CSE101: Introduction to Programming
Home Assignment 1

Introduction to Currency Exchange Web Service

Consider an overseas vacation that you are planning this year. It is best to go when the exchange rate is in your favor. When your dollars buy more in the foreign currency, you can do more on your vacation. This is why it would be nice to have a function that, given your current amount of cash in US dollars, tells you how much your money is worth in another currency. However, there is no set mathematical formula to compute this conversion. The value of one currency with respect to another is constantly changing. In fact, in the time that it takes you to read this piece of text, the exchange rate between the dollar and the INR has probably changed several times. How on Earth do we write a program to handle something like that?
One solution is to make use of a *web service*. A web service is a program that, when you send it web requests, automatically generates a web page with the information that you asked for. In our case, the web service will tell us the current exchange rate for most of the major international currencies. Your job will be to use string-manipulation methods to read the web page and extract the exact information we need.

Before we get into the details of the assignment, it would be good if you experiment with the web service. For this assignment, you will use a simulated currency exchange service that never changes values. This is important for testing; if the answer is always changing, it is hard to test that you are getting the right answers.

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

http://cs1110.cs.cornell.edu/2015fa/a1server.php?

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

from=*source* & to=*target* & q=*amount*

where *source* is a three-letter code for the original currency, *target* is a three-letter code for the new currency and *amount* is a float value for the amount of money in the original. For example, if you want to know the value of 2.5 dollars (USD) in Indian Rupee (INR), the query is

from=USD&to=INR&amt=2.5

The full URL for this query is
http://cs1110.cs.cornell.edu/2015fa/a1server.php?from=USD&to=INR&amt=2.5
Click on the link to see it in action.

You will note that the "web page" in your browser is just a single line in the following format:
{ "lhs" : "2.5 United States Dollars", "rhs" : "160.213875 Indian Rupees", "valid" : true, "error" : ""
}
This is what is known as a 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 and json module to pull out the relevant data out of the JSON string.

You should try a few more currency queries to familiarize yourself with the service.
Note: The currency exchange table, containing all the valid currency codes, is provided along as html file. Use a web browser to view it.

Note that if you enter an invalid query (for example, using a non-existent currency code like "AAA"), you will get the following response in error:

```
{"lhs":"", "rhs":"", "valid":false, "error":"Source currency code is invalid." }
```

Similarly, if you enter a query with two valid currency codes, but with an invalid quantity value, you will get the following error:

```
{"lhs":"", "rhs":"", "valid":false, "error":"Currency amount is invalid." }
```

For all error queries, the "lhs" and "rhs" values are blank, while "valid" is false. The value "error" is a specific error message describing the problem. This will be important for error handling in this assignment.

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 currency exchange service to write the following function:

```
def exchange(currency_from, currency_to, amount_from):
    """Returns: amount of currency received in the given exchange.
    In this exchange, the user is changing amount_from money in
    currency currency_from to the currency currency_to. The value
    returned represents the amount in currency currency_to.
```

```
    Return -1 for invalid currency code or invalid amount_from value

    The value returned has type float.

    Parameter currency_from: the currency on hand
    Precondition: currency_from is a string for a currency code.

    Parameter currency_to: the currency to convert to
    Precondition: currency_to is a string for a currency code

    Parameter amount_from: amount of currency to convert
    Precondition: amount_from is a float"""
```

This function will involve several steps. You will get the JSON string from the web service, break up the string to pull out the numeric value (as a substring), and then convert that substring to a float. For each of these steps, you should define a separate function and make call to it in appropriate function(s). These functions are described below in three parts.

Part A: Currency Query

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

```
def currency_response(currency_from, currency_to, amount_from):
    """Returns: a JSON string that is a response to a currency query.

    A currency query converts amount_from money in currency currency_from
    to the currency currency_to. The response should be a string of the form

        '{"lhs":"<old-amt>","rhs":"<new-amt>","valid":true, "error":""}'

    where the values old-amount and new-amount contain the value and name
    for the original and new currencies. If the query is invalid, both
    old-amount and new-amount will be empty, while "valid" will be followed
    by the value false.

    Parameter currency_from: the currency on hand
    Precondition: currency_from is a string

    Parameter currency_to: the currency to convert to
    Precondition: currency_to is a string

    Parameter amount_from: amount of currency to convert
    Precondition: amount_from is a float"""
```

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 class 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 amount value of exchanged currency. Refer to the guide on using json module to process the data encoded in JSON: http://docs.python-guide.org/en/latest/scenarios/json/
Note that you also need to deal with cases when the user query is invalid(eg. Currency code is invalid etc.) To perform this error check, you need to implement a function described below:

has_error(json)

Returns: True if the query has an error; False otherwise.
Given a JSON response to a currency query, this returns the opposite of the value following the keyword "valid". For example, if the JSON is
'{"lhs":"","rhs":"","valid":false,"error":"Source currency code is invalid."}'
then the query is not valid, so this function returns True (It does NOT return the message 'Source currency code is invalid').
Parameter json: a json string to parse
Precondition: json is the response to a currency query

Part C: Breaking Up Strings

A large part of this assignment is breaking up a JSON string. Conceptually, you want to separate the currency amount from the currency name. For example, if we are given the string "0.912968 Euros"
Then we want to break it up into "0.912968" and "Euros".

---

before_space(s)

Returns: Substring of s; up to, but not including, the first space
Parameter s: the string to slice
Precondition: s has at least one space in it

---

after_space(s)

Returns: Substring of s after the first space
Parameter s: the string to slice
Precondition: s has at least one space in it

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 **atleast 5 testcases 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 currency_response(currency_from, currency_to, amount_from) function.
2. testquery: Check validity of a query i.e write test cases for haserror(json) function
3. testexchange: Check correctness of exchange rate i.e write test cases for exchange(currency_from, currency_to, amount_from) function.

You might find it convenient to use a currency query URL to look up the correct answer, and then paste the answer into your test case.

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 both the partners

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