

Winter2021 Project 2 - Scraping and Crawling nps.gov

You will create a program to scrape and search for information about National Sites (Parks, Heritage Sites, Trails, and other entities) from <https://www.nps.gov>. You will also add the ability to look up nearby places using the [MapQuest API](#).

Knowledge Required

The contents of each part are covered in lecture as follows:

- Part 1 & Part 2 (Scraping): Week 7/8
- Part 3 (Crawling): In the Week 9
- Part 4 (API & caching): Week 7/8
- Part 5 (Interactive interface): Week 2

Get started

You will clone the starter code and the test code from GitHub

GitHub link: <https://github.com/umsi-amadaman/Project2Winter2021>

Hint: `git clone https://xxxx`

Starter code: `proj2_nps.py`.

Test file: `proj2_nps_test.py`.

What to submit

Accept the GitHub Classroom assignment invitation. This will create a GitHub repo in your personal account that you will use to track and submit your project. Clone the repo and work on it, committing and pushing as you go. Make sure to push your final submission to the GitHub repo before deadline, and submit the repo URL on Canvas (this is for backup). In addition, since you need to push only `proj2_nsp.py` to GitHub, use `.gitignore` to exclude unnecessary files. Make sure to create your `.gitignore` *before* adding (or generating) any new files, such as your cache file (e.g., `cache.json`), `__pycache__`, or `secrets.py`.

Also, please observe the following:

- Since we will grade both with and without cache, do not forget to test your solution code with and without a cache file. (When you want to run without a cache, delete the cache file and run.) Do *not* push your cache file to GitHub.
- Use the test file (run `python/python3 proj2_nps_test.py`) for testing Part 1, 2, 3 and 4 of your project.
- You can add any functions with docstrings but must leave the existing functions, including names and parameters, as they are. This is required for testing.
- 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.

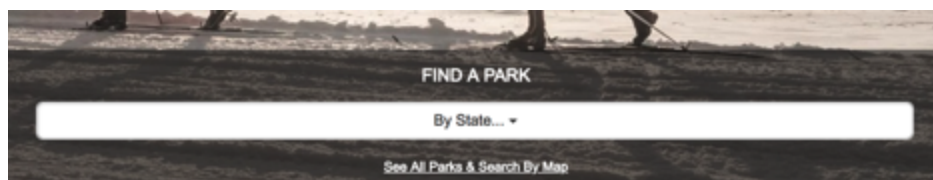
Part 1: Scrape state URLs

In part 1, you will scrape the page <https://www.nps.gov/index.htm> with the goal of being able to make a dictionary that maps state names to state page URLs.

The dictionary keys will be the state name (**in lower case**), and the value will be the URL for the state's page on `nps.gov`.

(e.g. `{'michigan': 'https://www.nps.gov/state/mi/index.htm', ...}`).

This dictionary should include all states listed on <https://www.nps.gov/index.htm>. The links to state pages can be accessed from the dropdown box under the label “FIND A PARK” (this is a clue to help you find the part of the page you will need to extract):



To pass the included tests, you will need to complete the implementation of `build_state_url_dict()` to return the correct dictionary.

Part 2: Create an instance of a national site

In part 2, you will scrape individual site pages (e.g.

<https://www.nps.gov/isro/index.htm> or <https://www.nps.gov/yell/index.htm>) with

the goal of being able to create instances of `NationalSite`. Each `NationalSite`

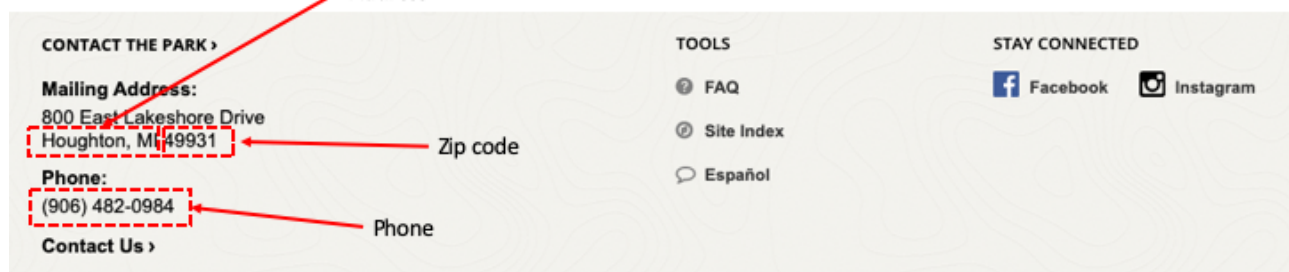
(instance) should have attributes on name, category (e.g., 'National Park,' 'National

Monument', or blank), address, zip code, and phone number. The required attributes for the `NationalSite` class can be seen in the starter code file in detail.

Header of the page



Footer of the page



`NationalSite` class should have a method `info()` that returns a string representation of itself.

The format is `<name> (<category>): <address> <zip> .`

Example: `Isle Royale (National Park): Houghton, MI 49931`

To pass the included tests, you will need to complete the implementation of `get_site_instance(site_url)` to return a `NationalSite` object.

Part 3: Crawling

In part 3, you will *crawl* `nps.gov` with the goal of being able to print out information about any National Site listed on the site, organized by state. The information will

include name, category, and mailing address. You will use functions that you implemented in Part 1 (`build_state_url_dict()`) and Part 2 (`get_site_instance()`) to achieve the goal of Part 3.

First, your program will ask a user to input a state name (case-insensitive).

Second, to pass the included tests, you will need to edit the function in the starter code `get_sites_for_state(state_url)` that takes a state page URL (e.g. "<https://www.nps.gov/state/az/index.htm>") and returns a list of `NationalSite` objects in the state page.

Third, based on the returned value from `get_sites_for_state(state_url)`, print national sites in the state in the following format: `[number] <name> (<type>): <address> <zip code>`.

Example: `[1] Isle Royale (National Park): Houghton, MI 49931`

Finally, implement caching so that you only have to visit each URL within `nps.gov` once (and subsequent attempts to visit, say <https://www.nps.gov/index.htm> or <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). Print `Using Cache` when you use cache data, and print `Fetching` when you make a HTTP request. (This will also have the side effect of dramatically speeding up your development time!)

Note

- Since you need to access multiple pages to create an object list, you are likely to print `Using Cache` or `Fetching` multiple times.
- You also need to modify functions in Part 1 and Part 2 to use caching.

Sample outputs

Case 1: Run without cache (upon first access)

```
(base) 0587340623:code tsuyoshikano$ python proj2_nps.py
Fetching
Enter a state name (e.g. Michigan, michigan) or "exit"
: Michigan
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
Fetching
-----
List of national sites in Michigan
-----
[1] Isle Royale (National Park): Houghton, MI 49931
[2] Keweenaw (National Historical Park): Calumet, MI 49913
[3] Motor Cities (National Heritage Area): Detroit, MI 48243
[4] North Country (National Scenic Trail): Lowell, MI 49331
[5] Pictured Rocks (National Lakeshore): Munising, MI 49862
[6] River Raisin (National Battlefield Park): Monroe, MI 48162
[7] Sleeping Bear Dunes (National Lakeshore): Empire, MI 49630
```

Case 2: Run with cache (upon subsequent accesses)

```

(base) 0587340623:code tsuyoshikano$ python proj2_nps.py
Using cache
Enter a state name (e.g. Michigan, michigan) or "exit"
: michigan
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
-----
List of national sites in michigan
-----
[1] Isle Royale (National Park): Houghton, MI 49931
[2] Keweenaw (National Historical Park): Calumet, MI 49913
[3] Motor Cities (National Heritage Area): Detroit, MI 48243
[4] North Country (National Scenic Trail): Lowell, MI 49331
[5] Pictured Rocks (National Lakeshore): Munising, MI 49862
[6] River Raisin (National Battlefield Park): Monroe, MI 48162
[7] Sleeping Bear Dunes (National Lakeshore): Empire, MI 49630

```

Part 4: Find nearby places

Implement a function `get_nearby_places(site_object)` that finds up to 10 places in or near the zip code of the national site's mailing address. The function will return a converted python dictionary from the MapQuest API.

In order to implement API request, go to [MapQuest API documentation](#) and find out the necessary URL and parameters. In order to make an API request, you need to obtain API key from [MapQuest](#) (click "Get your free API Key"). Once you get an API key, make `secrets.py` in the same folder with your project code and save the API key in `secrets.py`.

This is an example of `secrets.py`. (Replace "xxxxxxx" with your API key.)

```
API_KEY = "xxxxxxx"
```

For a MapQuest API request, you need to use the following five parameters.

- `key`: API key from `secrets.py`

- `origin`: Zip code of a national site (Use `NationalSite` instance attribute.)
- `radius`: Distance from the origin to search is 10 miles.
- `maxMatches`: The number of results returned in the response is 10.
- `ambiguities`: "ignore"
- `outFormat`: "json"

Based on the returned value from `get_nearby_places(site_object)`, print up to 10 places in the following format: `- <name> (<category>): <street address>, <city name>`. If a place doesn't have code or address, display "no category," as shown in Example 2.

Example 1: `- ALDI (Food Markets): 1850 N Telegraph Rd, Monroe`

Example 2: `- McIntyre Cemetery (no category): no address, no city`

Finally, implement caching so that you only have to call the API once each zip code. Print `Using Cache` when you use cache data, and print `Fetching` when you make an API request.

Sample output

```
- TA Monroe (Scales Public): I-75 Exit 15, Monroe
- Saint Joseph Cemetery (no category): no address, Monroe
- Doty Cemetery (no category): no address, no city
- Pilot Travel Center #284 (Scales Public): I-75 & Exit 18, Monroe
- McIntyre Cemetery (no category): no address, no city
- Custer Estates (Real Estate Agents): 1669 Mall Rd, Monroe
- Landings At Cedar Creek (no category): 1055 Cedar Creek Dr, Monroe
- Ruby Tuesday (no category): 2071 N Telegraph Rd, Monroe
- Benesh (Advertising Marketing): 1910 N Telegraph Rd, Monroe
- ALDI (Food Markets): 1850 N Telegraph Rd, Monroe
```

Part 5: Create an interactive search interface

For this last part, you will add the ability for users to enter their own queries and receive nicely formatted results. The steps of the interactive search interface are as follows.

[Step 1]

Ask a user to enter a state name (case-insensitive). If a user enters "exit", end the program. If a user enters an invalid state name, print an error and ask the user to input again.

Sample output of wrong input and exit

```
(base) 0587340623:code tsuyoshikano$ python proj2_nps.py
Using cache
Enter a state name (e.g. Michigan, michigan) or "exit" or "back":
: wrong name
[Error] Enter proper state name

Enter a state name (e.g. Michigan, michigan) or "exit":
: exit
(base) 0587340623:code tsuyoshikano$
```

[Step 2]

When a user inputs a proper state name, print a list of national sites in the state with a proper header (List of national sites in <state>) and numbering.

Sample output (when a user enters "michigan")

```
-----
List of national sites in michigan
-----
[1] Isle Royale (National Park): Houghton, MI 49931
[2] Keweenaw (National Historical Park): Calumet, MI 49913
[3] Motor Cities (National Heritage Area): Detroit, MI 48243
[4] North Country (National Scenic Trail): Lowell, MI 49331
[5] Pictured Rocks (National Lakeshore): Munising, MI 49862
[6] River Raisin (National Battlefield Park): Monroe, MI 48162
[7] Sleeping Bear Dunes (National Lakeshore): Empire, MI 49630
```

[Step 3]

Ask a user to enter a number in the list of Step 2. If a user enters "back", go to Step 1. If a user enters "exit", end the program. If a user enters an invalid number, print an error and ask the user to input again.

Sample output (an error case and "back" case)

```
Choose the number for detail search or "exit" or "back"
: 99999
[Error] Invalid input

-----
Choose the number for detail search or "exit" or "back"
: back
Enter a state name (e.g. Michigan, michigan) or "exit"
: 
```

[Step 4]

When a user inputs a proper number, print a list of (up to 10) nearby places with a header (Places near <national site>).

Sample output (when a user enters "michigan", then "6")

```
-----
Choose the number for detail search or "exit" or "back"
: 6
Fetching
-----
Places near River Raisin
-----
- TA Monroe (Scales Public): I-75 Exit 15, Monroe
- Saint Joseph Cemetery (no category): no address, Monroe
- Doty Cemetery (no category): no address, no city
- Pilot Travel Center #284 (Scales Public): I-75 & Exit 18, Monroe
- McIntyre Cemetery (no category): no address, no city
- Custer Estates (Real Estate Agents): 1669 Mall Rd, Monroe
- Landings At Cedar Creek (no category): 1055 Cedar Creek Dr, Monroe
- Ruby Tuesday (no category): 2071 N Telegraph Rd, Monroe
- Benesh (Advertising Marketing): 1910 N Telegraph Rd, Monroe
- ALDI (Food Markets): 1850 N Telegraph Rd, Monroe
```

[Step 5]

Repeat Step 3 and 4 until the user enter "exit".

Rubrics

Req	Part	Description	Category	Point Value
1	1	Function <code>build_state_url_dict</code> returns a dictionary. (Whether the code passes <code>test_1_1_return_type</code> or not)	Code	5
2	1	Function <code>build_state_url_dict</code> returns a dictionary that covers all the states listed in <code>nps.gov</code> . (56 items) (Whether the code passes <code>test_1_2_return_length</code> or not)	Code	10
3	1	Function <code>build_state_url_dict</code> returns a dictionary that maps state names to state page URLs. (e.g. <code>{'michigan': 'https://www.nps.gov/state/mi/index.htm', ...}</code>). (Whether the code passes <code>test_1_3_contents</code> or not)	Code	10
4	2	Function <code>get_site_instance</code> returns an instance of <code>NationalSite</code> that has proper <code>name</code> and <code>category</code> attributes. (Whether the code passes <code>test_2_1_basic</code> or not)	Code	10
5	2	Function <code>get_site_instance</code> returns an instance of <code>NationalSite</code> that has proper <code>address</code> and <code>zipcode</code> attributes. (Whether the code passes <code>test_2_2_address</code> or not)	Code	10
6	2	Function <code>get_site_instance</code> returns an instance of <code>NationalSite</code> that has proper <code>phone</code> attribute.	Code	10

		(Whether the code passes <code>test_2_3_phone</code> or not)		
7	2	Function <code>get_site_instance</code> returns an instance of <code>NationalSite</code> that has a method <code>info()</code> to print the site information. (e.g. <code>Isle Royale (National Park): Houghton, MI 49931</code>) (Whether the code passes <code>test_2_4_str</code> or not)	Code	10
8	3	Function <code>get_sites_for_state</code> returns a list (Whether the code passes <code>test_3_1_return_type</code> or not)	Code	5
9	3	Function <code>get_sites_for_state</code> returns a list with proper length. (e.g. Michigan has 7 sites and Wyoming has 10 sites.) (Whether the code passes <code>test_3_2_length</code> or not)	Code	10
10	3	Function <code>get_sites_for_state</code> returns a list of <code>NationalSite</code> instances. And the instance has proper attributes (i.e. name, category, address, zipcode, and phone). (Whether the code passes <code>test_3_3_contents</code> or not)	Code	10
11	4	Function <code>get_nearby_places</code> returns a dictionary (Whether the code passes <code>test_4_1_basic</code> or not)	Code	10
12	4	Function <code>get_nearby_places</code> returns a dictionary with the contents of a MapQuest API response that has keys <code>resultCount</code> and <code>options</code> with proper values. (Whether the code passes <code>test_4_2_contents</code> or not)	Code	10
13	5	[step 1] When a user inputs an invalid state name, print an error message and ask the user	Behavior	5

		to input again.		
14	5	[step 1] When a user inputs “exit”, end the program.	Behavior	5
15	5	[step 2] When a user input a valid state name, print a header with a state name, such as <code>List of national sites in michigan</code>	Behavior	5
16	5	[step 2] When a user input a valid state name, print a list of national sites in any format.	Behavior	5
17	5	[step 2] When a user input a state name, prints a list with the following format. <code>[number] <name> (<type>): <address></code> <code><zip></code> e.g. <code>[1] Isle Royale (National Park):</code> <code>Houghton, MI 49931</code>	Behavior	5
18	5	[step 2] Prints <code>Using cache</code> or <code>Fetching</code> appropriately, depending on whether the state has been accessed previously.	Behavior	10
19	5	[step 3] When a user inputs an invalid number, print an error and ask the user to input again	Behavior	5
20	5	[step 3] When a user inputs “exit”, end the program.	Behavior	5
21	5	[step 3] When a user inputs “back”, go back to step 1 (choose a state), and new state search works.	Behavior	5
22	5	[step 4] When a user inputs a valid number, to choose a National Site, print header with a site name, such as <code>Places near Isle Royale.</code>	Behavior	5
23	5	[step 4] When a user inputs a valid number, print nearby place list (probably 10 lines but may be less in some cases).	Behavior	5
24	5	[step 4] When a user inputs a valid number, print	Behavior	5

		nearby place list with the following format. - <name> (<category>): <street address>, <city name> e.g. - BP Station (Gas Stations): ST HWY 26, Houghton		
25	5	[step 4] Prints Using cache or Fetching appropriately, depending on whether the API request has been accessed previously.	Behavior	10
26	5	[step 5] After step 4, go to step 3 (ask for a National Site number again) and it works.	Behavior	5
27	ALL	Submitted GitHub repo has only <code>pj2_nps.py</code> and <code>pj2_nps_test.py</code> . Or, the repo has <code>.gitignore</code> that mentions other unnecessary files (such as <code>secrets.py</code> , a cache file, and <code>__pycache__</code>). (The second choice is for students who mistakenly pushed other files. You don't need to delete them from your GitHub repo.)	Behavior	5
28	ALL	Well-constructed code that follows our guidelines. — including docstrings!	Code	5
		Total		200