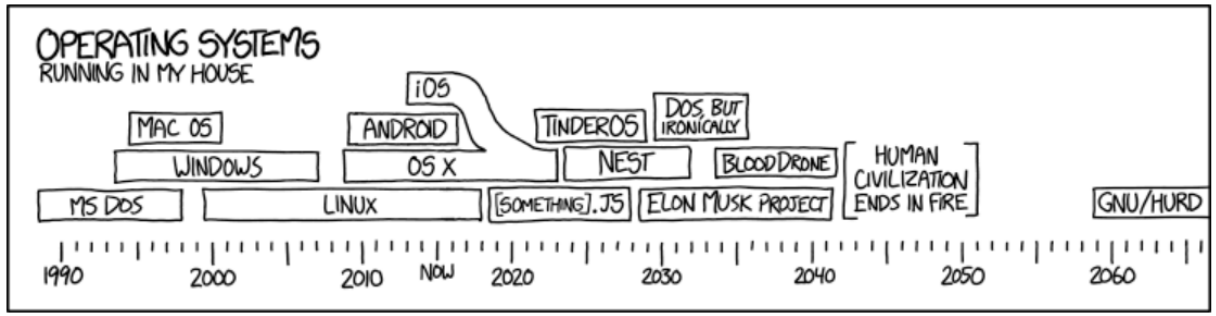


xkcd 149

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`

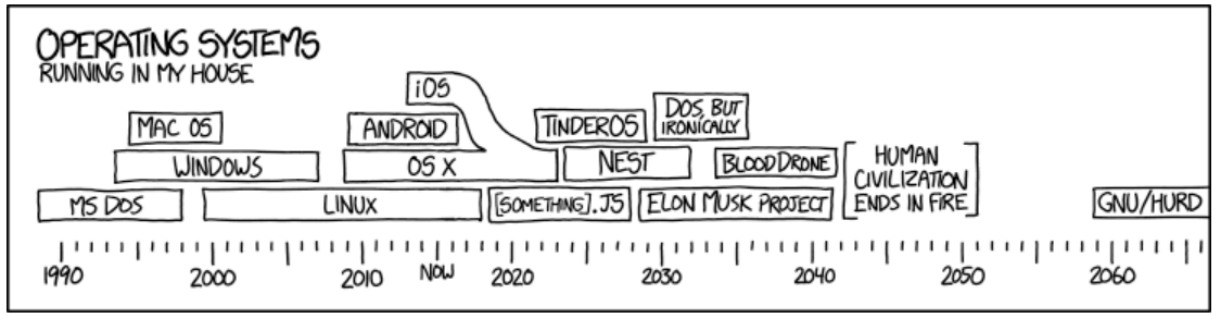
Practice Quiz & Final Questions



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).

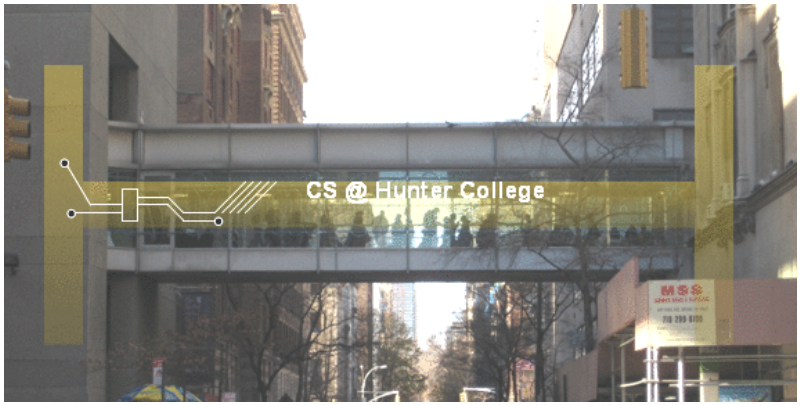
Practice Quiz & Final Questions



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 next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!

Before next lecture, don't forget to:

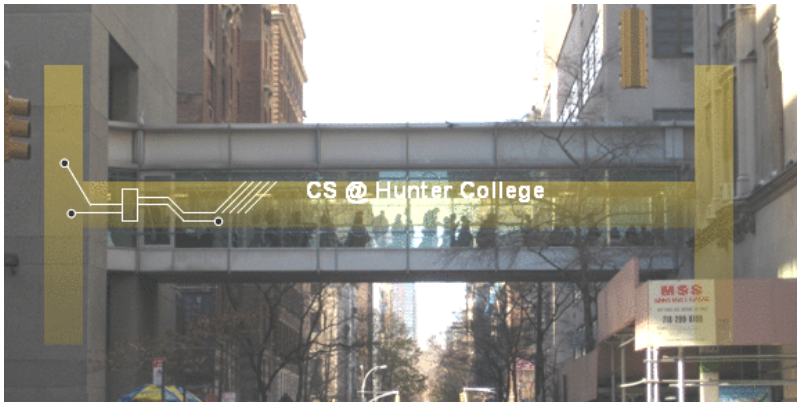
- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North

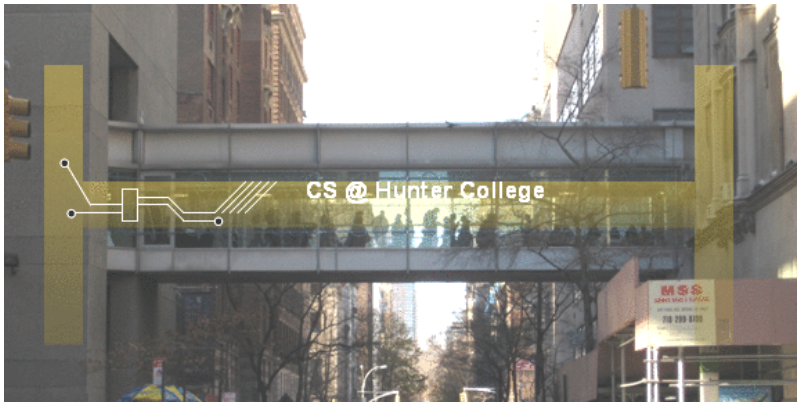
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 46-50)**

Weekly Reminders!



Before next lecture, don't forget to:

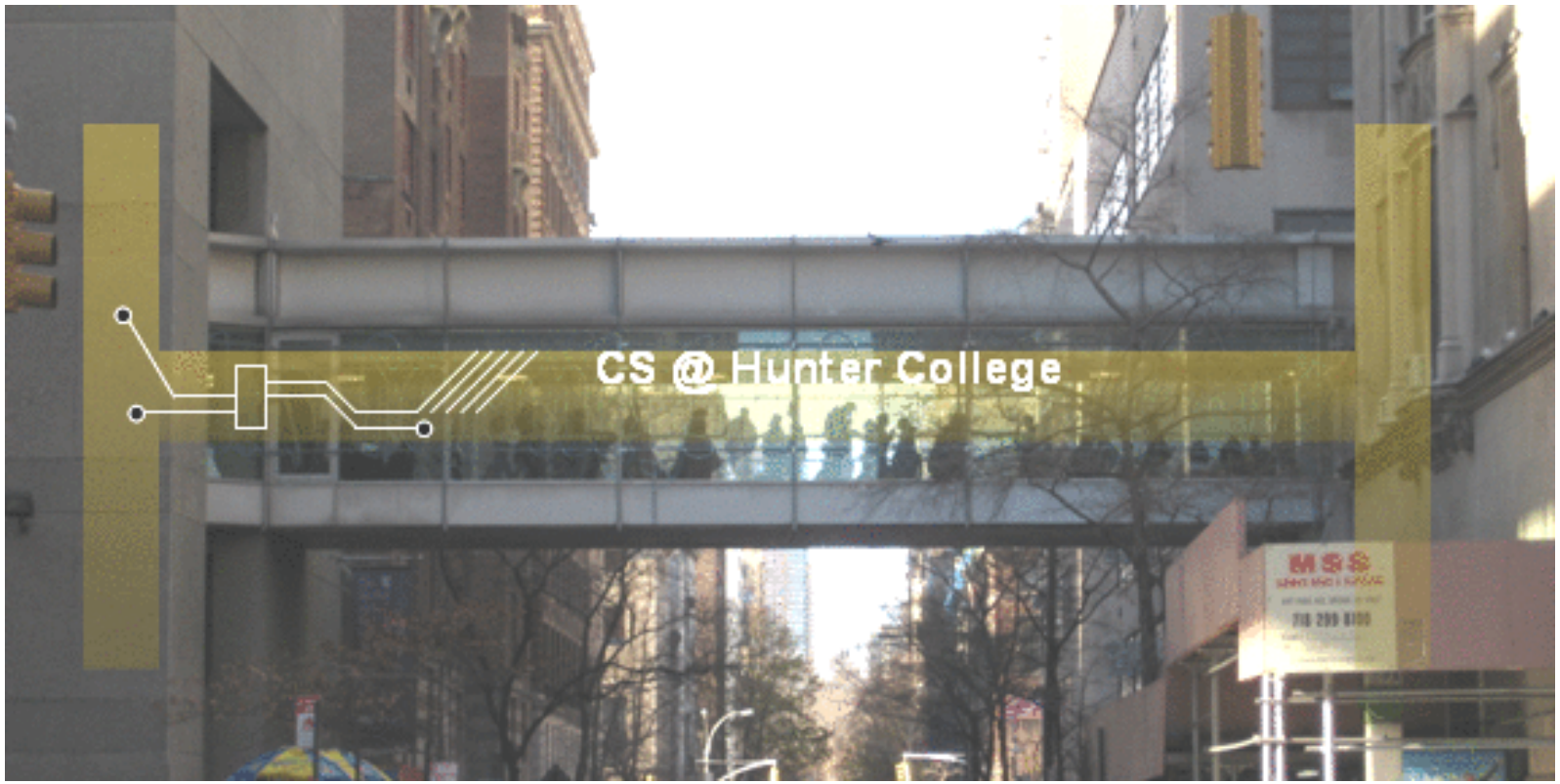
- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 46-50)**
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5pm

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 46-50)**
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10:15am on Tuesday)

Lecture Slips & Writing Boards



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