hw02

October 15, 2018

1 Homework 2: Arrays and Tables, Due Sunday, October 13, at 11:59pm

Reading: Textbook chapters 4 and 5.

Directly sharing answers is not okay, but discussing problems with the course staff or with other students is encouraged.

You should start early so that you have time to get help if you're stuck.

Please complete this notebook by filling in the cells provided. Before you begin, execute the following cell to load the provided tests. Each time you start your server, you will need to execute this cell again to load the tests.

```
In [1]: # Don't change this cell; just run it.

        import numpy as np
        from datascience import *

        from client.api.notebook import Notebook
        ok = Notebook('hw02.ok')
        _ = ok.auth(inline=True)
```

```
=====================================================================
Assignment: Homework 2: Arrays and Tables
OK, version v1.13.11
=====================================================================

Successfully logged in as wec149@ucsd.edu
```

Important: The `ok` tests don't always tell you that your answer is correct. More often, they help catch careless mistakes. It's up to you to ensure that your answer is correct. If you're not sure, ask someone (not for the answer, but for some guidance about your approach).

Once you're finished, you must do two things:

1.0.1 a. Turn into OK

Select "Save and Checkpoint" in the File menu and then execute the `submit` cell below. The result will contain a link that you can use to check that your assignment has been submitted successfully. If you submit more than once before the deadline, we will only grade your final submission.

```
In [2]: _ = ok.submit()
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


Saving notebook... Could not save your notebook. Make sure your notebook is saved before sending
Submit... 100% complete
Submission successful for user: wec149@ucsd.edu
URL: https://okpy.org/ucsd/dsc10/fa18/hw02/submissions/PZVpmn
```

1.0.2 b. Turn PDF into Gradescope

Select File > Download As > PDF via LaTeX in the File menu. Turn in this PDF file into the respective assignement at https://gradescope.com/. If you submit more than once before the deadline, we will only grade your final submission

1.1 1. Studying the Survivors

The Reverend Henry Whitehead was skeptical of John Snow's conclusion about the Broad Street pump. After the Broad Street cholera epidemic ended, Whitehead set about trying to prove Snow wrong. (The history of the event is detailed here.)

He realized that Snow had focused his analysis almost entirely on those who had died. Whitehead, therefore, investigated the drinking habits of people in the Broad Street area who had not died in the outbreak.

What is the main reason it was important to study this group?

1) Survivors could provide additional information about what else could have caused the cholera, potentially unearthing another cause.

2) If Whitehead had found that many people had drunk water from the Broad Street pump and not caught cholera, that would have been evidence against Snow's hypothesis.

3) Through considering the survivors, Whitehead could have identified a cure for cholera.

```
In [7]: # Set survivor_answer to 1, 2, or 3
        survivor_answer = 2
```

```
In [8]: _ = ok.grade('q1_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
```

```
    Failed: 0
[ooooooooook] 100.0% passed
```

Note: Whitehead ended up finding further proof that the Broad Street pump played the central role in spreading the disease to the people who lived near it. Eventually, he became one of Snow's greatest defenders.

1.2 2. Creating Arrays

Question 1. Make an array called `weird_numbers` containing the following numbers (in the given order):

1. The mathematical constant π.
2. The square root of 2.
3. The logarithm of 15, in base 3.
4. 25 degrees, in radians.

Hint: Take a look at the functions and constants in the `math` module.

```
In [164]: import math
          weird_numbers = make_array(math.pi, math.sqrt(2), math.log(15, 3), math.radians(25))
          weird_numbers

Out[164]: array([3.14159265, 1.41421356, 2.46497352, 0.43633231])

In [165]: _ = ok.grade('q2_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2. Make an array called `words` containing the following three strings: * I like cooking * my family * and my pets

```
In [16]: words = make_array("I like cooking", "my family", "and my pets")
         words

Out[16]: array(['I like cooking', 'my family', 'and my pets'], dtype='<U14')

In [17]: _ = ok.grade('q2_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Strings have a method called `join`. `join` takes one argument, an array of strings. It returns a single string. Specifically, the value of `a_string.join(an_array)` is a single string that's the concatenation ("putting together") of all the strings in `an_array`, **except** `a_string` is inserted in between each string.

Question 3. Use the array `words` and the method `join` to make two strings:

1. "I like cooking, my family, and my pets" (call this one `with_commas`)
2. "I like cooking my family and my pets" (call this one `with_spaces`)

Hint: If you're not sure what `join` does, first try just calling, for example, `"foo".join(numbers)` .

```
In [22]: with_commas = ", ".join(words)
         with_spaces = " ".join(words)
         # with_commas
         # with_spaces
```

```
In [23]: _ = ok.grade('q2_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

1.3 3. Indexing Arrays

These exercises give you practice accessing individual elements of arrays. In Python (and in many programming languages), elements are accessed by *index*, so the first element is the element at index 0.

Question 1. The cell below creates an array of strings. What is the index of the second element in the array?

```
In [24]: some_strings = make_array('first', 'second', 'third', 'fourth', 'fifth', 'last')

         index_of_second = 1
```

```
In [25]: _ = ok.grade('q3_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2. Using an array method, assign the last element of `some_strings` to `last_element`.

```
In [50]: last_element = some_strings.item(5)
         last_element
```

```
Out[50]: 'last'
```

```
In [51]: _ = ok.grade('q3_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 3. Suppose you have an array with 143 elements. Assign the index of the middle element to `mid_index` below.

```
In [32]: mid_index = round(143/2)
         mid_index
```

```
Out[32]: 72
```

```
In [33]: _ = ok.grade('q3_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests
```

```
---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

The following cell loads data about presidents into a table and prints out the data.

```
In [34]: president_data = Table.read_table("president_births.csv")
         president_data
```

```
Out[34]: Name                     | Birth      | Death      | Birth Year | Birth Days (since 1 CE)
         George Washington        | 1732-02-22 | 1799-12-14 | 1732       | 632286
         John Adams               | 1735-10-30 | 1826-07-04 | 1735       | 633632
         Thomas Jefferson         | 1743-04-13 | 1826-07-04 | 1743       | 636354
         James Madison            | 1751-03-16 | 1836-06-28 | 1751       | 639248
         James Monroe             | 1758-04-28 | 1831-07-04 | 1758       | 641848
         Andrew Jackson           | 1767-03-15 | 1845-06-08 | 1767       | 645091
         John Quincy Adams        | 1767-07-11 | 1848-02-23 | 1767       | 645209
         William Henry Harrison | 1773-02-09 | 1841-04-04 | 1773       | 647249
         Martin Van Buren         | 1782-12-05 | 1862-07-24 | 1782       | 650835
         Zachary Taylor           | 1784-11-24 | 1850-07-09 | 1784       | 651555
         ... (28 rows omitted)
```

More often, you don't know the number of elements in an array, its *length*. (For example, it might be a large dataset you found on the Internet.) The function `len` takes a single argument, an array, and returns the `length` of that array (an integer).

Question 4. The cell below loads an array called `president_death_years`. Assign the 9th from last element of death year to `death_year`.

```
In [48]: president_death_years = Table.read_table("president_births.csv").column('Death Year')
         death_year = president_death_years.item(len(president_death_years) - 8)
         death_year
```

```
Out[48]: 1945
```

```
In [49]: _ = ok.grade('q3_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

1.4 4. Basic Array Arithmetic

The following table contains six siblings and their weekly allowance from their parents:

Sibling	Weekly allowance ($)	Weekly expense ($)
Sarah	3	2
Dave	1	0.5
John	5	2
Ashley	6	3
June	10	1.5
Rob	2	1

Question 1. Load the allowances in an array called `allowances`.

```
In [53]: allowances = make_array(3,1,5,6,10,2)
         allowances
```

```
Out[53]: array([ 3,  1,  5,  6, 10,  2])
```

```
In [54]: _ = ok.grade('q4_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2. If every sibling is given a raise of $4/week by their parents, how much money does each sibling make per week? Update the allowances in the array `new_allowances_constant`.

```
In [56]: new_allowances_constant = allowances + 4
         new_allowances_constant
```

```
Out[56]: array([ 7,  5,  9, 10, 14,  6])
```

```
In [57]: _ = ok.grade('q4_2')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 3. If instead, the parents decided to give each sibling a 20% raise, how much money does each sibling make per week? Update the allowances in the array `new_allowances_percent`.

```
In [58]: new_allowances_percent = allowances * 1.2
         new_allowances_percent

Out[58]: array([ 3.6,  1.2,  6. ,  7.2, 12. ,  2.4])

In [59]: _ = ok.grade('q4_3')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 4. Calculate how much each sibling receives per day for allowance in dollars (be sure to round each allowance to the nearest cent!). Assign your answer to the name `allowances_by_day`.

```
In [63]: allowances_by_day = np.round(allowances/7,2)
         allowances_by_day

Out[63]: array([0.43, 0.14, 0.71, 0.86, 1.43, 0.29])

In [64]: _ = ok.grade('q4_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 5. Load the weekly expenses into the array `expenses`. Calculate the amount of remaining money after each sibling spends a portion of their allowance on expenses, assigning the amounts to the variable `remaining`.

```
In [66]: expenses = make_array(2,0.5,2,3,1.5,1)
         remaining = allowances - expenses
         remaining
```

```
Out[66]: array([1. , 0.5, 3. , 3. , 8.5, 1. ])
```

```
In [67]: _ = ok.grade('q4_5')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

1.5 5. Shark Attacks

Question 1. The first line assigns `sharks` to an array containing the number of shark attacks between 1930 and 2017. What's the smallest and largest number of shark attacks in a given year? What is the total number of shark attacks between 1930 and 2017?

```
In [87]: sharks = Table.read_table('./sharks.csv').column('Attacks')
         smallest = sharks.min()
         largest = sharks.max()
         total = sharks.sum()
         sharks
```

```
Out[87]: array([ 26,  29,  27,  22,  27,  32,  32,  30,  24,  25,  24,  27,  41,
                 28,  31,  16,  26,  30,  29,  31,  43,  32,  29,  36,  42,  43,
                 51,  41,  54,  93,  93,  78,  86,  61,  66,  51,  58,  48,  47,
                 30,  42,  28,  35,  27,  38,  49,  39,  26,  25,  25,  35,  49,
                 40,  50,  41,  37,  39,  35,  55,  53,  38,  38,  56,  56,  56,
                 76,  61,  57,  65,  66,  97,  92,  88,  92,  92, 103, 103, 112,
                122, 120, 101, 128, 117, 122, 127, 143, 130, 136])
```

```
In [76]: _ = ok.grade('q5_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 3
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2. What is the largest increase in attacks experienced between consecutive years? *Hint*: You'll need an array arithmetic function mentioned in the textbook.

```
In [83]: biggest_increase = max(np.diff(sharks))
         biggest_increase

Out[83]: 39

In [84]: _ = ok.grade('q5_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 3. What percentage of shark attacks, during the 88 years recorded in the array, occured in the first 44 years?

```
In [92]: pct_in_first_44_years = np.cumsum(sharks).item(43) / np.sum(sharks)
         pct_in_first_44_years

Out[92]: 0.354129174165167

In [93]: _ = ok.grade('q5_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 4. Suppose there were 30,000 beachgoers in 1930 and the number of beachgoers increased by 15% year-after-year. Create an array called `beachgoers` that contains the number of beachgoers in each year between 1930 and 2017. If any beachgoer is equally likely to be attacked by a shark, how dangerous was the *least* dangerous year for shark attacks? Assign the *percent* chance of being attacked by a shark to the variable `danger`.

Hint: To calculate `beachgoers`, you may find the function `np.arange` helpful.

```
In [104]: beachgoers = 30000 + 30000*0.15*np.arange(0,88,1)
          danger = 1/ beachgoers.max()
          beachgoers
          danger
```

```
Out[104]: 2.372479240806643e-06
```

```
In [105]: _ = ok.grade('q5_4')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

1.6 6. Tables

Question 1. Suppose you have 3 quarters, 4 dimes, 2 nickels, and 4 pennies in your pocket. Create a table, named `coins`, with two columns: `Coin Type` and `Quantity` that contains the inventory of your pocket.

```
In [106]: coins = Table().with_column(
              "Coin Type", make_array("quarters","dimes","nickes","pennies"),
              "Quantity", make_array(3,4,2,4),
          )
          coins
```

```
Out[106]: Coin Type | Quantity
          quarters  | 3
          dimes     | 4
          nickes    | 2
          pennies   | 4
```

```
In [107]: _ = ok.grade('q6_1')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 2. The file `inventory.csv` contains information about the inventory at a fruit stand at the end of the day. Each row represents the contents of one box of fruit. Load it as a table named `inventory`.

```
In [108]: inventory = Table.read_table('./inventory.csv')
          inventory

Out[108]: box ID | fruit name | price per item ($) | start count | sold count
          53686  | kiwi       | 2                  | 45          | 30
          57181  | strawberry | 3                  | 12          | 9
          25274  | apple      | 1.5                | 20          | 19
          48800  | orange     | 1                  | 35          | 30
          26187  | strawberry | 3                  | 25          | 25
          57930  | grape      | 0.5                | 17          | 17
          52357  | strawberry | 3                  | 10          | 3
          43566  | peach      | 4.5                | 40          | 20

In [109]: _ = ok.grade('q6_2')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 3. Does each box at the fruit stand contain a different fruit?

```
In [112]: # Set all_different to "Yes" if each box contains a different fruit or to "No" if mult
          all_different = 'No'
          all_different

Out[112]: 'No'

In [113]: _ = ok.grade('q6_3')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

----------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 4. How many pieces of fruit did the store sell in total that day?

```
In [119]: total_fruit_sold = inventory.column('sold count').sum()
          total_fruit_sold

Out[119]: 153

In [120]: _ = ok.grade('q6_4')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 5. What was the store's total revenue (the total price of all fruits sold) on that day?

```
In [124]: revenue = inventory.column('sold count')*inventory.column('price per item ($)')
          total_revenue = revenue.sum()
          total_revenue

Out[124]: 328.0

In [125]: _ = ok.grade('q6_5')

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 6. What was the stores's total revenue from strawberry sales that day?

```
In [134]: strawberries_columns = inventory.where('fruit name',are.equal_to('strawberry'))
          revenue = strawberries_columns.column('sold count')*strawberries_columns.column('price
          revenue_from_strawberries = revenue.sum()
          revenue_from_strawberries

Out[134]: 111.0

In [135]: _ = ok.grade('q6_6')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

Question 7. Create a table with a new column, called `remaining ($)` that indicates the total remaining value of the fruit still in each box at the end of the day. Assign this table the name `with_remaining`.

```
In [160]: remaining_table = inventory.column('start count') - inventory.column('sold count')
          value = remaining_table*inventory.column('price per item ($)')
          with_remaining = inventory.with_column('remaining ($)', value)
          with_remaining
```

```
Out[160]: box ID | fruit name | price per item ($) | start count | sold count | remaining ($)
          53686  | kiwi       | 2                  | 45          | 30         | 30
          57181  | strawberry | 3                  | 12          | 9          | 9
          25274  | apple      | 1.5                | 20          | 19         | 1.5
          48800  | orange     | 1                  | 35          | 30         | 5
          26187  | strawberry | 3                  | 25          | 25         | 0
          57930  | grape      | 0.5                | 17          | 17         | 0
          52357  | strawberry | 3                  | 10          | 3          | 21
          43566  | peach      | 4.5                | 40          | 20         | 90
```

```
In [161]: _ = ok.grade('q6_7')
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Running tests

---------------------------------------------------------------------
Test summary
    Passed: 1
    Failed: 0
[ooooooooook] 100.0% passed
```

```
In [162]: # For your convenience, you can run this cell to run all the tests at once!
          import os
          _ = [hw04.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]
```

```
        ---------------------------------------------------------------------------
```

```
        NameError                                 Traceback (most recent call last)

        <ipython-input-162-74f9d91be9d5> in <module>()
          1 # For your convenience, you can run this cell to run all the tests at once!
          2 import os
    ----> 3 _ = [hw04.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]


        <ipython-input-162-74f9d91be9d5> in <listcomp>(.0)
          1 # For your convenience, you can run this cell to run all the tests at once!
          2 import os
    ----> 3 _ = [hw04.grade(q[:-3]) for q in os.listdir("tests") if q.startswith('q')]


        NameError: name 'hw04' is not defined
```

```
In [163]: _ = ok.submit()
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>


Saving notebook... Saved 'hw02.ipynb'.
Submit... 100% complete
Submission successful for user: wec149@ucsd.edu
URL: https://okpy.org/ucsd/dsc10/fa18/hw02/submissions/l59VvV
```

1.7 Don't forget to submit to both OK and Gradescope!