Intellectual Property Prompt - Attempt 1

The following prompt will be based around data regarding intellectual property for the National Aeronautics and Space Administration (NASA) Agency. 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 1*** quiz. You may have the notebook open while answering the quiz.

1. This dataset provides a bit of information on some patents that NASA holds. Included is the name of the patent, who can access it, a date associated with it, a center associated with it, and a case number. The data is stored in a file called `NASA_data.json` which is in the same directory as this notebook. Your first task is to read in that data, convert it to a python object, and assign that object to a variable called `patent_data`.

In [13]:
```python
#replace None with your own answer. You may take as many lines as you would like to complete this problem.
import json

data = open("NASA_data.json", "r").read()
patent_data = json.loads(data)
```

2. For each patent, data has been stored under the following keys: `Case Number`, `NASA Center`, `Date`, `SRA Final`, and `Title`. Note that these keys are not at the highest level of the data.

   `Case Number` - identifier for the patent.

   `NASA Center` - a center associated with the patent.

   `date` - date associated with the patent, with a MM/DD/YYYY format.

   `SRA Final` - lists who is able to access it (ex. Academic Worldwide, Academic US, General US, Open Source, etc).

   `Title` - name of patent.

   For this task, you should determine how many unique centers are mentioned in patent_data. Assign this number to the variable `total_num_centers`.

In [14]:
```python
# replace pass and None with your own solutions.

unique_center = []
for thing in patent_data["Patent_Information"]["Results"]:
     if thing['NASA Center'] not in unique_center:
            unique_center.append(thing['NASA Center'])
total_num_centers = len(unique_center)
```

3. Provided is a light outline for a class called `Patent`. Your task is to fill out the rest of the class, based on the comments provided. We will provide them here as well for reference. Please note that the raw data may not use two digits to represent some months and days.

Initialize instance variable called `year`. The value can be extracted from `dict_data`. This should be a four digit string.

Initialize an instance variable called `month`. The value can be extracted from `dict_data`. This should be a two digit string.

Initialize an instance variable called `day`. The value can be extracted from `dict_data`. This should be a two digit string.

Define a new method called `get_date` which returns the date in the form of **YYYY-MM-DD**.

In [15]:
```python
class Patent:
    def __init__(self, abbreviated_organization_name, dict_data):
        self.org_name = abbreviated_organization_name
        self.title = dict_data["Title"]

        date = dict_data["Date"].split("/")

        # initialize instance variable called year. The value can be extracted from dict_data.
        # This should be a four digit string.
        self.year = date[2]

        # initialize an instance variable called month. The value can be extracted from dict_data.
        # This should be a two digit string.
        self.month = date[0]
        if int(self.month) < 10:
            self.month = "0" + self.month

        # initialize an instance variable called day. The value can be extracted from dict_data.
        # This should be a two digit string.
        self.day = date[1]
        if int(self.day) < 10:
            self.day = "0" + self.day

        self.id = dict_data['Case Number']
        self.access_limit = dict_data['SRA Final']

    # define a new method called "get_date" which returns the date using the following format: YYYY-MM-DD
    def get_date(self):
        return self.year + "-" + self.month + "-" + self.day
```

4. Create a list of instances of the `Patent` class, using the data stored in `patent_data`. Assign that list to the variable `patent_ints`.

In [16]:
```python
# replace None with your solution.
# Note that you do not have to use a list comprehension if you find that you are more comfortable with another format.
# However, you will need to assign the final list to patent_ints.
patent_ints = [Patent(data["Title"], data) for data in patent_data["Patent_Information"]["Results"]]
```

5. Finally, sort the list of instances by date, least recent (i.e. oldest) to most recent (i.e. youngest). For example, an instance with the date "2012-10-12" would have a lower index than one with "2012-10-23" and the patent with the date "2010-09-01" would have a lower index than one with "2010-10-11". Assign that list to the variable `ordered_dates`. Assign to the variable `most_recent_patent` the **case number** of the most recent patent. Assign to the variable `oldest_patent` the **case number** of the least recent patent.

In [17]:
```
# replace the None values with your own solutions.
ordered_dates = sorted(patent_ints, key= lambda p: p.get_date())
most_recent_patent = ordered_dates[-1].id
oldest_patent = ordered_dates[0].id
```

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 [18]:
```
import unittest
class MyTests(unittest.TestCase):
    def test_prompt_one_contents(self):
        data_sub_keys = list(patent_data['Patent_Information'].keys())
        self.assertIn("Results", data_sub_keys, "Testing that Results is a key in patent_data['Patent_Information']")
        self.assertIn("Metadata", data_sub_keys, "Testing that Metadata is a key in patent_data['Patent_Information']")
    def test_prompt_two_centers(self):
        self.assertIn("ARC", unique_center, "Testing that ARC is listed in unique_center.")
        self.assertIn("HDQS", unique_center, "Testing that HDQS is listed in unique_center.")
        self.assertIn("MSFC", unique_center, "Testing that MSFC is listed in unique_center.")
        self.assertIn("SSC", unique_center, "Testing that SSC is listed in unique_center.")
    def test_prompt_three_double_digits(self):
        example_ints = Patent(patent_data["Patent_Information"]["Metadata"]["Abbr_Origin"], patent_data["Patent_Information"]["Res
        self.assertEqual(example_ints.title, "Adaptive Relevance-Learning Software Component (ARNIE)", "Testing that the title att
        self.assertEqual(example_ints.year, '2001', "Testing that the year attribute is set correctly.")
        self.assertEqual(example_ints.month, '10', "Testing that the month attribute is set correctly.")
        self.assertEqual(example_ints.day, '19', "Testing that the day attribute is set correctly.")
    def test_prompt_three_single_digits(self):
        example_ints = Patent(patent_data["Patent_Information"]["Metadata"]["Abbr_Origin"], patent_data["Patent_Information"]["Res
        self.assertEqual(example_ints.title, "K9Client", "Testing that the title attribute is set correctly.")
        self.assertEqual(example_ints.year, '2004', "Testing that the year attribute is set correctly.")
        self.assertEqual(example_ints.month, '08', "Testing that the month attribute is set correctly.")
        self.assertEqual(example_ints.day, '05', "Testing that the day attribute is set correctly.")
    def test_prompt_four_contents(self):
        self.assertEqual(patent_ints[0].get_date(), "2001-10-19","Testing that the first item in patent_ints is matches the expect
    def test_prompt_four_length(self):
        self.assertEqual(len(patent_ints), 686, "Testing that the number of items in patent_ints matches the expected value.")
    def test_prompt_four_class(self):
        self.assertIsInstance(patent_ints[0], Patent, "Testing that the items in patent_ints are instances of the Patent class.")
    def test_prompt_five_sorting(self):
        self.assertEqual(ordered_dates[-2].get_date(), "2014-02-24", "Testing that the second to last item has the expected date."
        self.assertEqual(ordered_dates[1].get_date(), "1997-03-13", "Testing that the second item has the expected date.")




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

```
test_prompt_five_sorting (__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_double_digits (__main__.MyTests) ... ok
test_prompt_three_single_digits (__main__.MyTests) ... ok
test_prompt_two_centers (__main__.MyTests) ... ok

----------------------------------------------------------------------
Ran 8 tests in 0.005s

OK
```

Out[18]:
```
<unittest.main.TestProgram at 0x7f690c1569b0>
```

In [ ]:

In [ ]: