FIT5196-S2-2023 Assessment 1(35%)

This is an **individual assessment** and worth **35%** of your total mark for FIT5196.

Due date & time: Friday 25 Aug, 2023, 16:30PM

Background

Data, especially well-structured data, is the foundation of the current success of the Machine Learning and AI industry. The recent trending tool: ChatGPT, as a generative pre-trained language model, is also built upon training with numerous good-quality data.
However, in real-life scenarios, data is often in an unstructured format, commonly referred to as 'raw data'. Examples of raw data include text files obtained from system sales reports, PDF files downloaded from government databases, and images extracted from journal papers. The level of 'intelligence' attained by future or target models depends on the variety, accuracy, and level of structure present in the data they are fed.

Objectives

Parsing text files and reparing text data
for analysis. These s rate, reliable, and
meaningful. In this as and skills learned
from Week 2 to We s, extract specific
information, and perf tput the data in a
structured, machine- modelling tasks.
This assessment is a ark for FIT5196.

Task 1: Pa

Parsing data touche racting data from
different formats of ontain information
about reviews from a e the data file with
your STUDENT_ID, i.e., student_id.txt from the shared Google Drive folder (student_data). Using the wrong data will result in a ZERO mark as every student has a unique dataset to produce unique results. Three output files need to be named following the rules in the table below.

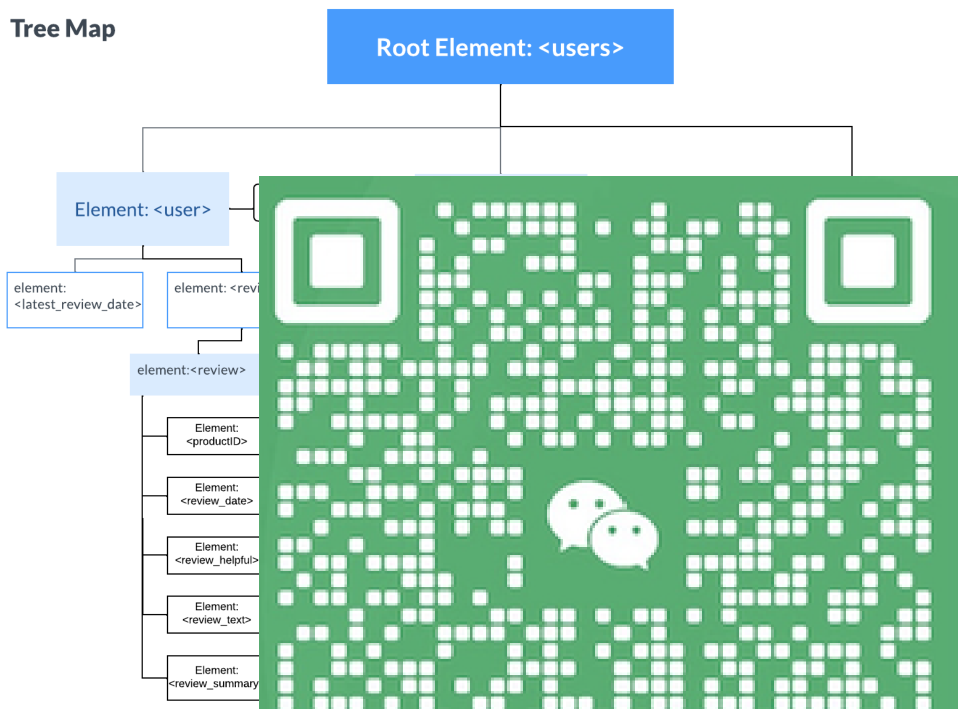
Input Files	Output Files (submission)
<student id>_task1_input#.txt	*<student id>.xml* *task1_<student id>.ipynb* *task1_<student id>.py*

Your input data contains information about reviews, i.e., **“reviewerID”**, **“productID”**, **“reviewer.NAME”**, **“No. helps”**, **“review_date”**, **“REVIEW”**, and the **“SUMMARY”**. Your task is to use **regular expressions** to extract all information regarding reviews from the text file, transform and represent the extracted data into a **XML** format with the following elements:

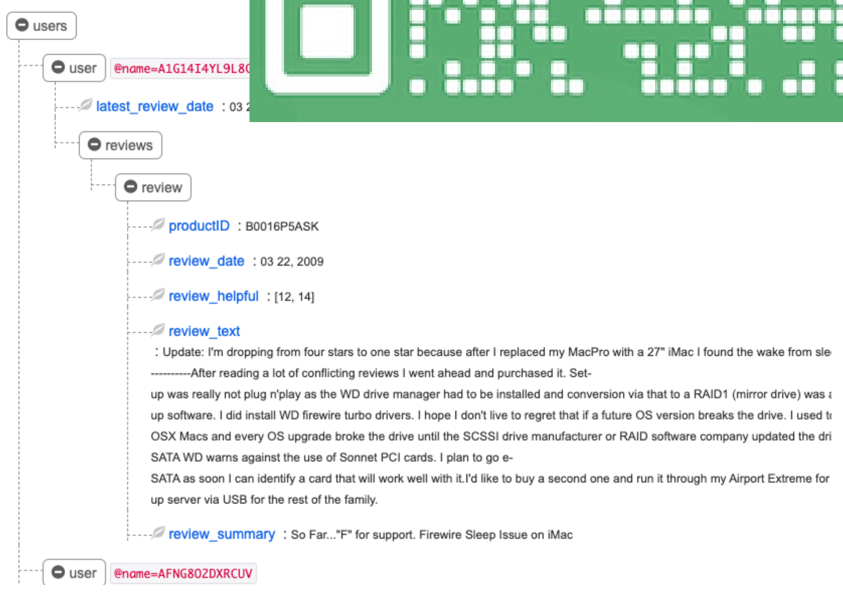
1. **users**: this tag wraps all the users, i.e. multiple <user> tag under <users>
2. **user**: this tag wraps all the reviews from a particular user and keeps the meta data for each user such as the latest review date and its username.
3. **reviews**: wraps all the reviews of a specific user
4. **review**: for each user, this tag wraps the **"productID"**, **"review_date"**, "**review_helpful**", and **"review_text"**, **"review_summary"** of the user tweet

Note: All the tag names are **case-sensitive** in the output XML file. You can refer to the sample here for the correct XML file structure.

Tree Map for Target XML:



Example: Sample Tr



```
users
  user  @name=A1G14I4YL9L8C
    latest_review_date : 03 2
    reviews
      review
        productID : B0016P5ASK
        review_date : 03 22, 2009
        review_helpful : [12, 14]
        review_text
        : Update: I'm dropping from four stars to one star because after I replaced my MacPro with a 27" iMac I found the wake from sle
        ----------After reading a lot of conflicting reviews I went ahead and purchased it. Set-
        up was really not plug n'play as the WD drive manager had to be installed and conversion via that to a RAID1 (mirror drive) was a
        up software. I did install WD firewire turbo drivers. I hope I don't live to regret that if a future OS version breaks the drive. I used t
        OSX Macs and every OS upgrade broke the drive until the SCSSI drive manufacturer or RAID software company updated the dri
        SATA WD warns against the use of Sonnet PCI cards. I plan to go e-
        SATA as soon I can identify a card that will work well with it.I'd like to buy a second one and run it through my Airport Extreme for
        up server via USB for the rest of the family.
        review_summary : So Far..."F" for support. Firewire Sleep Issue on iMac
  user  @name=AFNG8O2DXRCUV
```

Task 1 Guidelines

To complete the above task, please follow the steps below:

Step 0: Study the sample files
- Open and check your input txt file and find patterns for different data elements
- Use other online web applications such as xmlviewer to better understand the structure of the XML sample output.

Step 1: Txt file parsing
- Use python library to parse txt file
- Use Regex to extract the required attributes and their values as listed above

Step 2: Further process the extracted text from Step 1
- Remove the XML special characters from raw text (or replace with ' ', a white space)
- Save the data into a proper data format e.g. dataframe, dictionary...

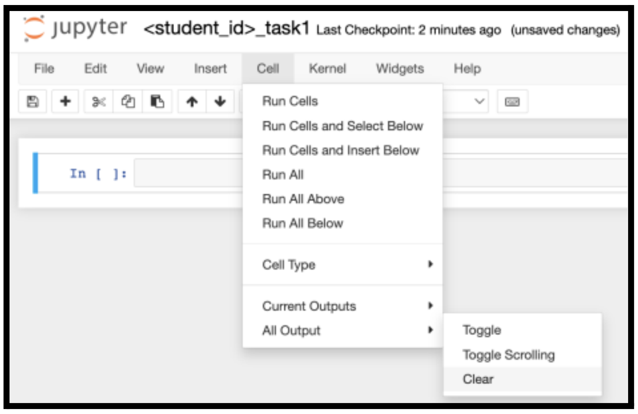
Step 3: XML file out ormat (make sure
- Use python li ... you check th ... hierarchy of your XML data)

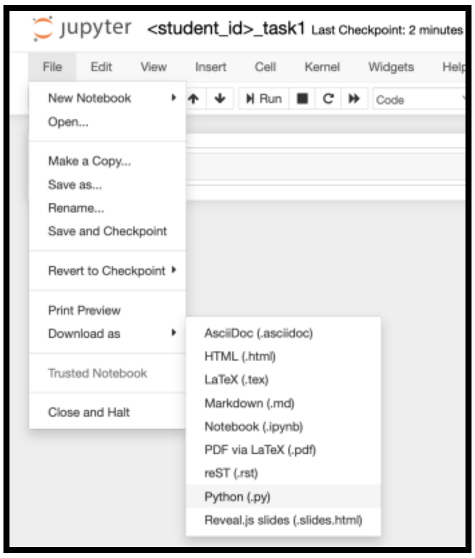
Submission Re

You need to submit 3 f
- A **task1_<** ... mation with all the elements l
- A Pytho ... **b** contains a well-docur ... sk 1. You need to clearly pre ... p process of your solution w ... can follow the suggested ... ook easy-to-read, as you wil ... e cell outputs are NOT cleared)
- A **task1_<student_id>.py** file. This file will be used for plagiarism check. (make sure the cell outputs are cleared before exporting)

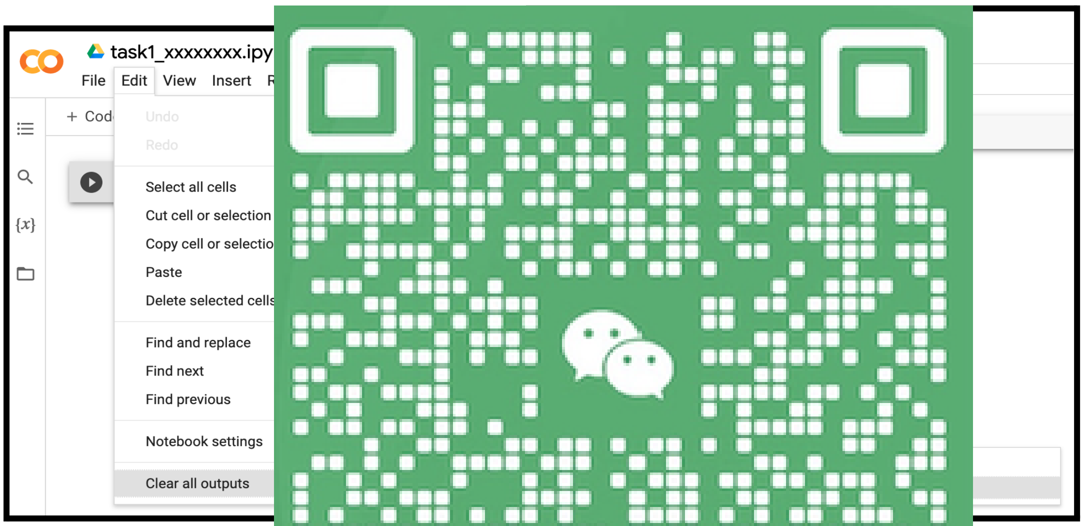
To generate a .py file, you need to clear all the cell outputs, and then download it.

In Jupyter notebook:


jupyter <student_id>_task1 Last Checkpoint: 2 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
In [ ]:
Run Cells
Run Cells and Select Below
Run Cells and Insert Below
Run All
Run All Above
Run All Below
Cell Type
Current Outputs
All Output
Toggle
Toggle Scrolling
Clear



In Google colab:



Requirements on the

- Methodolo
    - Yo gular expressions. Results from each step could help to demonstrate your solution better and be easier to understand.
    - You should present your solution in a proper way including all required steps. Skip any steps will cause
    - You need to select and use the appropriate Python functions for input, process and output.
    - Your solution should be an efficient one without redundant operations and unnecessary reading and writing the data.
- Report organisation and writing - 10%
    - The report should be organised in a proper structure to present your solutions to Task 1 with clear and meaningful titles for sections and subsections or sub-subsection if needed.

- Each step in your solution should be clearly described. For example, you can write to explain your idea of the solution, any specific settings, and the reason for using a particular function, etc.
- Explanation of your results including all intermediate steps is required. This can help the marking team to understand your solution and give partial marks if the final results are not fully correct.
- All your codes need proper (but not excessive) commenting.
- You can refer to the notebook templates provided as a guideline for a properly formatted notebook report.

Task 2: Text Pre-Processing (17/35)

This task touches on the next step of analysing textual data, converting the extracted text data into a numerical representation thus it can be used for a downstream modelling task. In this task, you are required to write Python code to pre-process a set of published papers (pdf) and convert the [illegible] resentation is the standard format of te[illegible] systems such as: recommender-system[illegible] n etc.). The most basic step for natural [illegible] s into numbers for machines to underst[illegible], though iterative, plays a significant rol[illegible] /algorithm.

Input Files [illegible]	[illegible])
*<student id>_paper_* [illegible]	[illegible] /

You are provided wit[illegible] blished in several popular AI confere[illegible] TUDENT_ID, i.e. ***<student_id>.txt*** in t[illegible] ***student_id>.pdf***) contains a table in which each row contains a paper with a unique ID and a URL where it can be downloaded.

You are asked to parse the table of paper URLs in python, and output the table into a csv file. Then programmatically download all papers, and parse the required abstract section from all papers. Then pre-process the abstract text and generate a vocabulary list and numerical representation for the corresponding text, which will be used in the model training by your colleagues. The information regarding output files is listed below:

- **paper_list.csv** contains the unique paper IDs along with their corresponding URLs.
- **vocab.txt** comprises unique stemmed tokens sorted alphabetically, presented in the format of **token_index:token**, as outlined in Guideline step 3.

- **countvec.txt** includes numerical representations of all tokens, organised by paper ID and token index, following the format **paper_id, token_index:frequency**, as outlined in Guideline step 4.

Carefully examine the sample files (**here**) for detailed information about the output structure.

VERY IMPORTANT NOTE: The sample outputs are just for you to understand the structure of the required output and the correctness of their content in task 2 is not guaranteed. So please do not try to reverse engineer the outputs as it will fail to generate the correct content.

Task 2 Guideline

To complete the above task, please follow the steps below:

Step 1: Programmatically download the pdfs

- Use the giver manual download will be penalis

Step 2: Read the pd

- Read the PD mplete the above task (hint: **p** es can help you complete this
- Replace the l tities (either using replace functi hyphen-separated word (e.g., ch o pdf format),

Step 3: Generate the

Before building the s preprocessing on **Abstract**. Please foll ange the order as the correct order of o cabulary.

1. Tokenize usin

r"[A-Za-z]\w+(?:[-'?]\w+)?"

2. Remove context-independent stop words (i.e., **stopwords_en.txt**)
3. Remove context-dependent stop words (unigram tokens appearing in 95% or more of the files).
4. Remove rare tokens (appearing in less than 3% of the files)
5. Remove tokens with less than 3 characters/symbols.
6. Stem unigram tokens using the Porter stemmer.
7. Generate the vocab.txt output with ascending ordered unigrams, with format:
   token1: token1_index
   token2: token2_index
   …

Step 4: Generate the sparse numerical representation and output as countvec.txt

1. Generate sparse representation by using the countvectorizer() function OR directly count the frequency using FreqDist().
2. Mapping the generated token with the stemmed token in step 3 if need
3. Output the sparse numerical representation into txt file with the format:

    paper_id1, token1_index:token1_frequency, token2_index:token2_frequency, token3_index:token3_frequency, …
    paper_id2, token2_index:token2_frequency, token5_index:token5_frequency, token7_index:token7_frequency, …
    paper_id3, token6_index:token6_frequency, token9_index:token9_frequency, token12_index:token12_frequency, …

Submission Requirements

You need to submit 6 files:

1. A **<student ic** from pdf input
2. A **<student_i** e following format, token:token_i betical order.
3. A **<student_** ains the sparse representatio

    paper_id, t _wordcount, …

    Please note: t ded in the sparse representatio
4. A **task2_<stu** ing the code and the methodolo
5. A **task2_<stu** e cell outputs are cleared before

Requirements on the

- Methodolo
    - Yo ular expressions.
    - You should present your solution in a proper way including all required steps.
    - You need to select and use the appropriate Python functions for input, process and output.
    - Your solution should be an efficient one without redundant operations and unnecessary reading and writing the data.
- Report organisation and writing - 10%
    - The report should be organised in a proper structure to present your solutions to Task 2 with clear and meaningful titles for sections and subsections or sub-subsection if needed.
    - Each step in your solution should be clearly described. For example, you can write to explain your idea of the solution, any specific settings, and the reason for using a particular function, etc.

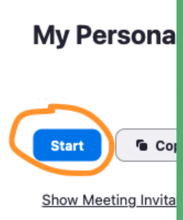
- Explanation of your results including all intermediate steps is required. This can help the marking team to understand your solution and give partial marks if the final results are not fully correct.
- All your codes need proper (but not excessive) commenting.
- You can refer to the notebook templates provided as a guideline for a properly formatted notebook report.

Task 3: Video Presentation (3/35)

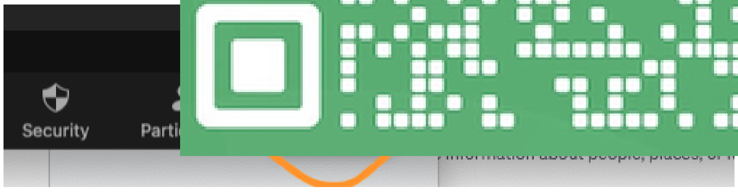
Presenting your methodology to the audience and explaining your logic is a crucial skill for a data analyst. In this task, you are required to **record a video** (less than 5 minutes) to go through the main logic of your codes in **BOTH Task 1 and 2.** You are required to show your notebook file with output while explaining in voice how your code works to generate the expected output. The explanation of methodology needs to be clear and allows the audience to understand how your python code parses the data, extract the data and pre-process the data. You can follow the steps below to record your video.

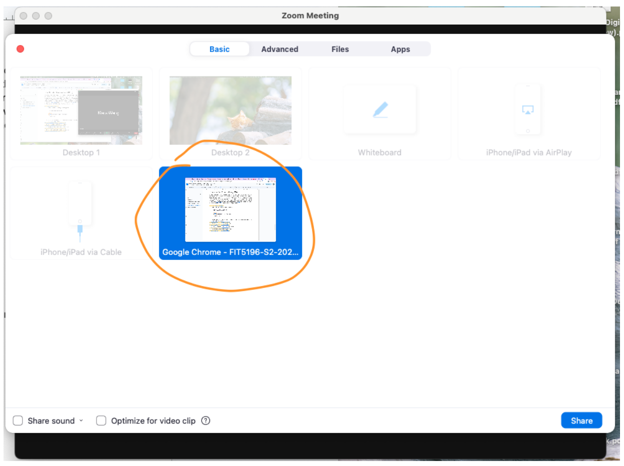
Zoom Recordi

1. Open the Zoo

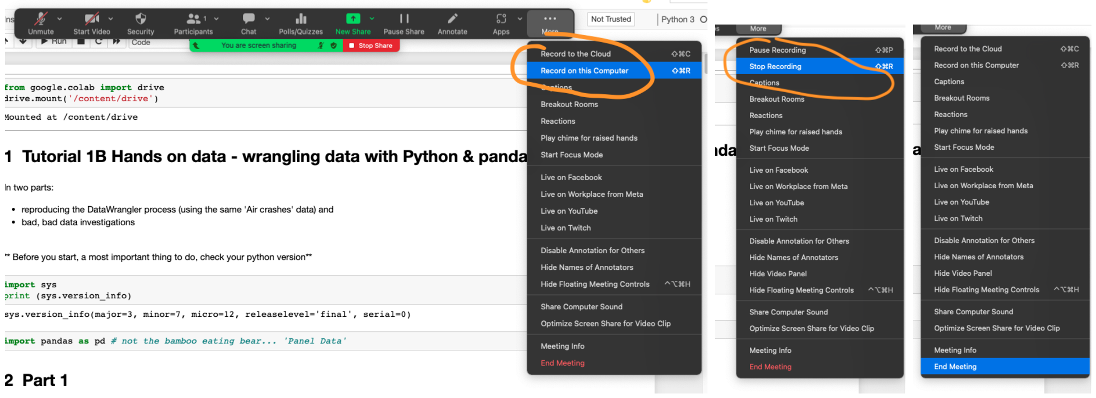


2. **Share your s** mera is turned on and shown on





3. **Start recording** and finish everything in 5 minutes, then press **stop recording** and **end meeting**. You can pause and resume multiple times if necessary.



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1 Tutorial 1B Hands on data - wrangling data with Python & panda

In two parts:

- reproducing the DataWrangler process (using the same 'Air crashes' data) and
- bad, bad data investigations

** Before you start, a most important thing to do, check your python version**

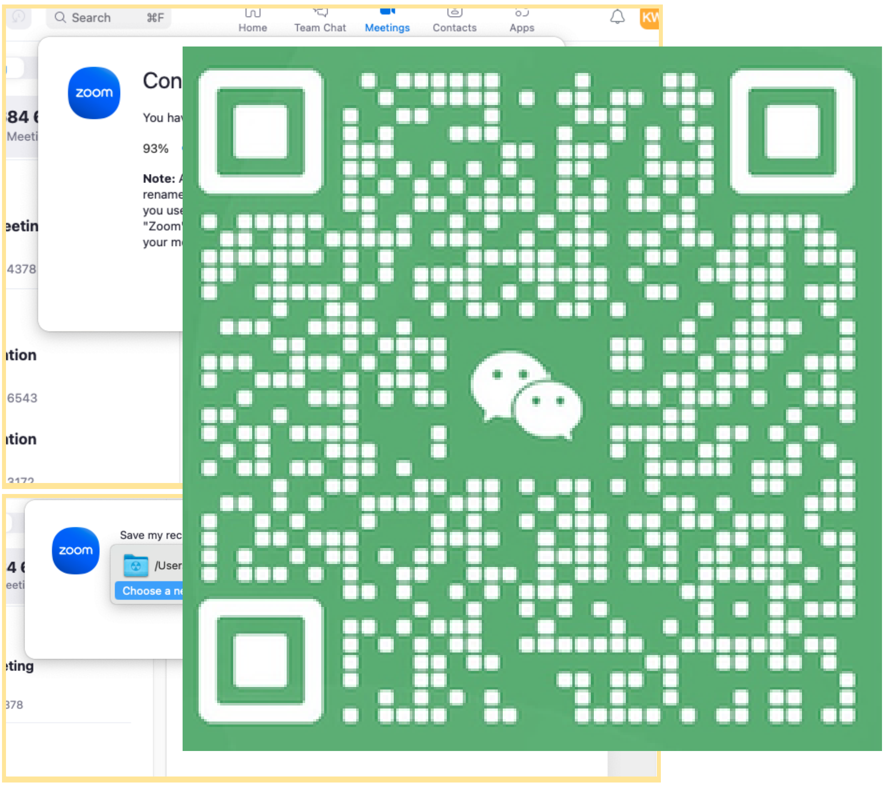
```
import sys
print (sys.version_info)
```

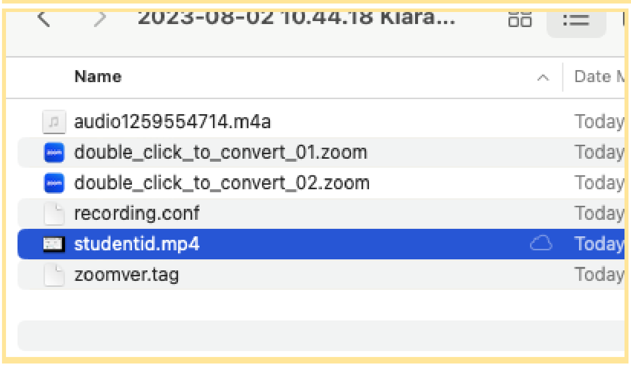
```
sys.version_info(major=3, minor=7, micro=12, releaselevel='final', serial=0)
```

```
import pandas as pd # not the bamboo eating bear... 'Panel Data'
```

2 Part 1

4. Wait for the video conversion and find the file, **rename** it with your **student id**.





Name	Date M
audio1259554714.m4a	Today
double_click_to_convert_01.zoom	Today
double_click_to_convert_02.zoom	Today
recording.conf	Today
studentid.mp4	Today
zoomver.tag	Today

5. **Submit it on Moodle** together with your jupyter notebook.

Submission Checklist:

- ☐ Please zip all the submission files for task 1 and 2 into a single file with the name **<student_id>_ass1.zip.** (any other format e.g. rar or 7z will be penalised)
- ☐ There are **8 files** in your compressed zip file
- ☐ Please submit the video with the name ***<student_id>.mp4***
- ☐ **<student_id>** should be replaced with **your monash student ID**.
- ☐ Please strictly follow the file naming standard. Any misnamed file will carry a penalty.
- ☐ Please make hile your **.py file**

  does not inclu
- ☐ Please ensure an achieve this by

  re-reading all **ead_csv** for CSV

  files or `Elem` anity checks and

  hence should

Note: All submissions hich automatically
checks for their simil found will trigger
the Faculty's relevan to and including
exclusion from the un