GGY 421 / GGY 593 Spatial Programming
Mid Exam

Par 1 & 2

Name: Connor Geis

Submission Instructions

- For Part 1, please answer the questions with a different color (e.g., blue) other than black.
- For Multiple Choice questions in Part 2, please highlight your selection like "A. int, 3."
- For the last fix-bug question in Part 2, please attached your screenshots to show your fixed program and program output.
- For each question in part 3, please make a screenshot of your code and the code execution result. Attach these screenshots directly in this Word document following each question.
- Required submission: 1) This Word document including your answers and screenshots, and 2) Your Python source code file. Create a jupyter note book (e.g., midexam_*yourlastname*). Put your answers to each question in a cell (i.e., each cell contains codes for answers to one question). You should be able to find the file with extension .ipynb under "C:\Users*yourusersname*".
- Submit the files to Canvas.

Part 1: Concept (20 points)

1. In Python, what is the difference and similarities between a *function* and a *module*? (100 words max) (5 points)

Functions offer a way to isolate a part of your program's functionality and make it reusable. Modules are a way to collect a number of functions in a single file, which you can then call and use in multiple projects and share with other programmers as a .py file. Functions and modules are similar in that they can be called in your program, but a function is specific to a task and can only be called within the application it is defined. A module defines classes, functions, attributes, etc, and can be imported from your desktop or in jupyter notebooks.

2. What is the different between a ***local variable*** and a ***global variable*** (feel free to use sample codes to facilitate your explanation)? (5 points).

A variable is a name assigned to a storage area that a program can manipulate. A global variable is a variable that is accessible globally, and they are ideally used for storing constants that will be utilized throughout the program. It is a variable defined outside of a subroutine or function, and it holds throughout the lifetime of the program, meaning it can be accessed throughout the program by any function defined within the program. A local variable is a variable that is only accessible to the current scope, such as temporary variables used within a single function definition. It can only be used inside the code block in which it is declared. It exists while the block of the function is under execution.

3. Use an example to explain the importance of loops in programming (100 words max) (5 points).

Loops are a programming element that repeats a portion of code until the desired process is complete or until a specified condition is reach. They are essential to save time and minimize human error and allow a programmer to use the same lines of code iteratively. For example, it may be easy to manually determine whether a value in a list of 7 numbers is even or odd, but if the list contains thousands or millions of values, this simple task would become very tedious, and creating a program that uses a simple loop would be beneficial.

4. What is a *default argument* of a *function*? When to use *default arguments*? (100 words max) (5 points)

Default arguments in Python functions are arguments that are assigned a default value by using the operator (=), i.e. keywordname=value. Functions with optional arguments offer more flexibility in how you can use them. Default values indicate that the function argument will take that value if no argument value is passed during the function call. They are used to make sure the function will still work, because you can call the function with or without the argument.

Part 2: Syntax (20 points)

1. When naming a variable, which of the following is the best? (2 points)

A. polylineLength B. polyline-length C. PolylineLength D. polyline length

2. Let variable a = 7, b =2.0, and $c = a / b$. What is the type and value of variable c? (2 points)

A. int, 3 B. float, 3.0 C. float, 3.5 D. string, 3.5

In [56]:
```
a = 7
b = 2.0
c = a/b
type(c)
```

Out[56]: float

In [57]:
```
a = 7
b = 2.0
c = a/b
type(c)
print(c)
```

```
3.5
```

3. When reading a text file, let *lines* = f.readlines(), what is the type of *lines*? (2 points)

A. string B. tuple C. list D. int

In [53]:
```
f = open ('./assignment3_ boundaries.txt', 'r')
lines = f.readlines()
type(lines)
```

Out[53]: list

4. Let variable *a* = 2, *b* = 0, what is the value of the expression: *not (a or not b)*? (2 points)

A. True B. False C. 2 D. cannot be determined

In [51]:
```
a = 2
b = 0
not(a or not b)
```

Out[51]: False

5. The type of a Python variable is dynamically determined when assigning value to the variable. Is this statement True or False? (2 points)

6. Let variable *a* = ("15", "10", "8"), what is the value of the expression: ***10 in a*** ? __false__ (2 points)

In [45]:
```
a = ("15", "10", "8")
10 in a
```

Out[45]: False

7. When using the *split* function to split string ***"ab;cd;ef;"*** with the separator semicolon(;), what is the length (# of elements) of the result list? _____4_____ (2 points) (**Hint**: make sure you test the answer by yourself)

In [48]:
```
1 myList = "ab;cd;ef;"
2 myList = myList.split(";")
3 print(myList)
4 print (len(myList))
```

```
['ab', 'cd', 'ef', '']
4
```

8. Fix the bugs (issues) in the following program to output the point coordinates exactly as

X: 15; Y: 36
X: 37; Y: 78
X: 39; Y: 42

Program to be fixed:

```
pointList = [(15,36),(37,78),(39,42)]
for point in pointlist
print ('X: {}, Y: {}'.format(point[1],point[2]))
```

If you would like to know about the function format(), you can find it here https://www.w3schools.com/python/ref_string_format.asp. But you can fix all the bugs without knowing the function.

Please attached your screenshots to show your fixed program and program output (6 points).

In [66]:
```
1 pointList = [(15,36),(37,78),(39,42)]
2 for point in pointList:
3     print ('X: {}; Y: {}'.format(point[0],point[1]))
4
```

```
X: 15; Y: 36
X: 37; Y: 78
X: 39; Y: 42
```

Part 3: Problem Solving (40 points)

Note: For each problem in this part, please attach the screenshots to show your codes and execution results.

1. Given the following numbers: 3, 6, 8, 12, 5, 9, 10. Answer the following questions:

1) Define a variable to store these numbers (2 point).

```
#part3 q1.

#define variable to store numbers
nums = [3, 6, 8, 12, 5, 9, 10]
```

2) Use a *for* loop to print out all even numbers (2 points).

In [5]:
```
#part3 q1.

#define variable to store numbers
nums = [3, 6, 8, 12, 5, 9, 10]

#print out all even numbers
for num in nums:
    if num % 2 == 0:
        print (str(num ) + " from nums is an even number.")
```

```
6 from nums is an even number.
8 from nums is an even number.
12 from nums is an even number.
10 from nums is an even number.
```

3) Define five functions named: ***myMean, myMax, myMin, myRange, myStd*** to calculate the average, maximum, minimum, range, and standard deviation of a given set of numbers. Each function takes a **list** as input and **return** the result. Test your functions with the given numbers (10 points)

In [28]:
```
nums = [3, 6, 8, 12, 5, 9, 10]

def myMean(nums):
    mean = sum(nums)/len(nums)
    return mean
myMean(nums)
```

Out[28]: 7.571428571428571

In [43]:
```
nums = [3, 6, 8, 12, 5, 9, 10]
def myMax(nums):
    maximum = (nums)[0]
    for i in nums[0:]:
        if i > maximum:
            maximum = i
    return maximum
myMax(nums)
```

Out[43]: 12

In [41]:
```
nums = [3, 6, 8, 12, 5, 9, 10]
def myMin(nums):
    minimum = (nums)[0]
    for i in nums[0:]:
        if i < minimum:
            minimum = i
    return minimum
myMin(nums)
```

Out[41]: 3

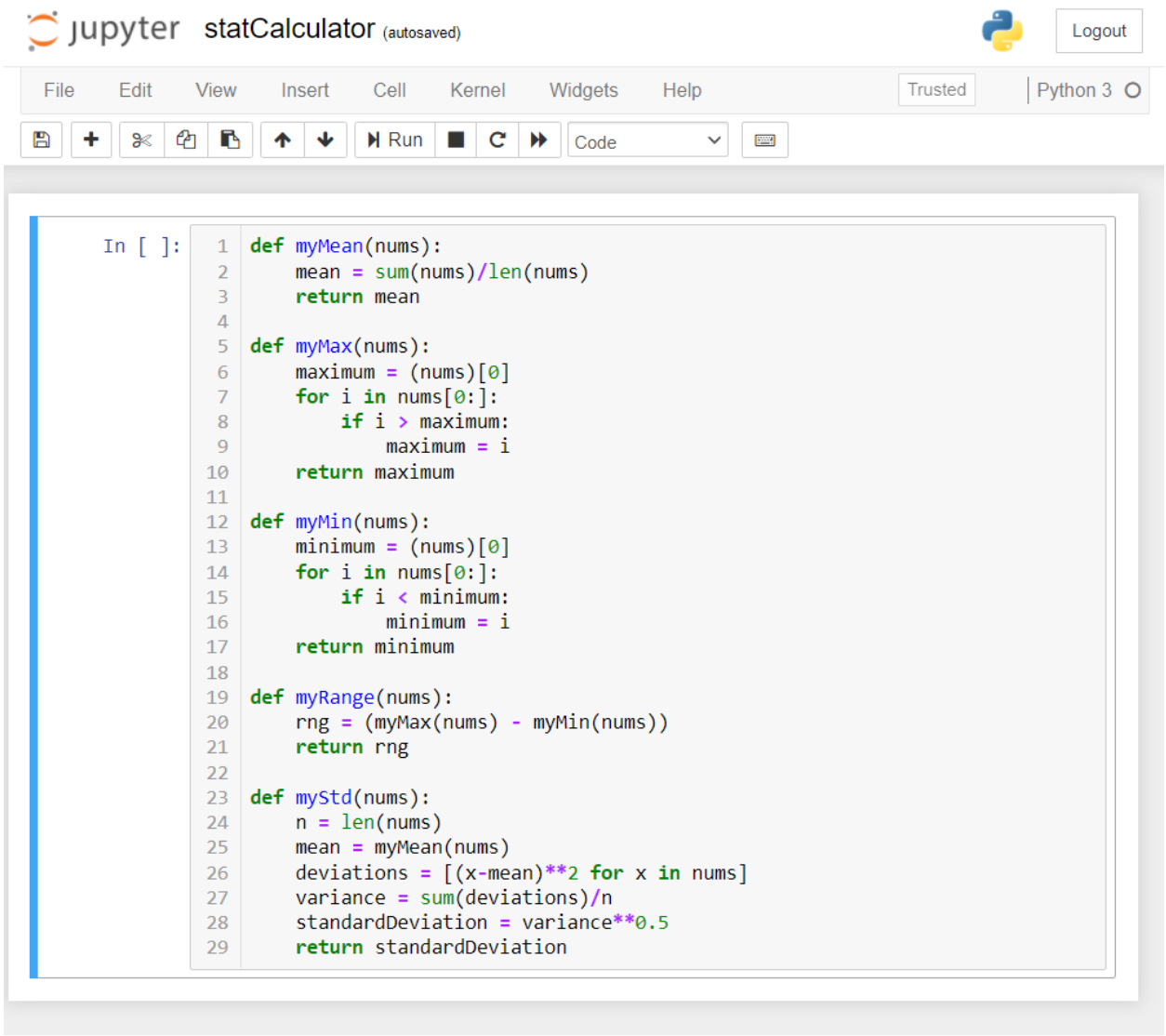
In [74]:
```
nums = [3, 6, 8, 12, 5, 9, 10]
def myRange(nums):
    rng = (myMax(nums) - myMin(nums))
    return rng
myRange(nums)
```

Out[74]: 9

In [68]:
```
nums = [3, 6, 8, 12, 5, 9, 10]
def myStd(nums):
    n = len(nums)
    mean = myMean(nums)
    deviations = [(x-mean)**2 for x in nums]
    variance = sum(deviations)/n
    standardDeviation = variance**0.5
    return standardDeviation
myStd(nums)
```

Out[68]: 2.8713930346059686

4) Develop a module named *StatCalculator* to include the five functions you just defined. Import the *StatCalculator* module in another Python file to calculate the **average** and **standard deviation** of the one thousand numbers (integers) from 1 to 1000. (6 points)




```
In [88]:
import statCalculator
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2

statCalculator.myMean(nums)
#statCalculator.myStd(nums)
```

```
Out[88]: 500.5
```

```
In [94]:
import statCalculator
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2

#statCalculator.myMean(nums)
statCalculator.myStd(nums)
```

```
Out[94]: 288.6749902572095
```

2. PointsExam1.txt stores longitudes and latitudes of 20 points. Read data from PointsExam1.txt. Answer the following questions:

1) Define a function that reads in data from PointsExam1.txt and returns a list that hold all points. The longitude and latitude of a point are represented by a tuple. The input of this function is a point file. The function returns a list of points. The structure of the function is below.

```
def getPointList(filePath):
# compile your codes here
        return pointList
```

When you call the function using the following scripts

```
pointList =getPointList ("C:/spatial programming/exam/PointsExam1.txt") # change the file path in your computer
print (pointList)
```

In [159]:

```
def getPointList(filepath):
    f = open(filepath, "r")
    f.readline()
    lines = f.readlines()
    pointList = []
    for line in lines:
        line = line.strip()
        cols = line.split(",")
        lon = cols[1]
        lat = cols[2]
        point = (lon,lat)
        pointList.append(point)
    return pointList
    getPointList('./PointsExam1.txt')
pointList = getPointList('./PointsExam1.txt')
print(pointList)
```

```
[('-76.61815', '36.2883'), ('-78.77221', '34.04806'),
('82.86309', '37.23665'), ('-79.62142', '34.73552'),
('84.50021', '33.58375'), ('-79.83924', '33.07177'),
('83.64016', '32.90659'), ('-81.22473', '37.07647'),
('80.19746', '35.7788'), ('82.99568', '32.95625'), ('8
3.61621', '34.7964'), ('84.0979', '33.17326'), ('83.58
11', '34.10279'), ('-76.30694', '35.72018'), ('-77.569
42', '34.25918'), ('-76.293', '32.4642'), ('-79.9748
7', '32.16343'), ('-76.29782', '37.53573'), ('-78.5512
7', '37.49596'), ('-76.18114', '34.6022')]
```

you should get
[(-76.61815, 36.2883), (-78.77221, 34.04806), (82.86309, 37.23665), (-79.62142, 34.73552), (84.50021, 33.58375), (-79.83924, 33.07177), (83.64016, 32.90659), (-81.22473, 37.07647), (80.19746, 35.7788), (82.99568, 32.95625), (83.61621, 34.7964), (84.0979, 33.17326), (83.5811, 34.10279), (-76.30694, 35.72018), (-77.56942, 34.25918), (-76.293, 32.4642), (-79.97487, 32.16343), (-76.29782, 37.53573), (-78.55127, 37.49596), (-76.18114, 34.6022)]
(Undergraduate: 5 points; Graduate: 4 points)

If you don't know how to answer the first question. Just assume you have a list of points as

```
pointList = [(-76.61815, 36.2883), (-78.77221, 34.04806), (82.86309, 37.23665), (-79.62142, 34.73552), (84.50021, 33.58375), (-79.83924, 33.07177), (83.64016, 32.90659), (-81.22473, 37.07647), (80.19746, 35.7788), (82.99568, 32.95625), (83.61621, 34.7964), (84.0979, 33.17326), (83.5811, 34.10279), (-76.30694, 35.72018), (-77.56942, 34.25918), (-76.293,
```

32.4642), (-79.97487, 32.16343), (-76.29782, 37.53573), (-78.55127, 37.49596), (-76.18114, 34.6022)]
So that it does not impact your answering the rest questions.

2) Loop through each point in the pointList you get from question one and print out whether the point is in the eastern hemisphere (i.e, longitude is greater than 0). In each iteration, call a function that determines whether a point is in the eastern hemisphere or not. The function takes a point as inputs and return whether it is in the eastern hemisphere (True) or not (False).
The function is structured as

```
def isEasternHemisphere (point):
        #compile your codes here and return True or False
```

You should print something like:
point 1 is not in the eastern hemisphere
point 2 is not in the eastern hemisphere
…….
point 20 is not in the eastern hemisphere

(Undergraduate: 5 points; Graduate: 4 points)

In [267]:
```
#Part3.Q2.2

pointList = [(-76.61815, 36.2883), (-78.77221, 34.04806), (82.86309, 37.2366

def isEasternHemisphere(pointList):
    for point in pointList:
        longitude = point[0]
        if longitude >= 0:
            return ("point is in the eastern hemisphere")
        elif longitude <= 0:
            return ("point is in the western hemisphere")
    return pointList
    isEasternHemisphere(pointList)
pList = isEasternHemisphere(pointList)
print(pList)
```

```
point is in the western hemisphere
```

3) Write a function that finds the eastern most point (the point that has the largest longitude) from the pointList you get from question one.
The function is structured as

```
def findEsternMostPoint (pointList):
        #compile your codes here and return the eastern most point
```

When you call

easternMostPoint = findEsternMostPoint(pointList)
print (easternMostPoint)

You should get

(84.50021, 33.58375)

(Undergraduate: 5 points; Graduate: 4 points)

In [209]:
```
pointList = [(-76.61815, 36.2883), (-78.77221, 34.04806), (82.86309, 37

def findEasternMostPoint(pointList):
    from operator import itemgetter
    max_x = max(pointList, key=itemgetter(0))
    return(max_x)
easternMostPoint = findEasternMostPoint(pointList)
print(easternMostPoint)
```

```
(84.50021, 33.58375)
```

4) Write a function that counts the number of points whose latitude is above a threshold. The function takes the list of points and the threshold as input. The function is structured as

```
def countPoints (pointList, threshold):
        #compile your codes here and return the count
```

When you call

```
count = countPoints (pointList, 35)
print count
```

You should get

7

(Undergraduate: 5 points; Graduate: 4 points)

In [244]:
```
pointList = [(-76.61815, 36.2883), (-78.77221, 34.04806), (82

def countPoint(pointList, threshold):
    latitudes = [item[1] for item in pointList]
    greater = [i for i in latitudes if i > threshold]
    return len(greater)

countPoint(pointList, 35)
```

Out[244]: 7

5) Write a function that separates the list of points you get from question one into two files that store points in the eastern and western. The function takes the list of points you get from question one and the outputs are two files called east.txt and west.txt. The function is structured as

```
def separate (pointList, eastFile, westFile):
        #compile your codes here
```

When you call

```
separate (pointList,"C:/spatial programming/exam/east.txt", "C:/spatial programming/exam/west.txt") # change the file paths in your computer
```

You should get two files (east.txt and west.txt). The screenshots are below.

east.txt

```
82.86309,37.23665
84.50021,33.58375
83.64016,32.90659
80.19746,35.7788
82.99568,32.95625
83.61621,34.7964
84.0979,33.17326
83.5811,34.10279
```

west.txt

```
-76.61815,36.2883
-78.77221,34.04806
-79.62142,34.73552
-79.83924,33.07177
-81.22473,37.07647
-76.30694,35.72018
-77.56942,34.25918
-76.293,32.4642
-79.97487,32.16343
-76.29782,37.53573
-78.55127,37.49596
-76.18114,34.6022
```

(**This question is only for graduate students: 4 points).**

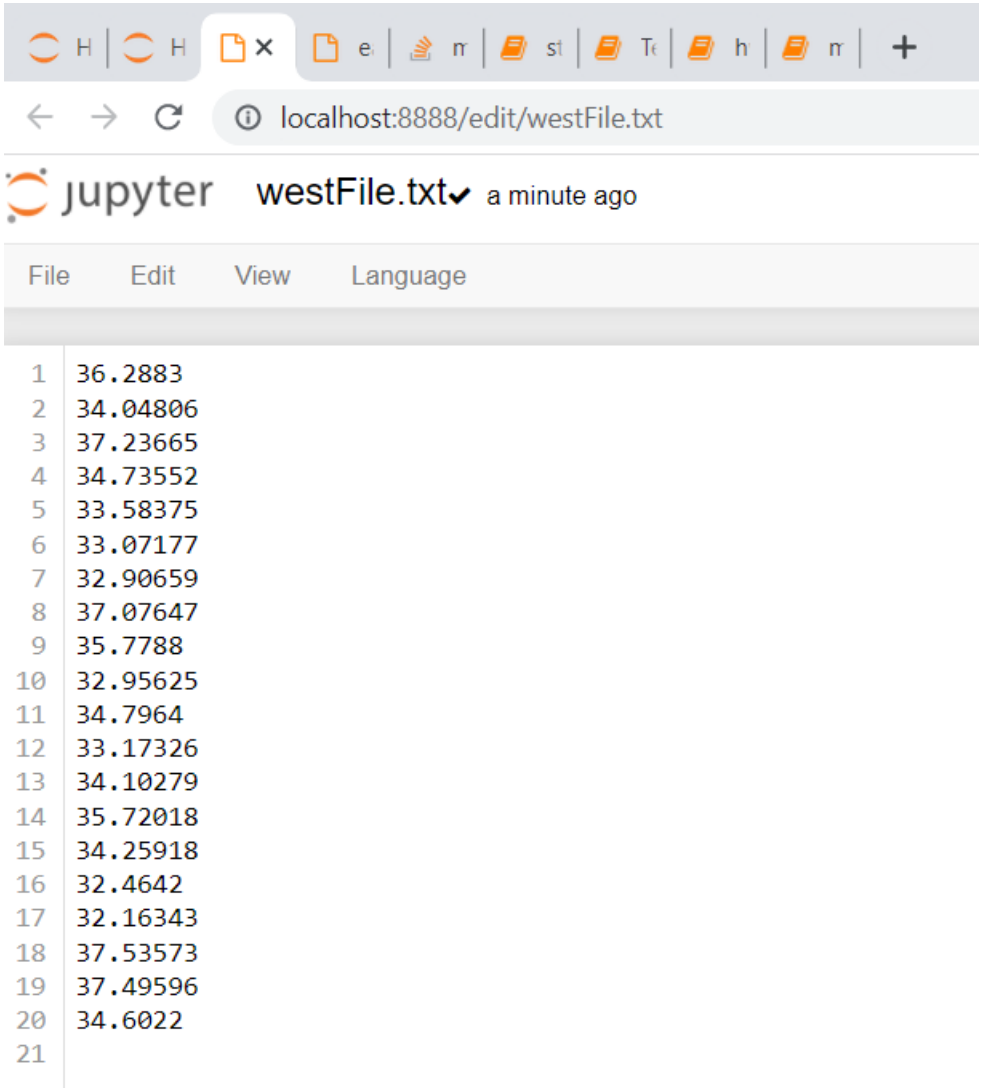
In [240]:

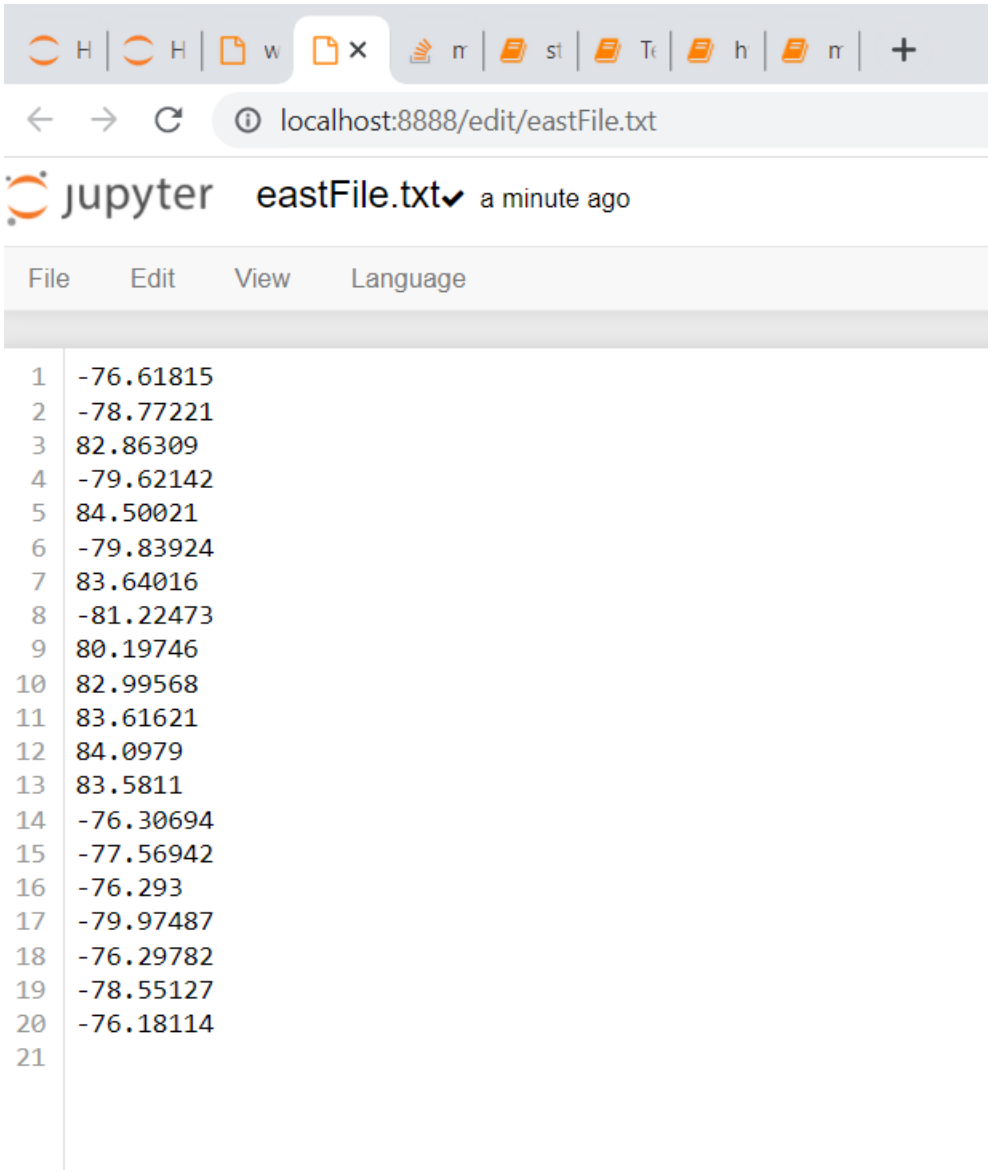
```
pointList = [(-76.61815, 36.2883), (-78.77221, 34.04806), (82

def seperate(pointList, eastFile, westFile):
    longitude = [item[0] for item in pointList]
    latitude = [item[1] for item in pointList]
    with open ('./eastFile.txt', 'w') as f:
        for item in longitude:
            f.write("%s\n" % item)
    with open ('./westFile.txt', 'w') as f:
        for item in latitude:
            f.write("%s\n" % item)
    return (eastFile, westFile)
seperate(pointList, './eastFile.txt', './westFile.txt')

```

Out[240]: ('./eastFile.txt', './westFile.txt')




localhost:8888/edit/eastFile.txt
jupyter
eastFile.txt
a minute ago
File
Edit
View
Language
1 -76.61815
2 -78.77221
3 82.86309
4 -79.62142
5 84.50021
6 -79.83924
7 83.64016
8 -81.22473
9 80.19746
10 82.99568
11 83.61621
12 84.0979
13 83.5811
14 -76.30694
15 -77.56942
16 -76.293
17 -79.97487
18 -76.29782
19 -78.55127
20 -76.18114
21