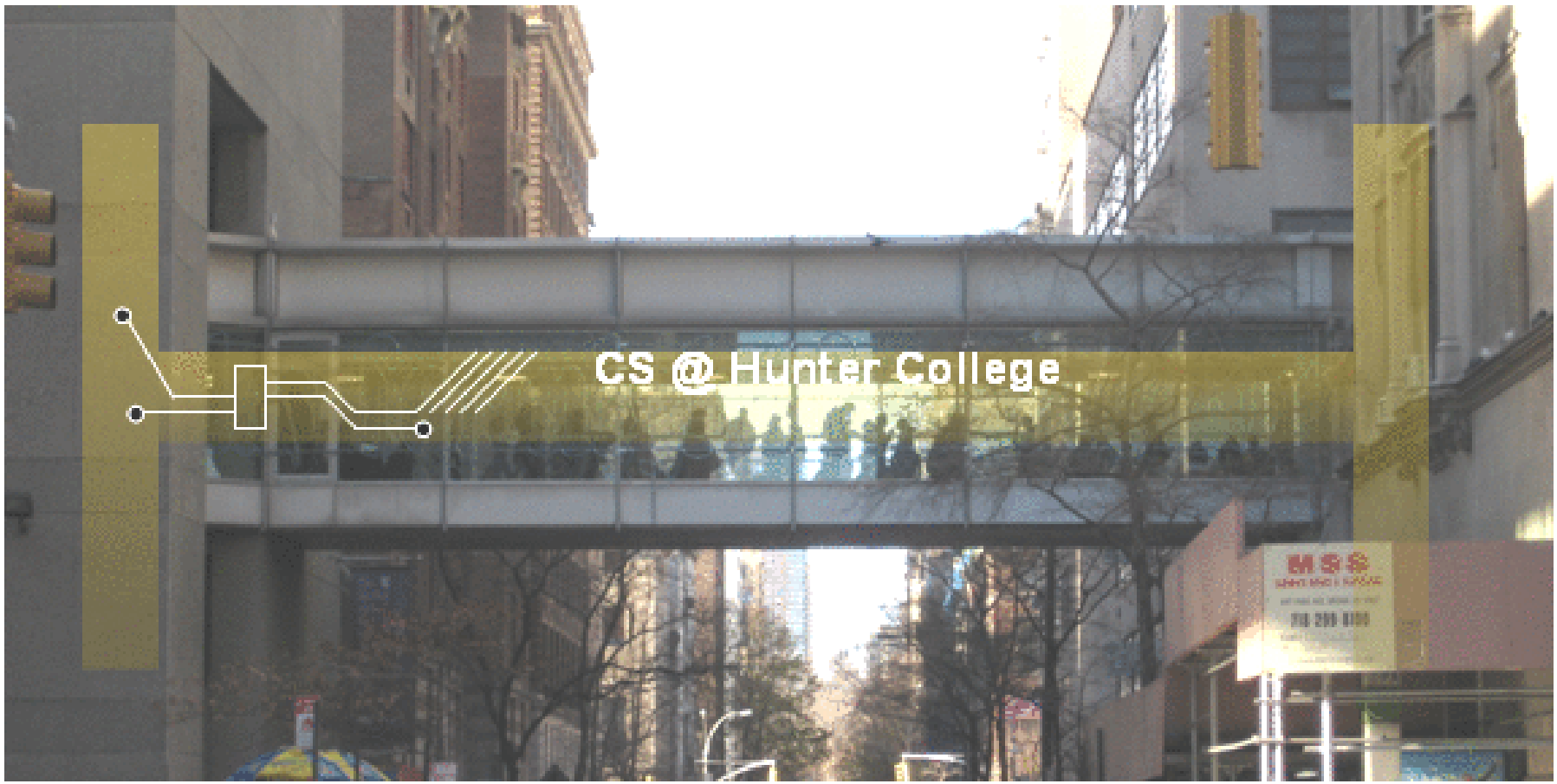
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Today's Topics

- More on Functions
- Recap: Open Data
- Top Down Design
- Github
- Design Challenge:

Today's Topics

- **More on Functions**
- Recap: Open Data
- Top Down Design
- Github
- Design Challenge:

Input Parameters & Return Values

- Functions can have **input parameters**.

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parentheses, both in the function de nition, and in the function call (invocation).

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parentheses, both in the function de nition, and in the function call (invocation).
- The \placeholders" in the function de nition: **formal parameters**.

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parentheses, both in the function de nition, and in the function call (invocation).
- The \placeholders" in the function de nition: **formal parameters**.
- The ones in the function call: **actual parameters**

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

- Functions can have **input parameters**.
- Surrounded by parentheses, both in the function de nition, and in the function call (invocation).
- The \placeholders" in the function de nition: **formal parameters**.
- The ones in the function call: **actual parameters**
- Functions can also **return values** to where it was called.

Input Parameters & Return Values

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- Functions can have **input parameters**.
- Surrounded by parenthesis, both in the function de nition, and in the function call (invocation).
- The \placeholders" in the function de nition: **formal parameters**.
- The ones in the function call: **actual parameters**.
- Functions can also **return values** to where it was called.

Challenge:

- What are the formal parameters? What is returned?

```
def enigma1(x,y,z):
    if x == len(y):
        return(z)
    elif x < len(y):
        return(y[0:x])
    else:
        s = cont1(z)
        return(s+y)
```

```
def cont1(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(a) `enigma1(7,"caramel","dulce de leche")`

Return: _____

(b) `enigma1(3,"cupcake","vanilla")`

Return: _____

(c) `enigma1(10,"pie","nomel")`

Return: _____

Python Tutor

```
def enigma1(x,y,z):
    if x == len(y):
        return(z)
    elif x < len(y):
        return(y[0:x])
    else:
        s = cont1(z)
        return(s+y)
```

(a) `enigma1(7,"caramel","dulce de leche")`

Return:

(b) `enigma1(3,"cupcake","vanilla")`

Return:

(c) `enigma1(10,"pie","nomel")`

Return:

```
def cont1(st):
    r = ""
    for i in range(len(st)-1,-1,-1):
        r = r + st[i]
    return(r)
```

(Demo with pythonTutor)

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.
- The copies are discarded when the function is done.

Input Parameters

```
def totalWithTax(food,tip):
    total = 0
    tax = 0.0875
    total = food + food * tax
    total = total + tip
    return(total)

lunch = float(input('Enter lunch total: '))
lTip = float(input('Enter lunch tip:' ))
lTotal = totalWithTax(lunch, lTip)
print('Lunch total is', lTotal)

dinner= float(input('Enter dinner total: '))
dTip = float(input('Enter dinner tip:' ))
dTotal = totalWithTax(dinner, dTip)
print('Dinner total is', dTotal)
```

Formal Parameters

Actual Parameters

- When called, the actual parameter values are copied to the formal parameters.
- All the commands inside the function are performed on the copies.
- The actual parameters do not change.
- The copies are discarded when the function is done.
- The time a variable exists is called its **scope**.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.
- The actual parameters do not change, but the inside elements might.

Input Parameters: What about Lists?

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

- When called, the actual parameter values are copied to the formal parameters.
- What is copied with a list?
- The address of the list, but not the individual elements.
- The actual parameters do not change, but the inside elements might.
- Easier to see with a demo.

Python Tutor

```
#Fall 2013 Final Exam, 5

def kuwae( inLst ):
    tot = 1
    for item in inLst:
        tot = tot * item
    return tot

def foo( inLst ):
    if ( inLst[-1] > inLst[0] ):
        return kuwae( inLst )
    else:
        return -1

foo( [2, 4, 6, 8] )

foo( [4002, 328, 457, 1] )
```

(Demo with pythonTutor)

Challenge:

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

- What are the formal parameters for the functions?
- What is the output of:

```
r = foo([1,2,3,4])
print("Return:  ", r)
```

- What is the output of:

```
r = foo([1024,512,256,128])
print("Return:  ", r)
```

Python Tutor

```
def bar(n):
    if n <= 8:
        return 1
    else:
        return 0

def foo(l):
    n = bar(l[-1])
    return l[n]
```

(Demo with pythonTutor)

Challenge:

Predict what the code will do:

```
#CSci 127 Teaching Staff
#Triangles two ways...
import turtle

def setUp(t, dist, col):
     t.penup()
     t.forward(dist)
     t.pendown()
     t.color(col)

def nestedTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
          nestedTriangle(t, side/2)

def fractalTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
               fractalTriangle(t, side/2)
```

```
def main():
     nessa = turtle.Turtle()
     setUp(nessa, 100, "violet")
     nestedTriangle(nessa, 160)

     frank = turtle.Turtle()
     setUp(frank, -100, "red")
     fractalTriangle(frank, 160)

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

IDLE

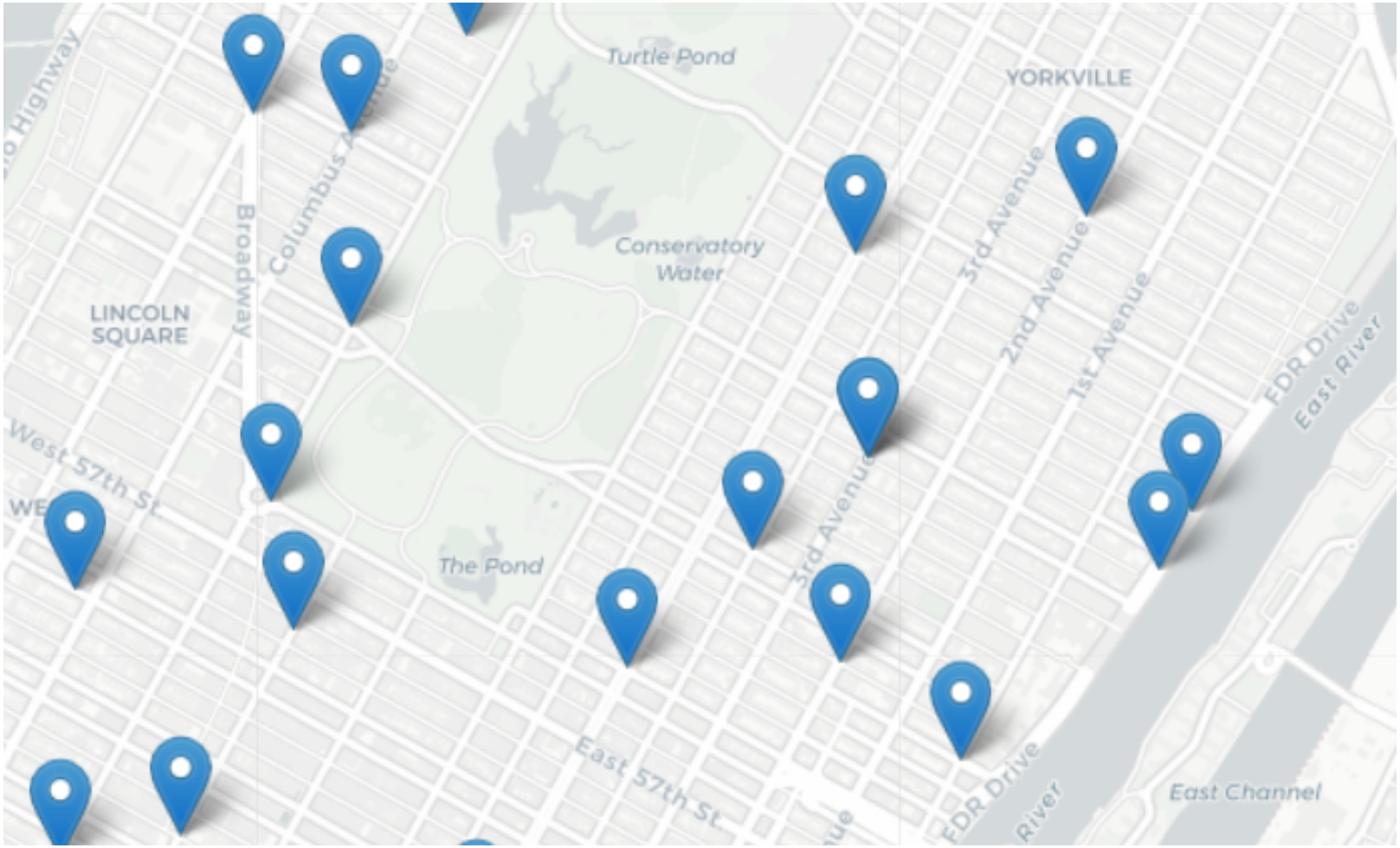
```
#CSci 127 Teaching Staff
#Triangles two ways...
import turtle

def setUp(t, dist, col):
     t.penup()
     t.forward(dist)
     t.pendown()
     t.color(col)

def nestedTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
          nestedTriangle(t, side/2)

def fractalTriangle(t, side):
     if side > 10:
          for i in range(3):
               t.forward(side)
               t.left(120)
               fractalTriangle(t, side/2)
```

(Demo with IDLE)

Today's Topics

- More on Functions
- **Recap: Open Data**
- Top Down Design
- Github
- Design Challenge:

OpenData Design Question



Design an algorithm that finds the closest collision.

(Sample NYC OpenData collision data file on back of lecture slip.)

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

How to approach this:

- Create a \To Do" list of what your program has to accomplish.
- Read through the problem, and break it into \To Do" items.
- Don't worry if you don't know how to do all the items you write down.
- Example:
  1. Find data set (great place to look: NYC OpenData).
  2. Ask user for current location.
  3. Open up the CSV le.
  4. Check distance to each to user's location.
  5. Print the location with the smallest distance.

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

How to approach this:

- Create a \To Do" list of what your program has to accomplish.
- Read through the problem, and break it into \To Do" items.
- Don't worry if you don't know how to do all the items you write down.
- Example:
  1. Find data set (great place to look: NYC OpenData).
  2. Ask user for current location.
  3. Open up the CSV le.
  4. Check distance to each to user's location.
  5. Print the location with the smallest distance.
- Let's use function names as placeholders for the ones we're unsure...

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

```
collisions = pd.read_csv(inF)
```

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

```
collisions = pd.read_csv(inF)
```

4. Check distance to each to user's location.

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

```
collisions = pd.read_csv(inF)
```

4. Check distance to each to user's location.

```
closestLat, closestLon = findClosest(collisions, lat, lon)
```

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

```
collisions = pd.read_csv(inF)
```

4. Check distance to each to user's location.

```
closestLat, closestLon = findClosest(collisions, lat, lon)
```

5. Print the location with the smallest distance.

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

```
collisions = pd.read_csv(inF)
```

4. Check distance to each to user's location.

```
closestLat, closestLon = findClosest(collisions, lat, lon)
```

5. Print the location with the smallest distance.

```
print("The closest is at lat:", lat, "and lon:", lon)
```

OpenData Design Question

Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

1. Find data set (great place to look: NYC OpenData).

```
import pandas as pd
inF = input('Enter CSV file name:')
```

2. Ask user for current location.

```
lat = float(input('Enter latitude:'))
lon = float(input('Enter longitude:'))
```

3. Open up the CSV file.

```
collisions = pd.read_csv(inF)
```

4. Check distance to each to user's location.

```
closestLat, closestLon = findClosest(collisions, lat, lon)
```

5. Print the location with the smallest distance.

```
print("The closest is at lat:", lat, "and lon:", lon)
```

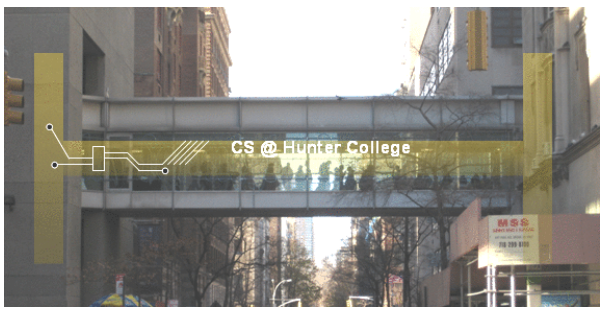
Today's Topics

- More on Functions
- Recap: Open Data
- **Top Down Design**
- Github
- Design Challenge:

Top-Down Design

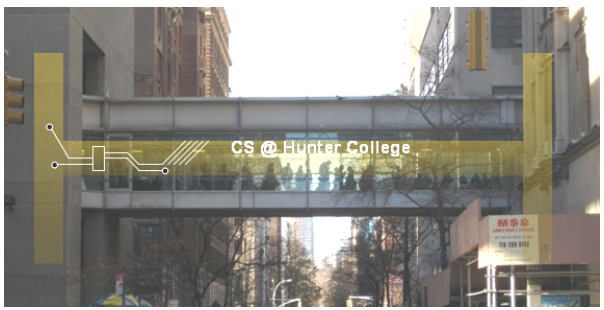
- The last example demonstrates **top-down design**: breaking into subproblems, and implementing each part separately.

Top-Down Design



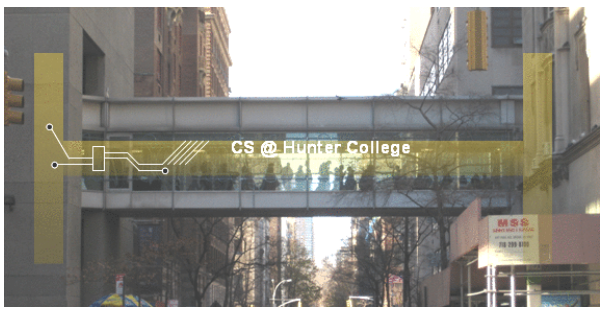
- The last example demonstrates **top-down design**: breaking into subproblems, and implementing each part separately.
  - Break the problem into tasks for a \To Do" list.

Top-Down Design



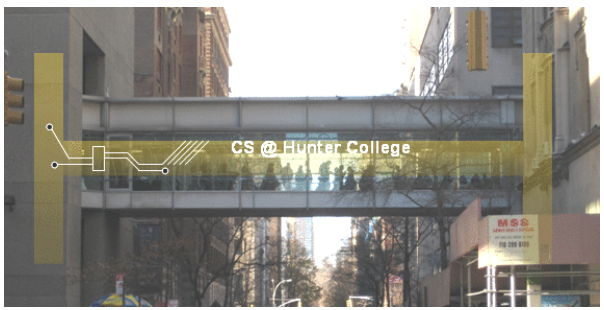
- The last example demonstrates **top-down design**: breaking into subproblems, and implementing each part separately.
  - Break the problem into tasks for a \To Do" list.
  - Translate list into function names & inputs/returns.

Top-Down Design



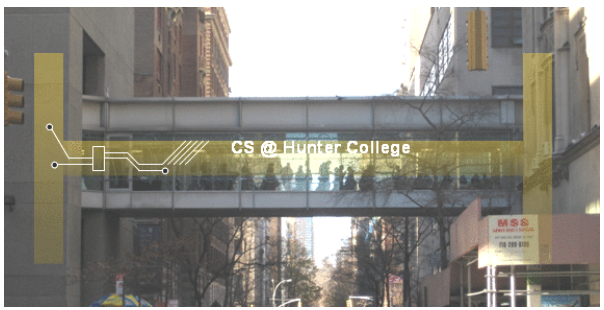
- The last example demonstrates **top-down design**: breaking into subproblems, and implementing each part separately.
  - Break the problem into tasks for a \To Do" list.
  - Translate list into function names & inputs/returns.
  - Implement the functions, one-by-one.

Top-Down Design



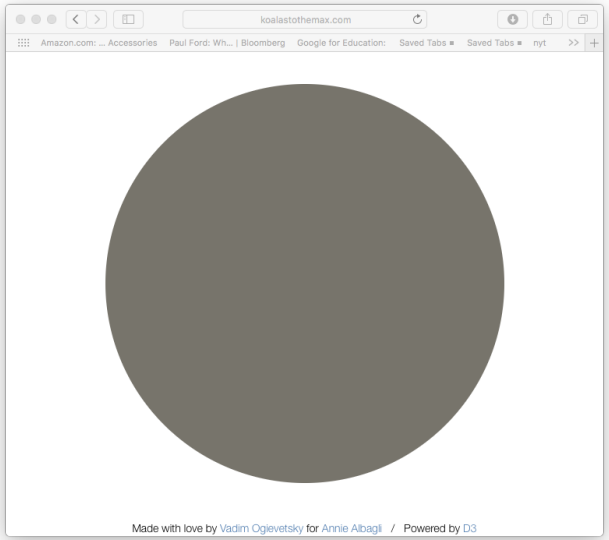
- The last example demonstrates **top-down design**: breaking into subproblems, and implementing each part separately.
  - Break the problem into tasks for a \To Do" list.
  - Translate list into function names & inputs/returns.
  - Implement the functions, one-by-one.
- Excellent approach since you can then test each part separately before adding it to a large program.

Top-Down Design



- The last example demonstrates **top-down design**: breaking into subproblems, and implementing each part separately.
  - Break the problem into tasks for a \To Do" list.
  - Translate list into function names & inputs/returns.
  - Implement the functions, one-by-one.
- Excellent approach since you can then test each part separately before adding it to a large program.
- Very common when working with a team: each has their own functions to implement and maintain.

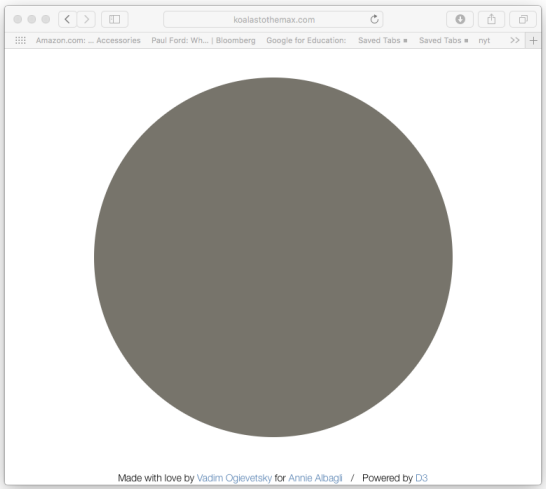
Challenge:



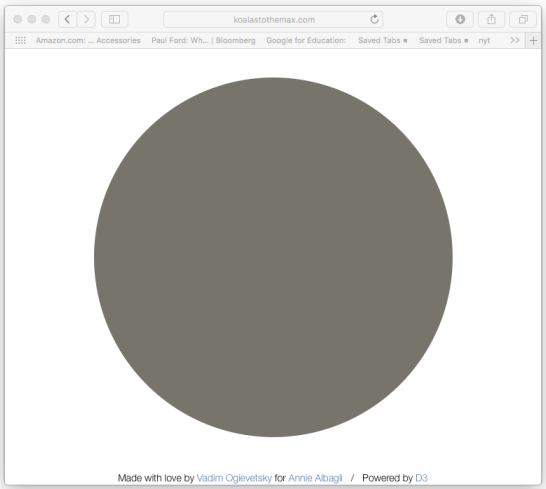
`http://koalastothemax.com`

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

Demo


koalastothemax.com
Made with love by Vadim Ogievetsky for Annie Albagli / Powered by D3

Demo


koalastothemax.com
Made with love by Vadim Ogievetsky for Annie Albagli / Powered by D3

Demo

Demo

Design: Koalas to the Max

Input: Image & mouse movements

Design: Koalas to the Max

Input: Image & mouse movements

Output: Completed image

Design: Koalas to the Max

Input: Image & mouse movements

Output: Completed image

Design:

Design: Koalas to the Max

Input: Image & mouse movements

Output: Completed image

Design:

- Every mouse movement,

Design: Koalas to the Max

Input: Image & mouse movements

Output: Completed image

Design:

- Every mouse movement,
  - Divide the region into 4 quarters.

Design: Koalas to the Max

Input: Image & mouse movements

Output: Completed image

Design:

- Every mouse movement,
- Divide the region into 4 quarters.
- Average the color of each quarter.

Design: Koalas to the Max

Input: Image & mouse movements

Output: Completed image

Design:

- Every mouse movement,
  - Divide the region into 4 quarters.
  - Average the color of each quarter.
  - Set each quarter to its average.

Averaging numpy arrays

Average each color channel of the image:

Averaging numpy arrays

Average each color channel of the image:

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
```

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
```

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
blueAve = np.average(region[:,:,2])
```

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
blueAve = np.average(region[:,:,2])
```

Set each pixel to the average value:

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
blueAve = np.average(region[:,:,2])
```

Set each pixel to the average value:

```
region[:,:,0] = redAve
```

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
blueAve = np.average(region[:,:,2])
```

Set each pixel to the average value:

```
region[:,:,0] = redAve
region[:,:,1] = greenAve
```

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
blueAve = np.average(region[:,:,2])
```

Set each pixel to the average value:

```
region[:,:,0] = redAve
region[:,:,1] = greenAve
region[:,:,2] = blueAve
```

Averaging numpy arrays

Average each color channel of the image:

```
redAve = np.average(region[:,:,0])
greenAve = np.average(region[:,:,1])
blueAve = np.average(region[:,:,2])
```

Set each pixel to the average value:

```
region[:,:,0] = redAve
region[:,:,1] = greenAve
region[:,:,2] = blueAve
```

Today's Topics

More on Functions

Recap: Open Data

Top Down Design

Github

Design Challenge:

Github

Used to collaborate on and share code, documents, etc.

Octocat

Github

Used to collaborate on and share code, documents, etc.

Supporting Open-Source Software: original source code is made freely available and may be redistributed and modi ed under the same licencse.

Octocat

Github

Used to collaborate on and share code, documents, etc.

Supporting Open-Source Software: original source code is made freely available and may be redistributed and modi ed under the same licencse.

More formally: git is a version control protocol for tracking changes and versions of documents.

Octocat

Github

Used to collaborate on and share code, documents, etc.

Supporting Open-Source Software: original source code is made freely available and may be redistributed and modi ed under the same licencse.

More formally: git is a version control protocol for tracking changes and versions of documents.

Github provides hosting for repositories (`repos') of code.

Octocat

Github

Octocat

Used to collaborate on and share code, documents, etc.

Supporting Open-Source Software: original source code is made freely available and may be redistributed and modi ed under the same licencse.

More formally: git is a version control protocol for tracking changes and versions of documents.

Github provides hosting for repositories (`repos') of code.

Also convenient place to host websites (i.e. huntercsci127.github.io).

Github

Used to collaborate on and share code, documents, etc.

Supporting Open-Source Software: original source code is made freely available and may be redistributed and modi ed under the same licencse.

More formally: git is a version control protocol for tracking changes and versions of documents.

Github provides hosting for repositories (`repos') of code.

Octocat

Also convenient place to host websites (i.e. huntercsci127.github.io).

In Lab6 you set up github accounts to copy (`clone') documents from the class repo. (More in future courses.)

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Find all current city job postings for internship positions.

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Input: CSV file from NYC OpenData.

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Input: CSV file from NYC OpenData.

Output: A list of internships offered by the city.

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Input: CSV le from NYC OpenData.

Output: A list of internships o ered by the city.

Process:

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Input: CSV le from NYC OpenData.

Output: A list of internships o ered by the city.

Process:

1. Open the le.

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Input: CSV le from NYC OpenData.

Output: A list of internships o ered by the city.

Process:

1. Open the le.
2. Select the rows that have \intern" in the business title.

Design Challenge

(data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t)

Input: CSV le from NYC OpenData.

Output: A list of internships o ered by the city.

Process:

1. Open the le.
2. Select the rows that have \intern" in the business title.
3. Print out those rows.

Recap

Functions are a way to break code into pieces, that can be easily reused.

Recap

Functions are a way to break code into pieces, that can be easily reused.

Functions can have input parameters that bring information into the function,

Recap

Functions are a way to break code into pieces, that can be easily reused.

Functions can have input parameters that bring information into the function,

And return values that send information back.

Recap

- Functions are a way to break code into pieces, that can be easily reused.
- Functions can have **input parameters** that bring information into the function,
- And **return values** that send information back.
- Top-down design: breaking into subproblems, and implementing each part separately.

Recap

- Functions are a way to break code into pieces, that can be easily reused.
- Functions can have **input parameters** that bring information into the function,
- And **return values** that send information back.
- Top-down design: breaking into subproblems, and implementing each part separately.
- Excellent approach: can then test each part separately before adding it to a large program.

Recap

- Functions are a way to break code into pieces, that can be easily reused.
- Functions can have **input parameters** that bring information into the function,
- And **return values** that send information back.
- Top-down design: breaking into subproblems, and implementing each part separately.
- Excellent approach: can then test each part separately before adding it to a large program.
- Github provides a platform for sharing work that allows collaboration (and version control).

Recap

- Functions are a way to break code into pieces, that can be easily reused.
- Functions can have **input parameters** that bring information into the function,
- And **return values** that send information back.
- Top-down design: breaking into subproblems, and implementing each part separately.
- Excellent approach: can then test each part separately before adding it to a large program.
- Github provides a platform for sharing work that allows collaboration (and version control).
- Log in to Gradescope to take Quiz 8