Homework 2 - Project #2: Reachable Cities

Name of student: Gentiana Mehmeti

Note on collaboration: I did not collaborate with other students for this homework.

I started the homework by using the instructions in the PDF provided and imported two modules, `urlopen` and `quote` from the `urllib` library. These modules are used for handling URLs and their encoding respectively. It then declares four variables: `found_cities`, `found_cities_name`, `cities_by_steps` which are all lists and are empty initially.

Please note: In the end the result may take some seconds to return values because of the loops used. The final output is one line which contains steps. You can notice it by Step 1, Step 2 etc.

First function (add_colum_names(city_params):

The code then defines a function `add_colum_names(city_params)` (the aim of the function is to add column names to a list of cities and return them as a dictionary), which takes a single argument `city_params`, which is a list. This function does the following:

1. Declares an empty dictionary called `result`.
2. Declares a list called `column_names` which contains 8 strings, representing the names of the columns of a table.
3. If the length of `city_params` is less than the length of `column_names`, the function calculates the number of empty spaces to add to `city_params` to make it the same length as `column_names`. Then it uses a `for` loop to append empty spaces to `city_params` for the number of times calculated in the previous step.
4. Uses another `for` loop to iterate over the `column_names` and `city_params` lists. On each iteration, it adds a key-value pair to the `result` dictionary, where the key is the column name and the value is the corresponding element from `city_params`.
5. Returns the `result` dictionary.

Second function (get_params(city):

Takes a single argument `city` a string representing the name of a city and it is used to get the details of a specific city by querying a database and returning the result as a dictionary with columns names. This function does the following:

1. Declares a variable `q` which is a string containing an SQL query. The query selects certain columns from a table named "City" and a table named "located", and returns the first result where the name of the city is equal to the passed city name.
2. Declares a variable `eq` which is the `q` variable encoded using the `quote()` function from the `urllib.parse` module.
3. Declares a variable `url` which is a string containing a URL. The URL is a PHP script that executes the SQL query on a specific database, the `eq` variable is appended to the URL as a query parameter.
4. Declares a variable `query_results` which is the result of the query passed in the URL, by using the `urlopen()` function from the `urllib.request` module.
5. Uses a for loop to iterate over the `query_results` which is a file-like object. Each line of the result is read and decoded from `bytes` to `string` using the `decode()` method. The result is stripped of whitespace characters using the `rstrip()` method.
6. Uses an if statement to check if the length of the `string_line` is greater than 0, if so it will do:
    - Declares a variable `columns` which is a list containing the values from the `string_line` split by tabs.
    - Declares a variable `with_column_names` which is the result of calling the function `add_colum_names()` with `columns` as the argument.
7. Returns the `with_column_names` variable.
8. Closes the `query_results` by calling the `close()` method on it.

The third function (getAllNeighbourCities(cities, d)):

This function is used to get the details of all cities that are located within a specific distance from a given list of cities by querying a database and returning the result as a list of dictionaries with columns names. It takes two arguments `cities` and `d` :

- `cities` is a list of dictionaries, where each dictionary represents a city and contains various information about the city.

- `d` is a float value representing the maximum distance between two cities.

This function does the following:

1. Declares an empty list `to_return` which will be used to store the results of the function.
2. Uses a `for` loop to iterate over the `cities` list.
3. Declares two variables `longitude` and `latitude` and initializes them to 0.
4. Uses an `if` statement to check if the `Longitude` key in the current city dictionary is not empty, if so it will set `longitude` to the value of this key as a float.
5. Uses another `if` statement to check if the `Latitude` key in the current city dictionary is not empty, if so it will set `latitude` to the value of this key as a float.
6. Declares a variable `q` which is a string containing an SQL query. The query select certain columns from a table named "City" and a table named "located" and returns the first result where the name of the city is different from the current city name and the distance between the two cities is less than or equal to `d`.
7. Declares a variable `eq` which is the `q` variable encoded using the `quote()` function.
8. Declares a variable `url` which is a string containing a URL. The URL is a PHP script that executes the SQL query on a specific database, the `eq` variable is appended to the URL as a query parameter.
9. Declares a variable `query_results` which is the result of the query passed in the URL, by using the `urlopen()` function from the `urllib.request` module.
10. Uses a for loop to iterate over the `query_results` which is a file-like object. Each line of the result is read and decoded from `bytes` to `string` using the `decode()` method. The result is stripped of whitespace characters using the `rstrip()` method.
11. Uses an if statement to check if the length of the `string_line` is greater than 0, if so it will do:
    - Declares a variable `columns` which is a list containing the values from the `string_line` split by tabs.
    - Uses an if statement to check if the first element of `columns` (which is the name of the city) is not in the `found_cities_name` list, if so it will do:
        - Appends the first element of `columns` to the `found_cities_name` list.

    - Declares a variable `with_column_names` which is the result of calling the function `add_colum_names()` with `columns` as the argument.
    - Appends `with_column_names` to the `to_return` and `found_cities` lists.
12. The `query_results.close()` is closing the connection to the database.
13. The `return to_return` statement is returning the result of the function, which is a list of dictionaries containing the details of all the cities that are located within the maximum distance specified by the `d` argument.

The fourth (final function) search(city, country, k, s, d):

The function above takes five arguments:

- `city` is a string representing the name of a city.
- `country` is a string representing the name of a country.
- `k` is an integer representing the maximum number of steps to take in the search.
- `s` is an integer representing the maximum number of cities to return in the search.
- `d` is a float value representing the maximum distance between two cities.

This function is used to search for all the cities that are located within a maximum distance `d` of a starting city, within a maximum number of steps `k`. It will return the names of the cities found in each step.

This function does the following:

1. Appends the result of calling the `get_params(city)` function with the `city` argument to the `found_cities` list and the name of the city to the `found_cities_name` list.
2. Declares a list `cities_to_loop` and assigns to it a list containing a single element which is the result of calling the `get_params(city)` function with the `city` argument.
3. Uses an if statement to check if `cities_to_loop` is not empty, if so it will do:
    - Declares a list `current_step_cities` which will be used to store the results of the function `getAllNeighbourCities()` on each iteration.
    - Uses a for loop to iterate over a range of integers from 1 to `k + 1`.
    - On the first iteration:

    - assigns to the `current_step_cities` the result of calling the `getAllNeighbourCities(cities_to_loop, d)` function.
    - Appends a dictionary containing the step number as key "Step" and the names of the cities as a value in key "Cities" to the `cities_by_steps` list.
  - On the other iterations:
    - assigns to the `current_step_cities` the result of calling the `getAllNeighbourCities(current_step_cities, d)` function.
    - Appends a dictionary containing the step number as key "Step" and the names of the cities as a value in key "Cities" to the `cities_by_steps` list.
4. Prints the `cities_by_steps` list
5. Calls the `search` function with the arguments ('Geneva', 'CH', 5, 4, 2) in our example provided, but you can use with other cities and other parameters values as well.