SI 506 - Programming I
Final Project Plan

SAMPLE SHEET

NOTES - DO READ:

1. **The contents of this file are subject to change. It is meant as an illustration of the work we expect from you.**
2. All the code related to the NYT has been omitted.
3. The code to demonstrate part 2 of the project plan relies on one file: *oxford_api_credentials.py*.
4. If you want to test the code, you need to create your own credentials for the OED and add the information in the corresponding places on this file.
5. Only then will you be able to run *SI506W18_practice_project.py*.
6. We have also provided a sample json object returned from the API so that you can practice some data exploration

Part 1

Which option are you choosing? **3**

Write a sentence or two describing what your project will do. (Even if you are choosing Option 1 or 2, you should articulate this yourself!)

Based on a search for something in the New York Times (NYT), I will find the five most popular keywords and then find their exact definitions from the Oxford English Dictionary (OED), and put those in a CSV file.

What 2 classes are you going to define? For each, you should list:

- What will one instance of the class represent?
- What input will the class constructor need in order to create an instance?
- What are all the instance variables the class will have, and what data will each of them contain?
- What methods will the class have? For each method, describe: will it take any input besides `self`? What will it return, if anything? What will it do (e.g. if it will change the value of an instance variable)?

1. Story class

One instance of this class will represent the contents of a news article published by the NYT.

The class constructor will need a dictionary holding all the information about one single article. This dictionary will be extracted from a request to the NYT REST API.

Each instance should hold info about the article — URL, headline, author's name, published date, key terms, lead (first paragraph), and the article's length.

Instance variables:

1. URL. String. The URL pointing to the article.
2. Headline. String. The article's title.
3. Author. The writer's name.
4. Published. String. A string representing the date and time the article was posted in the format mm/dd/yyyy.hh:mm. Hours will be represented from 0 to 23. This is available in the data.
5. Keywords. List. A list of strings where each element will represent a key term in the given article (from the NYT data).
6. Lead. String. The first paragraph of the article.
7. Length. Int. Total words in the article.

Methods:

1. Constructor **__init__.** The constructor method, or __init__ will have the input parameters **self**, and a dictionary that represents all the data of a single article. Each dictionary will be extracted directly from the JSON object retrieved from the NYT API. The constructor will parse out slices of data from this dictionary and assigned those slices to initialize instance variables as needed. The method will return None and won't print anything.
2. **most_common_word**. This method takes in only **self**. It calculates, out of all the words in a article's abstract, which is the most common word. Then it returns the string representing that word. In case there is no abstract, it will return the the value None.
3. String method, or __str__. It takes a class instance as an input (self). It returns a string representing identifying information about the article, Like so: "Story object posted by <author's name>, on <timestamp string>. Its keywords are <sequence of keywords as strings separated by commas> and a total of <article_length> words."

2. Word class

One instance of this class will represent a word that has been retrieved from the Oxford API.

This class will hold data such as: lexical categories (e.g. Noun, Verb, Dictionary), a series of definitions for every lexical category, and the domains to which each definition corresponds (the context in which this definition applies).

The class constructor will need a dictionary representing all the information about an individual Word. This dictionary will be extracted from a json object returned from a request to the Oxford API.

Instance variables:

1. Word_id. String. The word searched.
2. Lexical categories. List. A list of strings where each element of the list represents a lexical category (e.g. noun, verb, adjective) of the word searched. A word could have more than one possible lexical category, hence this value being a list.
3. Definitions. Dictionary. A dictionary where the keys represent a lexical category, and the associated values are lists of strings. Each element in each list is a definition for that word, corresponding to the lexical category.
4. Domains. Dictionary. A list of dictionaries where the keys are strings representing the domain to which a particular definition applies, and the values are strings representing the definition corresponding to that domain.

Methods:

1. Constructor, or **__init__**. It takes a class instance (self) and a dictionary as inputs, and returns None. The constructor parses out the dictionary to extract slices of data and assign those to the corresponding instance variables. This method does not print anything.
2. String, or **__str__**. It takes a class instance as input (self) and returns a string as output. The output string represents basic metadata to identify the Word object, like so: “The word <word> can be used as a <sequence of strings representing each of the lexical categories> has a total of <total number of definitions for all lexical categories> lexical categories. It is used in the these domains: <comma separated sequence strings indicating the domains of every definition for this word.”
3. **categories_domains_definitions**. It takes a class instance (self) and a string representing a lexical category as input. It returns a dictionary. The only key in this dictionary is the word searched, and the corresponding value is a list of strings where each item is a definition that matches the lexical category specified as an argument (e.g. verb, adverb, pronoun).

What functions OUTSIDE class definitions will you define (INCLUDING any functions that get and cache data from the internet)?

For each function, you should list: What it will be called, what input it will accept (note whether it will have any parameters with default values), what its return value will be, and a brief description of what the function does.

1. ***get_from_nyt*.**
    a. It will take one argument as input: a search term (string)
    b. It will rely on a global variable storing my NYT API key
    c. It will return a list of Story instances.
    d. Inside, the function will take the parameters to search whether the data exists in the cache. If the data is present, I will retrieve it as a json object. Otherwise, the function will make a call to the NYT API, and then update the cache in working memory. The update cache will have a new key-value pair, where the key will be the unique identifier of the query (see function get_unique_identifier), and the associated value will be the json object retrieved from this search. This json object will have a list of dictionaries, each bearing information about a NYT news article. (This is the information that the function will use to create, and ultimately return, a list of Story instances.)
    e. The function will print a message to indicate the exact moment the request is being made, when the cache is being overwritten, and whether the query was retrieved from the cache (which means the request had already been made before).
2. ***Get_from_oxford*.**
    a. It will take one argument: a search term (str).
    b. It will rely on a global variable storing my Oxford API key
    c. It will return a list of Word instances. The list will be created outside of the function, so that when get_from_oxford is called, the function merely expands the existing list of Word instances with the information from the most recent search.
    d. Inside, the function will take the parameters to search whether the requested data exists in the cache. If the data is already available, the function will retrieve it as a json object. Otherwise, the function will make a call to the Oxford API, and then update the cache in working memory. The update cache will have a new key-value pair, where the key will be the unique identifier of the query (see function get_unique_identifier_itunes), and the associated value will be the json object retrieved from this search. This json object will have a list of dictionaries, each bearing information about a Word. The function will take the information about multiple words and create with it a list of word instances.
    e. The function will print a message to indicate the exact moment the request is being made, when the moment the cache is being overwritten, and whether the query was retrieved from the cache (which means the request had already been made before).
3. ***params_unique_combination*.** You don't need to list it. We assume and expect you to use it.

What list of data will you be sorting? (Explain at least one, if you are sorting more than one list.) What is the type of the elements of the list, and how will you be sorting that list?

I will create a list of Story objects, where each object represents an article published on the NYT. One of the instance variables in these objects will be the most common word, which will represent the most common word in the article's abstract. I will sort the Story objects using this instance, and then make a call to the get_from_oxford function for every element in the sorted list.

What will your project's file output consist of? Explain the file output, what data it will contain. Include at least 1 sentence about what you will need to do to create the file.

The output of my file will consist on a CSV file. Each entry on this file will be data about a word, its definition, and the information of the article in the NYT were it was used. The file will contain the following columns:

1. Entry ID (a unique number starting from 1, and increases with every new entry into the file). INT
2. Word. STR
3. Lexical category. STR
4. Domain. STR
5. Definition. STR
6. Headline of article it appeared in. STR — headline of the article where the word was used.
7. Longest keyword. STR — the longest keyword in the article
8. Date published. STR — date the article was published in the format HH:MM.

Briefly explain how you will organize and plan/work through your code: what needs to go at the top of the file? What code will you write first? What code will you write next? etc. (OK to be as general as "I will write the class definition, and then…" no need to be specific about actual literal code, just describe how you plan to work through the process.)

At the top of the file, I will import the following modules:

codecs, sys
requests
json

requests_oauthlib
webbrowser

After the import statements, I will include my global variables:

OUTPUT_FNAME # for long term memory
CACHE_FNAME # for long term memory
CACHE_DICTION # for working memory
After the global variables, I will write a try/except block to make sure I load the cache file to working memory, if it already exists, or create a cache for the working memory if that file does not exist yet.

After this setup, I will write the class definitions and then the functions I described above.

Once classes and functions are defined I will proceed as follows:

1. I call get_data_from_nyt so that I can create a list of Story instances.
2. Sort the list of Story instances by the most frequent word on each.
3. Create an empty list of Word instances.
4. For every Story object, call get_data_from_oxford, and pass on as an argument the most frequent word of a given Story object
5. Open final project output file (CSV) in writing mode
6. Write the first line with a composite string that properly labels the columns of each of the 8 pieces of data described as the output of the program.
    a. The strings representing column names should be separated by commas
    b. At the end of the last column name (string), add a new line character “\n”
7. Iterate through the list of Word instances. For every instance, the program will
    a. Write to the output file a composite string with the pieces of data that correspond to each of the 8 columns (entry ID, word, lexical category, etc.)
    b. Call the string method in the current Word object by printing it.
8. Close the project output file