

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Today's Topics



- Recap: folium & koalas
- Indefinite Loops
- Searching Data
- Random Numbers

folium

Folium



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

Write code. \rightarrow *Run program.* \rightarrow *Open .html in browser.*

From Last Time: 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')
```

From Last Time: folium example

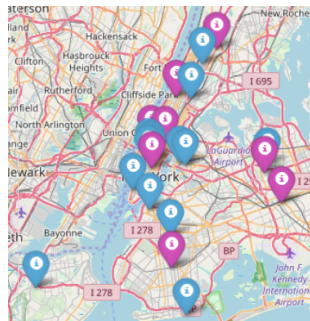
What does this code do?

```
import folium
import pandas as pd

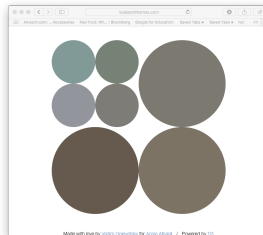
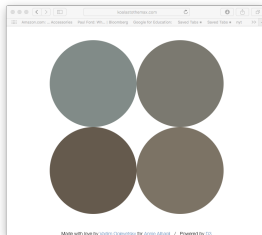
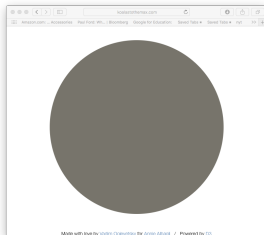
cuny = pd.read_csv('cunyLocations.csv')
mapCUNY = folium.Map(location=[40.75, -74.125])

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')
```

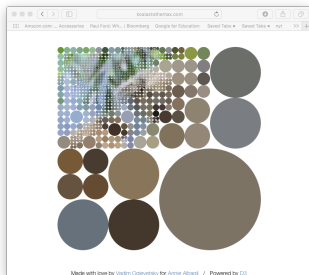


From Last Time: koalas



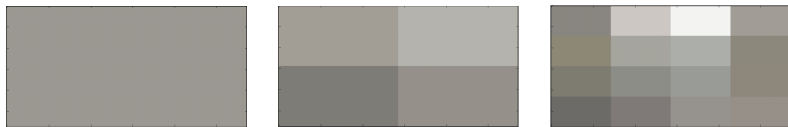
<http://koalastothemax.com>

From Last Time: koalas



- Top-down design puzzle:
 - ▶ What does koalastomax do?
 - ▶ What does each circle represent?
- Write a high-level design for it.
- Translate into a `main()` with function calls.

From Last Time: koalas



- Top-down design puzzle:
 - ▶ What does `koalastomax` do?
 - ▶ What does each circle represent?
- Write a high-level design for it.
- Translate into a `main()` with function calls.

From Last Time: koalas



```
69 def main():
70     inFile = input('Enter image file name: ')
71     img = plt.imread(inFile)
72
73     #Divides the image in 1/2, 1/4, 1/8, ... 1/2^8, and displays each:
74     for i in range(8):
75         img2 = img.copy()    #Make a copy to average
76         quarter(img2,i)      #Split in half i times, and average regions
77
78         plt.imshow(img2)     #Load our new image into pyplot
79         plt.show()          #Show the image (waits until closed to continue)
80
81     #Shows the original image:
82     plt.imshow(img)         #Load image into pyplot
83     plt.show()              #Show the image (waits until closed to continue)
84
85
```

- The `main()` is written for you.
- Only fill in two functions: `average()` and `setRegion()`.

From Last Time: koalas

Process:



Get template
from github



Fill in missing
functions



Test locally
idle3/python3



Submit to
Gradescope

In Pairs or Triples:

Predict what the code will do:

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))

print('The distance entered is', dist)
```

#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)
```

Python Tutor

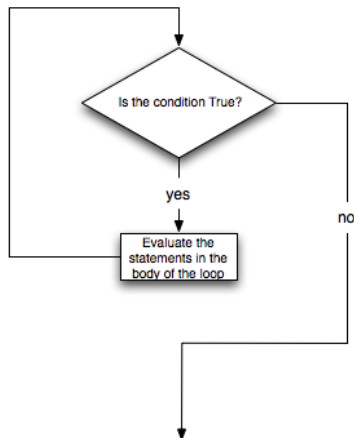
```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

```
#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)
```

(Demo with pythonTutor)

Indefinite Loops

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```



Indefinite Loops

```
dist = int(input('Enter distance: '))
while dist < 0:
    print('Distances cannot be negative.')
    dist = int(input('Enter distance: '))
print('The distance entered is', dist)
```

```
#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.

In Pairs or Triples:



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

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

Design Question: Find first alphabetically



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

Design Question: Find first alphabetically



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



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

In Pairs or Triples:

- *Predict what the code will do:*

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

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

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

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)
```

```
def search(nums, locate):
    found = False
    i = 0
    while not found and i < len(nums):
        print(nums[i])
        if locate == nums[i]:
            found = True
        else:
            i = i+1
    return(found)

nums= [1,4,10,6,5,42,9,8,12]
if search(nums,6):
    print('Found it! 6 is in the list!')
else:
    print('Did not find 6 in the list.')
```

(Demo with pythonTutor)

In Pairs or Triples:

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

Coding

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

```
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)
```

Python's random package

- Python has a built-in package for generating pseudo-random numbers.

- To use:

```
import random
```

- Useful command to generate whole numbers:

```
random.randrange(start, stop, step)
```

which gives a number chosen randomly from the specified range.

- Useful command to generate real numbers:

```
random.random()
```

which gives a number chosen (uniformly) at random from $[0.0, 1.0)$.

- Very useful for simulations, games, and testing.

```
import turtle
import random

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

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

Trinket

```
import turtle
import random

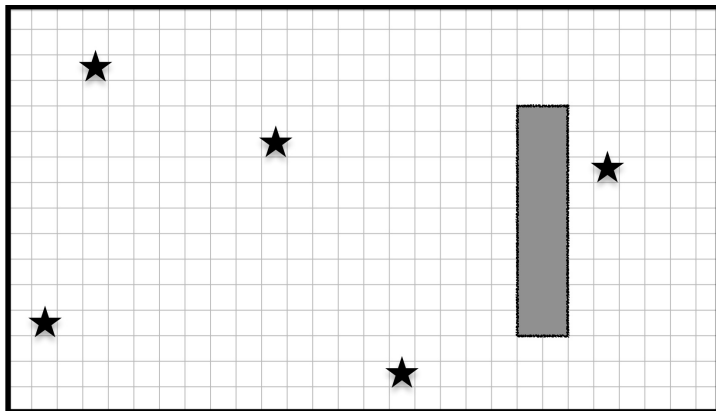
trex = turtle.Turtle()
trex.speed(10)

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

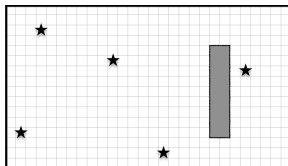
(Demo turtle
random walk)

Design Challenge

Collect all five stars (locations randomly generated):



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.
- If only turned left when you ran into a wall, what would happen?

Recap: Indefinite Loops & Random Numbers



- On lecture slip, write down a topic you wish we had spent more time (and why).
- Indefinite (while) loops allow you to repeat a block of code as long as a condition holds.
- Very useful for checking user input for correctness.
- Python's built-in random package has useful methods for generating random whole numbers and real numbers.
- To use, must include: `import random`.