Programming Assignment #1

Due: 2/19/23 @ 11:59pm

Assignment Link: https://classroom.github.com/a/tYtoMzMD

Please read through the entire writeup before beginning the programming assignment

Objectives

1. Review/Learn Scala basics
    a. Where to find / how to use Scala Docs
    b. How to edit / compile/debug code in Scala
2. Get comfortable working with large datasets
    a. Reading / processing data read from a CSV
    b. Diagnosing / debugging problems
    c. Breaking large problems into smaller, more digestible problems

Useful Links

1. The Scala API
    a. Scala Collections
    b. scala.collections.mutable.Map
2. Scala Tour
3. Scala Resources
4. ScalaTest: Writing your first test
5. Maps in Scala
6. Scala File I/O (Scala Cookbook Excerpt)
7. Scala Exercises

Submission Process, Late Policy and Grading

Submissions will open on Autolab by Sunday 02/12/23 by 11:59PM
Due date: 2/19/23 @ 11:59PM
Total points: 30

The project grade is the grade assigned to the latest (most recent) submission made to Autolab (or 0 if no submissions are made). Autolab will pull your submission from your GitHub repository, so you must make sure that any changes you want to be included in your grade have been **committed and pushed.**

- If your submission is made before the deadline, you will be awarded 100% of the points your project earns.
- If your submission is made up to 24 hours after the deadline, you will be awarded 75% of the points your project earns.
- If your submission is more than 24 hours after the deadline, but within 48 hours of the deadline, you will be awarded 50% of the points your project earns.
- If your submission is made more than 48 hours after the deadline, it will not be accepted.

You will have the ability to use three grace days throughout the semester, and at most two per assignment (since submissions are not accepted after two days). Using a grace day will negate the 25% penalty per day, but will not allow you to submit more than two days late. Please plan accordingly. You will not be able to recover a grace day if you decide to work late and your score is not sufficiently higher. Grace days are automatically applied to the first instances of late submissions, and are non-refundable. For example, if an assignment is due on a Friday and you make a submission on Saturday, you will automatically use a grace day, regardless of whether you perform better or not. Be sure to test your code before submitting, especially with late submissions in order to avoid wasting grace days.

Keep track of the time if you are working up until the deadline. Submissions become late after the set deadline. Keep in mind that submissions will close 48 hours after the original deadline and you will not be able to submit your code after that time.

Setup

In order to complete this project, you must have completed PA0. If you are working on a machine other than the one you used in PA0, you must at least complete steps 2 and 4 in order to get IntelliJ and GitHub working properly.

Once you have ensured your development environment is setup as in PA0, you can accept the PA1 assignment in GitHub Classroom (here), and create a new IntelliJ project from VCS with your newly created repository.

Instructions

In this PA you will be filling in the body of 4 functions to solve 4 different problems related to reading and parsing a CSV file, and computing certain properties from the read in data.

After you complete your coding, make sure to commit **and push** your work to GitHub, and submit PA1 in Autolab.

Hint: It is advised that you commit and push frequently rather than waiting until you've completed everything.

Hint: Although you will get feedback from Autolab about correctness of your solutions, you can also run the provided tests locally to get a similar level of feedback without the hassle of submitting to Autolab. Furthermore, augmenting the provided test cases with test cases of your own can help you find and squash issues proactively, and will be more targeted and informative than the provided tests and tests on Autolab.

0. Download NYS Open Data - Solar Installation Sites

We will be making use of a public dataset released by the NYS energy department (NYSERDA) of solar energy installation sites. You can download the dataset at the NYS Open Data Portal by clicking on **Export → CSV**. After you download it, place the resulting file in the data directory of your repository.

Your task with this dataset will be to sanitize and summarize the data file. There are a number of columns that are not of interest to us, so we will create an updated data file without these columns, while also obtaining some summary statistics.

Note: Although the specific tasks you will perform in this assignment are simplified to make them viable in the time allotted for the project, they are representative of common data processing tasks used for data exploration, visualization, and transformation, as well as for machine learning.

1. Define `splitArrayToRowArray` [15/30 points]

In the object `cse250.pa1.DataProcessor` define the Scala function:

```
splitArrayToRowArray(splitHeaderRow: Array[String]): Array[String]
```

with the following behavior:

- Assume that `splitHeaderRow` is the result from taking some line from the Solar Installations dataset and invoking `split(',')`.
- Given `splitHeaderRow`, place the data into an Array corresponding to the columns that would result from opening the original dataset file with a spreadsheet application.

Note that every row processed should produce a return result that contains the same number of column entries as the header row for the document. This means that each row, even if there are empty cells, should return an Array with 44 columns (even if some are empty strings).

Hint: review the documentation for the `split` method for the cases where there are empty entries in a row. Hint: Be mindful of rows that contain cells with commas (see the CSV representation rules below).

A good way to test this functionality is to ensure that the first row of the dataset, which contains the header, should return a copy of the row. The second row of the dataset, which contains successive blank entries, should still return a row with 44 entries, but should have some empty values.

2. Define `rowArrayToSolarInstallation` [5/30 points]

In the object `cse250.pa1.DataProcessor` define the Scala function:

```
rowArrayToSolarInstallation(rowArray: Array[String]): SolarInstallation
```

with the following behavior:

- Assume that the input `rowArray` is an Array containing 44 entries, corresponding to a row that was **correctly** processed through `splitArrayToRowArray`.
- Return a new `SolarInstallation` object that corresponds to the data in `rowArray`.

Note that `SolarInstallation` is only meant to hold a limited number of headers from the dataset. The headers that are required to be present are stored in the `Seq` named `SolarInstallation.REQUIRED_HEADERS`. A full list of all headers is stored in the `Seq` named `SolarInstallation.HEADERS`.

When you are finished, a `SolarInstallation` should contain exactly 29 entries (one piece of data associated with each header). This will cause the resulting updated output file after running the given code to contain exactly 29 columns in each row. See more on the `SolarInstallation` Objects section below.

3. Define `computeUniqueCities` [5/30 points]

In the object `cse250.pa1.DataProcessor` define the Scala function:

```
computeUniqueCities(dataset: Array[SolarInstallation]): Int
```

that determines the number of unique cities (corresponding to the City column) represented in the dataset. You should ignore any empty entries from your count, as well as the header.

4. Define `computeAverageCostForCity` [5/30 points]

In the object `cse250.pa1.DataProcessor` define the Scala function:

```
computeAverageCostForCity(
      dataset: Array[SolarInstallation], city: String): Double
```

that determines the average project cost of all projects corresponding to the passed in city. This corresponds to total project cost (corresponding to the Project Cost column) of all projects for that city divided by the total number of projects for that city.

Note that your answer produced should make sense, so you should assume that the cost of each project is positive (ignore projects that cost less than or equal to $0). If no valid data is found, you should return 0.0 as the average cost.

Additional Notes

The `SolarInstallation` Object

To represent a single data record, you must use the structure `cse250.objects.SolarInstallation` provided in the code skeleton.

```
/**
 * One specific solar installation site.
 */
class SolarInstallation
{
  /**
   * Key-value pairs representing data about the solar site.
   * See [[SolarInstallation.HEADERS]] for a list of allowable keys.
   * See [[SolarInstallation.REQUIRED_HEADERS]] for a list of mandatory keys.
   */
  val fields: mutable.Map[String, String] = new mutable.HashMap[String, String]
```

Note that the file containing `SolarInstallation` will be overwritten when your code is graded, so any changes you make will be ignored.

The information stored within a `SolarInstallation` should be stored in the fields attribute, which stores (key → value) pairs, where the header for the column (value in the respective column in the first row) is the key and the value is the data found within the row in the corresponding column. For example, the first installation (second row of the file) should be loaded as:

Key	Value
`Reporting Period`	`1/31/2023`
`Project Number`	`534735`
`Street Address`	
`City`	`Albany`
`Municipality Type`	`Town`
…	…
`Georeference`	`POINT (-73.8100987 42.7056536)`

Note: Street Address is an empty string, but is still in the `SolarInstallation` object.
Note: Not all fields are present, only those that are in `SolarInstallation.REQUIRED_HEADERS`

The `Main` Object

The `Main` object is the entry point for the program, and is what gets executed when you run the project in IntelliJ. As with `SolarInstallation.scala`, `Main.scala` will be replaced by Autolab during testing, so any changes you make will be ignored. However, it can be useful to read through `Main.scala` to get a better idea of the overall context in which the functions you define are being used.

The `DataProcessorTests` Object

The `DataProcessorTests` object is where the local test cases for this PA are defined. These are the tests that are run when you execute the test configuration in IntelliJ. You are encouraged to add your own additional test cases here to help debug issues you run into during development.

CSV Formatting

The formatting for the data file is CSV (comma-separated values). CSV files are a way to represent columns of data by separating entries within a row by a comma. As such, to explore a CSV file, you can open it in any spreadsheet application (like Excel) to see how each row is broken into columns. The goal of this lab is to break up each line in a CSV file into the corresponding columns it represents. The `Main` object calls `.split(",")` on each line, however this does not necessarily result in the split we want, and it is your job to fix it.

Two special cases arise that you must handle:

- If a cell contains a comma (,) within, the cell contents are enclosed in double quotes (") at the start and end. For example: Comma, Cell would be stored as "Comma, Cell".
- If a cell contains a double quote (") within, the cell contents are enclosed in double quotes (") at the start and end, and each double quote in the cell data is duplicated. For example, The "Best" Around would be stored as "The ""Best"" Around"

It is also possible that a cell contains both special cases. Note a data file containing the line:

```
First Cell,Second Cell,"The ""Best"" Around","Comma, Cell","""Object-Orientation, Abstraction, and Data Structures Using Scala"""
```

Corresponds to the following columns (ie if you opened it in Excel):

1	2	3	4	5
First Cell	Second Cell	The "Best" Around	Comma, Cell	"Object-Orientation, Abstraction, and Data Structures Using Scala"

Hint: If you run `.split(",")` on the above example, it will help demonstrate some of short-comings of using split that you will need to fix in your solution.

Other Tips

1. Working with such a large dataset can be a bit unwieldy and overwhelming at first. Feel free to use Excel or another similar program to create a smaller version of the file (i.e. only containing 10 or 100 rows). Try to get in the habit of testing/developing on smaller versions of a problem that are easier to reason about. Then once you are confident your solution works on the smaller problem, you can try it out on the full problem.
2. Write your own tests and examples. If there's a particular corner case you are struggling on, or are unsure how it would work, make a test that specifically focuses on that exact situation. If needed, modify the CSV to contain a row that tests the specific issue you are concerned with. You will be much more productive if you can isolate the problem you are working on to something simple.
3. Plan out/diagram your solution first. Come up with specific examples of interesting CSV rows (like the one shown above), and the expected output. Work through manually the steps you need to take to convert the input into the expected output before writing code.

Academic Integrity

As a gentle reminder, please re-read the academic integrity policy of the course. I will continue to remind you throughout the semester and hope to avoid any incidents.

What Constitutes a Violation of Academic Integrity?

These bullets should be obvious things not to do (but commonly occur):

- Turning in your friend’s code/write-up (obvious).
- Turning in solutions you found on Google with all the variable names changed (should be obvious). This is a copyright violation, in addition to an AI violation.
- Turning in solutions you found on Google with all the variable names changed and 2 lines added (should be obvious). This is also a copyright violation.
- Paying someone to do your work. You may as well not submit the work since you will fail the exams and the course.
- Posting to forums asking someone to solve the problem.

Note: Aggregating every [stack overflow answer|result from google|other source] because you "understand it" will likely result in full credit on assignments (if you aren't caught) and then failure on every exam. Exams don't test if you know how to use Google, but rather test your understanding (i.e., can you understand the problems to arrive at a solution on your own). Also, other students are likely doing the same thing and then you will be wondering why 10 people that you don’t know have your solution.

Other violations that may not be as obvious:

- Working with a tutor who solves the assignment with you. If you have a tutor, please contact me so that I may discuss with them what help is allowed.
- Sending your code to a friend to help them. If another student uses/submits your code, you are also liable and will be punished.
- Joining a chatroom for the course where someone posts their code once they finish, with the honor code that everyone needs to change it in order to use it.
- Reading your friend's code the night before it is due because you just need one more line to get everything working. It will most likely influence you directly or subconsciously to solve the problem identically, and your friend will also end up in trouble.

What Collaboration is Allowed?

Assignments in this course should be solved individually with only assistance from course staff and allowed resources. You may discuss and help one another with technical issues, such as how to get your compiler running, etc.

There is a gray area when it comes to discussing the problems with your peers and I do encourage you to work with one another to solve problems. That is the best way to learn and overcome obstacles. At the same time you need to be sure you do not overstep and not plagiarize. Talking out how you eventually reached the solution from a high level is okay:

> "I used a stack to store the data and then looked for the value to return."

but explaining every step in detail/pseudocode is not okay:

> "I copied the file tutorial into my code at the start of the function, then created a stack and pushed all of the data onto the stack, and finished by popping the elements until the value is found and use a return statement."

The first example is OK but the second is basically a summary of your code and is not acceptable, and remember that you shouldn't be showing any code at all for how to do any of it. Regardless of where you are working, you must always follow this rule: Never come away from discussions with your peers with any written work, either typed or photographed, and especially do not share or allow viewing of your written code.

What Resources are Allowed?

With all of this said, please feel free to use any [files|examples|tutorials] that we provide directly in your code (with proper attribution). Feel free to directly use anything from lectures or recitations. You will never be penalized for doing so, but should always provide attribution/citation for where you retrieved code from. Just remember, if you are citing an algorithm that is not provided by us, then you are probably overstepping.

More explicitly, you may use any of the following resources (with proper citation/attribution):

- Any example files posted on the course webpage (from lecture or recitation).
- Any code that the instructor provides.
- Any code that the TAs provide.
- Any code from the tour of Scala (https://docs.scala-lang.org/tour/tour-of-scala.html)
- Any code from Scala Collections (https://docs.scala-lang.org/overviews/collections-2.13/introduction.html)
- Any code from Scala API (https://www.scala-lang.org/api/2.13.0/)

Omitting citation/attribution will result in an AI violation (and lawsuits later in life at your job). This is true even if you are using resources provided.

Amnesty Policy

We understand that students are under a lot of pressure and people make mistakes. If you have concerns that you may have violated academic integrity on a particular assignment, and would like to withdraw the assignment, you may do so by sending me an email BEFORE THE VIOLATION IS DISCOVERED BY ME. The email should take the following format:

Dear Dr. Mikida,

I wish to inform you that on assignment X, the work I submitted was not entirely my own. I would like to withdraw my submission from consideration to preserve academic integrity.

J.Q. Student
Person #12345678
UBIT: jqstuden

When we receive this email, student J would receive a 0 on assignment X, but would not receive an F for the course, and would not be reported to the office of academic integrity.