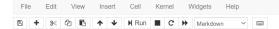
coursera



Jupyter Intellectual_Property_Patent_Submission Last Checkpoint: 01/28/2020 (autosaved)



Trusted Python 3 O



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.

```
M In [13]: #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.

```
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)
```

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]: # 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.

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]: # 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.

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.

```
walues involved in the corner solutions.

M In [18]: import unittest
class MyTests(unittest.TestCase):
    def test prompt_ome_contents(self):
        data_sub_koys_= list(patch_cdataf_reach_sub_koys_Tecting that Results is a key in patent_data["Patent_Information"]")
        self_assertIn("Metadataf_a, data_sub_koys, "Testing that Results is a key in patent_data["Patent_Information"]")
    def test_prompt_two_centers(self):
        self_assertIn("Metadataf_a, data_sub_koys, "Testing that Mctadata is a key in patent_data["Patent_Information"]")
        self_assertIn("Metadataf_a, data_sub_koys, "Testing that MC is listed in unique_center.")
        self_assertIn("Meta_unique_center, "Testing that MC is listed in unique_center.")
        self_assertIn("Self," unique_center, "Testing that Self_assertIn("Self," unique_center,")
        self_assertIn("Self," unique_center, "Testing that the unique_center.")
        self_assertIn("Self," unique_center, "Testing that the unique_center.")
        self_assertIqual(exampla_ints_vayar, "Well," "Testing that the year artiribute is set correctly,")
        self_assertIqual(exampla_ints_vayar, "Well," "Testing that the year artiribute is set correctly,")
        self_assertIqual(exampla_ints_day, '10," "Testing that the worm artiribute is set correctly,")
        self_assertIqual(exampla_ints_day, '10," "Testing that the day artiribute is set correctly,")
        self_assertIqual(exampla_ints_t,itle, "Neoling that the title artiribute is set correctly,")
        self_assertIqual(exampla_ints_t,itle, "Neoling that the title artiribute is set correctly,")
        self_assertIqual(exampla_ints_t,itle, "Neoling that the year artiribute is set correctly,")
        self_assertIqual(exampla_ints_t,itle, "Self_assert_day)
        self_assert_day(artiribute,ins,ont), 60," "Testing that the year artiribute is set correctly,")
        self_assert_day(artiribute,ins,ont), 60," "Testing that the post artiribute is set correctly,")
        self_assert_day(langla_ints,ont), 60," "Testing
```