CSE 101: Introduction to Programming
Home Assignment 1

Introduction to Weather Web Service

Consider an overseas vacation that you are planning this year. It is best to go when the weather condition is in your favor.

For this assignment, you will use a weather web service which provides the weather condition for a particular location in the world. Before we get into the details of the assignment, it would be good if you experiment with the web service.

To use the service, you employ special URLs that start with the following prefix:

http://api.openweathermap.org/data/2.5/forecast?

This prefix is followed by a weather query. A weather query has two pieces of information in the following format (without spaces; we have included spaces here solely for readability):

q=location & APPID=API Key

where location is name of any city in the world and API key is the interface to the web service. For example, if you want to know the weather condition of Delhi, the query is

q=Delhi&APPID=2ab136be1543b5789451a5994364c0d3

The full URL for this query is
http://api.openweathermap.org/data/2.5/forecast?q=Delhi&APPID=2ab136be1543b5789451a5994364c0d3
Click on the link to see it in action.

You will note that the “web page” in your browser is multiple line in the following format:
{"temp":300.71,"temp_min":300.71,"temp_max":301.189,"pressure":992.81,"sea_level":1017.24,"grnd_level":992.81,"humidity":96,"temp_kf":-0.48}

This is what is known as JSON representation of the answer. JSON is a way of encoding complex data so that it can be sent over the internet. You will use what you know about operations and methods in string module to pull out the relevant data out of JSON string.

You should try a few more weather queries to familiarize yourself with the service.
Note: You would need a API key to access a web service. To get your own API key, visit http://openweathermap.org/appid and sign up to get the API key. For a good response, use a key once in 10 minute.

Note that if you enter an invalid query, you will get the following response in error:
{"cod":"404","message":"city not found"}

In some cases the service returns a response that seems valid but it contains a wrong city name.

Tasks in the Assignment:

Task 1

For this task, maintain a module in a1.py and implement the functions described below.
Your primary goal in this assignment is to use the weather service to write the following function:

Part A: Weather Query

Your script would interact with the web service. In this part, you will implement a single function.

```
def weather_response(location, API_key):
    """Returns: a JSON string that is a response to a weather query.

    A weather query returns the JSON with weather attribute like temperature, pressure,
    humidity and wind speed etc.

    Parameter location: the city name
    Precondition: city name should be correct

    Parameter API_key: a valid key of weather service
    Precondition: for better response use one key once in 10 minute.
    """
```

While this function sounds complicated, it is not as bad as you think it is. You need to use the urlopen function from the module **urllib.request**. This function takes a string that represents a URL and returns an instance of the type addinfourl that represents the web page for that url. This object has the following methods:

Method	Specification
geturl()	**Returns:** The URL address of this web page as a string.
read()	**Returns:** The contents of this web page as a string.

Using one or both of these methods (you might not need them both) is enough to implement the function above.

Part B: Processing JSON data

Once you retrieve the json string for your query, you need to process this string in order to get the attribute.

In case you want to learn more about JSON data type you can refer to the guide on using json module to process the data encoded in JSON. However you will be using on String module functions for your work in this assignment. http://docs.pythonguide.org/en/latest/scenarios/json/

Note: You need to process the response using only string operation. Module json can not be used in this assignment.

Note that you also need to deal with cases when the user query is invalid(eg. location is invalid etc.) To perform this error check, you need to implement a function described below:

has_error(location,json)
Returns: True if the input location and response city name is not matched
Given a JSON response to a weather query, this returns the opposite of the value following the keyword "valid". For example, if the input is “123” and the part of JSON which contain **city** attribute i.e.

"city":{"id":7522437,"name":"Aegviidu vald","coord":{"lat":59.2794,"lon":25.6257},"country":"EE"}}
attribute “name: Aegviidu” does not match with input location “123”.

then the query is not valid, so this function returns True (city name in response is not equal to the input location)
Parameter json: a json string to parse
Precondition: json is the response to a weather query

PART 3: Getting attribute

The task is to get the attribute from the string for nth day at t^{th} time from current date. Current date can be accessed by datetime module in python.

a) Get the temperature:

```
get_temperature (json, n=0,t)
""" Return the temperature of nth day from the current date.
    Parameter json: a json string to parse
                    n: day from the the current day, valid values are 0 to 4
                    t : time, valid value are 3:00, 6:00, 9:00, 12:00, 15:00, 18:00, 21:00
    Precondition: json is the response to a weather query
"""
```

b) Get the humidity

```
get_humidity(json, n=0,t)
""" Return the humidity of nth day from the current date.
    Parameter json: a json string to parse
              n: day from the current day
              t : time, valid value are 3:00, 6:00, 9:00, 12:00, 15:00, 18:00, 21:00
    Precondition: json is the response to a weather query
"""
```

c) Get the pressure

```
get_pressure(json, n=0,t)
""" Return the pressure of nth day from the current date.
    Parameter json: a json string to parse
              n: day from the current day
              t : time, valid value are 3:00, 6:00, 9:00, 12:00, 15:00, 18:00, 21:00
    Precondition: json is the response to a weather query
"""
```

d) Get the Wind

```
get_wind(json, n=0,t)
""" Return the wind of nth day from the current date.
    Parameter json: a json string to parse
              n: day from the current day
              t : time, valid value are 3:00, 6:00, 9:00, 12:00, 15:00, 18:00, 21:00
    Precondition: json is the response to a weather query
"""
```

e) Get the sea level

```
get_sealevel(json, n=0,t)
""" Return the sea level of nth day from the current date.
    Parameter json: a json string to parse
              n: day from the current day
              t : time, valid value are 3:00, 6:00, 9:00, 12:00, 15:00, 18:00, 21:00
    Precondition: json is the response to a weather query
"""
```

Task 2

Once you implement the functionalities, you must test them. Implement the test code in a1test.py, using the unittest framework. This exercise would be similar to what you did for Lab3. You should define test procedures, with **at least 5 test cases for each**. The description of test procedure is given below:

1. testresponse: Check for a json response to the query i.e write test cases for weather_response(location, API_key) function.
2. testquery: Check validity of a query i.e write test cases for has error(json) function.
3. test_temp, test_humidity, test_pressure, test_wind, test_sealevel: Check correctness of attribute value i.e write test cases for all get attribute cases.

Grading Policy

In grading your code, we will focus on the following:

- Correct function specifications and/or formatting
- Adequate test cases quality of test cases(are they extensively testing your application?)
- Correctness of the code (does it pass our test cases?)

Submission Instructions

Make sure you mention the following details as comments at the top of both the modules.

1. Name and Roll number of student

Zip a1.py and a1test.py and upload the zipped file on the deadline link.