Holiday Prompt - Attempt 2

The following prompt will be based around data for holidays in different countries around the world. Please, follow the prompts to complete the assignment. Please use all variable names that we have requested in the prompts. When we have not requested a particular variable name, you are welcome to choose whatever variable name you would like. Once you have finished your solution for these prompts, you will be able to use it when answering some questions in the ***Full Entrance Assessment - Attempt 2*** quiz. You may have the notebook open while answering the quiz.

1. Provided is a file containing a json formatted string, detailing some holidays in 2016 that were celebrated in 41 different countries. The data is stored in a file called `holiday_data.json` which is in the same directory as this notebook. Your first task is to read in the data, convert it to a python object, and then assign the python object to the variable `world_hol_data`.

In [1]:
```
import json

data = open("holiday_data.json", "r").read()
world_hol_data = json.loads(data)
```

2. For each country, data has been stored using the following keys: `locale`, `region`, `date`, `description`, `type`, and `notes`.

   `locale` is a combination of an ISO 639-1 language code, such as "en" (English), and an ISO 3166-1 alpha-2 country code, such as "US" (United States), separated by a hyphen.

   `region` is a subdivision of locale, for those locales that have regional holidays. For example, Patriot's Day is only observed in the en-US locale in Massachusetts and Maine, so there are entries for each of those states.

   `date` is provided in YYYY-MM-DD format.

   `description` is text that simply describes or names the holiday.

   `type` is a collection of single-character indicators that describe the holiday. "N" stands for "national holiday", which means it is locale-wide. "R" stands for "religious holiday". "F" stands for "fixed holiday" which means that it occurs on the same day each year. "V" stands for "variable holiday" which means that it typically occurs on a particular day of the week or month - for example it may be a Monday or the third Thursday in the month.

   `notes` are provided in some cases for clarification, but are not to be used as part of the holiday `description`.

   For this task, you should determine how many holidays are "national holidays". Assign that number to the variable `total_national_hol`.

In [2]:
```
total_national_hol = 0

for cont in world_hol_data:
    for data in world_hol_data[cont]:
        if "N" in data["type"]:
            total_national_hol = total_national_hol + 1
print(total_national_hol)
```

```
616
```

3. Below, we have outlined a class to represent a single holiday, called Holiday. Complete the `__init__` method by adding two additional attributes: holiday_type and number_of_types. holiday_type should be a list where each item is the type of holiday (either "National", "Religious", "Fixed", or "Variable"). number_of_types should be an integer, and represents how many types a holiday is (for example, a holiday may be only Variable, so number_of_types would be 1).

In [3]:
```
#define the class here
class Holiday:
    def __init__(self, full_country_name, dataset):
        self.language = dataset['locale'].split("-")[0]  # example: "de" for german or 'en' for english
        self.country = full_country_name                 # example: "Australia" or "Czech Republic"
        self.region = dataset['region']                  # example: "" or "TAS"
        self.date = dataset['date']                      # example: "2016-01-23"
        self.name = dataset['description']               # example: "Nieuwjaar" or "Boxing Day"
        separated_types = [letter for letter in dataset["type"]]

        #fill in here for holiday_type and number_of_types
        #example of holiday_type: ["Variable"] or ["National", "Fixed"]
        #example of number_of_types: 1 or 2
        typeStr = dataset['type']

        self.holiday_type = []
        if "N" in typeStr: self.holiday_type.append("National")
        if "R" in typeStr: self.holiday_type.append("Religious")
        if "F" in typeStr: self.holiday_type.append("Fixed")
        if "V" in typeStr: self.holiday_type.append("Variable")

        self.number_of_types = len(typeStr)

        self.notes = dataset['notes']                    # example: "" or "3rd monday in Feb."

    def get_date(self):
        return self.date

    def __str__(self):
        return "{} celebrates {} on {}. It is a {} holiday.".format(self.country, self.name, self.date, " ".join(self.holiday_type
```

4. Below is a list of countries that we expect to be included in the dataset. Create a list of instances of the holiday class, using the world_hol_data, and assign the list to the variable called `all_hol`.

What is the name of the fourth holiday in all_hol? Note that when we say fourth, we mean from the human perspective.

What is the output when the fourth holiday is printed? Note that this is not asking for output that is formatted in the following manner: `<__main__.Holiday object at 0x10878de10>`. Instead, we are asking for the output that uses the **str** method.

In [5]:
```
countries = ['Austria', 'Australia', 'Belgium', 'Brazil', 'Belarus', 'Canada', 'Switzerland', 'Colombia', 'Czech Republic', 'Germa
all_hol = []
for country in countries:
    for data in world_hol_data[country]:
```

```
        all_hol.append(Holiday(country, data))

print(str(all_hol[3]))
```

```
Austria celebrates Karfreitag on 2016-03-25. It is a National Religious Variable holiday.
```

5. Finally, determine the five most common days to have a holiday on across all countries. We have initialized a dictionary for you to use, but if you would like to use another data type then feel free to. Assign this list to the variable `top_five_hol_days`. Note that we would expect each item in `top_five_hol_days` to be a string where the date is in YYYY-MM-DD format.

In [6]:
```
hol_freq = {}
for inst in all_hol:
    if inst.date not in hol_freq:
        hol_freq[inst.date] = 1
    else:
        hol_freq[inst.date] += 1

top_five = sorted(hol_freq.items(), key = lambda x: x[1], reverse = True)[0:5]
top_five_hol_days = []
for day in top_five:
    top_five_hol_days.append(day[0])
print(top_five_hol_days)
```

```
['2016-03-25', '2016-12-26', '2016-01-01', '2016-03-28', '2016-05-01']
```

Tests you can run to check if you have likely answered the prompts correctly.

These tests will not contain answers to the questions you will fill in later. To provide some additional feedback, these tests will check some of the variables or values involved in the correct solutions.

In [7]:
```
import unittest
class MyTests(unittest.TestCase):
    def test_prompt_one_contents(self):
        self.assertEqual(len(world_hol_data), 41, "Testing that then length of world_hol_data is matches the expected value.")
        self.assertIn('Austria', world_hol_data, "Testing that Austria is in world_hol_data.")
        self.assertIn('Estonia', world_hol_data, "Testing that Estonia is in world_hol_data.")
        self.assertIn('South Africa', world_hol_data, "Testing that South Africa is in world_hol_data.")
    def test_prompt_two(self):
        self.assertTrue((total_national_hol > 500 and total_national_hol < 700), "Testing that the value assigned to total_nationa
    def test_prompt_three_Autumn_Feast(self):
        example_ints = Holiday("Narnia", {'locale': 'en-NA', 'region': '', 'date': 'XXXX-11-05', 'description': "Autumn Feast", 't
        self.assertEqual(example_ints.holiday_type, ["Variable"], "Testing that the holiday_type attribute is set correctly.")
        self.assertEqual(example_ints.number_of_types, 1, "Testing that the number_of_types attribute is set correctly.")
    def test_prompt_three_Spring_Festival(self):
        example_ints = Holiday("Narnia", {'locale': 'en-NA', 'region': '', 'date': 'XXXX-04-8', 'description': "Spring Festival",
        self.assertEqual(example_ints.holiday_type, ["National", "Variable"], "Testing that the holiday_type attribute is set corr
        self.assertEqual(example_ints.number_of_types, 2, "Testing that the number_of_types attribute is set correctly.")
    def test_prompt_four_contents(self):
        self.assertEqual(all_hol[-1].get_date(), "2016-12-26", "Testing that the date of the last item in all_hol is matches the e
        self.assertEqual(all_hol[-1].name, "Day of Goodwill", "Testing that the name of the last item in all_hol is matches the ex
    def test_prompt_four_length(self):
        self.assertEqual(len(all_hol), 749, "Testing that the number of items in all_hol matches the expected value.")
    def test_prompt_four_class(self):
        self.assertIsInstance(all_hol[0], Holiday, "Testing that the items in all_hol are instances of the Holiday class.")
    def test_prompt_five_sorting_contents(self):
        self.assertEqual(top_five_hol_days[0], '2016-03-25', "Testing that the first item has the expected date.")
    def test_prompt_five_sorting_len(self):
        self.assertEqual(len(top_five_hol_days), 5, "Testing that the len of top_five_hol_days is five.")



unittest.main(argv=['first-arg-is-ignored'], exit=False, verbosity=2)
```

```
test_prompt_five_sorting_contents (__main__.MyTests) ... ok
test_prompt_five_sorting_len (__main__.MyTests) ... ok
test_prompt_four_class (__main__.MyTests) ... ok
test_prompt_four_contents (__main__.MyTests) ... ok
test_prompt_four_length (__main__.MyTests) ... ok
test_prompt_one_contents (__main__.MyTests) ... ok
test_prompt_three_Autumn_Feast (__main__.MyTests) ... ok
test_prompt_three_Spring_Festival (__main__.MyTests) ... ok
test_prompt_two (__main__.MyTests) ... ok

----------------------------------------------------------------------
Ran 9 tests in 0.005s

OK
```

Out[7]:
```
<unittest.main.TestProgram at 0x7fdc6af2b320>
```

In [ ]:
```

```