Deadline: October 17, 11:59 PM
Project instructions & requirements

Note: *Throughout this project, we are providing some direction for how you should complete each part of the project. If you complete the requirements in another way, with another program design, that is OK -- as long as it fulfills the stated requirements and/or passes required tests.*

We also recommend, like for Project 2, you create your own GitHub milestones and issues for this project!

Part 0 [200 points]

- There are 10 images on the page `http://newmantaylor.com/gallery.html`. Some of them have "alt text", which is the text that is displayed or spoken because of browser limitations, or because someone is using a screen reader, for example. Scrape this page and print out the alt text for each image. If there is no alt text, print "No alternative text provided!" The code you write should be general enough to work for any similar page with 10 images like this (not just this one), just by changing the URL to a different one.
- **IMPORTANT NOTE:** We may test your code on a version of `gallery.html` that has different alt-text. For example, it may be that the 8th image is missing alt text and the 7th image has the alt text "Waving Kitty 7", OR the first 3 images will be missing alt text, and the fourth image will have the alt-text "Waving Kitty 1"... There will still be 10 images, no matter what. You will receive 20 points for each correct alt text output.

- There are no *tests* for this, but the way the page is, you should see the following when this part of your program runs:

  Waving Kitty 1

  No alternative text provided!

  Waving Kitty 3

  Waving Kitty 4

  Waving Kitty 5

  Waving Kitty 6

  No alternative text provided!

  Waving Kitty 8

Waving Kitty 9

Waving Kitty 10

Part 1 [100 points]

- Access and cache data, starting from `https://www.nps.gov/index.htm`. You will ultimately need the HTML data from all the parks from Arkansas, California, and Michigan. So, you should save on your computer data from the following pages, in files with the following names:
    - Main page data, `https://www.nps.gov/index.htm`, in a file `nps_gov_data.html`
    - Arkansas, `https://www.nps.gov/state/ar/index.htm`, in a file `arkansas_data.html`
    - California, `https://www.nps.gov/state/ca/index.htm`, in a file `california_data.html`
    - Michigan, `https://www.nps.gov/state/mi/index.htm`, in a file `michigan_data.html`
- You should commit and push each of these `.html` files to your final Git repository.

- *Note* that this is a much less complex 'caching' system to save data from the internet on your computer than you may be accustomed to from accessing REST API data. A system like the one discussed in our textbook can *also*certainly be used for HTML data like this, but in this case, we're using a shortcut. Later in the course you'll see more options for structuring your code in an easily-reusable way, and you may also come up with some yourself.

- You should not hardcode the above URLs to get the html. Instead, you should write code that begins scraping `https://www.nps.gov/index.htm`. This code should be written such that you can pretty easily decide to add a new state, such as NY, to the states you want data from, and it would work.
- You can access, and thus cache, data from those three listed pages for AK, CA, and MI parks, by starting with a `BeautifulSoup` object of the HTML on the page `https://www.nps.gov/index.htm` -- and to get full points on this question, you should do that, rather than simply e.g. `resp_text = requests.get("https://www.nps.gov/state/mi/index.htm").text`, etc.
- If you access and cache the data without starting with a BeautifulSoup instance from the `https://www.nps.gov/index.htm` data, you will lose at least 50 points from this problem.

- We have provided comments as structure to proceed through Part 1. We suggest that you follow them. However, if you choose not to, as long as you end up with the result that is required (the files), and you are *not* accessing the internet to get HTML data from those 4 pages every time you run the program, you will get credit for Part 1.

Part 2 [400 points]

This part is probably the most challenging of the project, because it affects other parts of the project and requires design decisions! We suggest making a careful plan using GitHub issues or writing it out in English before beginning it, and thinking about Part 3 as well as you do so.

- Define a class `NationalSite` that accepts a BeautifulSoup object as input to its constructor, representing 1 National Park / National Lakeshore / etc (e.g. what you see here or what you see here)
- A `NationalSite` instance should have the following instance variables:
    - `location` (state, or a city, or states ... whatever location description is provided)
    - `name` (e.g. `"Alcatraz Island"`, `"Channel Islands"`...)
    - `type` (e.g. `"National Lakeshore"`, `"National Monument"`... if there is no specified type, this value should be the special value `None`)
    - `description` (e.g. `"Established in 1911 by presidential proclamation, Devils Postpile National Monument protects and preserves the Devils Postpile formation, the 101-foot high Rainbow Falls, and pristine mountain scenery. The formation is a rare sight in the geologic world and ranks as one of the world's finest examples of columnar basalt. Its columns tower 60 feet high and display an unusual symmetry."` -- if there is no description, this instance variable should have the value of the empty string, `""`)
- A `NationalSite` instance should also have the following methods:
    - A string method `__str__` that returns a string of the format **National Park/Site/Monument Name | Location**
    - A `get_mailing_address` method that returns a string representing the mailing address of the park/site/etc. Because a multi-line string will make a CSV more difficult, you should separate the lines in the address with a forward slash, like this: `/`. However you decide to get this information and relatively-sensibly put it together is fine. In fact, some addresses may have information included in them twice, e.g. "Yosemite National Park, CA 95389 / Yosemite National Park / CA / 95389", while some will not -- that is also OK! There is enough information to send mail if possible there, which is all that matters for our purposes: **is there some address info that will be returned in a single-line string from this function? If so, that is success.**

        - **HINT:** This address info can be found by clicking the **Basic Information** link that each park/site/monument specification has; even parks that have many locations have a specific mailing address. It looks like this for Old Spanish in California.

        - **NOTE:** If a park has no mailing address, the return value of this function should be the empty string ("").

    - A `__contains__` method that checks whether the additional input to the method is included in the string of the park's name. If the input *is* inside the name of the park, this method should return `True`; otherwise, it should return `False`.
    - *Note* that you may make additional design decisions when you define your class `NationalSite` to help you write these methods successfully -- e.g. you could add other instance variables or other methods if you wanted to/found them useful.
    - After you complete this, you should try creating an instance of your `NationalSite` class with the following code, to test and see if your class definition worked properly:

```
  soup_inst = BeautifulSoup(f.read(),'html.parser')```
```

Part 3 [200 points]

- Create a list of `NationalSite` objects from each one of these 3 states: Arkansas, California, and Michigan. They should be saved in the following variables, respectively:
    - `arkansas_natl_sites`
    - `california_natl_sites`
    - `michigan_natl_sites`

- (You may accumulate these lists in any way you prefer.)

- Write 3 CSV files, `arkansas.csv`, `california.csv`, `michigan.csv` -- one for each state's national parks/sites/etc, each of which has 5 columns:
    - Name
    - Location
    - Type
    - Address
    - Description

Remember to handle e.g commas and multi-line strings so that data for 1 field all ends up inside 1 spreadsheet cell when you open the CSV!

For any park/site/monument/etc where a value is `None`, you should put the string `"None"` in the CSV file.

Grading

To look at how each part of this HW will be graded, and to tell what score you will get when you submit it, there are few things you should note:

- Make sure you follow the instructions in this README

- Note the number of points pertaining to each section if you are curious about how many points each part is worth (though also note that some of the later parts depend upon earlier ones!)
- Run the tests file, `python si507f17_project_3_tests.py`, in this directory, to ensure that you pass the tests there
- Note that the tests do *not* cover every possible thing and are simply a guideline -- to tell whether everything is covered, you should ensure that the instructions are follows and your code works on sample data you've accessed in the way it should

To submit

[correct GitHub submission 100 points]

- Fork this repository (this is the GitHub repository link you should end up submitting)
- Clone the fork to your computer
- Make changes, add and commit them, and eventually push all your commits to GitHub before the deadline
- The files that should be in your repository when you submit should be at least:
    - This `README.md`, which you should *not* edit personally (unless you have incorporated changes from instructors via Git)
    - The file that runs your code, `si507f17_project3_code.py` (you may have other code files that you import into this one if you like, but this is the one that should ultimately run, to be tested and create file output)
    - The test file, `si507f17_project3_tests.py`, which you should *not* edit personally (unless you have incorporated changes from instructors via Git)
    - the 3 CSV files you have generated by running your code
    - the 4 HTML data files you generated & saved for data caching purposes (you may cache other data as well if you like, but these 3 are required)
- To submit the project to Canvas, submit the URL to this project, which should be exactly like this: `https://github.com/`**your-github-username**`/SI507-Project3.git`