

# CSci 127: Introduction to Computer Science



[hunter.cuny.edu/csci](http://hunter.cuny.edu/csci)

# Frequently Asked Questions

From email

# Frequently Asked Questions

From email

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

*We will start with that today.*

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

*We will start with that today.*

- **Why are we looking at NYC historical population and CUNY enrollment data?**

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

*We will start with that today.*

- **Why are we looking at NYC historical population and CUNY enrollment data?**

*We are showing you how to access and analyze data. The tools we are exploring can be applied to many different datasets.*

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

*We will start with that today.*

- **Why are we looking at NYC historical population and CUNY enrollment data?**

*We are showing you how to access and analyze data. The tools we are exploring can be applied to many different datasets.*

*We will explore many more in the coming weeks!*

- **What is the difference between [ ] and ( )?**

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

*We will start with that today.*

- **Why are we looking at NYC historical population and CUNY enrollment data?**

*We are showing you how to access and analyze data. The tools we are exploring can be applied to many different datasets.*

*We will explore many more in the coming weeks!*

- **What is the difference between [ ] and ( )?**

*Parenthesis ( ) generally follow function names, e.g. print().*

*You may also find them in mathematical and boolean expressions, e.g. (  $x == 2*(y+3)$  ) and (  $x < 10$  )*

# Frequently Asked Questions

From email

- **How do I know the height and width of an image?**

*When you read an image file using pyplot, you can access the number of rows (height) and the number of columns (width) using the shape attribute of a numpy array.*

*We will start with that today.*

- **Why are we looking at NYC historical population and CUNY enrollment data?**

*We are showing you how to access and analyze data. The tools we are exploring can be applied to many different datasets.*

*We will explore many more in the coming weeks!*

- **What is the difference between [ ] and ( )?**

*Parenthesis ( ) generally follow function names, e.g. print().*

*You may also find them in mathematical and boolean expressions,*

*e.g. (  $x == 2*(y+3)$  ) and (  $x < 10$  )*

*We use square brackets [ ] to index or slice,*

*i.e. take a piece, of a string, list or numpy array: my\_string[2:5]*

# Today's Topics



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

# Today's Topics



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

# 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()
```

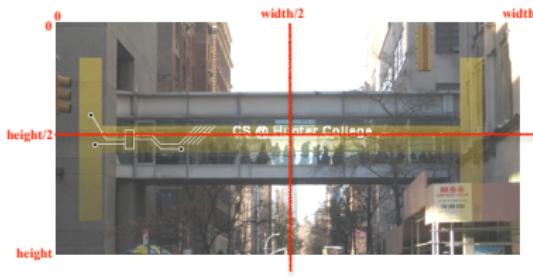
# Challenge: Cropping Images

```
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()
```



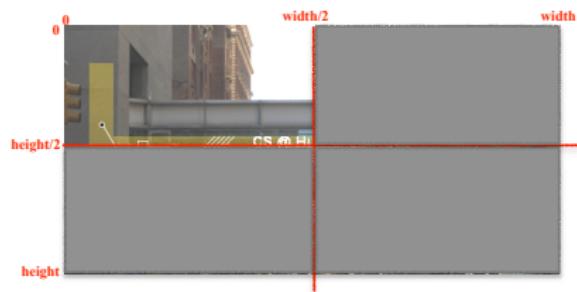
# Challenge: Cropping Images

```
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()
```



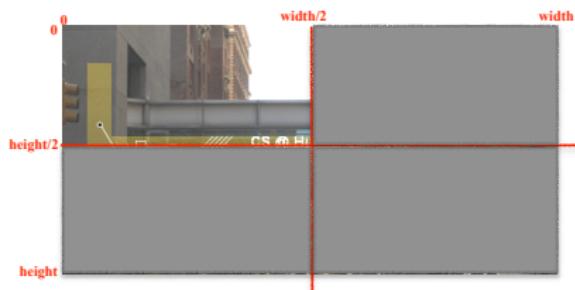
# Challenge: Cropping Images

```
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

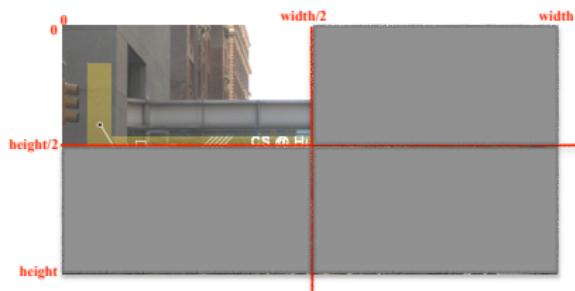
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- 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()
```

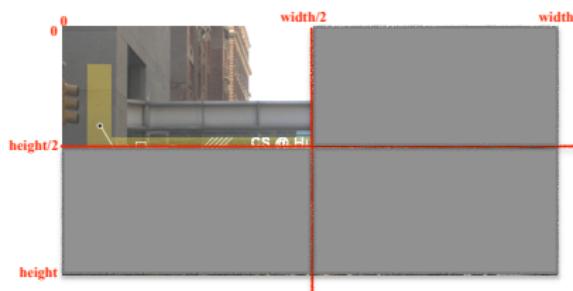


- How would you select the lower left corner?

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

# Challenge: Cropping Images

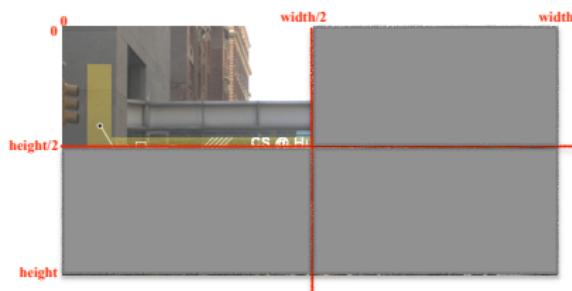
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- 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?

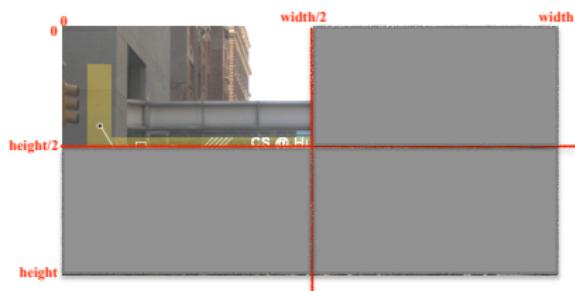
```
img2 = img[height//2:, :width//2]
```

- How would you select the upper right corner?

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

# Challenge: Cropping Images

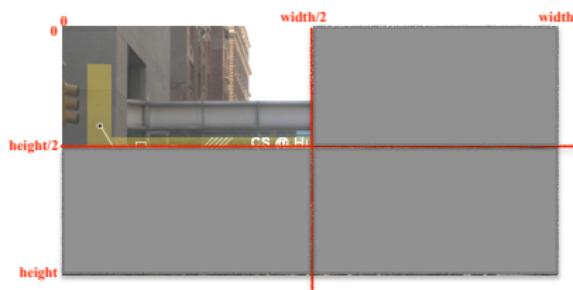
```
import matplotlib.pyplot as plt
import numpy as np
img = plt.imread('csBridge')
plt.imshow(img)
plt.show()
height = img.shape[0]
width = img.shape[1]
img2 = img[:height//2, :width//2]
plt.imshow(img2)
plt.show()
```



- 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()
```



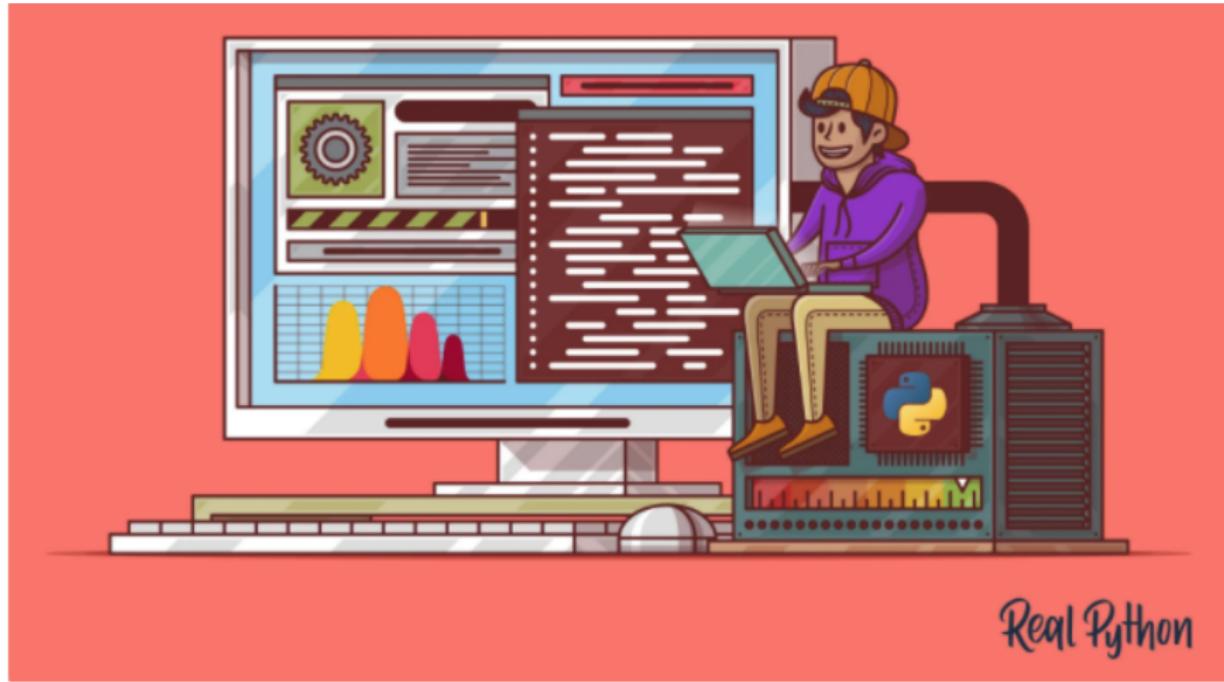
- How would you select the lower left corner?  
`img2 = img[height//2:, :width//2]`
- How would you select the upper right corner?  
`img2 = img[:height//2, width//2:]`
- How would you select the lower right corner?  
`img2 = img[height//2:, width//2:]`

# Today's Topics



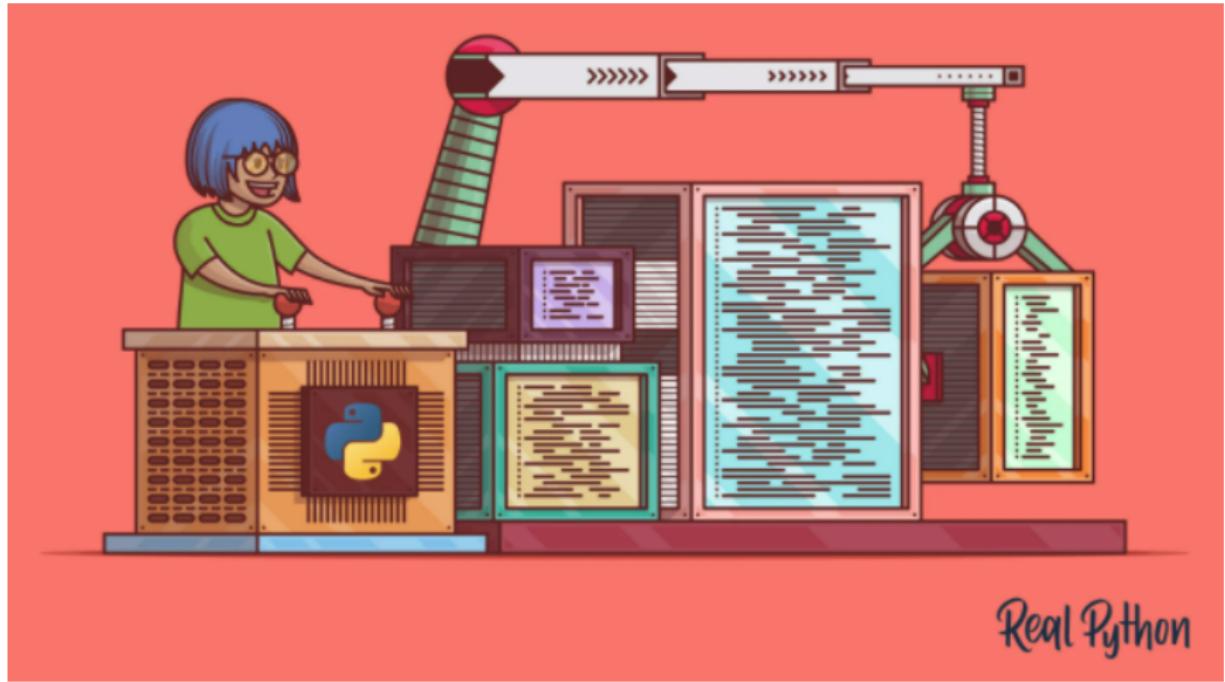
- Recap: Slicing & Images
- **Introduction to Functions**
- NYC Open Data

# Scripts

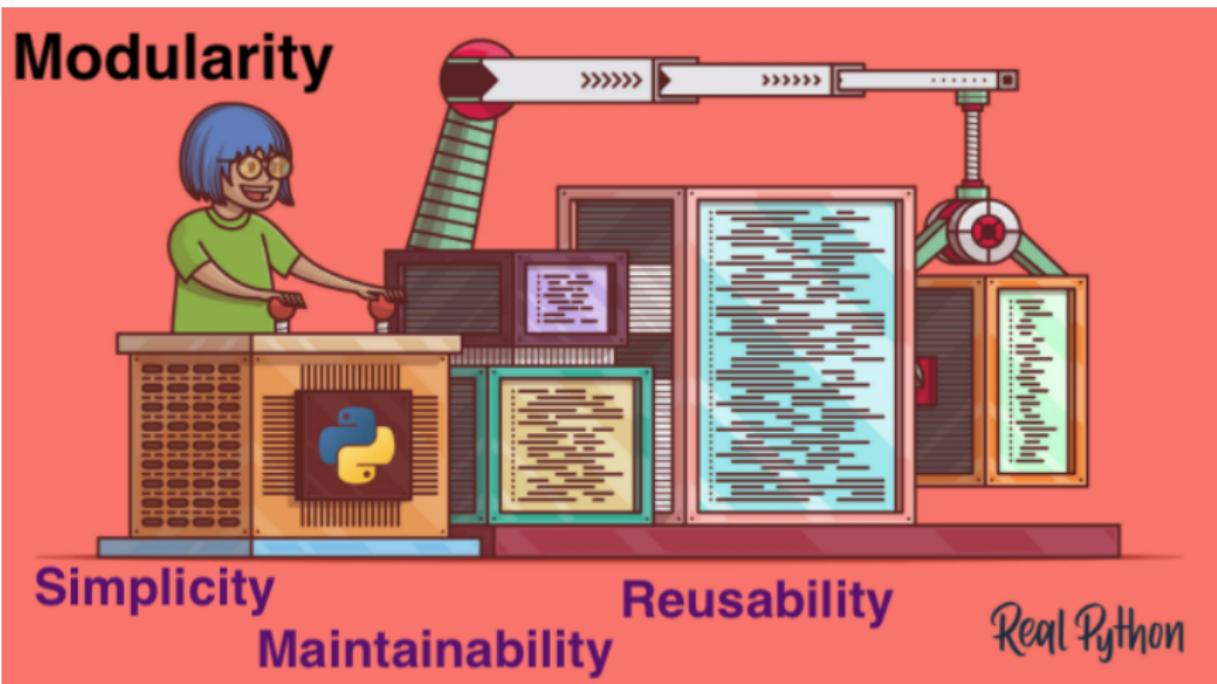


Real Python

# Modularity



# Modularity



# 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()`
- Naming conventions same as variables

# 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()`
- Naming conventions same as variables
- 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()`
- Naming conventions same as variables
- 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()`
- Naming conventions same as variables
- 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()`
- Naming conventions same as variables
- 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

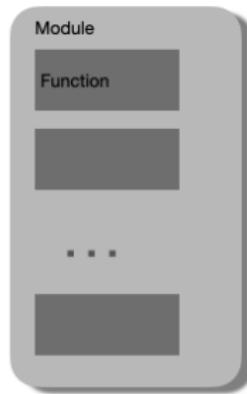
```
#Name: your name here
#Date: October 2017
#This program, uses functions,
#     says hello to the world!

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

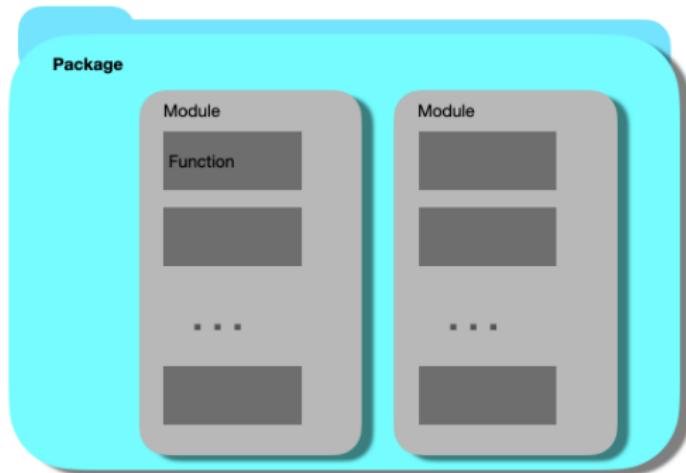
if __name__ == "__main__":
    main()
```

(Demo with pythonTutor)

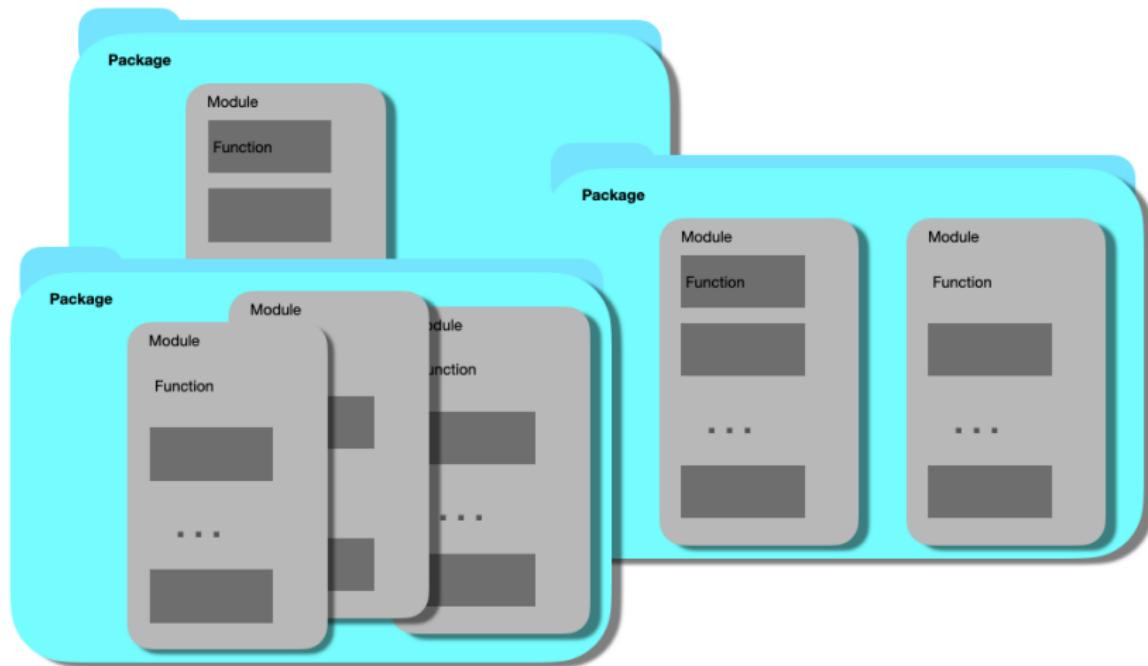
# functions - modules - packages



# functions - modules - packages



# functions - modules - packages



# Stand-alone program

Stand-alone program

```
#include mdl
```



\*\*\*

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

## Challenge:

*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)

# Scope

```
def eight():
    x = 5+3
    print(x)
```

```
def nine():
    x = "nine"
    print(x)
```

- You can have multiple functions.

# Scope

```
def eight():
    x = 5+3
    print(x)

def nine():
    x = "nine"
    print(x)
```

- You can have multiple functions.
- Each function defines the **scope** of its local variables

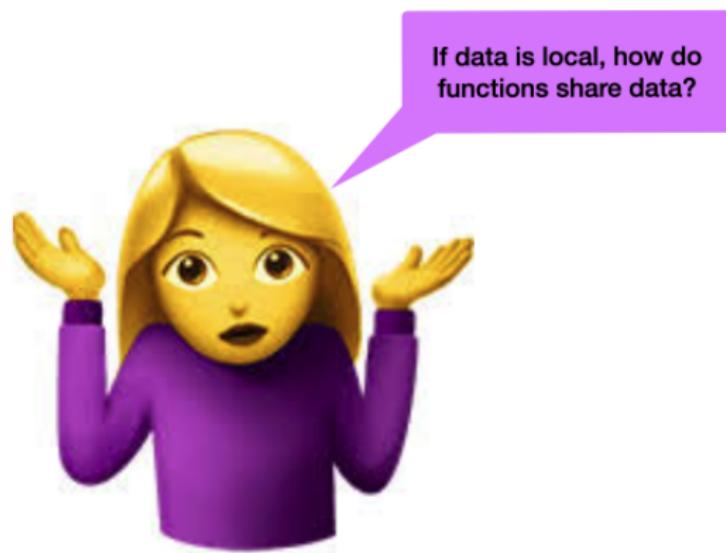
# Scope

```
def eight():
    x = 5+3
    print(x)

def nine():
    x = "nine"
    print(x)
```

- You can have multiple functions.
- Each function defines the **scope** of its local variables
- A variable defined inside a function is **local**, i.e. defined only inside that function.

# Local Data?



# 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 definition, 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 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**.

# 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.

# Challenge:

*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:

*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:

*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:

*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:

*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: '))
    mString = monthString(n)
    print('The month is', mString)
```

## (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**

# Accessing Structured Data: NYC Open Data



The image shows the homepage of the NYC Open Data website. The background is blue. On the left, there is a large white text area containing the title "Open Data for All New Yorkers" and a brief description of what the data can be used for. Below this is a search bar. On the right, there is a graphic featuring four stylized human figures and several speech bubbles containing icons related to various city services like education, health, and transportation.

## 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



- Freely available source of data.

# Accessing Structured Data: NYC Open Data



The image shows the homepage of the NYC Open Data website. The background is blue. At the top left, there is a large white title: "Open Data for All New Yorkers". Below the title, there 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." To the right of the text, there is a row of four stylized human figures (two men and two women) of different ethnicities. Above the figures, there is a cluster of speech bubbles containing icons related to data: a heart, a house, a map, a graph, a soccer ball, a computer monitor, and other symbols.

Search Open Data for things like 311, Buildings, Crime

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

# Accessing Structured Data: NYC Open Data



The image shows the homepage of the NYC Open Data website. The background is blue. At the top left, there is a large white title: "Open Data for All New Yorkers". Below the title, there 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." To the right of the text, there is a row of four stylized human figures (two men and two women) in business attire. Above the figures, there is a cluster of speech bubbles containing various icons related to city life, such as a heart, a house, a map, a graph, a soccer ball, and a computer monitor.

Search Open Data for things like 311, Buildings, Crime

- 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



The image shows the homepage of the NYC Open Data website. The background is blue. At the top left, there is a large white title: "Open Data for All New Yorkers". 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." To the right of the text, there is a row of four stylized human figures (two men and two women) and a cluster of speech bubbles containing icons related to data: a heart, a house, a map, a graph, a soccer ball, a person at a computer, and a leaf.

Search Open Data for things like 311, Buildings, Crime

- 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



The image shows the homepage of the NYC Open Data website. The background is blue. At the top left, there is a large white text area containing the title "Open Data for All New Yorkers". 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." To the right of the text, there is a graphic featuring four stylized human figures (two men and two women) standing below a cluster of speech bubbles. The speech bubbles contain icons representing various data categories such as a heart, a building, a map, a graph, a soccer ball, a house, a person at a computer, and a leaf. At the bottom left, there is a search bar with the placeholder text "Search Open Data for things like 311, Buildings, Crime".

## 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

- 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



Home Data About ▾ Learn

## 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	StartDateTi...	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



Home Data About Learn Alerts Contact Us Blog



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/mome/permits/when-permit-required.page>

EventID	EventType	StartDateTime	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



Home Data About Learn Alerts Contact Us Blog



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/mome/permits/when-permit-required.page>

Find in this Dataset  
More Views Filter Visualize Export Discuss Embed About

EventID	EventType	StartDateTime	EndDateTime	EnteredOn	EventAge	ParkingHeld	Borough	Com...	Police...	Category	SubCategory	Count...	ZipCode
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

NYC OpenData

Home Data About Learn Alerts Contact Us Blog



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/mome/permits/when-permit-required.page>

The screenshot shows a data visualization interface for a 'Film Permits' dataset. At the top, there are navigation links: Home, Data, About, Learn, Alerts, Contact Us, Blog, a search bar, and a 'Sign In' button. Below the header, there's a brief description of what permits are used for. To the right of the description are several small icons: a magnifying glass, a refresh, a back/forward, and a search bar labeled 'Find in this Dataset'. Below these are buttons for 'More Views', 'Filter', 'Visualize', 'Export', 'Discuss', and 'Embed'. A horizontal line separates this from the main data table. The table has 17 columns with headers: EventID, EventType, StartDateTl, EndDateTime, EnteredOn, EventAg..., ParkingHeld, Borough, Com..., Police..., Categ..., SubC..., Count..., ZipCo..., and a final column with ellipses. There are 8 rows of data, each representing a permit application. The data includes details like the permit type (Shooting Permit), dates, location (e.g., STARR AVENUE b...), and associated entities (Mayor's Office). The last column contains additional identifiers like 'United Sta...' and zip codes.

EventID	EventType	StartDateTl	EndDateTime	EnteredOn	EventAg...	ParkingHeld	Borough	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...	ELDER 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...	ELDER 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

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

EventID	EventType	StartDateTm	EndDateTm	EnteredOn	EventMgt.	ParkingInfo	Boro...	Com...	Permit...	Categ...	SubC...	Crater...	ZipCo...
455063	Shooting Permit	12/05/2018 0700..	12/06/2018 0900..	12/05/2018 1230..	Mayor's Off...	STARR AVENUE Bl..	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 0700..	12/06/2018 0500..	12/06/2018 0201..	Mayor's Off...	SAGGS STREET Bl..	Brooklyn	1	84	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 0700..	12/06/2018 0700..	12/06/2018 0944..	Mayor's Off...	SOUTH OXFORD ..	Brooklyn	2, 8	76, 88	Still Photo..	Not Applic...	United Sta...	11217, 11..
454920	Shooting Permit	12/06/2018 1000..	12/06/2018 1150..	12/06/2018 0232..	Mayor's Off...	13 AVENUE betw..	Queens	1, 3, 7	108, 7, 98	Film	Feature	United Sta...	10802, 11...
454914	Shooting Permit	12/06/2018 0800..	12/06/2018 1100..	12/06/2018 0359..	Mayor's Off...	ELDERT STREET Bl..	Brooklyn	4, 5	104, 75, 89	Television	Episodic s...	United Sta...	11203, 11..
454909	Shooting Permit	12/05/2018 0800..	12/05/2018 0600..	12/04/2018 0245..	Mayor's Off...	ELDERT STREET Bl..	Brooklyn	4	83	Television	Episodic s...	United Sta...	11237
454905	Shooting Permit	12/06/2018 0700..	12/06/2018 1000..	12/06/2018 0217..	Mayor's Off...	36 STREET betwe..	Queens	1	114	Television	Color+sp...	United Sta...	11101, 11..

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

# Example: OpenData Film Permits

EventID	EventType	StartDate/Time	EndDate/Time	EnteredOn	EventMg.	ParkingHeld	Block#	Borough	Comm.	Police	CompCo	SubCo	Create	ZipCode
455063	Shooting Permit	12/05/2018 0700..	12/06/2018 0900..	12/05/2018 1230..	Mayor's Offc.	STARK AVENUE Bl..	Queens	2	108	Television	Epidemic s..	United Sta..	11/01	
454967	Shooting Permit	12/06/2018 0700..	12/06/2018 0500..	12/06/2018 0201..	Mayor's Offc.	SAGGS STREET Bl..	Brooklyn	1	84	Television	Epidemic s..	United Sta..	11/22	
454941	Shooting Permit	12/06/2018 0700..	12/06/2018 0700..	12/06/2018 0244..	Mayor's Offc.	SOUTH OXFORD ..	Brooklyn	2, 9	76, 88	Still Photo	Not Applic..	United Sta..	11/21/11..	
454920	Shooting Permit	12/06/2018 1000..	12/06/2018 1150..	12/06/2018 0232..	Mayor's Offc.	13 AVENUE betw..	Queens	1, 3, 7	106, 7, 98	Film	Feature	United Sta..	10/02, 11..	
454914	Shooting Permit	12/06/2018 0800..	12/06/2018 1100..	12/06/2018 0239..	Mayor's Offc.	ELDERT STREET Bl..	Brooklyn	4, 5	104, 75, 89	Television	Epidemic s..	United Sta..	12/03, 11..	
454909	Shooting Permit	12/05/2018 0800..	12/05/2018 0600..	12/04/2018 0245..	Mayor's Offc.	ELDERT STREET Bl..	Brooklyn	4	83	Television	Epidemic s..	United Sta..	11/23	
454905	Shooting Permit	12/06/2018 0700..	12/06/2018 1030..	12/06/2018 0217..	Mayor's Offc.	36 STREET betwe..	Queens	1	114	Television	Cable-sys..	United Sta..	11/05, 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

Film Permits

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

EventID	EventType	StartDateTm	EndDateTm	EnteredOn	EventMgt.	ParkingHeld	Boro...	Cou...	Permit...	Compl...	SubCo...	Crater...	ZipCo...
455063	Shooting Permit	12/05/2018 0700..	12/06/2018 0900..	12/05/2018 1235..	Mayor's Off...	STARK AVENUE Bl..	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 0700..	12/06/2018 0500..	12/06/2018 0911..	Mayor's Off...	SAGGS STREET Bl..	Brooklyn	1	84	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 0700..	12/06/2018 0700..	12/06/2018 0944..	Mayor's Off...	SOUTH OXFORD ..	Brooklyn	2, 9	76, 88	Self-Permit	Not Applic...	United Sta...	11217, 11...
454920	Shooting Permit	12/06/2018 1000..	12/06/2018 1155..	12/06/2018 0232..	Mayor's Off...	13 AVENUE betw..	Queens	1, 3, 7	106, 7, 98	Film	Feature	United Sta...	10082, 11...
454914	Shooting Permit	12/06/2018 0600..	12/06/2018 1100..	12/06/2018 0239..	Mayor's Off...	ELDERY STREET Bl..	Brooklyn	4, 5	104, 75, 89	Television	Episodic s...	United Sta...	11203, 11...
454909	Shooting Permit	12/05/2018 0800..	12/05/2018 0600..	12/04/2018 0245..	Mayor's Off...	ELDERY STREET Bl..	Brooklyn	4	83	Television	Episodic s...	United Sta...	11227
454905	Shooting Permit	12/06/2018 0700..	12/06/2018 1030..	12/06/2018 0217..	Mayor's Off...	36 STREET betwe..	Queens	1	114	Television	Color+epis...	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

EventID	EventType	StartDate	EndDate	EnteredOn	EventMgt	ParkingHeld	Block	Comm.	Police	CompCo	Create	ZipCode
455063	Shooting Permit	12/05/2018 0700..	12/06/2018 0900..	12/05/2018 1230..	Mayor's Offic...	STARR AVENUE Bl..	Queens	2	108	Television	Episodic s...	United Stat... 11101
454967	Shooting Permit	12/06/2018 0700..	12/06/2018 0500..	12/06/2018 0901..	Mayor's Offic...	SAGGS STREET Bl..	Brooklyn	1	84	Television	Episodic s...	United Stat... 11222
454941	Shooting Permit	12/06/2018 0700..	12/06/2018 0700..	12/06/2018 0944..	Mayor's Offic...	SOUTH OXFORD ..	Brooklyn	2, 9	76, 88	Self Prod..	Not Applic...	United Stat... 11217, 11...
454920	Shooting Permit	12/06/2018 1000..	12/06/2018 1150..	12/06/2018 0232..	Mayor's Offic...	13 AVENUE betw..	Queens	1, 3, 7	108, 7, 98	Film	Feature	United Stat... 10082, 11...
454914	Shooting Permit	12/06/2018 0800..	12/06/2018 1100..	12/06/2018 0239..	Mayor's Offic...	ELDERY STREET Bl..	Brooklyn	4, 5	104, 75, 89	Television	Episodic s...	United Stat... 11203, 11...
454909	Shooting Permit	12/05/2018 0800..	12/05/2018 0600..	12/04/2018 0245..	Mayor's Offic...	ELDERY STREET Bl..	Brooklyn	4	83	Television	Episodic s...	United Stat... 11237
454905	Shooting Permit	12/06/2018 0700..	12/06/2018 1030..	12/06/2018 0217..	Mayor's Offic...	36 STREET betwe..	Queens	1	114	Television	Cable-sys...	United Stat... 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

EventID	EventType	StartDate	EndDate	EnteredOn	EventMg.	ParkingHeld	Block	Com.	Police	CompAg	SubCtg	Create	ZipCode
454983	Shooting Permit	12/05/2018 0700..	12/06/2018 0900..	12/05/2018 1230..	Mayor's Offic...	STARR AVENUE Bl...	Queens	2	108	Television	Epidemic s...	United Stat...	11101
454967	Shooting Permit	12/06/2018 0700..	12/06/2018 0500..	12/06/2018 0901..	Mayor's Offic...	SAGGS STREET Bl...	Brooklyn	1	84	Television	Epidemic s...	United Stat...	11222
454941	Shooting Permit	12/06/2018 0700..	12/06/2018 0700..	12/06/2018 0944..	Mayor's Offic...	SOUTH OXFORD ..	Brooklyn	2, 9	76, 88	Self Photo..	Not Applic...	United Stat...	11217, 11...
454920	Shooting Permit	12/06/2018 1100..	12/06/2018 1150..	12/06/2018 0232..	Mayor's Offic...	13 AVENUE betw...	Queens	1, 3, 7	108, 7, 98	Film	Feature	United Stat...	10002, 11...
454914	Shooting Permit	12/06/2018 0600..	12/06/2018 1100..	12/06/2018 0239..	Mayor's Offic...	ELDERY STREET Bl...	Brooklyn	4, 5	104, 75, 89	Television	Epidemic s...	United Stat...	11203, 11...
454909	Shooting Permit	12/05/2018 0800..	12/05/2018 0600..	12/04/2018 0245..	Mayor's Offic...	ELDERY STREET Bl...	Brooklyn	4	83	Television	Epidemic s...	United Stat...	11227
454905	Shooting Permit	12/06/2018 0700..	12/06/2018 1030..	12/06/2018 0217..	Mayor's Offic...	36 STREET betwe...	Queens	1	114	Television	Cable-epis...	United Stat...	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

Film Permits

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

EventID	EventType	StartDateTm	EndDateTm	EnteredOn	EventMgt.	ParkingHeld	Boro...	Cou...	Permis...	Compl...	SubCo...	Crater...	ZipCo...
455083	Shooting Permit	12/05/2018 0700..	12/06/2018 0900..	12/05/2018 1230..	Mayor's Off...	STARR AVENUE Bl...	Queens	2	108	Television	Episodic s...	United Sta...	11101
454967	Shooting Permit	12/06/2018 0700..	12/06/2018 0500..	12/06/2018 0201..	Mayor's Off...	SAGGS STREET Bl...	Brooklyn	1	84	Television	Episodic s...	United Sta...	11222
454941	Shooting Permit	12/06/2018 0700..	12/06/2018 0700..	12/06/2018 0544..	Mayor's Off...	SOUTH OXFORD ..	Brooklyn	2, 9	76, 88	Self Prod.	Not Applic...	United Sta...	11217, 11...
454920	Shooting Permit	12/06/2018 1000..	12/06/2018 1150..	12/06/2018 0232..	Mayor's Off...	13 AVENUE betw...	Queens	1, 3, 7	106, 7, 98	Film	Feature	United Sta...	10802, 11...
454914	Shooting Permit	12/06/2018 0800..	12/06/2018 1100..	12/06/2018 0239..	Mayor's Off...	ELDERY STREET Bl...	Brooklyn	4, 5	104, 75, 89	Television	Episodic s...	United Sta...	12020, 11...
454909	Shooting Permit	12/05/2018 0800..	12/05/2018 0900..	12/04/2018 0245..	Mayor's Off...	ELDERY STREET Bl...	Brooklyn	4	83	Television	Episodic s...	United Sta...	11227
454905	Shooting Permit	12/06/2018 0700..	12/06/2018 1000..	12/06/2018 0217..	Mayor's Off...	36 STREET betwe...	Queens	1	114	Television	Color-Epis...	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

NYC OpenData

Home Data About Learn Alerts Contact Us Blog Sign In

## Film Permits

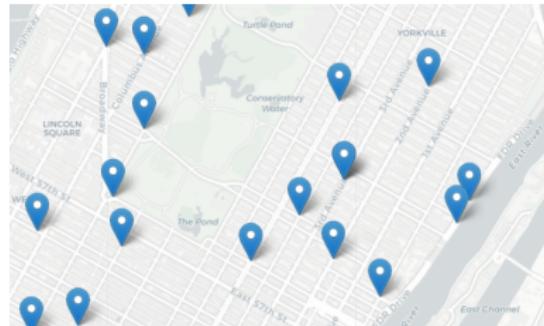
Permits are generally required when asserting the exclusive use of city property, like a sidewalk, a street, or a park. See <http://www1.nyc.gov/site/mome/permits/when-permit-required.page>

													Find in this Dataset
EventID	EventType	StartDateTL	EndDateTL	EnteredOn	EventAge	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...	ELDERST 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...	ELDERST 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 collision that is closest to input location.

DATE	TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET	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, -73.85497]	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, -73.95335]	ROGERS AVE	UNION STREET		0
12/31/16	20:23	BROOKLYN	11209	40.62578	-74.02415	[40.6257805, -74.02415]	80 STREET	5 AVENUE		0
12/31/16	20:20	QUEENS	11375	40.71958	-73.83977	[40.719584, -73.83977]	ASCAN AVENUE	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, -73.90026]	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 Read 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 Read the CSV file.
  - ④ Check distance from each collision to user’s location.

# Design Question

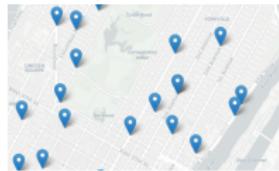
Design an algorithm that uses NYC OpenData collision data and computes the closest collision to the location the user provides.

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

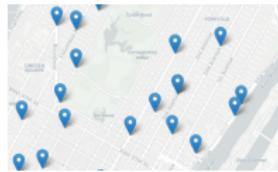
# Recap

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

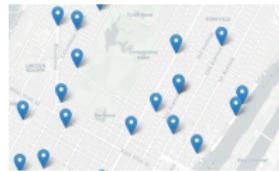


# Recap

- **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



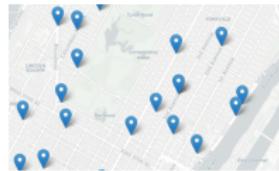
- **Functions** are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:  
Example: `print("Hello", "World")`

# Recap



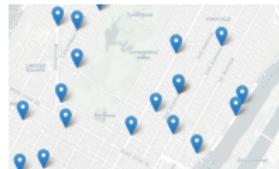
- **Functions** are a way to break code into pieces, that can be easily reused.
- You **call** or **invoke** a function by typing its name, followed by any inputs, surrounded by parenthesis:  
Example: `print("Hello", "World")`
- Can write, or **define** your own functions,

# 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

# Class Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

# Class Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Take the Lab Quiz



# Class Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Take the Lab Quiz
- Submit this class's 5 programming assignments (programs 31-35)