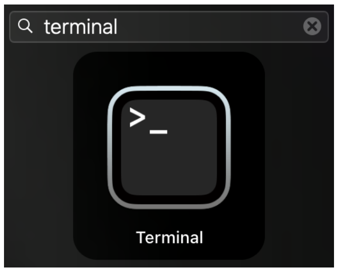
Github Link: github.com/bellaachang/TWB-DeviSansthan-DSS-SPR23

Video Tutorial:

Step-by-step: Mac Setup

1) Install a terminal



The terminal is a program that allows you to interact with your computer by entering commands.

If you're on a Mac, you already have a program called `Terminal` or something similar on your computer. Open that up and you should be good to go.

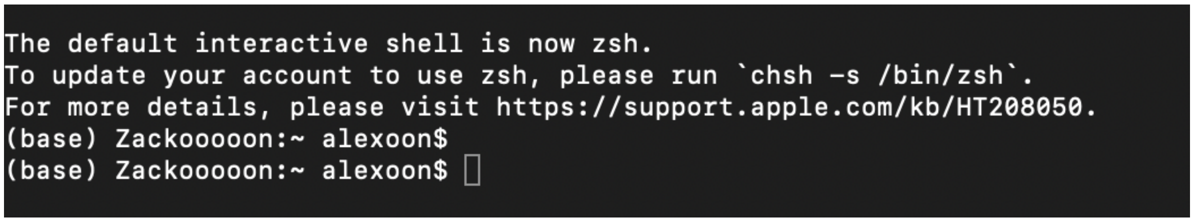
2) Install Python 3

Python 3 is the programming language we used in our work. If you already have an older version of Python installed, please make sure to download and install Python3. You can check your Python version with `python3 —version`. [Download Python3 Here]()

Download and install Python 3 (64-bit). You may need to right-click the download icon and select "Open". After installing, please close and reopen your Terminal.

3) Open terminal and see current location

Open "Terminal" and you'll see something like this



```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Zacksooooon:~ alexoon$
(base) Zacksooooon:~ alexoon$ 
```

The ~ sign tells us that we are the "home" directory, which is the outermost folder of your file organization system. In the example above, my home directory is named "alexoon"

4) Open the project folder

We now want to go to the folder in which your program was saved. You can check where it was saved through your file explorer.

If we go to our file explorer and type in the name of the program that was downloaded (ours was getit.py), **[INSERT NAME]**, then **single**-click on the file/folder name, it'll show at the bottom how to get from the "home" directory to the file.

To go from outer folders to the other folders listed above, we will use the following commands:

```
cd ~
cd Desktop/TWB
```

Now, you will be in the innermost folder (in this example, it is called "TWB").

5) Download necessary libraries

Before we start coding, we will need to download the necessary libraries required to run our code.

Now, to download libraries, do:

```
pip install -r requirements.txt
```

Now, all the required libraries should be installed!

Step-by-step: Windows Setup

1) Install a terminal

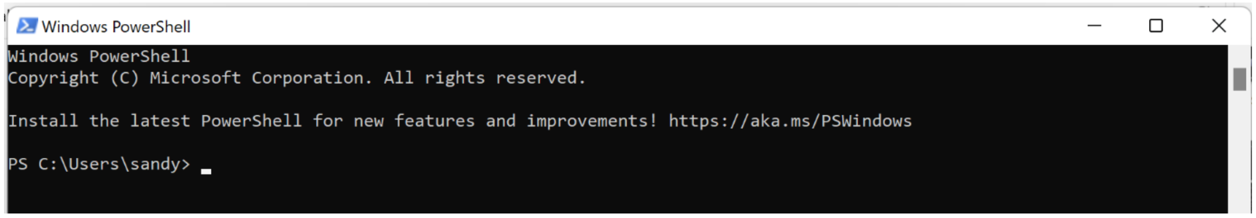
PowerShell comes pre-installed on Windows and requires no extra setup. You can simply launch it from the Start menu. Simple commands like `cd` and `ls` will work (`python` will work after the setup), which encompass most of the Bash commands needed.

2) Install Python 3

Open the Microsoft Store and search for "python". Install Python 3.9 by the Python Software Foundation (this should be the first result). You can then skip the rest of this section. (Important: If you later decide to reinstall Python differently, **uninstall it from the Microsoft Store first.)**

3) Open terminal and see current location

Open "Terminal" and you'll see something like this



Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sandy> _
```

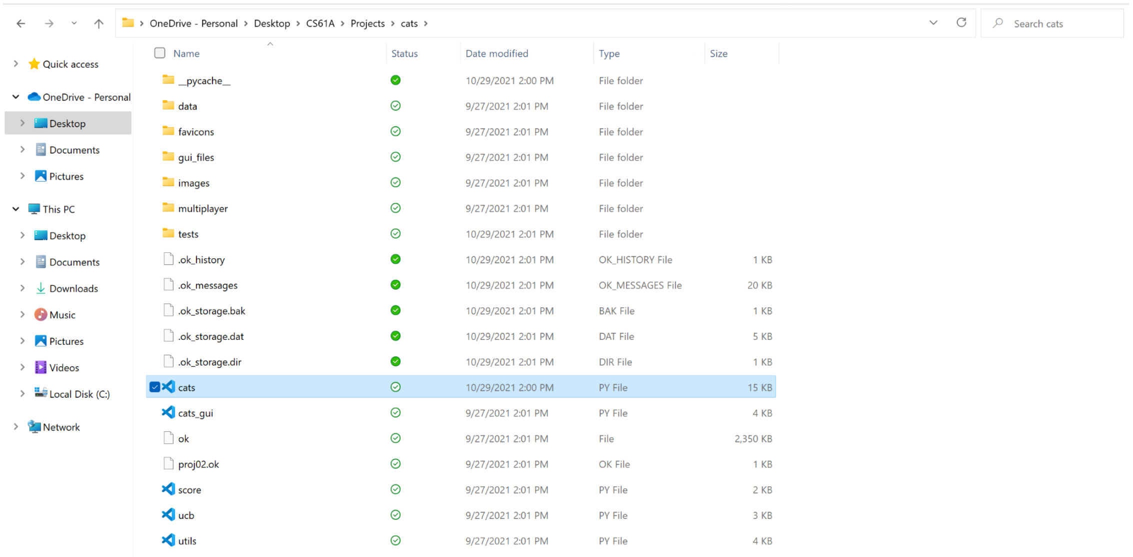
This is our “home” directory, which is the outermost folder of your file organization system. In the example above, my home directory is named “sandy”

4) Open the project folder

We now want to go to the folder in which your program was saved. You can check where it was saved through your file explorer.

If we go to our file explorer and type in the name of the program that was downloaded (ours was cats.py), **[INSERT NAME]**, then **single**-click on the file/folder name, it’ll show at the top how to get from the “home” directory to the file.

(In this case, the path is OneDrive → Personal → Desktop → CS61A → Projects → cats → cats.py)



Name	Status	Date modified	Type	Size
__pycache__	●	10/29/2021 2:00 PM	File folder	
data	✓	9/27/2021 2:01 PM	File folder	
favicons	✓	9/27/2021 2:01 PM	File folder	
gui_files	✓	9/27/2021 2:01 PM	File folder	
images	✓	9/27/2021 2:01 PM	File folder	
multiplayer	✓	9/27/2021 2:01 PM	File folder	
tests	✓	10/29/2021 2:01 PM	File folder	
.ok_history	●	10/29/2021 2:01 PM	OK_HISTORY File	1 KB
.ok_messages	●	10/29/2021 2:01 PM	OK_MESSAGES File	20 KB
.ok_storage.bak	●	10/29/2021 2:01 PM	BAK File	1 KB
.ok_storage.dat	●	10/29/2021 2:01 PM	DAT File	5 KB
.ok_storage.dir	●	10/29/2021 2:01 PM	DIR File	1 KB
cats	✓	10/29/2021 2:00 PM	PY File	15 KB
cats_gui	✓	9/27/2021 2:01 PM	PY File	4 KB
ok	✓	9/27/2021 2:01 PM	File	2,350 KB
proj02.ok	✓	9/27/2021 2:01 PM	OK File	1 KB
score	✓	9/27/2021 2:01 PM	PY File	2 KB
ucb	✓	9/27/2021 2:01 PM	PY File	3 KB
utils	✓	9/27/2021 2:01 PM	PY File	4 KB

You’ll only need to do step 5 once. Once you have done it, you won’t need to do it even if you update the data. Step 6 should be carried out whenever you want to analyze updated data! If it isn’t the first time going through this manual on your computer and you aren’t using new data, you can *skip to part 7.*

5) Download required dependencies

Before we can execute the question generation toolkit, we will need to download the dependencies required to run our code.

To download dependencies, do:

```
pip install -r requirements.txt
```

Now, all the required dependencies should be installed!

Step-by-step: Question Generation Process

6) Prompt ChatGPT

ChatGPT is a large language model that can quickly generate human-like responses to language prompts, which is perfect for creating a variety of questions for the Literacy Now app.

6a) Text-based Question Prompt

See the following for an example of a text-based question prompt. This particular prompt was used to generate **division-based** questions:

You are generating questions for an educational app aimed toward educating 8 year olds. Generate simple division questions similar to "Rida has Rf 730. She wants to give them equally to her two brothers for buying books. How much money does each brother get?" The solution must be a whole number. All numbers used must be less than 100.

Note that you will upload multiple CSVs because each CSV corresponds to a different type of text based question. For example, the above prompt was related to division, so you would upload addition-based questions and subtraction-based questions separately.

6b) Image-based Question Prompt

Go to this link to see how we generated the list of all English phonemes and elementary example words for each of them:
https://chat.openai.com/share/ba3a63d0-f39d-4e0f-bc0b-a7f1366d70d1

If the link is not working, paste these prompts (in order):

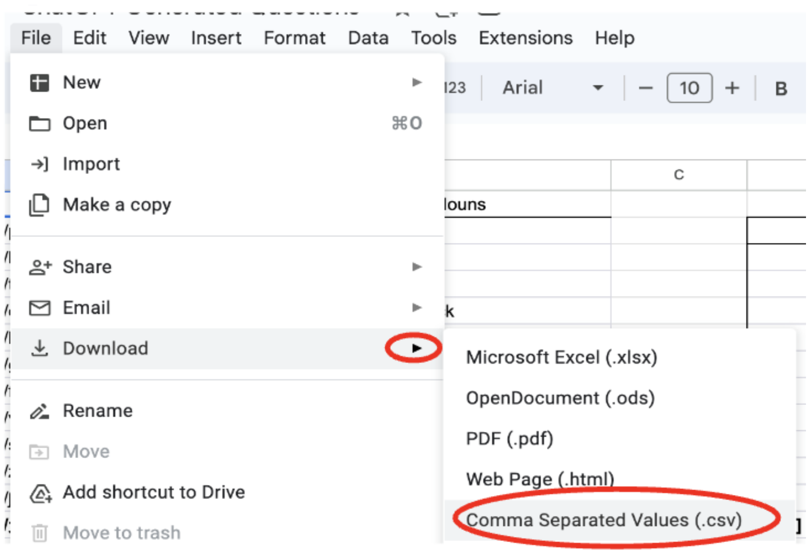
1. *Generate a table with two columns, the first corresponding to all the English phonemes and the second is a long list of one-syllable nouns that contain that phoneme*
2. *Generate a table with two columns, the first corresponding to all the English phonemes you generated above and the second column as a long list of elementary-level, one-syllable nouns that contain that phoneme.*

3. *Now, recreate the table with the same requirements, except with 5 examples of each one-syllable noun*

7) Download Chat GPT Output

To convert the Chat GPT output to a CSV file, we copied the output into a Google Sheets document. Then, we downloaded it as a Comma Separated Values (CSV) file. Make sure the name of the Google Slides have no spaces in it (“division-questions” instead of “division questions”), or else there may be some errors when loading in the data.
Make sure to upload it in this form:

- No headers
- Quotes around the questions
- No answer choices



8) Load in Data

Finally, before we can use the question generation pipelines, we just need to import the downloaded CSVs from Step 7.

Note: We have provided you with several question databases, so importing new ones or adding on to the existing CSVs in the folder is only necessary if you would like to generate further or more complex questions.

Once you have obtained the CSVs from Step 7, drag and drop the data files into their corresponding subfolder within the “data” folder (“image-based” or “text-based” respectively). As mentioned above, each subfolder in the “data” folder will already be populated with CSV files we have generated with ChatGPT. Feel free to paste contents from your new files into the existing files or replace them entirely.

If you choose to replace any files in the “data” folder, we want to update the file titles within our code! This may seem very intimidating, but don’t worry, it is just a couple copy-pastes.

There are two options: you can (1) change the data file names in your “data” folder — this is the easiest way to go about it, or (2) you can update the file “paths” (where it is located in your computer) in our code! We don’t recommend this option unless you are inputting a new data path we have not generated already (e.g. image-based math questions).

For (1): Change file name in your folder
The exact file names we use are:

- Text-based folder
    - addition-qs.csv
    - division-qs.csv
    - multiplication-qs.csv
    - perimeter-qs.csv
    - subtraction-qs.csv
- Image-based folder
    - phonemes-list.csv

For (2): Update file paths in code
To update the file paths, you can do so in a text editor. Specific instructions are found in this section of the document here.

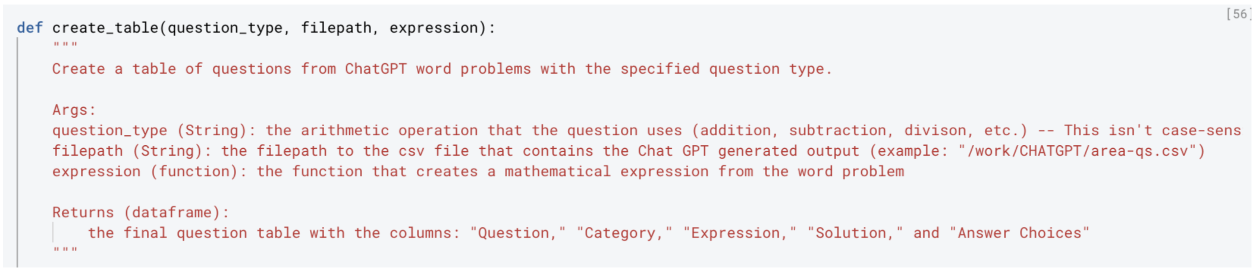
9) Generate Solutions

Now that we have our questions, it’s time to make the answers the possible solutions. Since this step is different for text-based and image-based questions, we have separated the steps for those in two sub-sections.

9a) For Text-Based Solutions (in a text editor)

*Note: We generated text-based questions and solutions for **math**-related questions so far, but the same process can be applied to create english-related questions. There are more detailed explanations in the code itself.*

We have created a **create_table()** function that creates a table of questions from ChatGPT word problems with the specified question type. This function takes in three arguments (question_type, filepath, expression) and returns a question table with the columns: "Question," "Category," "Expression," "Solution," and "Answer Choices"



```
def create_table(question_type, filepath, expression):
    """
    Create a table of questions from ChatGPT word problems with the specified question type.

    Args:
    question_type (String): the arithmetic operation that the question uses (addition, subtraction, divison, etc.) -- This isn't case-sens
    filepath (String): the filepath to the csv file that contains the Chat GPT generated output (example: "/work/CHATGPT/area-qs.csv")
    expression (function): the function that creates a mathematical expression from the word problem

    Returns (dataframe):
        the final question table with the columns: "Question," "Category," "Expression," "Solution," and "Answer Choices"
    """
```

The create_table() function and other math expression functions are included in the utils document.

9b) For Image-Based Solutions (in a text editor)

Note: We generated image-based questions and solutions for ***english****-related questions so far that use phonemes, but the same process can be generalized to create different types of questions. There are more detailed explanations in the code itself.*

1. Clean up the dataframe of phonemes from Chat GPT output

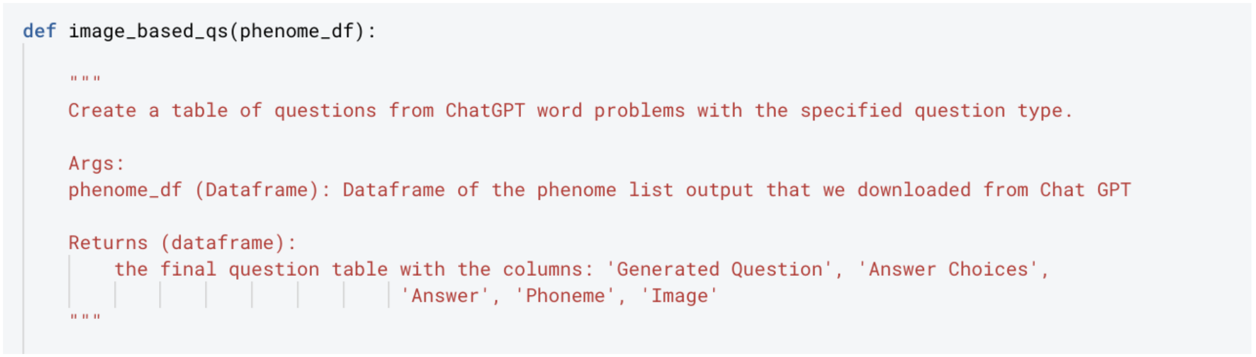
```
#phoneme_list is the dataframe of phonemes from the Chat GPT output
phoneme_list = phoneme_list.rename(columns={"Example Nouns": "Examples"})

#Drop the rows that don't contain enough examples
phoneme_list= phoneme_list[phoneme_list.Examples.str.contains("Few common nouns") == False]

# turn the examples into an array to make the future code easier
phoneme_list["Word List"] = phoneme_list["Examples"].str.split(",")
phoneme_list["Word List"]

phoneme_list
```

2. Use the image_based_qs function to create a table of with a question, answer choices, and answer for each phonemes
    a. Note that this function is included in the utils.py document



```
def image_based_qs(phenome_df):

    """
    Create a table of questions from ChatGPT word problems with the specified question type.

    Args:
    phenome_df (Dataframe): Dataframe of the phenome list output that we downloaded from Chat GPT

    Returns (dataframe):
        the final question table with the columns: 'Generated Question', 'Answer Choices',
                                                    'Answer', 'Phoneme', 'Image'
    """
```

Web-scraping Tool

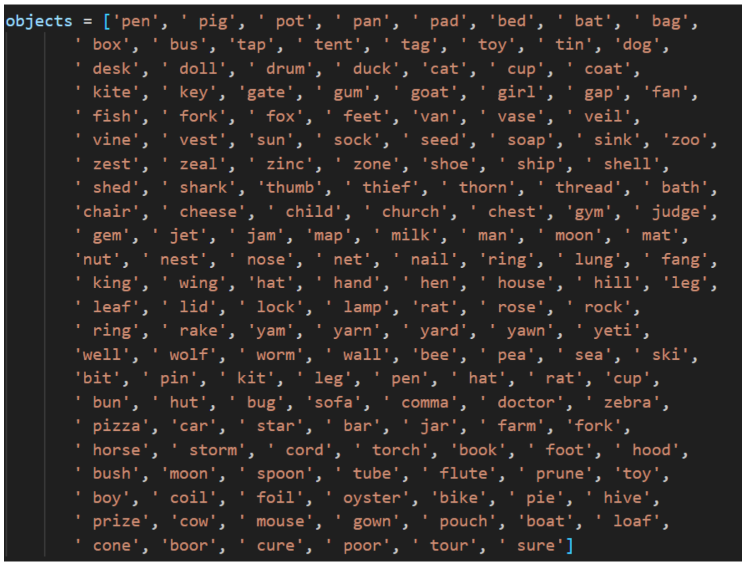
10a) Import Libraries and Create List of Image-based Answers

Here are some libraries and modules we think might be helpful in accomplishing the task of automating image retrieval with a web scraping tool.

```
from bs4 import BeautifulSoup
import requests

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import urllib
import os
```

Next, input the identified answers from **9b** as a list in your chosen IDE. This will allow you to webscrape an image that corresponds to each word and download it.



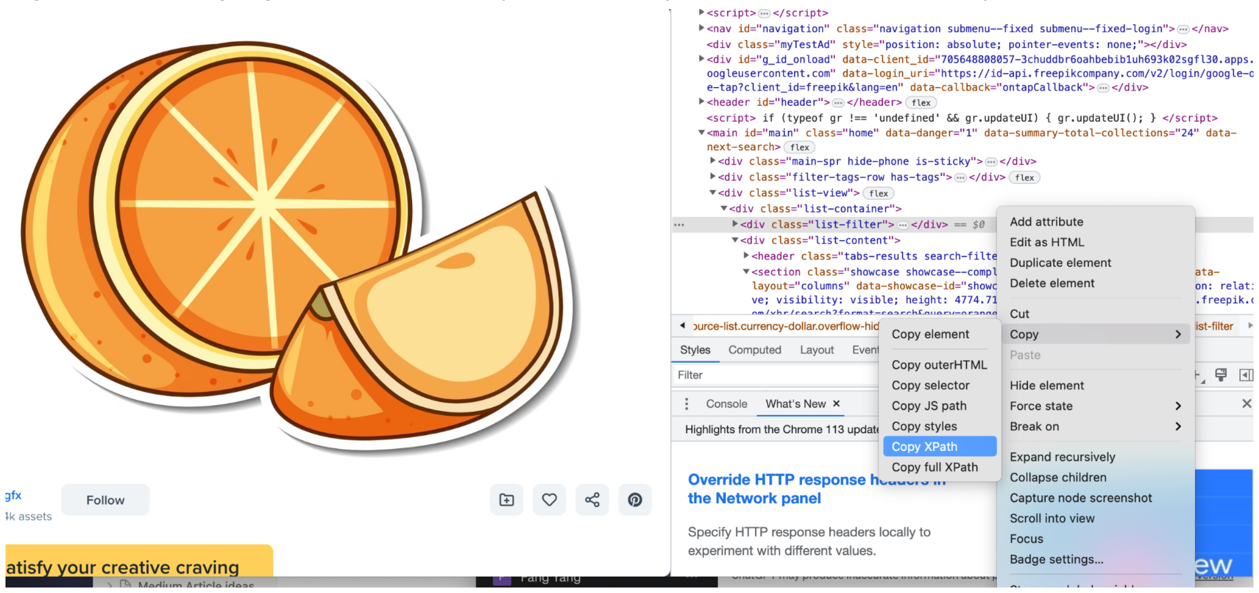
```
objects = ['pen', ' pig', ' pot', ' pan', ' pad', 'bed', ' bat', ' bag',
        ' box', ' bus', 'tap', ' tent', ' tag', ' toy', ' tin', 'dog',
        ' desk', ' doll', ' drum', ' duck', 'cat', ' cup', ' coat',
        ' kite', ' key', 'gate', ' gum', ' goat', ' girl', ' gap', 'fan',
        ' fish', ' fork', ' fox', ' feet', 'van', ' vase', ' veil',
        ' vine', ' vest', 'sun', ' sock', ' seed', ' soap', ' sink', 'zoo',
        ' zest', ' zeal', ' zinc', ' zone', 'shoe', ' ship', ' shell',
        ' shed', ' shark', 'thumb', ' thief', ' thorn', ' thread', ' bath',
        'chair', ' cheese', ' child', ' church', ' chest', 'gym', ' judge',
        ' gem', ' jet', ' jam', 'map', ' milk', ' man', ' moon', ' mat',
        'nut', ' nest', ' nose', ' net', ' nail', 'ring', ' lung', ' fang',
        ' king', ' wing', 'hat', ' hand', ' hen', ' house', ' hill', 'leg',
        ' leaf', ' lid', ' lock', ' lamp', 'rat', ' rose', ' rock',
        ' ring', ' rake', 'yam', ' yarn', ' yard', ' yawn', ' yeti',
        'well', ' wolf', ' worm', ' wall', 'bee', ' pea', ' sea', ' ski',
        'bit', ' pin', ' kit', ' leg', ' pen', ' hat', ' rat', 'cup',
        ' bun', ' hut', ' bug', 'sofa', ' comma', ' doctor', ' zebra',
        ' pizza', 'car', ' star', ' bar', ' jar', ' farm', 'fork',
        ' horse', ' storm', ' cord', ' torch', 'book', ' foot', ' hood',
        ' bush', 'moon', ' spoon', ' tube', ' flute', ' prune', 'toy',
        ' boy', ' coil', ' foil', ' oyster', 'bike', ' pie', ' hive',
        ' prize', 'cow', ' mouse', ' gown', ' pouch', 'boat', ' loaf',
        ' cone', 'boor', ' cure', ' poor', ' tour', ' sure']
```

10b) Choose a Website and Method to Retrieve Content

- Select a website with an approachable user interface and images that are easy to understand.
- Right click on page elements and select “Inspect” to explore classes within these elements. The goal is to identify where the image lies within the website’s code.
- Think about different ways these classes can be extracted in the next step by researching approaches that use libraries such as BeautifulSoup and Selenium.

10c) Extract Website Content

- In the developer’s tool, look at the highlighted line that pops up after right clicking the inspect element.
- Right click the highlighted line and copy the XPath (no need to do full XPath)



```
▶<script>⋯</script>
▶<nav id="navigation" class="navigation submenu--fixed submenu--fixed-login">⋯</nav>
 <div class="myTestAd" style="position: absolute; pointer-events: none;"></div>
▶<div id="g_id_onload" data-client_id="705648808057-3chuddbr6oahbebib1uh693k02sgfl30.apps.
 oogleusercontent.com" data-login_uri="https://id-api.freepikcompany.com/v2/login/google-o
 e-tap?client_id=freepik&lang=en" data-callback="ontapCallback">⋯</div>
▶<header id="header">⋯</header> flex
 <script> if (typeof gr !== 'undefined' && gr.updateUI) { gr.updateUI(); } </script>
▼<main id="main" class="home" data-danger="1" data-summary-total-collections="24" data-
 next-search> flex
  ▶<div class="main-spr hide-phone is-sticky">⋯</div>
  ▶<div class="filter-tags-row has-tags">⋯</div> flex
  ▼<div class="list-view"> flex
    ▼<div class="list-container">
…     ▶<div class="list-filter">⋯</div> == $0
      ▼<div class="list-content">
        ▶<header class="tabs-results search-filte
        ▼<section class="showcase showcase--compl
         layout="columns" data-showcase-id="showc
         ve; visibility: visible; height: 4774.71
```

Highlights from the Chrome 113 update

Override HTTP response headers in the Network panel

Specify HTTP response headers locally to experiment with different values.

- Repeat your process for the other words by creating a for loop that iterates over the rest of the words in the list

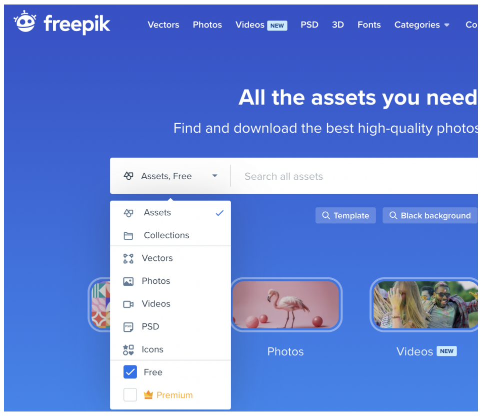
10d) Download Image Data to Specified File Location

- Finally, make sure to create a folder for the images to be stored in. You can also add on to the folder we have included for you, which has already been populated with images.
- To download the extracted content as an image, you need to first get the image data from the extracted content. Functions like get_attribute from web scraping libraries or the library urllib can assist you in getting this image data.
- Once you have this information, save the image with a consistent naming convention to the folder location you have specified. The code below shows how to save an image to the “ques_img” folder we have provided you with.

```
with open(f"ques_img/{object}.png", "wb") as f:
        f.write(IMAGEDATA)
```

Tips:

- Use the website freepik.com for non-copyrighted, free graphic images
    - Click the free button to ensure that you are using free images (shown in the image to the right)
- To ensure that the image you webscrap aligns with what you want, web scrape images of nouns. It's easier to find a picture that describes "pan" than "run."
- Make sure to specify "clipart" after the image of what you want. This ensures that the images all follow the same style of the questions that are currently being asked on the Literacy Now app.



Updating file names in code:

1) Install a text editor

The **Python interpreter** that you just installed allows you to *run* Python code. You will also need a **text editor**, where you will *write* Python code.

Visual Studio Code (VS Code) is the most popular choice among the staff for this course for writing Python. Some other editors that are used among staff are listed below as well.

If you're using Windows and followed our Python setup process, VS Code will work best for you (since it has WSL support). After installing VS Code, install the Remote Development extension pack. Then, you can use the instructions in this section of the VS Code docs to open WSL folders in VS Code.

We highly recommend using VS Code.

Another nice feature of VS Code is that it features an "embedded terminal". So, when running terminal commands for this class, you can manage everything in VS Code rather than navigating back and forth between VS Code and a separate terminal application. You can open an embedded terminal by going to `Terminal > New Terminal` in VS Code's navigation bar.

Warning: Please do not use word processors such as Microsoft Word to edit programs, as it'll add/remove extra characters.

For your reference, here are some guides on using popular text editors:

- Visual Studio Code: A full-featured desktop editor with many extensions available to support different languages.
- Atom: A more lightweight desktop editor.

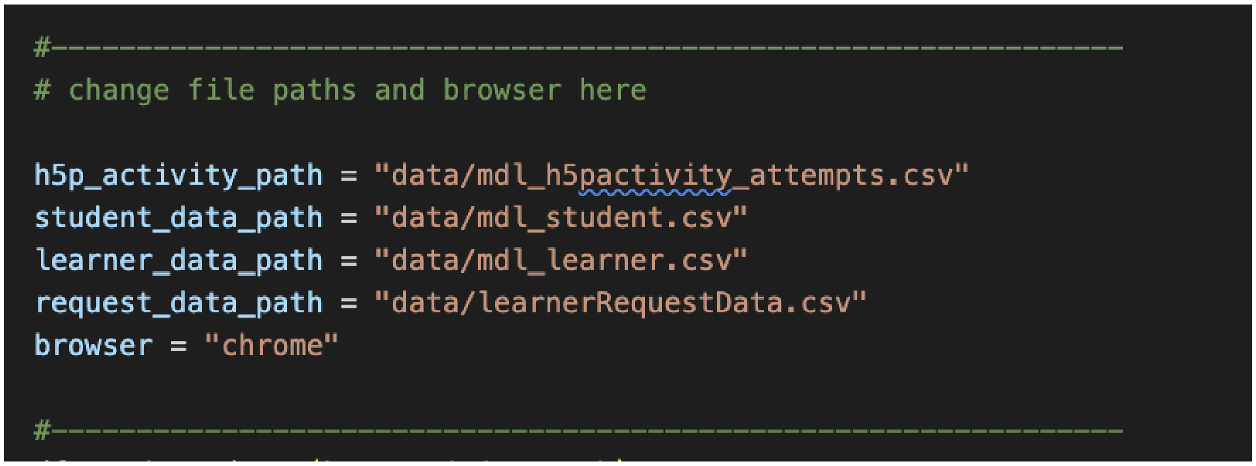
2) Change path name

For example, for the demographicsDashboard.py, we need 4 file "paths" to be updated. Make sure the name of the file after "data/" is exactly matching with that of the corresponding file in your "data" folder.

The four categories of data this dashboard takes in are: h5p_activity_attempts, student, learner, and learnerRequestData:

```
h5p_activity_path = "data/mdl_h5pactivity_attempts.csv"
student_data_path = "data/mdl_student.csv"
learner_data_path = "data/mdl_learner.csv"
request_data_path = "data/learnerRequestData.csv"
```

In our case here, our student dataset is saved as "mdl_student.csv" in the directory '/Users/alexoon/Desktop/DSS-x-TWB-Devi-S/data/mdl_student.csv' — so this is what we set as student_data_path in performanceDashboard.py (can be shortened to data/mdl_student.csv as the code is within the DSS-x-TWB-Devi-S folder).



```
#--------------------------------------------------------------
# change file paths and browser here

h5p_activity_path = "data/mdl_h5pactivity_attempts.csv"
student_data_path = "data/mdl_student.csv"
learner_data_path = "data/mdl_learner.csv"
request_data_path = "data/learnerRequestData.csv"
browser = "chrome"

#--------------------------------------------------------------
```