Lecture 2: Dealing with Data

COSC 526: Introduction to Data Mining

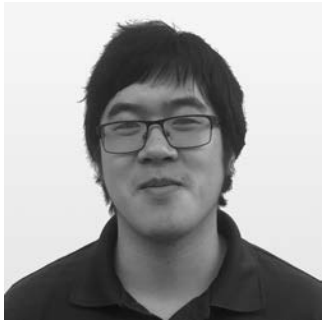
THE UNIVERSITY OF
TENNESSEE
KNOXVILLE

Instructors:

Michela Taufer

Assistants to the Instructor:

Ian Lumsden



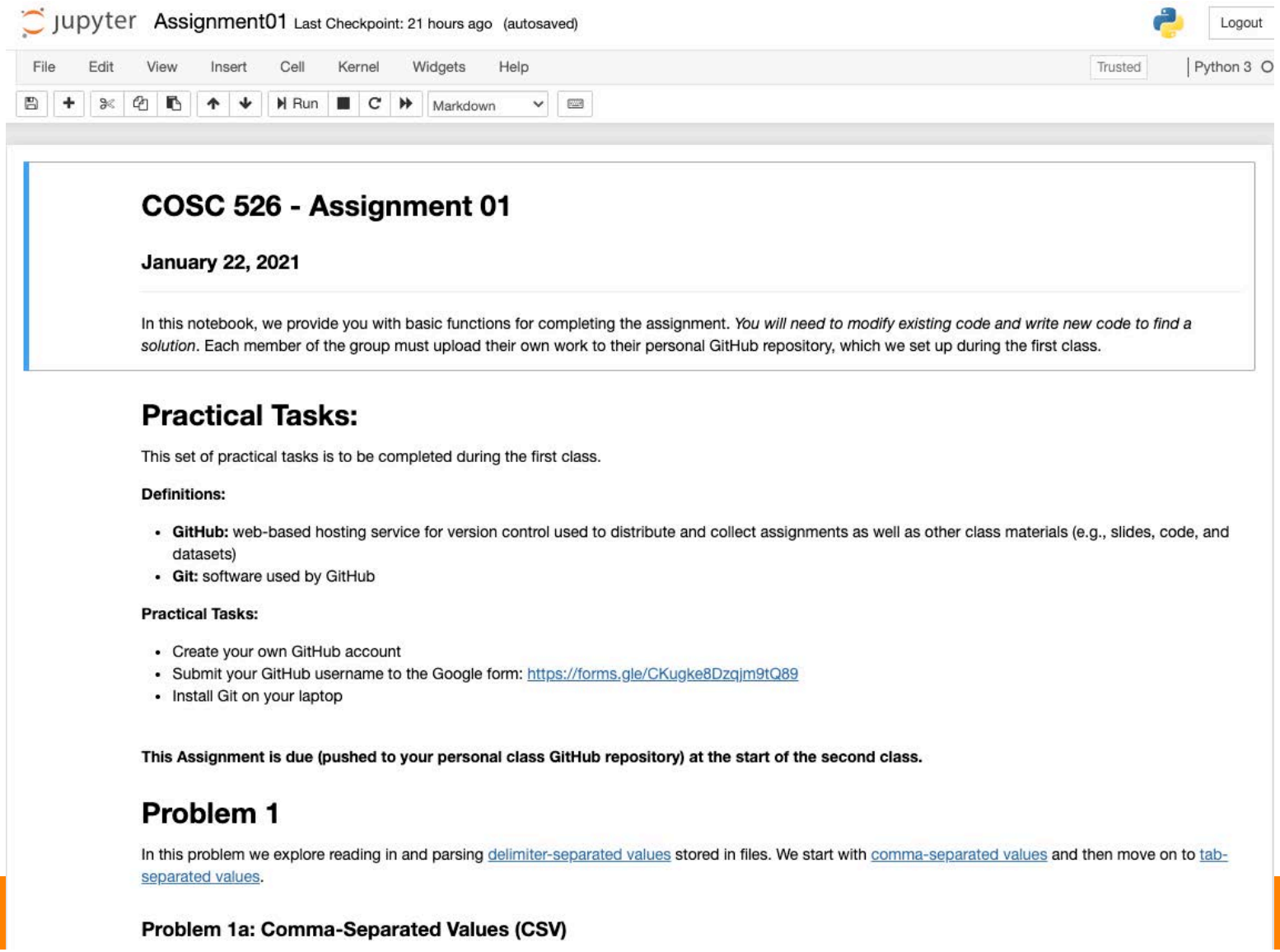
Nigel Tan

Kae Suarez

Paula Olaya

Leo Valera

Text Representation and Handling


jupyter Assignment01 Last Checkpoint: 21 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
COSC 526 - Assignment 01
January 22, 2021
In this notebook, we provide you with basic functions for completing the assignment. You will need to modify existing code and write new code to find a solution. Each member of the group must upload their own work to their personal GitHub repository, which we set up during the first class.
Practical Tasks:
This set of practical tasks is to be completed during the first class.
Definitions:
GitHub: web-based hosting service for version control used to distribute and collect assignments as well as other class materials (e.g., slides, code, and datasets)
Git: software used by GitHub
Practical Tasks:
Create your own GitHub account
Submit your GitHub username to the Google form: https://forms.gle/CKugke8Dzqjm9tQ89
Install Git on your laptop
This Assignment is due (pushed to your personal class GitHub repository) at the start of the second class.
Problem 1
In this problem we explore reading in and parsing delimiter-separated values stored in files. We start with comma-separated values and then move on to tab-separated values.
Problem 1a: Comma-Separated Values (CSV)

Assignment 1: Dealing with Text

- Reading in, parsing, and processing delimiter-separated values stored in files – **comma-separated values (csv)** and **tab-separated values** (**tsv**)
  - Count (and print) the number of rows of data (header is excluded) in the csv file
  - Count (and print) the number of columns of data in the csv file
  - Calculate (and print) the average of the values that are in the "age" column - You can assume each age in the file is an integer, but the average should be calculated as a float

comma-separated values (csv)

In [ ]:
```
def parse_delimited_file(filename, delimiter=","):
    # Open and read in all lines of the file
    # (I do not recommend readlines for LARGE files)
    # `open`: ref [1]
    # `readlines`: ref [2]
    with open(filename, 'r', encoding='utf8') as dsvfile:
        lines = dsvfile.readlines()

    # Strip off the newline from the end of each line
    # Using list comprehension is the recommended pythonic way to iterate through lists
    # HINT: refs [3,4]


    # Split each line based on the delimiter (which, in this case, is the comma)
    # HINT: ref [5]


    # Separate the header from the data
    # HINT: ref [6]


    # Find "age" within the header
    # (i.e., calculating the column index for "age")
    # HINT: ref [7]


    # Calculate the number of data rows and columns
    # HINT: [8]
    num_data_rows = 0
    num_data_cols = 0

    # Sum the "age" values
    # HINT: ref [9]


    # Calculate the average age
    ave_age = 0

    # Print the results
    # `format`: ref [10]
    print("Number of rows of data: {}".format(num_data_rows))
    print("Number of cols: {}".format(num_data_cols))
    print("Average Age: {}".format(ave_age))

# Parse the provided csv file
parse_delimited_file('data.csv')
```

References:

- 1: open
- 2: readlines
- 3: list comprehension
- 4: rstrip
- 5: split
- 6: splice
- 7: "more on lists"
- 8: len
- 9: int
- 10: format

tab-separated values (tsv)

```
    # Print the results
    # `format`: ref [10]
    print("Number of rows of data: {}".format(num_data_rows))
    print("Number of cols: {}".format(num_data_cols))
    print("Average Age: {}".format(ave_age))

# Parse the provided csv file
parse_delimited_file('data.csv')
```

In [ ]:
```
# Further reading on optional arguments, like "delimiter": http://www.diveintopython.net/power_of_introspection/optiona
parse_delimited_file('data.tsv', delimiter="\t")
```


Assignment 1: Dealing with different encoding

Standards for encoding text:

- ASCII
  - Use numeric codes to represent characters
- Unicode:
  - Map every character to a specific code U+<hex-code>, ranging from U+0000 to U+10FFFF
  - Define Unicode transformation formats: UTF-8, UTF-16, and UTF-32

Converting the unicode-formatted names into ascii-formatted names

- Use the provided tranliteration **dictionary** that maps several common unicode characters to their ascii transliteration to convert the unicode strings to ascii

convert unicode strings to ascii

```
In [ ]: translit_dict = {
            "ä" : "ae",
            "ö" : "oe",
            "ü" : "ue",
            "Ä" : "Ae",
            "Ö" : "Oe",
            "Ü" : "Ue",
            "ł" : "l",
            "ō" : "o",
        }

        with open("data.csv", 'r', encoding='utf8') as csvfile:
            lines = csvfile.readlines()

        # Strip off the newline from the end of each line


        # Split each line based on the delimiter (which, in this case, is the comma)


        # Separate the header from the data


        # Find "name" within the header


        # Extract the names from the rows
        unicode_names = []

        # Iterate over the names
        translit_names = []
        for unicode_name in unicode_names:
            # Perform the replacements in the translit_dict
            # HINT: ref [1]
            False

        # Write out the names to a file named "data-ascii.txt"
        # HINT: ref [2]


        # Verify that the names were converted and written out correctly
        with open("data-ascii.txt", 'r') as infile:
            for line in infile:
                print(line.rstrip())
```

References:

- 1: replace
- 2: file object methods

Commit and push in your personal repository