cycle_4_lecture_notes

February 7, 2021

Plan for this cycle: - look at more examples of reading from files and dealing with data - Start thinking more about data structures - make our first use of a data plotting library

A recap - last cycle, we saw and practiced: - reading data from a file - dealing with errors both *defensively* and using *exceptions* - writing functions for working with a particular data structure

Today, we'll do more of this with messier and more complicated data, and include some visualisation. Part of the goal of looking at the visualisation is to get more experience using a Python **library**

Our data

We want to write a system to deal with commuter information (this example uses fictional data). On each line of our file there will be three pieces of information, separated by commas: `id_number,origin_name,destination_name`

Where - `id_number` is the identifier for an individual person, and - `origin_name` is where they started from in a day's commute, and - `destination_name` is where they went.

Let's call a single line a 'journey'

Here is an example of a lines in a file that we might see:

`90999,Milngavie,Glasgow 892341,Glasgow,Carlisle 892341,Glasgow,Edinburgh`

What would be a good data structure to store this information?

We could talk about a couple different options: - a dictionary with the `id_number` as a key, and a list of tuples as values - a list of 3-tuples - anything else anyone can think of?

Key questions to think about: - Why is a dictionary with an entry per line unsuitable? - Should the id numbers be strings or numbers?

Let's look at an example of the first option:

[ ]:
```
myData = {'90999': [('Milngavie','Glasgow')], '892341': [('Glasgow',␣
↪'Carlisle'), ('Glasgow', 'Edinburgh')]}
```

[ ]:
```
Or an example of the second:
```

[ ]:
```
myData = [('90999','Milngavie','Glasgow'), ('892341','Glasgow','Carlisle'),␣
↪('892341','Glasgow','Edinburgh')]
```

I'm going to proceed with that first option: a dictionary with the `id_number` as a key. and a list of tuples as values

Now that we've picked our data structure, let's work on some functions to process it.

What might we want to do? - read in data - find the list of places a single person commutes to - count the number of journeys between different places - etc

This idea of a data structure and a suite of functions built to operate on it is a fundamental notion that you'll see over and over in computing science - it's the idea that leads to *objects*

We will write a couple functions, and then look at plotting (split over multiple videos).

First, we will write a function to count the total number of journeys recorded for each person, and a function to read in values.

[7]:
```
myData = {'90999': [('Milngavie','Glasgow')],
          '892341': [('Glasgow', 'Carlisle'), ('Glasgow', 'Edinburgh')]}



def readJourneysFromFile(filename):
#     id_num,origin,destination
    dataStruct = {}
    try:
        for line in open(filename, 'r'):
            split = line.strip().split(",")
            if len(split) < 3:
                print('There is a problem with line ' + line.strip())
            else:
                person = split[0]
                if person not in dataStruct:
                    dataStruct[person] = []
                theTuple = (split[1], split[2])
                dataStruct[person].append(theTuple)
        return dataStruct
    except Exception as e:
        print('Failure while reading from ' + filename)
        print('Due to exception ' + str(e))
        return dataStruct




def countJourneysByPerson(dataStruct):
    personToNumber = {}
    for person in dataStruct:
        listOfTrips = dataStruct[person]
        personToNumber[person] = len(listOfTrips)
    return personToNumber


myJourneys = readJourneysFromFile('sampleJourneys.csv')
```

```
# for idNum in myJourneys:
#     print (idNum + ": " + str(myJourneys[idNum]))


counts = countJourneysByPerson(myJourneys)
for count in counts:
     print (count + ": " + str(counts[count]))
```

```
102: [('Glasgow', 'Edinburgh')]
103: [('Glasgow', 'Edinburgh')]
104: [('Glasgow', 'Edinburgh')]
105: [('Glasgow', 'Edinburgh')]
106: [('Glasgow', 'Edinburgh')]
107: [('Glasgow', 'Edinburgh')]
108: [('Glasgow', 'Edinburgh')]
109: [('Glasgow', 'Edinburgh')]
110: [('Glasgow', 'Edinburgh')]
111: [('Glasgow', 'Edinburgh')]
112: [('Glasgow', 'Edinburgh')]
113: [('Glasgow', 'Edinburgh')]
114: [('Glasgow', 'Edinburgh')]
115: [('Glasgow', 'Edinburgh')]
116: [('Glasgow', 'Edinburgh')]
117: [('Glasgow', 'Edinburgh')]
118: [('Glasgow', 'Edinburgh')]
119: [('Glasgow', 'Edinburgh')]
120: [('Glasgow', 'Edinburgh')]
121: [('Glasgow', 'Edinburgh')]
122: [('Glasgow', 'Edinburgh')]
123: [('Glasgow', 'Edinburgh')]
124: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Bearsden')]
125: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Bearsden')]
126: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Bearsden')]
127: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Bearsden')]
128: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Bearsden')]
129: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Bearsden')]
130: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Stirling')]
131: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Stirling')]
132: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Stirling')]
133: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Stirling')]
134: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Stirling')]
135: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Stirling')]
136: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Carlisle'), ('Glasgow', 'Bowling'),
('Glasgow', 'Aberdeen')]
137: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Dumfries'), ('Glasgow', 'Bowling'),
('Glasgow', 'Bridge of Allan')]
138: [('Glasgow', 'Edinburgh'), ('Glasgow', 'Iona'), ('Glasgow', 'Bowling'),
```

```
('Glasgow', 'Bridge of Allan')]
139: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling'), ('Edinburgh',
'Cupar'), ('Edinburgh', 'Castle Douglas'), ('Edinburgh', 'Dunblane')]
140: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling'), ('Edinburgh',
'Cupar'), ('Edinburgh', 'Aberdeen'), ('Edinburgh', 'Dunblane')]
141: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling'), ('Edinburgh',
'Cupar'), ('Edinburgh', 'Musselburgh')]
142: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling'), ('Edinburgh',
'Cupar'), ('Edinburgh', 'Musselburgh')]
143: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
144: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
145: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
146: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
147: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
148: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
149: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
150: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
151: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
152: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
153: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
154: [('Edinburgh', 'Glasgow'), ('Edinburgh', 'Stirling')]
155: [('Edinburgh', 'Glasgow')]
156: [('Edinburgh', 'Glasgow')]
157: [('Edinburgh', 'Glasgow')]
158: [('Edinburgh', 'Glasgow')]
159: [('Edinburgh', 'Glasgow')]
160: [('Edinburgh', 'Glasgow')]
161: [('Edinburgh', 'Glasgow')]
162: [('Edinburgh', 'Glasgow')]
163: [('Edinburgh', 'Glasgow')]
164: [('Edinburgh', 'Glasgow')]
165: [('Edinburgh', 'Glasgow')]
166: [('Edinburgh', 'Glasgow')]
167: [('Edinburgh', 'Glasgow')]
168: [('Edinburgh', 'Glasgow')]
169: [('Edinburgh', 'Glasgow')]
170: [('Edinburgh', 'Glasgow')]
171: [('Edinburgh', 'Glasgow')]
172: [('Edinburgh', 'Glasgow')]
173: [('Edinburgh', 'Glasgow')]
174: [('Edinburgh', 'Glasgow')]
175: [('Edinburgh', 'Glasgow')]
176: [('Edinburgh', 'Glasgow')]
177: [('Edinburgh', 'Glasgow')]
178: [('Edinburgh', 'Glasgow')]
179: [('Edinburgh', 'Glasgow')]
180: [('Edinburgh', 'Glasgow')]
181: [('Edinburgh', 'Glasgow')]
```

```
182: [('Edinburgh', 'Glasgow')]
183: [('Edinburgh', 'Glasgow')]
184: [('Edinburgh', 'Glasgow')]
185: [('Edinburgh', 'Glasgow')]
186: [('Edinburgh', 'Glasgow')]
187: [('Edinburgh', 'Glasgow')]
188: [('Edinburgh', 'Glasgow')]
189: [('Edinburgh', 'Glasgow')]
190: [('Edinburgh', 'Glasgow')]
191: [('Edinburgh', 'Glasgow')]
192: [('Edinburgh', 'Glasgow')]
193: [('Edinburgh', 'Glasgow')]
194: [('Edinburgh', 'Glasgow')]
195: [('Edinburgh', 'Glasgow')]
196: [('Glasgow', 'Milngavie')]
197: [('Glasgow', 'Milngavie')]
198: [('Glasgow', 'Milngavie')]
199: [('Glasgow', 'Milngavie')]
200: [('Glasgow', 'Milngavie')]
201: [('Glasgow', 'Milngavie')]
202: [('Glasgow', 'Milngavie')]
203: [('Glasgow', 'Milngavie')]
204: [('Glasgow', 'Milngavie')]
205: [('Glasgow', 'Milngavie')]
206: [('Glasgow', 'Milngavie')]
207: [('Glasgow', 'Milngavie')]
208: [('Glasgow', 'Milngavie')]
209: [('Glasgow', 'Milngavie'), ('Glasgow', 'Stirling')]
210: [('Glasgow', 'Milngavie'), ('Glasgow', 'Stirling')]
211: [('Glasgow', 'Milngavie'), ('Glasgow', 'Stirling'), ('Glasgow',
'Hamilton')]
212: [('Glasgow', 'Milngavie'), ('Glasgow', 'Stirling'), ('Glasgow',
'Hamilton')]
213: [('Glasgow', 'Milngavie'), ('Glasgow', 'Stirling'), ('Glasgow',
'Hamilton')]
214: [('Glasgow', 'Milngavie'), ('Glasgow', 'Alloa'), ('Glasgow', 'Hamilton')]
215: [('Glasgow', 'Milngavie'), ('Glasgow', 'Alloa'), ('Glasgow', 'Hamilton')]
216: [('Glasgow', 'Milngavie'), ('Glasgow', 'Alloa'), ('Glasgow', 'Livingston')]
217: [('Glasgow', 'Milngavie'), ('Glasgow', 'Alloa'), ('Glasgow', 'Greenock')]
218: [('Glasgow', 'Milngavie'), ('Glasgow', 'Alloa'), ('Glasgow', 'Gourock')]
219: [('Glasgow', 'Milngavie'), ('Glasgow', 'Stirling'), ('Glasgow', 'Dunoon')]
220: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Stirling'), ('Edinburgh',
'Berwick')]
221: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Stirling'), ('Edinburgh',
'Berwick')]
222: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Stirling')]
223: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Stirling')]
224: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Stirling')]
```

```
225: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Alloa')]
226: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Alloa')]
227: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Alloa')]
228: [('Edinburgh', 'Livingston'), ('Edinburgh', 'Alloa')]
229: [('Edinburgh', 'Livingston')]
230: [('Edinburgh', 'Livingston')]
231: [('Edinburgh', 'Livingston')]
232: [('Edinburgh', 'Livingston')]
233: [('Edinburgh', 'Livingston')]
234: [('Edinburgh', 'Livingston')]
: [('', '')]
102: 1
103: 1
104: 1
105: 1
106: 1
107: 1
108: 1
109: 1
110: 1
111: 1
112: 1
113: 1
114: 1
115: 1
116: 1
117: 1
118: 1
119: 1
120: 1
121: 1
122: 1
123: 1
124: 2
125: 2
126: 2
127: 2
128: 2
129: 2
130: 2
131: 2
132: 2
133: 2
134: 2
135: 2
136: 4
137: 4
138: 4
```

139: 5
140: 5
141: 4
142: 4
143: 2
144: 2
145: 2
146: 2
147: 2
148: 2
149: 2
150: 2
151: 2
152: 2
153: 2
154: 2
155: 1
156: 1
157: 1
158: 1
159: 1
160: 1
161: 1
162: 1
163: 1
164: 1
165: 1
166: 1
167: 1
168: 1
169: 1
170: 1
171: 1
172: 1
173: 1
174: 1
175: 1
176: 1
177: 1
178: 1
179: 1
180: 1
181: 1
182: 1
183: 1
184: 1
185: 1
186: 1

```
187: 1
188: 1
189: 1
190: 1
191: 1
192: 1
193: 1
194: 1
195: 1
196: 1
197: 1
198: 1
199: 1
200: 1
201: 1
202: 1
203: 1
204: 1
205: 1
206: 1
207: 1
208: 1
209: 2
210: 2
211: 3
212: 3
213: 3
214: 3
215: 3
216: 3
217: 3
218: 3
219: 3
220: 3
221: 3
222: 2
223: 2
224: 2
225: 2
226: 2
227: 2
228: 2
229: 1
230: 1
231: 1
232: 1
233: 1
234: 1
```

: 1

Let's think about another couple functions we could write: 1. a function that counts the number of people who have done each unique journey 2. a function that counts the number of journeys that end in each destination

[4]:
```
# we will assume each person does each journey at most once
def countPeoplePerJourney(dataStruct):
#     map from journey (tuples) to a count
#     keys: pairs, values integers
    countDict = {}
    for person in dataStruct:
        journeys = dataStruct[person]
        for journey in journeys:
            if journey not in countDict:
                countDict[journey] = 0
            countDict[journey] = countDict[journey] + 1
    return countDict


def countJourneysPerDestination(dataStruct):
#     map from destinations to a count
#   keys: string, values integers
    countDict = {}
    for person in dataStruct:
        journeys = dataStruct[person]
        for journey in journeys:
            (origin, destination) = journey
            if destination not in countDict:
                countDict[destination] = 0
            countDict[destination] = countDict[destination] + 1
    return countDict



# Important question: how might we add error-checking or exception-catching to␣
 ↪our functions above?

myData = {'90999': [('Milngavie','Glasgow')],
          '892341': [('Glasgow', 'Carlisle'), ('Glasgow', 'Edinburgh')],
          '909679': [('Milngavie','Glasgow')]}

print(countPeoplePerJourney(myData))
print(countJourneysPerDestination(myData))
```

```
{('Milngavie', 'Glasgow'): 2, ('Glasgow', 'Carlisle'): 1, ('Glasgow',
'Edinburgh'): 1}
{'Glasgow': 2, 'Carlisle': 1, 'Edinburgh': 1}
```

Important extra task: think about errors and exceptions - what else could you add?

But for the moment, let's look at plotting. We are going to use a library called matplotlib for plotting.

https://matplotlib.org/

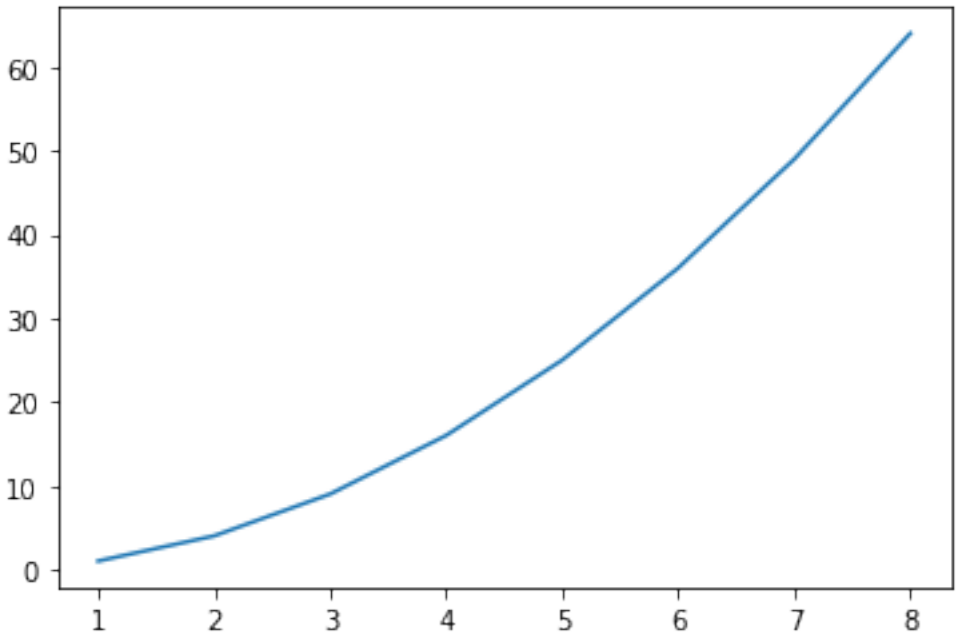
A big part of the reason for this: I want you to practice using an existing library, and understanding documentation.

Let's look at a basic example of a scatterplot in matplotlib:

[10]:
```
import matplotlib.pyplot as plt

xVals = [1, 2, 3, 4, 5, 6, 7, 8]
yVals = [1, 4, 9, 16, 25, 36, 49, 64]

plt.plot(xVals, yVals)
plt.show()
```



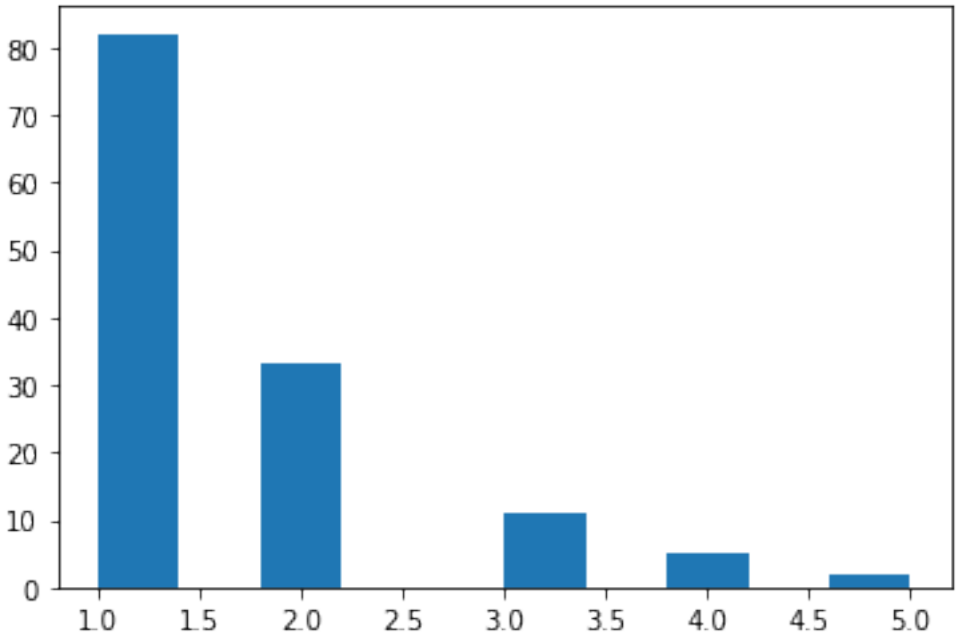
There are loads of different plotting styles possible using matplotlib! Feel free to have look and try things out.

We'll have a quick look at **histograms** and **scatterplots**

Let's use the histogram function to plot a histogram of the number of journeys for each person. We already have a function to get the count of number of different journeys for each person!

[26]:
```
# this gets us just the values from the dictionary -
#here, it will be just the counts of journeys, not the ids
justCounts = counts.values()

plt.hist(justCounts)
plt.show()
```

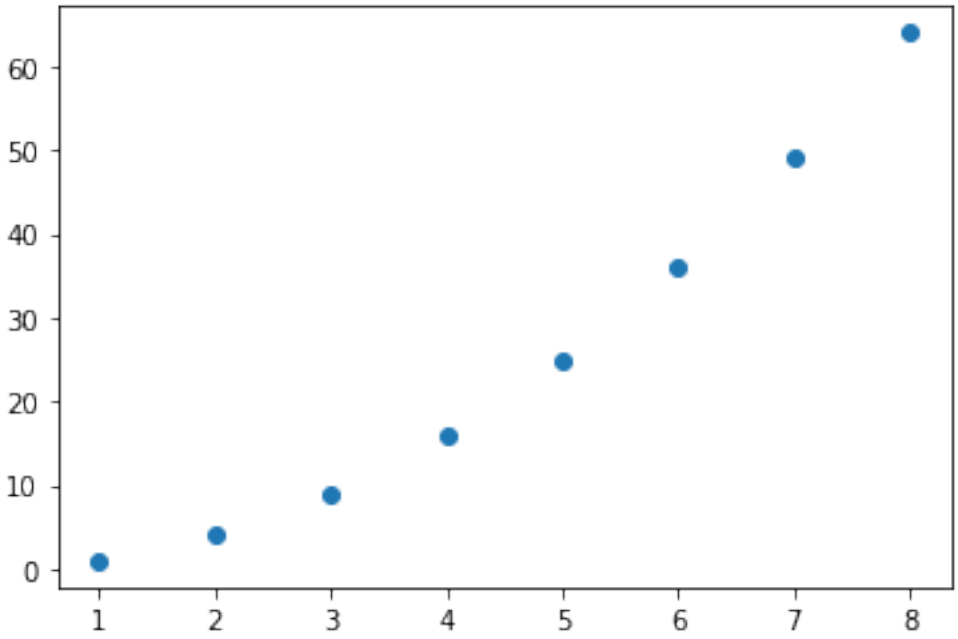


Now let's look at a scatterplot. We can find the docs for `matplotlib`'s `scatter` here: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html

(Note the many pretty examples near the bottom of the docs page)

First, let's do a generic example with our `xVals` and `yVals` from before:

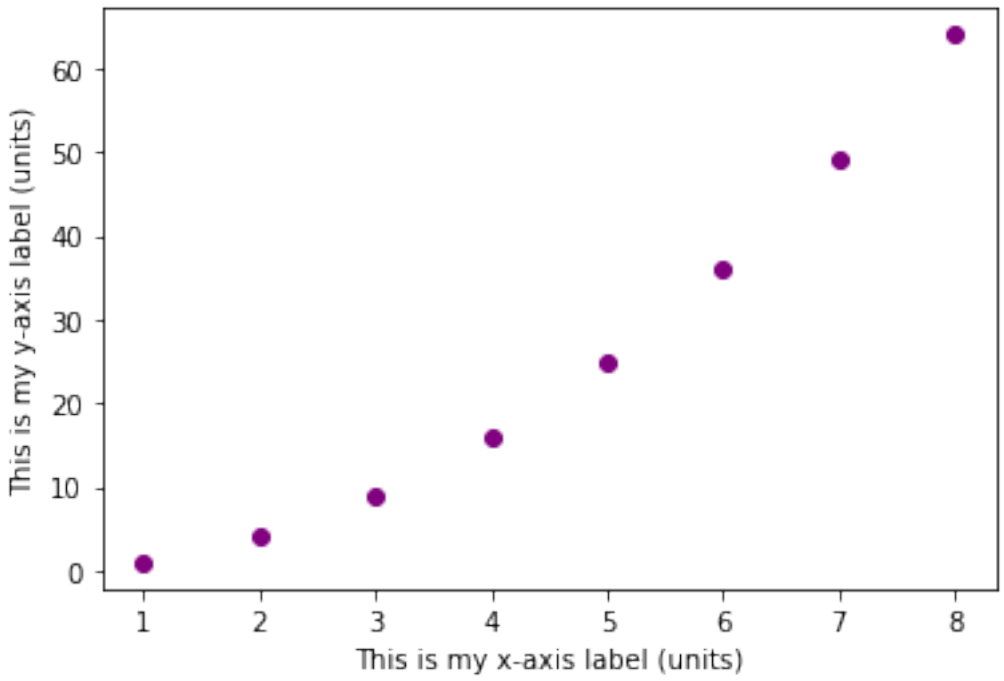
[11]:
```
plt.scatter(xVals, yVals)
plt.show()
```



It's good practice to have x and y axis labels, so we should usually add those!

We may also want to change the colour or the shape of our markers. There are many, many possibilities of how to make our plots more informative and beautiful. I encourage you to have a look around the matplotlib documentation - you'll have a good change to use some of your plotting skills in your assessed assignment.

[12]:
```
plt.scatter(xVals, yVals, color = 'purple')
plt.xlabel('This is my x-axis label (units)')
plt.ylabel('This is my y-axis label (units)')
plt.show()
```


This is my y-axis label (units)
0
10
20
30
40
50
60
1
2
3
4
5
6
7
8
This is my x-axis label (units)

[ ]: