Reading Text Files

Atmospheric Sciences Relevance

Most of the time any data that we work with will come from some sort of a data file. There are a few ways that files can be opened and worked with depending on how they are structured. For tabular data, files with data seperated by commas, the easiest way is with pandas. For gridded data, multi-dimensional data organized with metadata and spatial coordinates the easiest thing to use is numpy. Today we will be using the built in ***open()*** function.

The data that we are working with today is actual radiosonde data I collected in Ocean City, Maryland in January 2025. This particular radiosonde was in the air for ~90 minutes and got to ~40 mb before it lost connection. It was cold and exteremly windy that day so you better appreciate this data cause it was a pain to get.

I've started some code for you that will open the file, and put each line into a list within the list ***data_list***. Two lines are partially there that you will need to change to get to work. Before you change anything, run what I've given you to see what you get then answer the first question.

In [ ]:
```
# this is just the file name you dont need to change that
file_name = 'OCC01_1700UTC_010625_UND01_profile.tsv'
data = open (file_name, 'r')

data_list = []


#this for loop puts each line of the file into the list data_list
for line in data:
    line = line.strip()
    # line = line.replace('\t', ',') #uncomment
    # line = ???.split(',') #uncomment and fix this line
    data_list.append(line)



# this will print the first 20 indicies of data_list
print(data_list[:20])
```

1. Copy and paste two lines of output below. What do you notice about the output? Do you think you could use it as is right now?

In [ ]:
```
No it is hard to read and decifer. ['Time [sec]\tP [hPa]\tT [oC]\tWs [m/s]\tWd [o]\tAlt [m]\tLon [o]\tLat [o]\tDew
[oC]', '0\t1009.3\t4.5\t1.5\t30\t30\t38.332900\t-75.089120\t1.7', '1\t1009.2\t2.1\t3.8\t235\t31\t-75.089025\t38.332695\t-1.1', '2\t1009.1\t1.7\t3.8\t235\t32\t-75.089093\t38.332656\t-1.3', '3\t1009.0\t1.7\t3.8\t235\t33\t-75.089161\t38.332616\t-0.7', '4\t1008.8\t1.7\t3.8\t235\t34\t-75.089229\t38.332577\t-0.2', '5\t1008.7\t1.7\t3.8\t235\t35\t-75.089297\t38.332537\t0.2', '6\t1008.6\t1.7\t1.9\t235\t36\t-75.089365\t38.332498\t0.5', '7\t1008.3\t1.7\t7.3\t55\t36\t-75.089433\t38.332458\t0.8', '8\t1008.0\t1.7\t7.3\t55\t37\t-75.089501\t38.332419\t0.9', '9\t1007.7\t1.7\t7.3\t55\t39\t-75.089569\t38.332379\t0.9', '10\t1007.4\t1.6\t7.3\t55\t43\t-75.089636\t38.332340\t1.0', '11\t1007.1\t1.6\t7.3\t55\t46\t-75.089704\t38.332300\t1.0', '12\t1006.8\t1.6\t7.3\t55\t50\t-75.089772\t38.332261\t1.0', '13\t1006.2\t1.5\t7.3\t55\t53\t-75.089840\t38.332221\t1.0', '14\t1005.7\t1.5\t7.3\t55\t57\t-75.089908\t38.332182\t0.9', '15\t1005.1\t1.5\t7.3\t55\t61\t-75.089985\t38.332157\t0.8', '16\t1004.5\t1.4\t7.3\t55\t67\t-75.090062\t38.33213
```

Now that you've answered question one go back and update the code.

2. Copy and Paste two lines of output below. What do you notice about the output? Do you think you could use it as is now?

In [ ]:
```
Yes, because it is easier to read and decifer. [['Time [sec]', 'P [hPa]', 'T [oC]', 'Ws [m/s]', 'Wd [o]', 'Alt [m]', 'Lon [o]', 'Lat [o]', 'Dew [oC]'],
['0', '1009.3', '4.5', '1.5', '30', '30', '38.332900', '-75.089120', '1.7'],
['1', '1009.2', '2.1', '3.8', '235', '31', '-75.089025', '38.332695', '-1.1'],
['2', '1009.1', '1.7', '3.8', '235', '32', '-75.089093', '38.332656', '-1.3'],
['3', '1009.0', '1.7', '3.8', '235', '33', '-75.089161', '38.332616', '-0.7'],
['4', '1008.8', '1.7', '3.8', '235', '34', '-75.089229', '38.332577', '-0.2'],
['5', '1008.7', '1.7', '3.8', '235', '35', '-75.089297', '38.332537', '0.2'],
['6', '1008.6', '1.7', '1.9', '235', '36', '-75.089365', '38.332498', '0.5'],
['7', '1008.3', '1.7', '7.3', '55', '36', '-75.089433', '38.332458', '0.8'],
['8', '1008.0', '1.7', '7.3', '55', '37', '-75.089501', '38.332419', '0.9'],
['9', '1007.7', '1.7', '7.3', '55', '39', '-75.089569', '38.332379', '0.9'],
['10', '1007.4', '1.6', '7.3', '55', '43', '-75.089636', '38.332340', '1.0'],
['11', '1007.1', '1.6', '7.3', '55', '46', '-75.089704', '38.332300', '1.0'],
['12', '1006.8', '1.6', '7.3', '55', '50', '-75.089772', '38.332261', '1.0'],
['13', '1006.2', '1.5', '7.3', '55', '53', '-75.089840', '38.332221', '1.0'],
['14', '1005.7', '1.5', '7.3', '55', '57', '-75.089908', '38.332182', '0.9'],
```

Now that our code is usable, lets sperate it into more meaningful lists so that we can actually work with the data.

First we need to figure out what values corresponde with what variables.

3. Use list indexing to figure out where each variable is. Write the index of each variable below (i.e. "time is at index __")

In [ ]:
```
# write code to find the index of each variable here
data_list[0]
time is at index 0
pressure is at index 1
temp 2
```

Now that we know where each varible is, lets seperate each variable into its own list. I have started a for loop for you. Some lines of code are unfinished. Finish them so that the code works correctly and you only have the values in the list, no variable name.

In [ ]:
```
press_list = []
temp_list = []
lon_list = []
time_list = []
winddir_list = ???
dew_list = ???
 # missing windspeed list
 # missing latitude list
 # missing altitude list

for i in range(len(???)):


    time_list.append(data_list[i][0])
    temp_list.append(data_list[i][???])
    lon_list.append(data_list[???][6])
    press_list.append(data_list???)
    altitude_list.append(???[i][???])
    dew_list.append(???)
    windspeed_list.append(???)
    lat_list.append(???)
    winddir_list.append(???)
```

Now that we have added the data to meaningful lists, let's make sure they look like they make sense.

4. Write code to print the first 13 values of each list. Copy and paste the output below.

In [ ]:
```
print(temp_list[:13])
print(time_list[???])
print(lon_list[???])
print(???)
print(???)
print(???)
print(???)
print(???)
print(???)

#paste out put here
```
[['Time [sec]', 'P [hPa]', 'T [oC]', 'Ws [m/s]', 'Wd [o]', 'Alt [m]', 'Lon [o]', 'Lat [o]', 'Dew [oC]'], ['0', '1009.3', '4.5', '1.5', '30', '30', '38.332900', '-75.089120', '1.7'],
['1', '1009.2', '2.1', '3.8', '235', '31', '-75.089025', '38.332695', '-1.1'],
['2', '1009.1', '1.7', '3.8', '235', '32', '-75.089093', '38.332656', '-1.3'],
['3', '1009.0', '1.7', '3.8', '235', '33', '-75.089161', '38.332616', '-0.7'],

Now we get to start playing with the data. This is the fun part! We'll start with something we've done a few times now, averaging the values inside the list.

5. Write a function, ***average()***, that takes the argument ***list_name*** and finds the average value in the box below. Do not print in your function.

In [ ]:
```
def average(list_name):
    # fill in the function
    return avg
```

Now that we have a function lets use it to find the averages. There are 5 lists that it makes sense to find the averages for. Decide which lists to find the average for and find it below.

6. List the variables you found averages for and the average for that variable.

set it up like this -- > variable_name = average(variable_list)

In [ ]:
```
#now print the variables that you found averages for
def average(list_name):
    total=0
    for i in range (len(list_name)):
        total=total+float(list_name[i])
    avg=total/len(list_name)

    return avg
temp_average=average(temp_list)
dew_average=average(dew_list)
press_average=average(press_list)
windspeed_average=average(windspeed_list)
```

7. Do the averages you got make sense? Explain your answer.

In [ ]:
No not all of them because a wind speed of over 30,000 would be insane.

8. When data is being collected, sensors might not work the entire time. In this case, if the sensor missed a measurement, it is saved as 999999. To make wind speed and wind direction make sense, we need to loop through the list and ignore the value if it is 999999. Write a function, ***clean_list()*** that tests for bad values and ignores them if present. I'll let you decide what parameters the function should take.

In [ ]:
```
def clean_list(???):
    new_list = []
    for ???:
        if ???[i] == ???:
            pass
        else:
            new_list.append(???)
    return new_list

new_winddir_list = clean_list(???)
new_windspeed_list = clean_list(???)
```

In [ ]:
```
print(average(new_winddir_list))
print(average(new_windspeed_list))
```

Let's also find the the maximum and minimum value for each list. For this we can do it on all the lists except latitude and longitude. We'll focus on finding the max first.

9. Complete the ***maximum()*** function below.

In [ ]:
```
def maximum(list_name):
    max_val = ???
    for i in ???:
        if list_name[i] ???    ???:
            ??? = list_name[i]
    return ???
```

Once you've completed the maximum function run the code below.

10. List which variables have max values that make sense and which ones don't make sense. If you get a ValueError that's ok. Think about what could have caused that and how to fix it.

In [ ]:
```
temp_max = maximum(temp_list)
print('Temp:',??? )

dew_max = maximum(dew_list)
print('Dew:', ??? )
```

```
press_max = maximum(???)
print('Press:', press_max )

windspeed_max = ???(windspeed_list)
print('Windspeed:', windspeed_max)

time_max = ???
print('Time:', time_max )

altitude_max = maximum(altitude_list) # don't worry about an error her
print('Alt:',altitude_max)
```

You should have gotten an error that looks like this "ValueError: could not convert string to float: 'M'".

11. What list did this occur with and what could 'M' stand for?

In [ ]:
Altitude list. M could stand for reaching the missing value.

12. We need to filter out any 'M' from the list. We already have a ***clean_list()*** function that filters out 999999's. Edit ***clean_list()*** to get rid of any 'M' as well as any 999999.

In [ ]:
```
def clean_list(???):
    new_list = []
    for ???:
        if ???[i] == ???:
            pass
        else:
            new_list.append(???)
    return new_list
```

13. Now that ***clean_list()*** handles both missing value place holders, edit ***maximum()*** to call ***clean_list()*** and calculate the maximum after cleaning the list.

In [ ]:
```
def maximum(list_name):
    max_val = ???
    cln_list = ???
    for i in range(len(???)):
        if cln_list[i] > ???:
            max_val = ???
    return max_val
```

In [ ]:
```
temp_max = maximum(temp_list)
print('Temp:',temp_max )

dew_max = maximum(dew_list)
print('Dew:', dew_max )

press_max = maximum(press_list)
print('Press:', press_max )

windspeed_max = maximum(windspeed_list)
print('Windspeed:', windspeed_max)

time_max = maximum(time_list)
print('Time:', time_max )

altitude_max = maximum(altitude_list)
print('Alt:',altitude_max)
```

14. Do these values make more sense now that we have gotten rid of the missing value place holders?

In [1]:
Yes, because even the altitude is now a number (22869.0). All values all make sense now.

15. Now we can find the minimum. Using the lessons we learned from finding the maximum, write a function ***minimum()*** that finds the smallest value of a variable.

In [ ]:
```
def minimum(list_name):
        #write the function
    return min_val
```

In [ ]:
```
temp_min = minimum(temp_list)
print('Temp:',temp_min )
dew_min = minimum(dew_list)
print('Dew:', dew_min )
press_min = minimum(press_list)
print('Press:', press_min )
windspeed_min = minimum(windspeed_list)
print('Windspeed:', windspeed_min)
time_min = minimum(time_list)
print('Time:', time_min )
altitude_min = minimum(altitude_list)
print('Alt:',altitude_min)
```

16. Do all of these values make sense and seem resonable?

Yes, the values do make sense and seem reasonable (for all minimum values).