

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Today's Topics



- Recap: Slicing & Images
- Introduction to Functions
- NYC Open Data
- Design Challenge

Today's Topics



- **Recap: Slicing & Images**
- Introduction to Functions
- NYC Open Data
- Design Challenge

Challenge: Cropping Images

Crop an image to select the top quarter (upper left corner)



Challenge: Cropping Images

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```

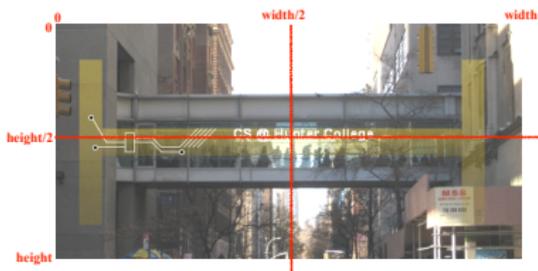
Challenge: Cropping Images

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



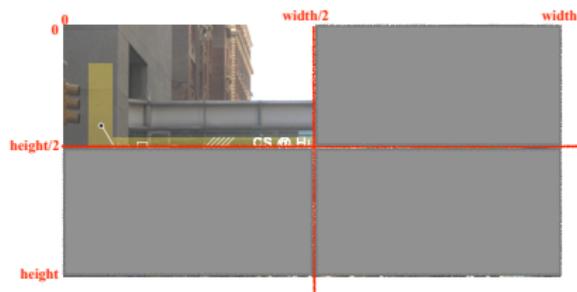
Challenge: Cropping Images

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



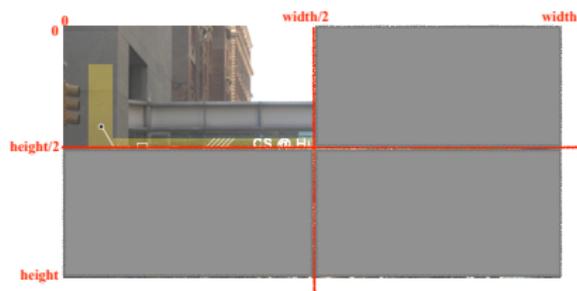
Challenge: Cropping Images

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



Challenge: Cropping Images

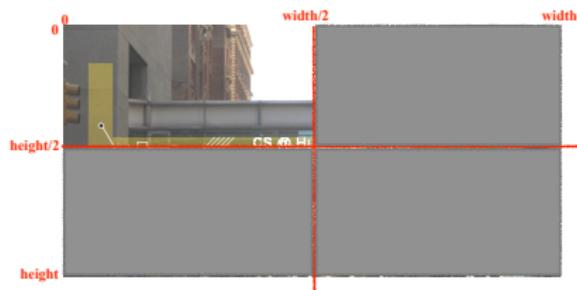
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- How would you select the lower left corner?

Challenge: Cropping Images

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```

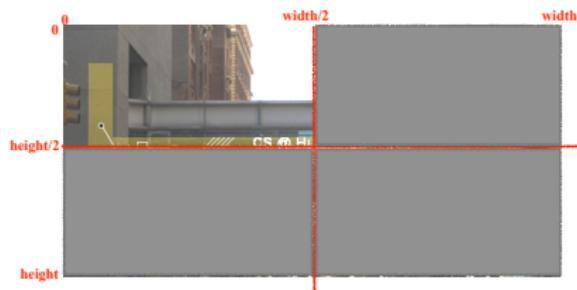


- How would you select the lower left corner?

```
img2 = img[height//2:, :width//2]
```

Challenge: Cropping Images

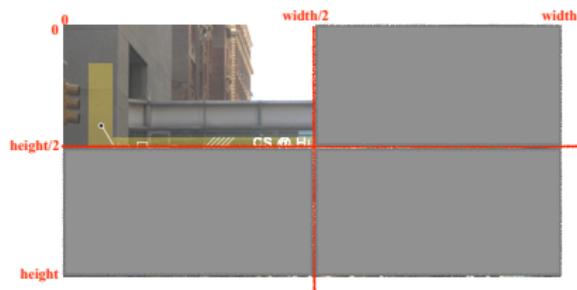
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- How would you select the lower left corner?
`img2 = img[height//2:, :width//2]`
- How would you select the upper right corner?

Challenge: Cropping Images

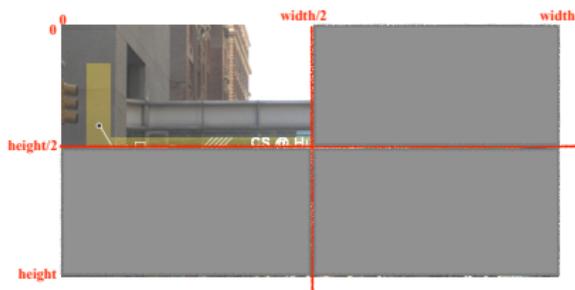
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- How would you select the lower left corner?
`img2 = img[height//2:, :width//2]`
- How would you select the upper right corner?
`img2 = img[:height//2, width//2:]`

Challenge: Cropping Images

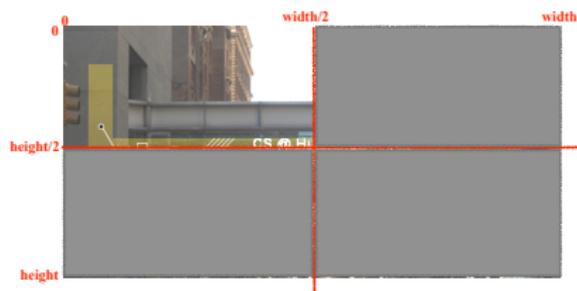
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- How would you select the lower left corner?
`img2 = img[height//2:, :width//2]`
- How would you select the upper right corner?
`img2 = img[:height//2, width//2:]`
- How would you select the lower right corner?

Challenge: Cropping Images

```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- How would you select the lower left corner?
`img2 = img[height//2:, :width//2]`
- How would you select the upper right corner?
`img2 = img[:height//2, width//2:]`
- How would you select the lower right corner?
`img2 = img[height//2:, width//2:]`

Today's Topics



- Recap: Slicing & Images
- **Introduction to Functions**
- NYC Open Data
- Design Challenge

Functions

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

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.

Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`

Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`

Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- Many languages require that all code must be organized with functions.
- The opening function is often called `main()`
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

“Hello, World!” with Functions

```
#Name:  your name here  
#Date:  October 2017  
#This program, uses functions,  
#      says hello to the world!
```

```
def main():  
    print("Hello, World!")
```

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

Python Tutor

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

(Demo with pythonTutor)

Challenge Problem:

Predict what the code will do:

```
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)
```

Python Tutor

```
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)
```

(Demo with pythonTutor)

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

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

```
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 definition, and in the function call (invocation).
- The “placeholders” in the function definition: **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 definition, and in the function call (invocation).
- The “placeholders” in the function definition: **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 definition, and in the function call (invocation).
- The “placeholders” in the function definition: **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 parentheses, both in the function definition, and in the function call (invocation).
- The “placeholders” in the function definition: **formal parameters**.
- The ones in the function call: **actual parameters**.
- Functions can also **return values** to where it was called.

Challenge Problem:

Circle the actual parameters and underline the formal parameters:

```
def prob4():
    verse = "jam tomorrow and jam yesterday,"
    print("The rule is,")
    c = mystery(verse)
    w = enigma(verse,c)
    print(c,w)
def mystery(v):
    print(v)
    c = v.count("jam")
    return(c)
def enigma(v,c):
    print("but never", v[-1])
    for i in range(c):
        print("jam")
    return("day.")
prob4()
```

Challenge Problem:

Circle the actual parameters and underline the formal parameters:

```
def prob4():  
    verse = "jam tomorrow and jam yesterday,"  
    print("The rule is,")  
    c = mystery(verse)  
    w = enigma(verse, c)  
    print(c, w)  
def mystery(v):  
    print(v)  
    c = v.count("jam")  
    return(c)  
def enigma(v, c):  
    print("but never", v[-1])  
    for i in range(c):  
        print("jam")  
    return("day.")  
prob4()
```

Actual Parameters

Formal Parameters

Challenge Problem:

Predict what the code will do:

```
def prob4():
    verse = "jam tomorrow and jam yesterday,"
    print("The rule is,")
    c = mystery(verse)
    w = enigma(verse,c)
    print(c,w)
def mystery(v):
    print(v)
    c = v.count("jam")
    return(c)
def enigma(v,c):
    print("but never", v[-1])
    for i in range(c):
        print("jam")
    return("day.")
prob4()
```

Python Tutor

```
def prob4():
    verse = "jam tomorrow and jam yesterday."
    print("The rule is.")
    c = mystery(verse)
    w = enigma(verse,c)
    print(c,w)
def mystery(v):
    print(v)
    c = v.count("jam")
    return(c)
def enigma(v,c):
    print("but never", v[-1])
    for i in range(c):
        print("jam")
    return("day.")
prob4()
```

(Demo with pythonTutor)

Challenge Problem:

Predict what the code will do:

```
#Greet loop example
```

```
def greetLoop(person):  
    print("Greetings")  
    for i in range(5):  
        print("Hello", person)
```

```
greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle
```

```
def happy():  
    print("Happy Birthday to you!")
```

```
def sing(P):  
    happy()  
    happy()  
    print("Happy Birthday dear " + P + "!")  
    happy()
```

```
sing("Fred")  
sing("Thomas")  
sing("Hunter")
```

Python Tutor

```
#Greet loop example

def greetLoop(person):
    print("Greetings")
    for i in range(5):
        print("Hello", person)

greetLoop("Thomas")
```

```
# From "Teaching with Python" by John Zelle

def happy():
    print("Happy Birthday to you!")

def sing(P):
    happy()
    happy()
    print("Happy Birthday dear " + P + "!")
    happy()

sing("Fred")
sing("Thomas")
sing("Hunter")
```

(Demo with pythonTutor)

Challenge Problem:

Fill in the missing code:

```
def monthString(monthNum):
    """
    Takes as input a number, monthNum, and
    returns the corresponding month name as a string.
    Example: monthString(1) returns "January".
    Assumes that input is an integer ranging from 1 to 12
    """

    monthString = ""

    #####
    ### FILL IN YOUR CODE HERE      ###
    ### Other than your name above, ###
    ### this is the only section    ###
    ### you change in this program. ###
    #####

    return(monthString)

def main():
    n = int(input('Enter the number of the month: '))
    mString = monthString(n)
    print('The month is', mString)
```

IDLE

```
def monthString(monthNum):
    """
    Takes as input a number, monthNum, and
    returns the corresponding month name as a string.
    Example: monthString(1) returns "January".
    Assumes that input is an integer ranging from 1 to 12
    """

    monthString = ""

    #####
    ## FILL IN YOUR CODE HERE ##
    ## Other than your name above, ##
    ## this is the only section ##
    ## you change in this program. ##
    #####

    return(monthString)

def main():
    n = int(input('Enter the number of the month: '))
    nString = monthString(n)
    print('The month is', nString)
```

(Demo with IDLE)

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 modified.



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

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 modified.
- More formally: `git` is a version control protocol for tracking changes and versions of documents.
- Github provides hosting for repositories (**'repos'**) of code.

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 modified.
- 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



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

Recap: Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

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

Recap: Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Recap: Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`

Recap: Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Recap: Functions

```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#    says hello to the world!

def main():
    print("Hello, World!")

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

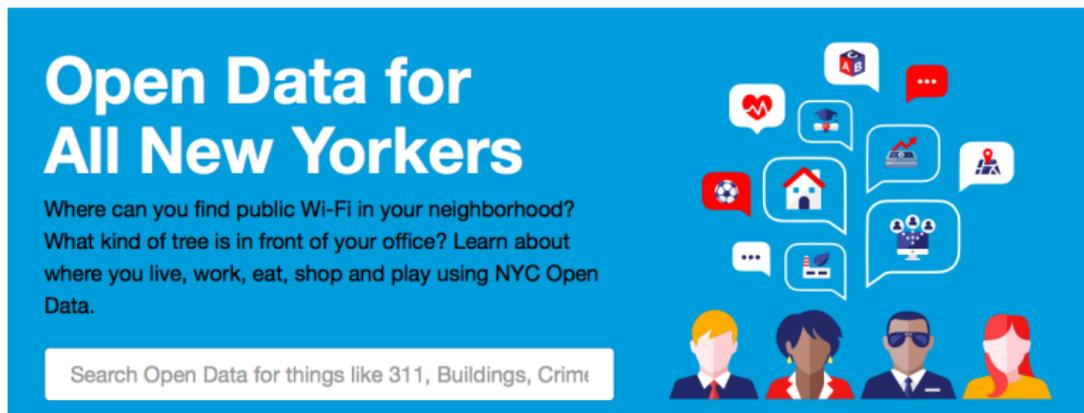
- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis: Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

Today's Topics



- Recap: Slicing & Images
- Introduction to Functions
- **NYC Open Data**
- Design Challenge

Accessing Structured Data: NYC Open Data



Open Data for All New Yorkers

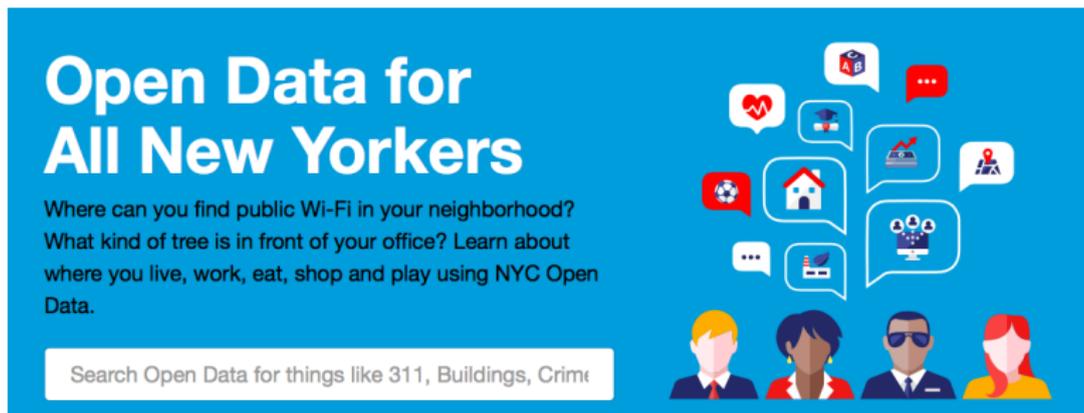
Where can you find public Wi-Fi in your neighborhood?
What kind of tree is in front of your office? Learn about where you live, work, eat, shop and play using NYC Open Data.

Search Open Data for things like 311, Buildings, Crime

The banner features a blue background with white text and icons. On the right, there are several speech bubbles containing icons for a heart, a graduation cap, a house, a soccer ball, a bar chart, a location pin, a Wi-Fi symbol, and a group of people. At the bottom right, there are four stylized human figures representing diverse individuals.

- Freely available source of data.

Accessing Structured Data: NYC Open Data



Open Data for All New Yorkers

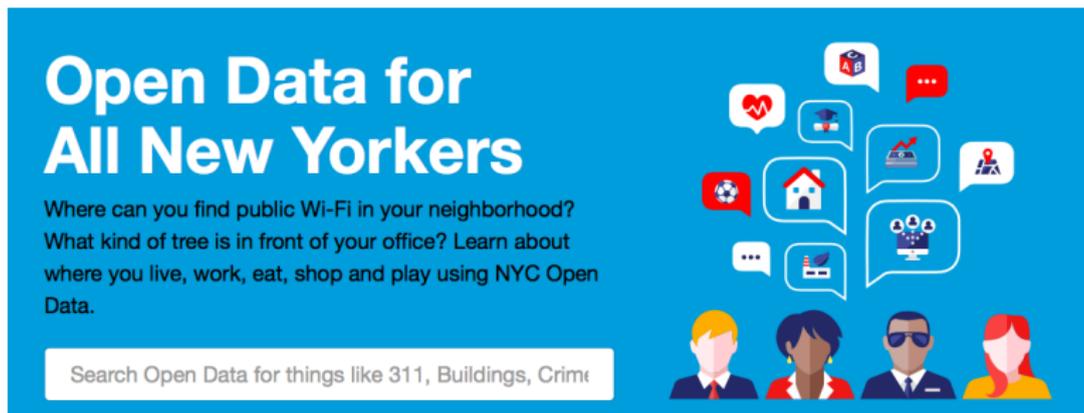
Where can you find public Wi-Fi in your neighborhood?
What kind of tree is in front of your office? Learn about where you live, work, eat, shop and play using NYC Open Data.

Search Open Data for things like 311, Buildings, Crim

The banner features a blue background with white text. On the right side, there are several white speech bubbles containing icons for a heart, a graduation cap, a house, a soccer ball, a bar chart, a location pin, a Wi-Fi symbol, and a group of people. Below the speech bubbles are four stylized human figures representing diverse individuals.

- Freely available source of data.
- Maintained by the NYC data analytics team.

Accessing Structured Data: NYC Open Data

A promotional graphic for NYC Open Data. It features a blue background with white text and icons. The text reads: "Open Data for All New Yorkers", "Where can you find public Wi-Fi in your neighborhood? What kind of tree is in front of your office? Learn about where you live, work, eat, shop and play using NYC Open Data.", and "Search Open Data for things like 311, Buildings, Crim". There are several icons representing different data categories: a heart with a pulse line, a graduation cap, a house, a soccer ball, a factory, a person with a location pin, a person with a magnifying glass, a person with a gear, and a person with a speech bubble. At the bottom, there are four stylized human figures representing diverse people.

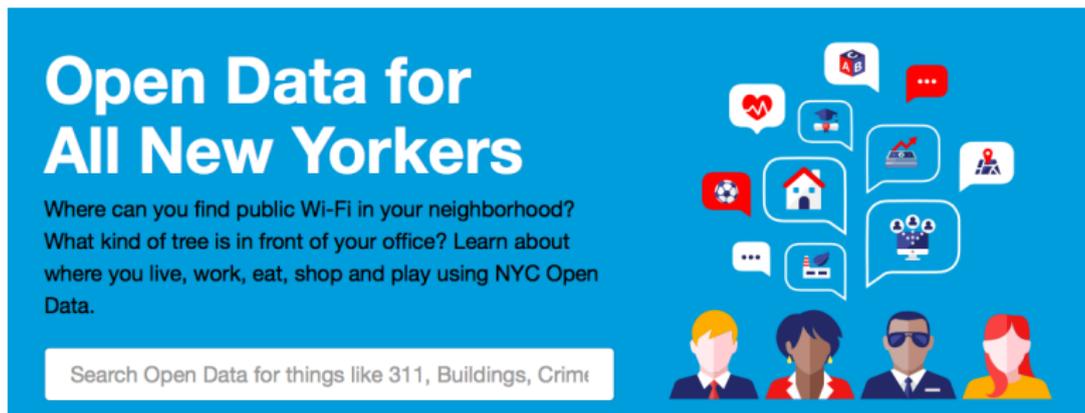
Open Data for All New Yorkers

Where can you find public Wi-Fi in your neighborhood? What kind of tree is in front of your office? Learn about where you live, work, eat, shop and play using NYC Open Data.

Search Open Data for things like 311, Buildings, Crim

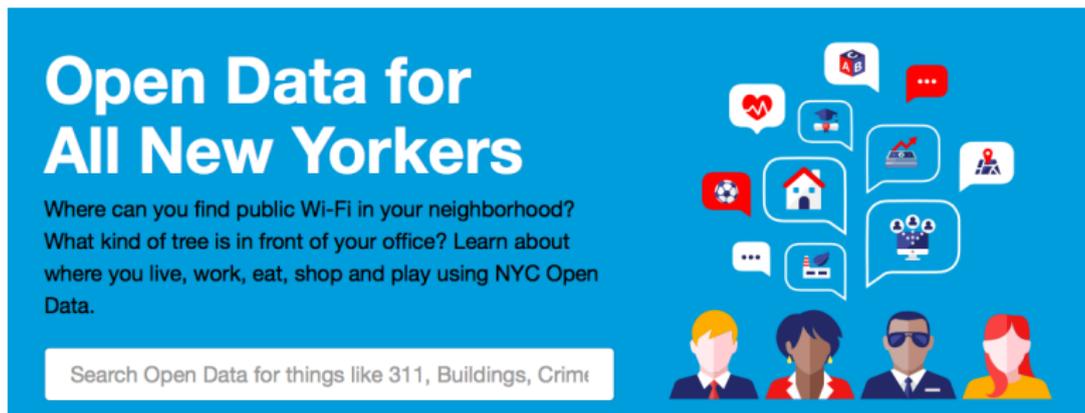
- Freely available source of data.
- Maintained by the NYC data analytics team.
- We will use several different ones for this class.

Accessing Structured Data: NYC Open Data

A promotional graphic for NYC Open Data. It features a blue background with the title "Open Data for All New Yorkers" in large white text. Below the title, there are two lines of white text: "Where can you find public Wi-Fi in your neighborhood?" and "What kind of tree is in front of your office? Learn about where you live, work, eat, shop and play using NYC Open Data." At the bottom left, there is a white search bar with the text "Search Open Data for things like 311, Buildings, Crim". On the right side, there are several colorful icons representing different data categories: a heart with a pulse line, a graduation cap, a house, a soccer ball, a bar chart with an upward arrow, a location pin, a group of people, and a person with a magnifying glass. At the bottom right, there are four stylized human figures representing diverse people.

- Freely available source of data.
- Maintained by the NYC data analytics team.
- We will use several different ones for this class.
- Will use `pandas`, `pyplot` & `folium` libraries to analyze, visualize and map the data.

Accessing Structured Data: NYC Open Data

A promotional graphic for NYC Open Data. It features a blue background with the title "Open Data for All New Yorkers" in large white text. Below the title is a paragraph of text: "Where can you find public Wi-Fi in your neighborhood? What kind of tree is in front of your office? Learn about where you live, work, eat, shop and play using NYC Open Data." At the bottom left is a white search bar with the text "Search Open Data for things like 311, Buildings, Crim". On the right side, there are several colorful icons representing different data categories: a heart with a pulse line, a graduation cap, a house, a soccer ball, a bar chart with an upward arrow, a location pin, a person with a magnifying glass, a person with a speech bubble, and a person with a gear. At the bottom right, there are four stylized human figures representing diverse people.

- Freely available source of data.
- Maintained by the NYC data analytics team.
- We will use several different ones for this class.
- Will use `pandas`, `pyplot` & `folium` libraries to analyze, visualize and map the data.
- Lab 7 covers accessing and downloading NYC OpenData datasets.

Example: OpenData Film Permits

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/mome/permits/when-permit-required.page>

EventID	EventType	StartDateTL...	EndDateTime	EnteredOn	EventAg...	ParkingHeld	Borou...
455063	Shooting Permit	12/06/2018 07:00...	12/06/2018 09:00...	12/05/2018 12:36...	Mayor's Offic...	STARR AVENUE b...	Queens
454967	Shooting Permit	12/06/2018 07:00...	12/06/2018 05:00...	12/04/2018 09:11...	Mayor's Offic...	EAGLE STREET be...	Brooklyn
454941	Shooting Permit	12/06/2018 07:00...	12/06/2018 07:00...	12/04/2018 05:44...	Mayor's Offic...	SOUTH OXFORD ...	Brooklyn
454920	Shooting Permit	12/06/2018 10:00...	12/06/2018 11:59...	12/04/2018 03:28...	Mayor's Offic...	13 AVENUE betw...	Queens
454914	Shooting Permit	12/06/2018 08:00...	12/06/2018 11:00...	12/04/2018 03:05...	Mayor's Offic...	ELDERT STREET b...	Brooklyn
454909	Shooting Permit	12/05/2018 08:00...	12/05/2018 06:00...	12/04/2018 02:45...	Mayor's Offic...	ELDERT STREET b...	Brooklyn
454905	Shooting Permit	12/06/2018 07:00...	12/06/2018 10:00...	12/04/2018 02:17...	Mayor's Offic...	35 STREET betwe...	Queens

Example: OpenData Film Permits

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/mome/permits/when-permit-required.page>

[More Views](#) [Filter](#) [Visualize](#) [Export](#) [Discuss](#) [Embed](#) [About](#)

EventID	EventType	StartDateT...	EndDateTime	EnteredOn	EventAg...	ParkingHeld	Borou...	Com...	Police...	Categ...	SubC...	Count...	ZipCo...
455063	Shooting Permit	12/06/2018 07:00...	12/06/2018 09:00...	12/05/2018 12:36...	Mayor's Offic...	STARR AVENUE b...	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 07:00...	12/06/2018 05:00...	12/04/2018 09:11...	Mayor's Offic...	EAGLE STREET be...	Brooklyn	1	94	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 07:00...	12/06/2018 07:00...	12/04/2018 05:44...	Mayor's Offic...	SOUTH OXFORD ...	Brooklyn	2, 6	76, 88	Still Photo...	Not Applic...	United Sta...	11217, 11...
454920	Shooting Permit	12/06/2018 10:00...	12/06/2018 11:59...	12/04/2018 03:28...	Mayor's Offic...	13 AVENUE betw...	Queens	1, 3, 7	109, 7, 90	Film	Feature	United Sta...	10002, 11...
454914	Shooting Permit	12/06/2018 08:00...	12/06/2018 11:00...	12/04/2018 03:05...	Mayor's Offic...	ELDERT STREET b...	Brooklyn	4, 5	104, 75, 83	Television	Episodic s...	United Sta...	11207, 11...
454909	Shooting Permit	12/05/2018 08:00...	12/05/2018 06:00...	12/04/2018 02:45...	Mayor's Offic...	ELDERT STREET b...	Brooklyn	4	83	Television	Episodic s...	United Sta...	11237
454905	Shooting Permit	12/06/2018 07:00...	12/06/2018 10:00...	12/04/2018 02:17...	Mayor's Offic...	35 STREET betwe...	Queens	1	114	Television	Cable-epis...	United Sta...	11101, 11...

- What's the most popular street for filming?

Example: OpenData Film Permits

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/mome/permits/when-permit-required.page>

[More Views](#) [Filter](#) [Visualize](#) [Export](#) [Discuss](#) [Embed](#) [About](#)

EventID	EventType	StartDateT...	EndDateTime	EnteredOn	EventAg...	ParkingHeld	Borou...	Com...	Police...	Categ...	SubC...	Count...	ZipCo...
455063	Shooting Permit	12/06/2018 07:00...	12/06/2018 09:00...	12/05/2018 12:36...	Mayor's Offic...	STARR AVENUE b...	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 07:00...	12/06/2018 05:00...	12/04/2018 09:11...	Mayor's Offic...	EAGLE STREET be...	Brooklyn	1	94	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 07:00...	12/06/2018 07:00...	12/04/2018 05:44...	Mayor's Offic...	SOUTH OXFORD ...	Brooklyn	2, 6	76, 88	Still Photo...	Not Applic...	United Sta...	11217, 11...
454920	Shooting Permit	12/06/2018 10:00...	12/06/2018 11:59...	12/04/2018 03:28...	Mayor's Offic...	13 AVENUE betw...	Queens	1, 3, 7	109, 7, 90	Film	Feature	United Sta...	10002, 11...
454914	Shooting Permit	12/06/2018 08:00...	12/06/2018 11:00...	12/04/2018 03:05...	Mayor's Offic...	ELDERT STREET b...	Brooklyn	4, 5	104, 75, 83	Television	Episodic s...	United Sta...	11207, 11...
454909	Shooting Permit	12/05/2018 08:00...	12/05/2018 06:00...	12/04/2018 02:45...	Mayor's Offic...	ELDERT STREET b...	Brooklyn	4	83	Television	Episodic s...	United Sta...	11237
454905	Shooting Permit	12/06/2018 07:00...	12/06/2018 10:00...	12/04/2018 02:17...	Mayor's Offic...	35 STREET betwe...	Queens	1	114	Television	Cable-epis...	United Sta...	11101, 11...

- What's the most popular street for filming?
- What's the most popular borough?

Example: OpenData Film Permits

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/mome/permits/when-permit-required.page>

EventID	EventType	StartDateT...	EndDateTime	EnteredOn	EventAg...	ParkingHeld	Borou...	Com...	Police...	Categ...	SubC...	Count...	ZipCo...
455063	Shooting Permit	12/06/2018 07:00...	12/06/2018 09:00...	12/05/2018 12:36...	Mayor's Offic...	STARR AVENUE b...	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 07:00...	12/06/2018 05:00...	12/04/2018 09:11...	Mayor's Offic...	EAGLE STREET be...	Brooklyn	1	94	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 07:00...	12/06/2018 07:00...	12/04/2018 05:44...	Mayor's Offic...	SOUTH OXFORD ...	Brooklyn	2, 6	76, 88	Still Photo...	Not Applic...	United Sta...	11217, 11...
454920	Shooting Permit	12/06/2018 10:00...	12/05/2018 11:59...	12/04/2018 03:28...	Mayor's Offic...	13 AVENUE betw...	Queens	1, 3, 7	109, 7, 90	Film	Feature	United Sta...	10002, 11...
454914	Shooting Permit	12/06/2018 08:00...	12/06/2018 11:00...	12/04/2018 03:05...	Mayor's Offic...	ELDERT STREET b...	Brooklyn	4, 5	104, 75, 83	Television	Episodic s...	United Sta...	11207, 11...
454909	Shooting Permit	12/05/2018 08:00...	12/05/2018 06:00...	12/04/2018 02:45...	Mayor's Offic...	ELDERT STREET b...	Brooklyn	4	83	Television	Episodic s...	United Sta...	11237
454905	Shooting Permit	12/06/2018 07:00...	12/06/2018 10:00...	12/04/2018 02:17...	Mayor's Offic...	35 STREET betwe...	Queens	1	114	Television	Cable-epis...	United Sta...	11101, 11...

- What's the most popular street for filming?
- What's the most popular borough?
- How many TV episodes were filmed?

Example: OpenData Film Permits

NYC OpenData

Home Data About Learn Alerts Contact Us Blog Q Sign In

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/nycopendata/when-permits-required.page>

Find in this Dataset

More Views Filter Visualize Export Discuss Embed About

EventID	Event Type	Start Date/Time	End Date/Time	Event Date	Event Agency	Parking/Block	Borough	Count	Permits	Category	Sub-Category	Event Status	Zip Code
45363	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/05/2018 12:35	Mayor's Office	STARBUCKS AVE b...	Queens	2	108	Television	Episode S...	United Sta...	11101
45467	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/06/2018 09:11	Mayor's Office	GALLS STREET b...	Brooklyn	1	84	Television	Episode S...	United Sta...	11222
45481	Shooting Permit	12/06/2018 07:00	12/06/2018 07:00	12/06/2018 05:44	Mayor's Office	SOUTH OXFORD ...	Brooklyn	2, 6	76, 88	Still Photo	Not Applic...	United Sta...	11217, 11...
45400	Shooting Permit	12/06/2018 13:00	12/06/2018 11:00	12/06/2018 03:20	Mayor's Office	13 AVENUE betw...	Queens	1, 3, 7	108, 7, 96	Film	Feature	United Sta...	10002, 11...
45494	Shooting Permit	12/06/2018 08:00	12/06/2018 11:00	12/06/2018 09:05	Mayor's Office	ELBERT STREET b...	Brooklyn	4, 6	104, 76, 89	Television	Episode S...	United Sta...	11207, 11...
45489	Shooting Permit	12/05/2018 08:00	12/05/2018 06:00	12/06/2018 02:45	Mayor's Office	ELBERT STREET b...	Brooklyn	4	83	Television	Episode S...	United Sta...	11227
45485	Shooting Permit	12/06/2018 07:00	12/06/2018 10:00	12/06/2018 02:17	Mayor's Office	35 STREET betw...	Queens	1	114	Television	Cable-epic...	United Sta...	11101, 11...

- Download the data as a CSV file and store on your computer.

Example: OpenData Film Permits

NYC OpenData

Home Data About Learn Alerts Contact Us Blog Q Sign In

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a sign, or a park. See <http://www1.nyc.gov/site/nycopendata/when-permits-required.page>

More Views Filter Visualize Export Discuss Embed About

EventID	EventType	StartDate	EndDate	EventDate	EventTag	ParkingInfo	Boro	Com.	Permis.	Comp.	SubC.	Exam.	ZipCo.			
45363	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/05/2018 12:35	Mayor's Offi...	STARBUCK AVENUE b...	Queens	2	108	Television	Episodic S...	United Sta...	11101			
45467	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/06/2018 09:11	Mayor's Offi...	GALLS STREET b...	Brooklyn	1	84	Television	Episodic S...	United Sta...	11222			
45491	Shooting Permit	12/06/2018 07:00	12/06/2018 07:00	12/06/2018 05:44	Mayor's Offi...	SOUTH OXFORD ...	Brooklyn	2	76	88	Self-Produc...	Not Applica...	United Sta...	11217, 11...		
45400	Shooting Permit	12/06/2018 13:00	12/06/2018 11:00	12/06/2018 03:20	Mayor's Offi...	13 AVENUE betwe...	Queens	1	17	108	7	96	Film	Feature	United Sta...	10002, 11...
45414	Shooting Permit	12/06/2018 08:00	12/06/2018 11:00	12/06/2018 03:05	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4	5	104	76	89	Television	Episodic S...	United Sta...	11207, 11...
45489	Shooting Permit	12/05/2018 08:00	12/05/2018 06:00	12/06/2018 02:45	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4	83	Television	Episodic S...	United Sta...	11227			
45485	Shooting Permit	12/06/2018 07:00	12/06/2018 10:00	12/06/2018 02:17	Mayor's Offi...	35 STREET betwe...	Queens	1	114	Television	Cable-epic...	United Sta...	11101, 11...			

- Download the data as a CSV file and store on your computer.
- Python program:

```
#CSci 127 Teaching Staff  
#March 2019  
#OpenData Film Permits
```

```
#Import pandas for reading and analyzing CSV data:  
import pandas as pd  
csvFile = "filmPermits.csv" #Name of the CSV file  
tickets = pd.read_csv(csvFile)#Read in the file to a dataframe
```

Example: OpenData Film Permits

NYC OpenData Home Data About Learn Alerts Contact Us Blog Q Sign

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/nycopen/permits/when-permits-required.page>

Find in this Dataset

More Views Filter Visualize Export Discuss Embed About

EventID	EventType	StartDateT...	EndDateTime	EnteredOn	EventAg...	ParkingInf...	Borne...	Com...	Perlic...	Camp...	SubC...	Exam...	ZipCo...
45363	Shooting Permit	12/05/2018 07:00...	12/05/2018 09:00...	12/05/2018 12:35...	Mayor's Offi...	STARBUCK AVENUE S...	Queens	2	108	Television	Episode S...	United Sta...	11101
45467	Shooting Permit	12/06/2018 07:00...	12/06/2018 09:00...	12/06/2018 09:11...	Mayor's Offi...	GALLE STREET bet...	Brooklyn	1	84	Television	Episode S...	United Sta...	11222
45481	Shooting Permit	12/06/2018 07:00...	12/06/2018 07:00...	12/06/2018 05:44...	Mayor's Offi...	SOUTH OXFORD ...	Brooklyn	2	76, 88	9/8 Photo...	Not Applic...	United Sta...	11217, 11...
45400	Shooting Permit	12/06/2018 13:00...	12/06/2018 11:00...	12/06/2018 03:20...	Mayor's Offi...	13 AVENUE betwe...	Queens	1, 2, 7	108, 7, 96	Film	Feature	United Sta...	10002, 11...
45494	Shooting Permit	12/06/2018 08:00...	12/06/2018 11:00...	12/06/2018 03:05...	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4, 5	104, 76, 83	Television	Episode S...	United Sta...	11207, 11...
45489	Shooting Permit	12/05/2018 08:00...	12/05/2018 06:00...	12/06/2018 02:45...	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4	83	Television	Episode S...	United Sta...	11227
45485	Shooting Permit	12/06/2018 07:00...	12/06/2018 10:00...	12/06/2018 02:17...	Mayor's Offi...	35 STREET betwe...	Queens	1	114	Television	Cable-epi...	United Sta...	11101, 11...

- Download the data as a CSV file and store on your computer.
- Python program:

```
#CSci 127 Teaching Staff  
#March 2019  
#OpenData Film Permits
```

```
#Import pandas for reading and analyzing CSV data:  
import pandas as pd  
csvFile = "filmPermits.csv" #Name of the CSV file  
tickets = pd.read_csv(csvFile)#Read in the file to a dataframe  
print(tickets) #Print out the dataframe
```

Example: OpenData Film Permits

NYC OpenData

Home Data About Learn Alerts Contact Us Blog Q Sign In

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/nycopen/permits/when-permits-required.page>

More Views | Filter | Visualize | Export | Discuss | Embed | About

EventID	EventType	StartDate	EndDate	EventDate	EventName	ParkingHeld	Block	Comm.	Permits	Comp.	SubC.	Event	ZipCode
45363	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/05/2018 12:35	Mayor's Offi...	STARBUCKS AVE...	Queens	2	108	Television	Episodic L...	United St...	11101
45467	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/06/2018 09:11	Mayor's Offi...	GALLIE STREET b...	Brooklyn	1	84	Television	Episodic L...	United St...	11222
45491	Shooting Permit	12/06/2018 07:00	12/06/2018 07:00	12/06/2018 05:44	Mayor's Offi...	SOUTH OXFORD ...	Brooklyn	2	76, 88	TV Photo...	Not Applica...	United St...	11217, 11...
45400	Shooting Permit	12/06/2018 13:00	12/06/2018 11:00	12/06/2018 03:20	Mayor's Offi...	13 AVENUE betw...	Queens	1, 2, 7	108, 7, 98	Film	Feature	United St...	10002, 11...
45414	Shooting Permit	12/06/2018 08:00	12/06/2018 11:00	12/06/2018 03:05	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4, 5	108, 76, 89	Television	Episodic L...	United St...	11207, 11...
45489	Shooting Permit	12/05/2018 08:00	12/05/2018 06:00	12/06/2018 02:45	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4	83	Television	Episodic L...	United St...	11227
45485	Shooting Permit	12/06/2018 07:00	12/06/2018 10:00	12/06/2018 02:17	Mayor's Offi...	35 STREET betwe...	Queens	1	114	Television	Cable-epic...	United St...	11101, 11...

- Download the data as a CSV file and store on your computer.
- Python program:

```
#CSci 127 Teaching Staff  
#March 2019  
#OpenData Film Permits
```

```
#Import pandas for reading and analyzing CSV data:  
import pandas as pd  
csvFile = "filmPermits.csv" #Name of the CSV file  
tickets = pd.read_csv(csvFile)#Read in the file to a dataframe  
print(tickets) #Print out the dataframe  
print(tickets["ParkingHeld"]) #Print out streets (multiple times)
```

Example: OpenData Film Permits

NYC OpenData Home Data About Learn Alerts Contact Us Blog Q Sign In

Film Permits
Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a sign, or a park. See <http://www1.nyc.gov/site/nycopen/permits/when-permits-required.page>

More Views Filter Visualize Export Discuss Embed About

EventID	EventType	StartDate	EndDate	EventDate	EventAg.	ParkingHeld	Block	Com.	Permis.	Categ.	SubC.	Exam.	ZipCo.
45363	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/05/2018 12:35	Mayor's Offi...	STARBUCKS AVENUE S...	Queens	2	108	Television	Episodic L...	United Sta...	11101
45467	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/06/2018 09:11	Mayor's Offi...	GALLIE STREET bet...	Brooklyn	1	84	Television	Episodic L...	United Sta...	11222
45491	Shooting Permit	12/06/2018 07:00	12/06/2018 07:00	12/06/2018 05:44	Mayor's Offi...	SOUTH OXFORD ...	Brooklyn	2	76, 88	Still Photo...	Not Applic...	United Sta...	11217, 11...
45400	Shooting Permit	12/06/2018 13:00	12/06/2018 11:00	12/06/2018 03:20	Mayor's Offi...	13 AVENUE betwe...	Queens	1, 3, 7	108, 7, 96	Film	Feature	United Sta...	10002, 11...
45414	Shooting Permit	12/06/2018 08:00	12/06/2018 11:00	12/06/2018 03:05	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4, 5	104, 76, 89	Television	Episodic L...	United Sta...	11207, 11...
45489	Shooting Permit	12/05/2018 08:00	12/05/2018 06:00	12/04/2018 02:45	Mayor's Offi...	ELBERT STREET b...	Brooklyn	4	83	Television	Episodic L...	United Sta...	11227
45485	Shooting Permit	12/06/2018 07:00	12/06/2018 10:00	12/06/2018 02:17	Mayor's Offi...	35 STREET betwe...	Queens	1	114	Television	Cable-epic...	United Sta...	11101, 11...

- Download the data as a CSV file and store on your computer.
- Python program:

```
#CSci 127 Teaching Staff  
#March 2019  
#OpenData Film Permits
```

```
#Import pandas for reading and analyzing CSV data:
```

```
import pandas as pd  
csvFile = "filmPermits.csv" #Name of the CSV file  
tickets = pd.read_csv(csvFile)#Read in the file to a dataframe  
print(tickets) #Print out the dataframe  
print(tickets["ParkingHeld"]) #Print out streets (multiple times)  
print(tickets["ParkingHeld"].value_counts()) #Print out streets & number of times used
```

Example: OpenData Film Permits

NYC OpenData Home Data About Learn Alerts Contact Us Blog

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a park, or a park. See <http://www1.nyc.gov/site/nycopendata/when-permits-required.page>

Find in this Dataset

EventID	EventType	StartDateTime	EndDateTime	EnteredAt	EventAg.	ParkingHeld	Borne.	Com.	Permis.	Comp.	SubC.	Exam.	ZipCo.
45363	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/05/2018 12:35	Mayor's Offi...	STARBUCK AVENUE S...	Queens	2	108	Television	Episode S...	United Sta...	11101
45467	Shooting Permit	12/06/2018 07:00	12/06/2018 09:00	12/04/2018 09:11	Mayor's Offi...	GALLE STREET bet...	Brooklyn	1	84	Television	Episode S...	United Sta...	11222
45481	Shooting Permit	12/06/2018 07:00	12/06/2018 07:00	12/04/2018 05:44	Mayor's Offi...	SOUTH OXFORD ...	Brooklyn	2	76, 88	Still Photo	Not Applic...	United Sta...	11217, 11...
45400	Shooting Permit	12/06/2018 13:00	12/06/2018 11:00	12/04/2018 03:20	Mayor's Offi...	13 AVENUE betwe...	Queens	1, 3, 7	108, 7, 96	Film	Feature	United Sta...	10002, 11...
45414	Shooting Permit	12/06/2018 08:00	12/06/2018 11:00	12/04/2018 03:05	Mayor's Offi...	ELDRST STREET b...	Brooklyn	4, 5	104, 76, 83	Television	Episode S...	United Sta...	11207, 11...
45489	Shooting Permit	12/05/2018 08:00	12/05/2018 06:00	12/04/2018 02:45	Mayor's Offi...	ELDRST STREET b...	Brooklyn	4	83	Television	Episode S...	United Sta...	11227
45485	Shooting Permit	12/06/2018 07:00	12/06/2018 10:00	12/04/2018 02:17	Mayor's Offi...	35 STREET betwe...	Queens	1	114	Television	Cable-epi...	United Sta...	11101, 11...

- Download the data as a CSV file and store on your computer.
- Python program:

```
#CSci 127 Teaching Staff
```

```
#March 2019
```

```
#OpenData Film Permits
```

```
#Import pandas for reading and analyzing CSV data:
```

```
import pandas as pd
```

```
csvFile = "filmPermits.csv" #Name of the CSV file
```

```
tickets = pd.read_csv(csvFile)#Read in the file to a dataframe
```

```
print(tickets) #Print out the dataframe
```

```
print(tickets["ParkingHeld"]) #Print out streets (multiple times)
```

```
print(tickets["ParkingHeld"].value_counts()) #Print out streets & number of times used
```

```
print(tickets["ParkingHeld"].value_counts()[:10]) #Print 10 most popular
```

Example: OpenData Film Permits

Film Permits

Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/mome/permits/when-permit-required.page>

Find in this Dataset

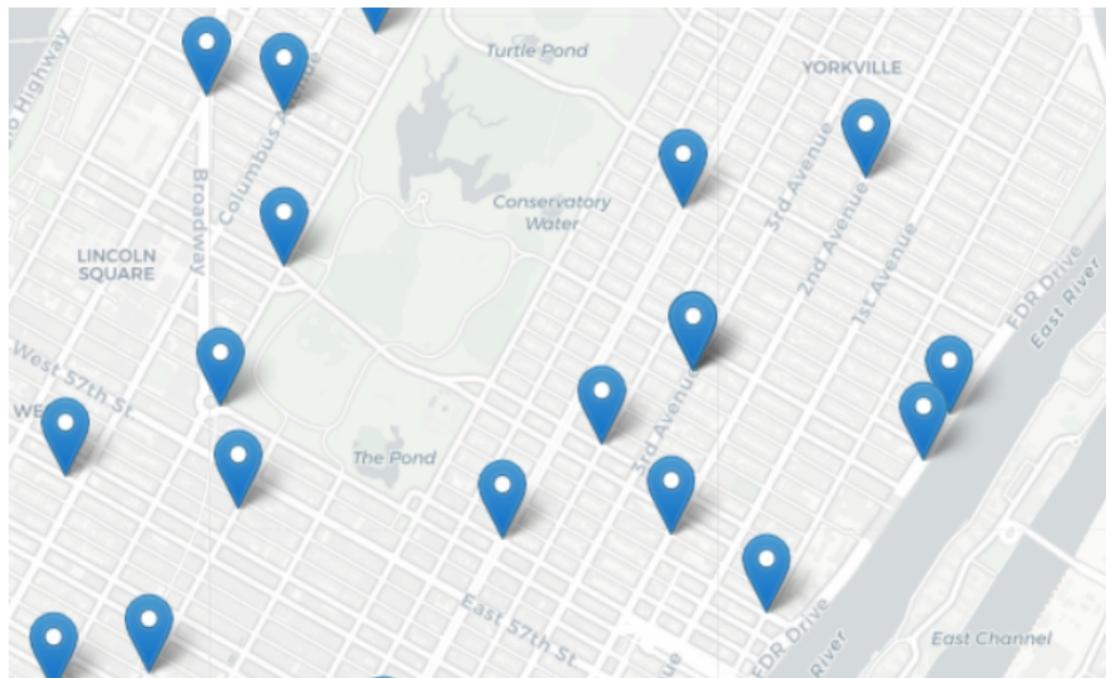
More Views Filter Visualize Export Discuss Embed About

EventID	EventType	StartDateT...	EndDateTime	EnteredOn	EventAg...	ParkingHeld	Borou...	Com...	Police...	Categ...	SubC...	Count...	ZipCo...
455063	Shooting Permit	12/06/2018 07:00...	12/06/2018 09:00...	12/05/2018 12:36...	Mayor's Offic...	STARR AVENUE b...	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 07:00...	12/06/2018 05:00...	12/04/2018 09:11...	Mayor's Offic...	EAGLE STREET be...	Brooklyn	1	94	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 07:00...	12/06/2018 07:00...	12/04/2018 05:44...	Mayor's Offic...	SOUTH OXFORD ...	Brooklyn	2, 6	76, 88	Still Photo...	Not Applic...	United Sta...	11217, 11...
454920	Shooting Permit	12/06/2018 10:00...	12/06/2018 11:59...	12/04/2018 03:28...	Mayor's Offic...	13 AVENUE bew...	Queens	1, 3, 7	109, 7, 90	Film	Feature	United Sta...	10002, 11...
454914	Shooting Permit	12/06/2018 08:00...	12/06/2018 11:00...	12/04/2018 03:05...	Mayor's Offic...	ELBERT STREET b...	Brooklyn	4, 5	104, 75, 83	Television	Episodic s...	United Sta...	11207, 11...
454909	Shooting Permit	12/05/2018 08:00...	12/05/2018 06:00...	12/04/2018 02:45...	Mayor's Offic...	ELBERT STREET b...	Brooklyn	4	83	Television	Episodic s...	United Sta...	11237
454905	Shooting Permit	12/06/2018 07:00...	12/06/2018 10:00...	12/04/2018 02:17...	Mayor's Offic...	35 STREET betwe...	Queens	1	114	Television	Cable-epis...	United Sta...	11101, 11...

Can approach the other questions in the same way:

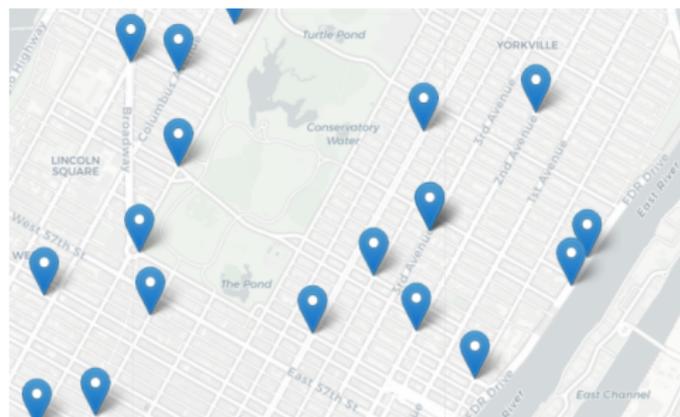
- What's the most popular street for filming?
- What's the most popular borough?
- How many TV episodes were filmed?

Design Question



Design an algorithm that finds the closest collision.

Design Question



Design an algorithm that finds the closest collision.

DATE	TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET N	CROSS STREET	OFF STREET	NUMBER OF
12/31/16	9:56						2 AVENUE			0
12/31/16	9:55	BRONX	10462	40.83521	-73.85497	(40.8352098	UNIONPORT	OLMSTEAD AVENUE		0
12/31/16	9:50						JESUP AVENUE			0
12/31/16	9:40	BROOKLYN	11225	40.66911	-73.95335	(40.6691137	ROGERS AVE	UNION STREET		0
12/31/16	20:23	BROOKLYN	11209	40.62578	-74.02415	(40.6257805	80 STREET	5 AVENUE		0
12/31/16	20:20	QUEENS	11375	40.71958	-73.83977	(40.719584,	ASCAN AVEN	QUEENS BOULEVARD		0
12/31/16	20:15	BROOKLYN	11204				60 STREET	BAY PARKWAY		0
12/31/16	20:10			40.66479	-73.82047	(40.6647944,	-73.8204653)			0
12/31/16	20:10						69 STREET	37 AVENUE		0
12/31/16	20:05	BRONX	10457	40.85429	-73.90026	(40.8542925	RYER AVENUE	EAST 181 STREET		0

Design Question

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

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.

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.

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.

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:

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).

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.

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 file.

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 file.
 - ④ Check distance from each collision to user's location.

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 file.
 - ④ Check distance from each collision to user's location.
 - ⑤ Save the location with the smallest distance.

Today's Topics



- Recap: Slicing & Images
- Introduction to Functions
- NYC Open Data
- **Design Challenge**

Design Challenge

MY HOBBY: EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS

CHOTCHKIES RESTAURANT

~ APPETIZERS ~

MIXED FRUIT	2.15
FRENCH FRIES	2.75
SIDE SALAD	3.35
HOT WINGS	3.55
MOZZARELLA STICKS	4.20
SAMPLER PLATE	5.80

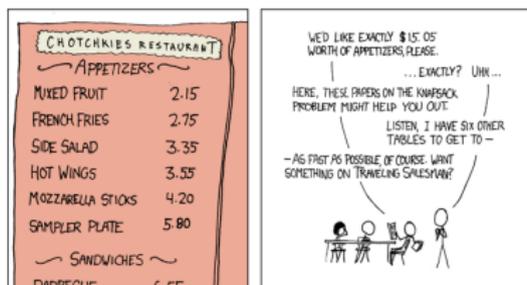
~ SANDWICHES ~

BARBECUE	6.55
----------	------



Design Challenge

MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS



- Possible solutions:

Design Challenge

MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS



- Possible solutions:
 - ▶ 7 orders of mixed fruit, or

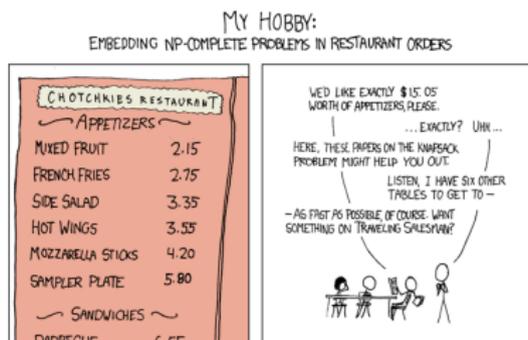
Design Challenge

MY HOBBY:
EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS



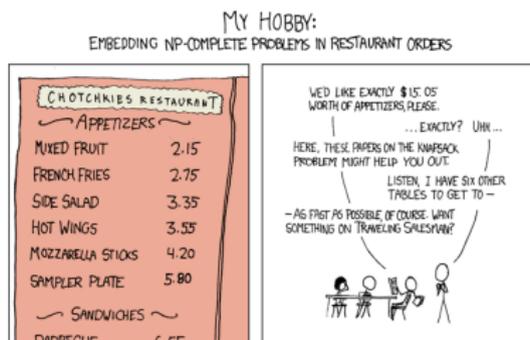
- Possible solutions:
 - ▶ 7 orders of mixed fruit, or
 - ▶ 2 orders hot wings, 1 order mixed fruit, and 1 sampler plate.

Design Challenge



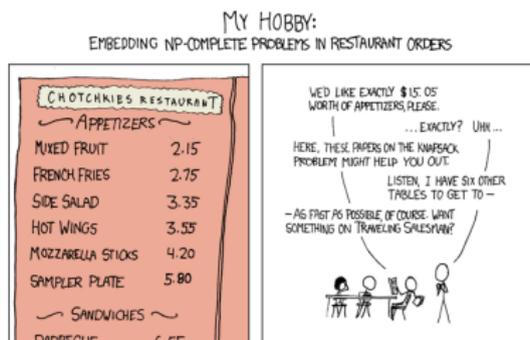
- Possible solutions:
 - ▶ 7 orders of mixed fruit, or
 - ▶ 2 orders hot wings, 1 order mixed fruit, and 1 sampler plate.
- **Input:** List of items with prices and amount to be spent.

Design Challenge



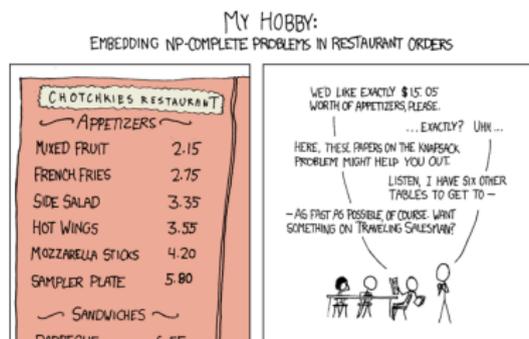
- Possible solutions:
 - ▶ 7 orders of mixed fruit, or
 - ▶ 2 orders hot wings, 1 order mixed fruit, and 1 sampler plate.
- **Input:** List of items with prices and amount to be spent.
- **Output:** An order that totals to the amount or empty list if none.

Design Challenge



- Possible solutions:
 - ▶ 7 orders of mixed fruit, or
 - ▶ 2 orders hot wings, 1 order mixed fruit, and 1 sampler plate.
- **Input:** List of items with prices and amount to be spent.
- **Output:** An order that totals to the amount or empty list if none.
- Possible algorithms: For each item on the list, divide total by price. If no remainder, return a list of that item. Repeat with two items, trying 1 of the first, 2 of the first, etc. Repeat with three items, etc.

Design Challenge



- Possible solutions:
 - ▶ 7 orders of mixed fruit, or
 - ▶ 2 orders hot wings, 1 order mixed fruit, and 1 sampler plate.
- **Input:** List of items with prices and amount to be spent.
- **Output:** An order that totals to the amount or empty list if none.
- Possible algorithms: For each item on the list, divide total by price. If no remainder, return a list of that item. Repeat with two items, trying 1 of the first, 2 of the first, etc. Repeat with three items, etc.
- “NP-Complete” problem: possible answers can be checked quickly, but not known how to compute quickly.

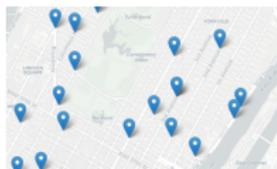
Today's Topics



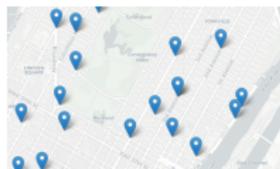
- Recap: Slicing & Images
- Introduction to Functions
- NYC Open Data
- Design Challenge

Recap

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

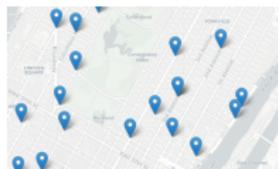


Recap



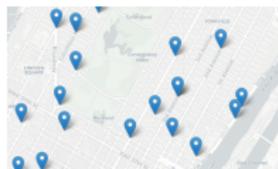
- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:

Recap



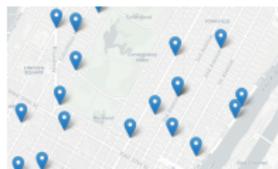
- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`

Recap



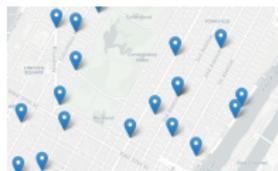
- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

Recap



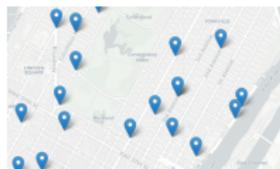
- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.

Recap



- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.
- Accessing Formatted Data: NYC OpenData

Recap



- Functions are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:
Example: `print("Hello", "World")`
- Can write, or **define** your own functions, which are stored, until invoked or called.
- Accessing Formatted Data: NYC OpenData
- [Log in to Gradescope for Quiz 7.](#)