SI 507 Project 2 | Fall 2018

Project Overview

You will create a program to scrape and search information about National Sites (Parks, Heritage Sites, Trails, and other entities) from nps.gov. You will also add the ability to look up nearby places using the Google Places API and to display National Sites and Nearby Places on a map using plotly.

Starter code:
proj2_nps.py
secrets.py
Test file:
proj2_nps_test.py
* Part of your project will be graded using this test file.

You only need to submit proj2_nps.py file.

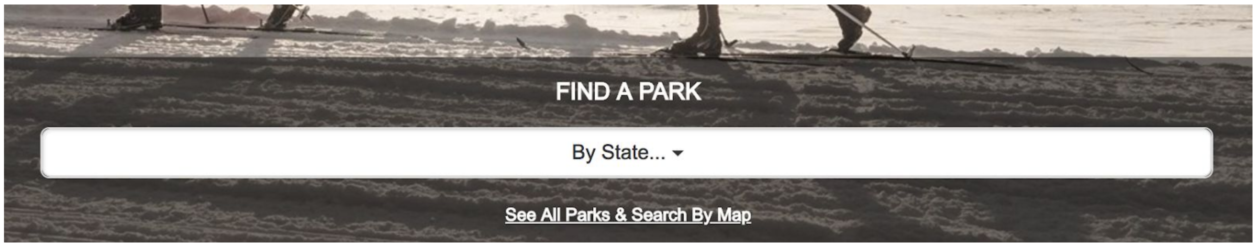
Also please observe the following:

- Do not change the name of the file proj2_nps.py
- Do not change any of the contents of the file proj2_nps_test.py
    - You can create other files, including other test files, if you would like, but you may not change this file or rename the main program file.

Failure to follow these guidelines may result in point deductions.

Part 1 (80 points)

In part 1 you will scrape nps.gov with the goal of being able to print out information about any National Site listed on the site, organized by state. Information will include the site name, site type, and the physical (or mailing) address. Your program will start crawling at https://www.nps.gov/index.htm, and from there crawl pages for particular states and then pages for particular sites. The links to state pages can be accessed from the dropdown box labeled “FIND A PARK.”



To pass the included tests, you will need to create a function `get_sites_for_state(state_abbr)` that takes a state abbreviation and returns a list of NationalSites that are in that state. The required attributes for the NationalSite class can be seen in the skeleton code file (proj2_nps.py).

At the basic level, each NationalSite (instance) should be created with a name, type (e.g., 'National Park,' 'National Monument', 'National Historic Site'), and description. All of these can be found on the landing page for a particular state (e.g., https://www.nps.gov/state/mi/index.htm).

In addition, you should visit the detail page for each site to extract additional information--in particular the physical address of the site. To do this, you will have to crawl one level deeper into the site, and extract information from the site-specific pages (e.g., https://www.nps.gov/isro/index.htm).

NationalSites should return a string representation of themselves (using `__str__( )`) of the following form: <name> (<type>): <address string>

For example:
```
Isle Royale (National Park): 800 East Lakeshore Drive, Houghton, MI
49931
```

Finally, though you should really consider doing this first to dramatically speed up your development time, implement caching so that you only have to visit each URL within nps.gov once (and subsequents attempts to visit, say https://www.nps.gov/state/mi/index.htm or https://www.nps.gov/isro/index.htm are satisfied using the cache rather than another HTTP request).

Grading

- [30 points] Implement basic searching by state and creation of NationalSites with `name` and `type`. Pass `TestStateSearch.test_basic_search( )`
- [30 points] Implement adding address information to NationalSites by crawling. Pass `TestStateSearch.test_addresses( )`

- [10 points] Implement __str__( ) as specified. Pass `TestStateSearch.test_str( )`
- [10 points] Add caching so that you never have to visit a page on the nps site more than once.

Part 2 (40 points)

Implement a function `get_nearby_places(site_object)` that looks up a site by name using the Google Places API and returns a list of up to 20 nearby places, where "nearby" is defined as within 10km (note: 20 results is the default maximum number returned by the Google Places API without paging).

Getting the list of nearby places will require two calls to the google API: one to get the GPS coordinates for a site (tip: do a text search for <site.name> <site.type> to ensure a more precise match--it turns out there are lots of places called "Death Valley" that aren't National Parks!), and another one to get the nearby places. Documentation on the Google Places API can be found here: https://developers.google.com/places/web-service/search.

You will need to get a Google API key following instructions here. Implementing caching for this portion of the project is STRONGLY recommended.

`get_nearby_places(site_object)` should return a list of NearbyPlace objects.

At a minimum, a NearbyPlace needs to have the name of the place as an attribute. You may find it useful to add other attributes as well. A NearbyPlace should include a `__str__( )` method, which simply prints the Place name.

Note:

- If you do a search on a NationalSite using <site name> <site type> (e.g., "Death Valley National Park" or "Motor Cities National Heritage Area") and Google Places does not return any results (or returns results, but none of them have the specific name you searched for), you the list of "Nearby Places" should be an empty list.

Grading:

- [30 points] Return a list of up to 20 NearbyPlaces from `get_nearby_places(site_object)`. Each place has a properly configured `name` attribute. Pass `test_nearby_search( )`.
- [10 points] adding caching so that you only do a particular nearby search once (items in your cache don't need to expire, even though technically the data could change given enough time.)

Part 3 (40 Points)

Use plotly to display maps of NationalSites within a state and NearbyPlaces near at NationalSite.

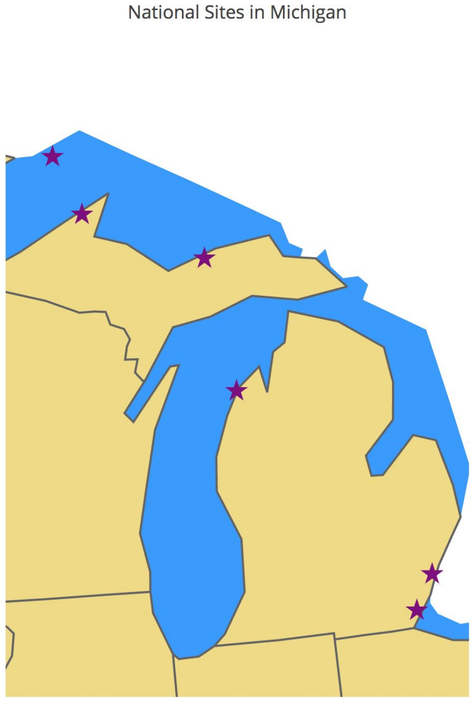
Implement two functions:

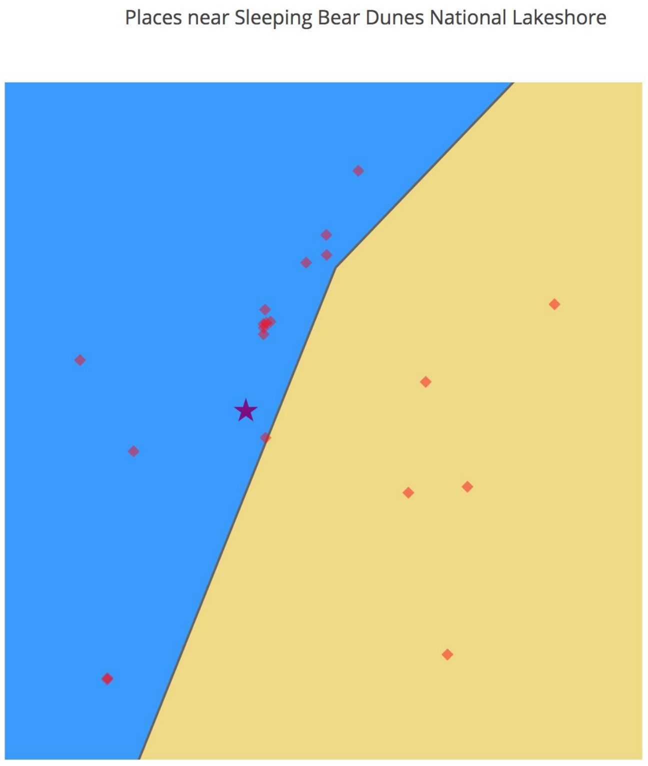
`plot_sites_for_state(state_abbr)` and `plot_nearby_for_site(site_object)`

Here are some details about each function:

- `plot_sites_for_state(state_abbr)`:
  - Takes a state abbreviation
  - Creates a map scatter plot on the plotly site that contains all of the NationalSites found for that state *that Google Places was able to find GPS coordinates for*.
    - Any Sites that don't have GPS coordinates should be removed before creating the map
  - The map should be centered and scaled appropriately so that all of the sites are visible and that there is a reasonable amount of "padding" around the edges of the map (i.e., so that all of the sites are comfortably within the map frame and not all the way at the edge)
  - All Sites should be displayed with the same type of marker, and each should display the name of the site when a user hovers over the marker (this is the default behavior in plotly if each data point has a 'text' field correctly set).
- `plot_nearby_for_site(site_object)`
  - Takes a NationalSite object
  - Creates a map scatter plot on the plotly site that contains all of the NearbyPlaces for the specified site.
    - If a NationalSite is provided that Google Places can't find GPS coordinates for, the map should not be created (you can handle the error however you deem appropriate)
  - The map should be centered and scaled appropriately so that all of the places are visible and that there is a reasonable amount of "padding" around the edges of the map (i.e., so that all of the sites are comfortably within the map frame and not all the way at the edge)
  - The NationalSite should be displayed with a *different* marker than the NearbyPlaces. Note that the NationalSite may be returned as a result by Google Places, in which case it needs to be removed before the map is plotted. The NationalSite and all NearbyPlaces should display their name when a user hovers over the marker in plotly.

Here are examples of each:
On the left is the result of calling `plot_sites_for_state('mi')`.
On the right is the result of calling `plot_nearby_for_site(NationalSite('National Lakeshore', 'Sleeping Bear Dunes'))`





Don't worry about the fact that some of the markers appear to be off by a few fractions of a degree. This seems to have something to do with the projection we are using for plotly in our code ('albers usa') which doesn't agree with the coordinates being produced by Google. If the data is correct and the maps are more or less in the right area, you will not get points off.

Grading:

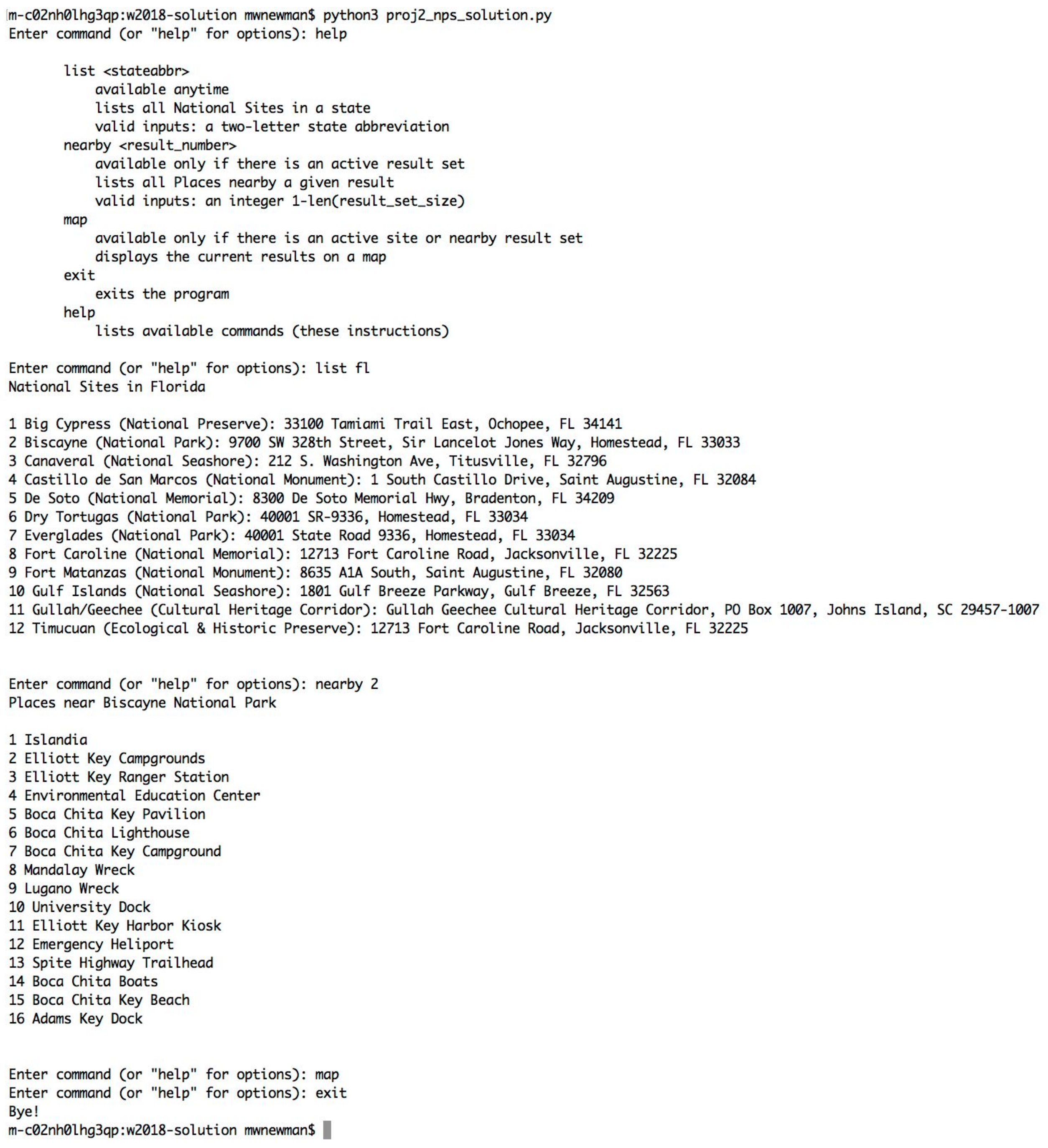
- [20 points] Correct implementation of `plot_sites_for_state(state_abbr)`
- [20 points] Correct implementation of `plot_nearby_for_site(site_object)`

Part 4 (40 Points)

Make the program interactive. Here is a list of commands your program should accept and how it should handle them:

```
        list <stateabbr>
        available anytime
        lists all National Sites in a state
        valid inputs: a two-letter state abbreviation
    nearby <result_number>
        available only if there is an active result set
        lists all Places nearby a given result
        valid inputs: an integer 1-len(result_set_size)
    map
        available only if there is an active result set
        displays the current results on a map
    exit
        exits the program
    help
        lists available commands (these instructions)
```

Note: a “result set” here refers to a list of NationalSites for a state or a list of NearbyPlaces for a NationalSite. You can implement this concept however you like, as long as the above semantics are preserved. **This part shouldn’t be run when running the test, but it needs to be run when running the proj2_nps.py file.** Here is a sample run of the program:



```
m-c02nh0lhg3qp:w2018-solution mwnewman$ python3 proj2_nps_solution.py
Enter command (or "help" for options): help

       list <stateabbr>
           available anytime
           lists all National Sites in a state
           valid inputs: a two-letter state abbreviation
       nearby <result_number>
           available only if there is an active result set
           lists all Places nearby a given result
           valid inputs: an integer 1-len(result_set_size)
       map
           available only if there is an active site or nearby result set
           displays the current results on a map
       exit
           exits the program
       help
           lists available commands (these instructions)

Enter command (or "help" for options): list fl
National Sites in Florida

1 Big Cypress (National Preserve): 33100 Tamiami Trail East, Ochopee, FL 34141
2 Biscayne (National Park): 9700 SW 328th Street, Sir Lancelot Jones Way, Homestead, FL 33033
3 Canaveral (National Seashore): 212 S. Washington Ave, Titusville, FL 32796
4 Castillo de San Marcos (National Monument): 1 South Castillo Drive, Saint Augustine, FL 32084
5 De Soto (National Memorial): 8300 De Soto Memorial Hwy, Bradenton, FL 34209
6 Dry Tortugas (National Park): 40001 SR-9336, Homestead, FL 33034
7 Everglades (National Park): 40001 State Road 9336, Homestead, FL 33034
8 Fort Caroline (National Memorial): 12713 Fort Caroline Road, Jacksonville, FL 32225
9 Fort Matanzas (National Monument): 8635 A1A South, Saint Augustine, FL 32080
10 Gulf Islands (National Seashore): 1801 Gulf Breeze Parkway, Gulf Breeze, FL 32563
11 Gullah/Geechee (Cultural Heritage Corridor): Gullah Geechee Cultural Heritage Corridor, PO Box 1007, Johns Island, SC 29457-1007
12 Timucuan (Ecological & Historic Preserve): 12713 Fort Caroline Road, Jacksonville, FL 32225


Enter command (or "help" for options): nearby 2
Places near Biscayne National Park

1 Islandia
2 Elliott Key Campgrounds
3 Elliott Key Ranger Station
4 Environmental Education Center
5 Boca Chita Key Pavilion
6 Boca Chita Lighthouse
7 Boca Chita Key Campground
8 Mandalay Wreck
9 Lugano Wreck
10 University Dock
11 Elliott Key Harbor Kiosk
12 Emergency Heliport
13 Spite Highway Trailhead
14 Boca Chita Boats
15 Boca Chita Key Beach
16 Adams Key Dock


Enter command (or "help" for options): map
Enter command (or "help" for options): exit
Bye!
m-c02nh0lhg3qp:w2018-solution mwnewman$
```

Grading:

- [8 points] Implement ‘list’ command correctly
- [8 points] Implement ‘nearby’ command
- [8 points] Implement ‘map’ command
- [4 points] Implement ‘help’ command
- [4 points] Implement ‘exit’ command
- [8 points] Handle bad inputs elegantly