IST 256: Application Programming for Information Systems

Professors Fudge / Ramnarine-Rieks

Group 28 (Julia Flores, Jose Velasquez & Bernard Akinboyewa)

Project iTravel App

Problem: Having a singular location to access hotel, restaurant, tourist attraction and currency exchange information based on a destination of travel.

Input: Broken into several functions to simplify code.

1st Function: Currency Exchange defined as *currency\_exch(exch\_opt)*

* Argument to initiate function is exch\_opt, which is derived from user input on whether they choose to lookup exchange information.
* Function will then require a user to input an amount to be exchanged
* Function will then require user to select a base currency for the exchange, based on supported currencies
* Function will the require user to select an output currency, based on the supported currencies

2nd Function: Exchange rate defined as *get\_exchange\_rate(base\_currency, target\_currency)*

* Argument to initiate function is the base\_currency and target\_currency

3rd Function is the use of the Google Places Module within python defined as *ping\_gplaces(option, latlng)*

* Arguments to initiate this function are option and latlng, where option changes based on what will be searched, using the module, and latlng being the coordinates for the search.

4th Function Google Maps geocoding defined as *geocode(location)*

* Argument to initiate the location (i.e. city, state) based off user input

Main code inputs:

* Location for travel destination
* Option for hotel search
* Option for restaurant search
  + Option for specific type of restaurant
  + Input for type of restaurant
* Option for transportation search (i.e. taxi)
* Option for tourist attraction search
* Option for currency exchange lookup

Output:

1st Function: Currency Exchange defined as *currency\_exch(exch\_opt)*

* Prints the amount of target currency based off exchange rate times amount entered

2nd Function: Exchange rate defined as *get\_exchange\_rate(base\_currency, target\_currency)*

* Returns JSON data

3rd Function is the use of the Google Places Module within python defined as *ping\_gplaces(option, latlng)*

* Prints specific details about *option* (i.e. attributions, name, rating, address, local phone number, international phone number, website url (if available) and google places url)

4th Function Google Maps geocoding defined as *geocode(location)*

* Returns coordinates of *location*

Main code outputs:

* A combination of outputs from functions
* Statements to identify errors in input
* Statements to identify errors within code processing

Algorithm:

1st Function

1. Define function as currency\_exch, with exch\_opt as arguments
2. Create dictionary for supported currencies*,* with currency code as dictionary key and currency name as dictionary value (*SUPPORTED\_CURRENCIES*)
3. Create dictionary for currency code, with numbers being dictionary keys and currency code being dictionary values (*CURRENCY\_CODES)*
4. Initiate bool to control code flow
5. Create input (as float) for amount to be converted (amount)
6. Print statement to prompt user to choose a code
7. Start indefinite loop to control the base currency input
   1. Start definite loop
      1. Print each item in CURRENCY\_CODE dictionary in format key, value
   2. Create variable for user input to choose a code for the base currency (base\_currency\_code)
   3. Bool if selected option in CURRENCY\_CODE dictionary keys, break loop
   4. Bool else print error message
   5. Create variable and assign *CURRENCY\_CODES[base\_currency\_code] (base\_currency)*
8. Repeat steps 7 & 8 but for target currency instead of base currency
9. Create variable to call function 2 *(exchange\_rate)*

\*\*\*THIS FUNCTION DOES NOT RETURN SPECIFIC DATA, INSTEAD IT IS USED TO RUN A SEPARATE CODE\*\*\*

2nd Function

1. Define function as *get\_exchange\_rate*, with arguments *base\_currency* and target currency
2. Bool if *base\_currency* is not in *SUPPORTED\_CURRENCIES.keys()* raise ValueError
3. Bool if *target\_currency* is not in *SUPPORTED\_CURRENCIES.keys()* raise ValueError
4. Bool if *base\_currency* == *target\_currency*, return 1
5. Create variable to assign url for api (*api\_url*)
6. Create variable to send request to api (*api\_response*)
7. Bool if api\_response.statuse code == 200, return rates of target currency

3rd Function

1. Define function as *ping\_gplace*, with arguments of *option* and *latlng*
2. Use *try to attempt the following*
   1. Print statement that identifies what is about to be shown
   2. Print an empty line to simplify reading of results
   3. Create variable to initiate search by using the google\_places module to talk to api (*query\_result*)
   4. Bool if query\_result has attributions
      1. Print attributions
   5. Start definite loop to iterate over results in query\_results
      1. Get details
      2. Print place name
      3. Print place rating
      4. Print place address
      5. Print place local phone number
      6. Print place international phone number
      7. Print place website
      8. Print place url
      9. Print blank line to separate this result from next
3. Except IndexError
   1. Print message relating to Index Error
4. Except JSONDecodeError
   1. Print message relating to JSOnDecodeError
5. Except HTTPError
   1. Print message relating to HTTPError
6. Except RequestException error
   1. Print message relating to RequestException error

4th Function

1. Define function as *geocode* with argument of *location*
2. Create variable and assign API URL *(url)*
3. Create variable to assign dictionary of parameters required by API (*options*)
4. Create variable to request data from api, using *url* and *options* as arguments *(response)*
5. Create variable to assign JSON data from API (*geodata*)
6. Create variable to assign filtered data from *geodata (coords)*
7. Return cords

Main Code:

1. Import required modules from python *(request, json, googleplaces)*
2. Try the following code
   1. Create variable and assign google API key *(apikey)*
   2. Create variable and call the google places api
   3. Create variable to prompt user input for a destination *(destination)*
   4. Create variable to call *geocode* function *(latlng\_dest)*
   5. Create indefinite loop
      1. Create variable to prompt user input for hotel search *(hotel)*
      2. Bool if *hotel* variable is == to ‘y’
         1. Print blank line
         2. Variable *hotel* = ‘hotel’
         3. Create variable to call *ping\_gplaces* function *(hotel\_query)*
         4. Break loop
      3. Bool if *hotel* == ‘n’
         1. Break
      4. Bool else
         1. Print error message
   6. Create indefinite loop
      1. Create variable to prompt user input for restaurant search *(restaurant)*
      2. Bool if *restaurant* variable is == to ‘y’
         1. Create variable to prompt user input of specific type (*res\_spec*)
         2. Bool if res\_spec == ‘y’
            1. Create variable to prompt user input for type of restaurant (res\_type)
            2. Print empty line
            3. Create variable for search parameter (*res\_search*)
            4. Create variable to call *ping\_gplaces* function (*res\_query*)
            5. Break loop
         3. Bool if *res\_spec* == ‘n’
            1. Print empty line
            2. Create variable for default search parameters *(res\_search)*
            3. Create variable to call *ping\_gplaces* function *(res\_query)*
            4. Break loop
         4. Else print error message
      3. Bool if *restaurant* == ‘n’
         1. None
         2. Break
      4. Else print error message
   7. Repeat step e, but replace *hotel* with *transportation*
   8. Repeat step e, but replace *hotel* with *tourist\_attract*
   9. Create indefinite loop
      1. Create variable to prompt user for input on currency exchange (*exch\_opt*)
      2. Bool if *exch\_opt* == ‘y’
         1. Call *currency\_exchange* function
         2. Break loop
      3. Bool if *exch\_opt* == ‘n’
         1. Break
      4. Bool if *exch\_opt* not ‘y’ or ‘n’
         1. Print error message
3. Except ValueError
   1. Print error message