Homework 8: Data Manipulation

November 2018

Part 1 (Tuesday): NUMPY

Task 1: NUMPY Arrays

Let’s start by exploring numpy through the interactive terminal. Before typing the command python, make sure to navigate to the CS3 workspace.

Begin by calling import numpy as np. This imports the numpy module and allows us to call it as np. This saves some key strokes when using numpy a lot in our code.

Start by creating a numpy array from -5 to 5 named array1 (excluding the number 5). Creating a numpy array requires a collection or an iterable, like a list or a range, to be passed as an argument to the np.array() function. Fortunately, numpy by default returns an array when using the arange function.

You can use the mathematical operators we have been familiar with between a series and a single number. Test out the various mathematical operators by trying to add, subtract, multiply and divide array1 by a number.

Now, create another numpy array with the numbers from 10 up to but not including 20 and store it in a variable called array2.

Numpy also supports these same operators between two numpy arrays. Again, try adding, subtracting, multiplying and dividing each array by one another. This provides convenient manipulation of large arrays of data in Python.

When performing these operations, the two arrays need to be the same shape. Try adding array1 to a new series np.array(range(20)). Notice that the error. If you happen to get 'nan', this is a special signifier for float values which stands for Not A Number. Therefore, it’s important that you ensure that the dimensions of the array on which you are performing operations matches.
You do not need to submit anything for Task 1.

**Task 2: Operating on Arrays**

Add the two arrays together and store the new series in a variable called summation.

Numpy also supports comparison operators like `==`, `!=`, `>`, `>=`, `<`, `<=` with either another series or with float values. For example, `series == 5` will create a resulting series which is True for any value in `series` equal to 5 and False otherwise.

Using the modulo operator (`%`) and a comparison operator, write an expression to find the values in array1 that are evenly divisible by 2 and assign this to a variable called `divisible_by_two`.

*Hint:* You can use an array of booleans (True/False) to match items you want pulled from the array. For example,

```python
>>> are_positive = array > 0
>>> array[are_positive]
```

Using the variable `series` and `divisible_by_two`, Try creating an array with just the elements that are even.

You do not need to submit anything for Task 2.

**Task 3: Other Methods**

The numpy array data structure includes many other helpful methods. Still using the interactive terminal, try out these methods. When performing these operations in your programs, you just need to replace `array` with the `np.array()` variable you have in your own code.

1. Compute the sum of the array, using `array.sum()`
2. Compute the mean and median
3. Compute the max and min of the series using `array.max()` or `array.min()`
4. You can sort the elements of a series extremely quickly using `array.sort()`. For example, sort 1 million random numbers: `np.sort(np.random.rand(1000000))`

You do not need to submit anything for Task 3.
Part 2 (Thursday): Pandas

In this task, we’ll write a small program that performs some operations on some data. Instead of numpy, we will use the Pandas data frame structure. Keep your interactive terminal open, we’ll be using it to try out new functions. We will analyze happiness data on countries worldwide. Download the .csv file here and save it in your CS3 workspace.

Pandas has a data structure called a Series which is similar to the numpy array. Many of the same operations (min, max, +, ==, etc.) can be used on Series in the same way as numpy arrays.

Task 1: Pandas

In this part of the assignment, we will be writing a python script that you will submit to Canvas. If you want to practice, use the interactive terminal. The commands are the same!

Your first step will be to read the csv file using Pandas. can be done with the following code:

```python
>>> data_frame = pd.read_csv('data.csv')
```

You can view the first few rows of data by typing:

```python
>>> data_frame.head()
```

It can also be helpful to glance at summary statistics of the DataFrame. You can do this by calling:

```python
>>> data_frame.describe()
```

Task 2: Dataframe Indexing

In a data frame, the columns can be indexed by their name. For example, the data set contains the column Country that contains the name of all of the countries. We can observe this column by running:

```python
>>> print(data_frame["Country"]).
```

To observe multiple columns, we can chain them together as follows:

```python
>>> print(data_frame[['Country','Happiness Rank']])
```
Task: In the main function, print a table with the Country Name, its Happiness Rank, and its Happiness Score.

In the data frame, the rows can also be indexed numerically. You can do that with:

```python
>>> data_frame[start index: end index]
```

A specific row of data can be printed out with:

```python
>>> data_frame.iloc[index]
```

Task: In your main function, print out the top ten rows.

---

**Task 3: Sorting**

Pandas provides multitude of tools for analyzing data. The data frame objects support sorting functionality with the `sort_values` method:

```python
>>> data_frame.sort_values(columns_to_sort_by)
```

Here, the variable `columns_to_sort_by` is a list of column names that indicate the order in which you would like to sort. For example, if you wanted to sort by "Happiness Score" you could type:

```python
>>> data_frame.sort_values(["Happiness Score"])
```

Often times, we want just the top or bottom items in a sorted list. Pandas provides convenient methods, `nlargest()`, `nsmallest()` for data frames. Each method requires two parameters: the number of items in the list to return and the columns to sort by. For example, you can find the 5 rows of data with the highest "Freedom" rating with:

```python
>>> data_frame.nlargest(5, ["Freedom"])
```

Task: Write a line of code that prints the names of the 10 countries which have the highest happiness score.
Task 4: Filtering

You can filter a data frame very easily using Pandas. The following code provides the row corresponding to the statistics for the United States.

```python
>>> data_frame[data_frame["Country"] == "United States"]
```

This line of code finds the rows where the "Country" column is equivalent to the "United States" and returns a data frame that is a subset of the original data frame.

Task: You can also link queries together using the & and | symbols. Create a DataFrame that includes all the European countries by combining the rows of Western, Central and Eastern Europe with:

```python
>>> data_frame[(data_frame["Region"] == "Central and Eastern Europe") | (data_frame["Region"] == "Western Europe")]
```

Task: Also, create a DataFrame for the countries of North America with a Freedom index greater than 0.5 by writing:

```python
>>> data_frame[(data_frame["Region"] == "North America") & (data_frame["Freedom"] > 0.5)]
```

Important: with these chained expressions, you must include each condition inside parentheses. Your program will throw an error otherwise.

Task: Write a function called multi_query that takes in 3 parameters: the data frame, a region, and a happiness score. And returns a DataFrame that contains just the rows of data corresponding to that region and that have a happiness score above the provided value.
Hours Worked

When you’re done, add the number of hours worked, whether you came to TA hours, and the names of any students you worked with at the top of the file in a comment. For extra credit, describe in 2-4 sentences how you could use Pandas and/or Numpy in your Project 1 code, and write some code to illustrate such.

______________________________

Handin

Name your submission file hw08.py and submit it through Canvas before Monday at 11:59PM.