1 Multiple Choice

Problem Setup: The **geometric mean** of a list of numbers $\{y_i\}_{i=1}^{n}$ is defined to be

$$\overline{y}_{\text{geom}} = (y_1 \times y_2 \times \cdots \times y_n)^{\frac{1}{n}}$$

i.e. the geometric mean is computed by first computing the product of the numbers, and then raising the product to the power $(1/n)$ where n is the number of observations. João would like to write a Python function called `geom_mean()` that takes in a single input `y = [y1, ..., yn]` and outputs the geometric mean of `y`. To that end, he has written the following code, and has nothing written above it:

```
def geom_mean(y):

    """
    return the geometric mean of y
    """

    n = len(y)

    prod_y = 1

    for k in __Blank 1__:

        prod_y __Blank 2__ k

    return (prod_y) __Blank 3__ (1/n)
```

Problem: Assuming all blanks are filled in correctly, what would be the output of running `geom_mean (1, 2, 3)`?

A. `0.5503`

B. `1.8171`

C. `2.0000`

D. An Error

E. None of the above

Solution: Note that the function `geom_mean` takes in a **single** input `y`, which must be a list or an array. As such, trying to call `geom_mean(1, 2, 3)` will return an error since we are trying to call the function on 3 arguments whereas the function itself only takes in one argument.

Said differently, the correct answer would be `1.8171` if we had called `geom_mean([1, 2, 3])`.

Problem Setup: Consider the following data matrix:

`grade`	`sleep`	`major`	`fav_color`
A+	7.8	PSTAT	Green
B	6.9	PSYCH	Gold
A-	7.0	SOC	Red
B	5.5	PSTAT	Gold
C+	6.7	PSTAT	Purple

We are also provided with the following data dictionary:

- **`grade`**: letter grade
- **`sleep`**: amount of sleep (in hours)
- **`major`**: major
- **`fav_color`**: favorite color

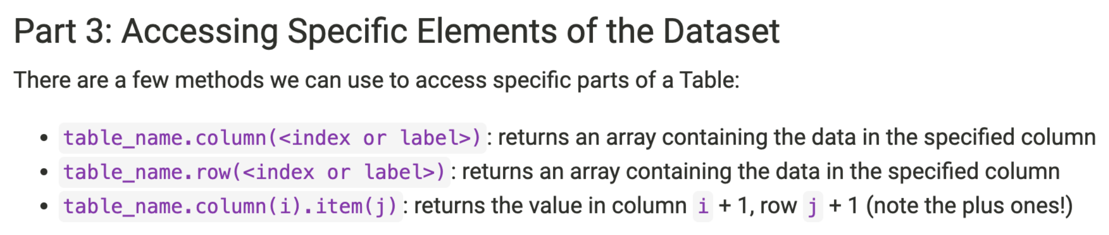
Suppose the above data matrix has been imported into Python as a `datascience` table called `students`. Also assume the `datascience` module has been imported, and that it has been imported without any nickname.

Problem: What would be the result of running the code

```
students.column(2).item(3)
```

A. 7.0
B. 5.5
C. SOC
[✓ D. PSTAT]
E. None of the above.

Solution: Recall that Python starts indexing at zero. As such, `students.column(i).item(j)` actually returns the element in the $(i+1)^{\text{th}}$ position of the $(j+1)^{\text{th}}$ column: see the screenshot from Lab 5 below.



Part 3: Accessing Specific Elements of the Dataset

There are a few methods we can use to access specific parts of a Table:

- `table_name.column(<index or label>)`: returns an array containing the data in the specified column
- `table_name.row(<index or label>)`: returns an array containing the data in the specified column
- `table_name.column(i).item(j)`: returns the value in column `i` + 1, row `j` + 1 (note the plus ones!)

Hence, `students.column(2).item(3)` returns the 4th element of the 3rd column; i.e. `PSTAT`.

Problem: Which of the answer choices below best describes what the following code is doing:

```
students.row(students.column(3)== "Gold")[0]
```

A. It returns the favorite colors of students whose favorite color was Gold.

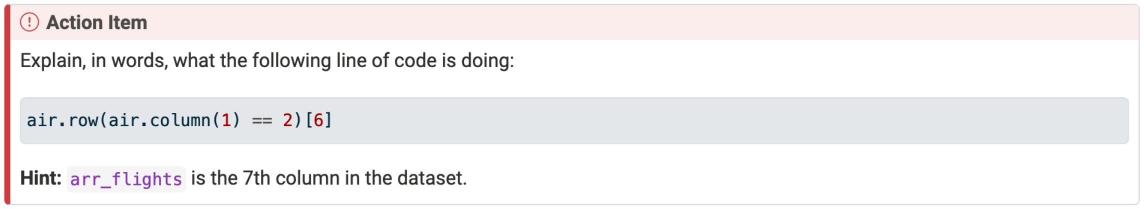
B. It returns the grades of students whose favorite color was Gold.

C. It returns the number of students whose favorite color was Gold.

D. It returns an error.

E. None of the above.

Solution: This is a near-direct copy of code from Lab5. Specifically, recall the following Action Item:



Action Item

Explain, in words, what the following line of code is doing:

```
air.row(air.column(1) == 2)[6]
```

Hint: `arr_flights` is the 7th column in the dataset.

In this problem, `students.column(3)== "Gold"` gives us a vector of `True` and `False` elements, where an element is `True` if the corresponding value of the `fav_color` column has value `Gold`. Passing this into `students.rows()` reveals the rows of the `students` table corresponding to students whose favorite color is `Gold`; since we are subsetting/indexing the `0`th element of this, we are returning the grades (i.e. the *first* column in the `students` table) of students whose favorite color is Gold.

Problem: What does the output of **`len`**`(students.labels)` represent?

A. The number of variables

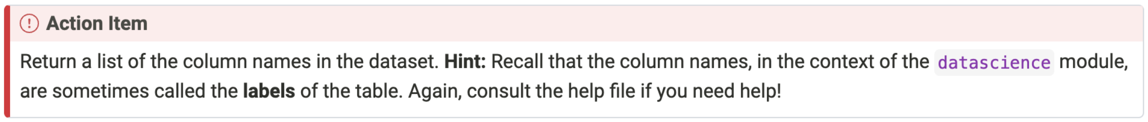
B. The number of observational units

C. The number of explanatory variables.

D. The total number of elements in the table

E. None of the above.

Solution: Recall the following Action Item from Lab 05:



Action Item

Return a list of the column names in the dataset. **Hint:** Recall that the column names, in the context of the `datascience` module, are sometimes called the **labels** of the table. Again, consult the help file if you need help!

What we saw with that task is that `students.labels` returns a list of all of the *column names* of the `students` table; i.e. it returns a list of all of the *variables* in the associated data matrix. We also know that **len**() gives the length of a given list/array; as such, **len** (students.labels) returns the number of columns in the data matrix- i.e. the number of *variables* in the data matrix.

By the way, it is not at all correct to say that it gives the number of "explanatory variables", as explanatory variables only arise in the context of regression.

. . . .

Problem: Consider the function g(), defined as follows:

```
def g(x):

    """
    return negative one times x
    """

    -1 * x
```

What will be returned by calling g(-1)?

A. -1

B. 1

C. An Error

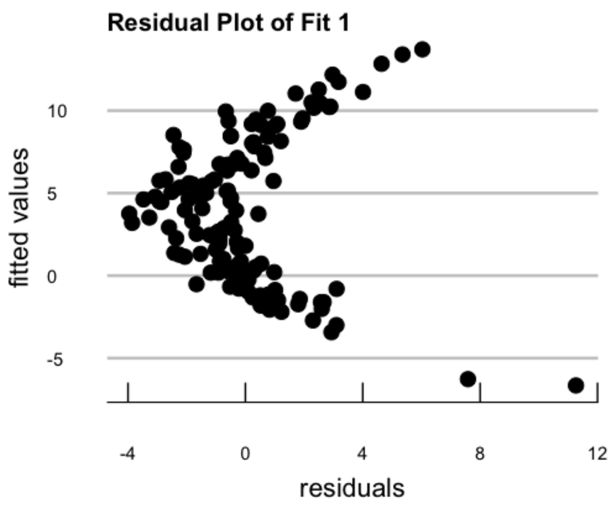
D. Nothing

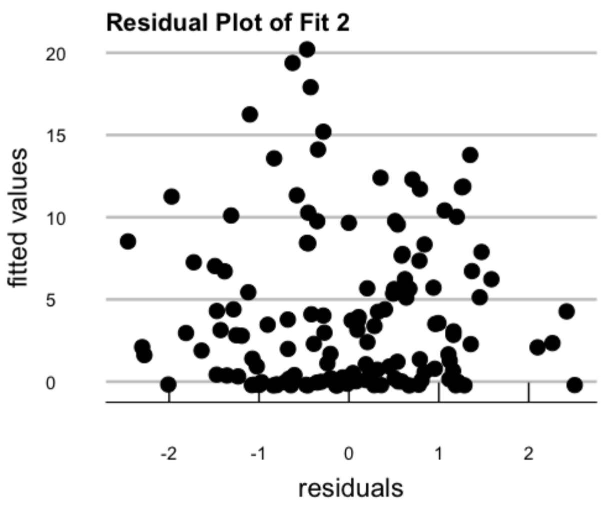
E. None of the above.

Solution: As was discussed many times throughout this course, Python functions do not return anything unless you specifically tell them to (by including a **return** statement). The function g(), as it is defined above, is missing a **return** statement and will therefore not return anything when it is called on an input. (Yes, there is the word "return" in the docstring, but recall that the docstring is simply a multiline comment and hence Python does not execute anything written in it.)

. . . .

Problem: A variable y is regressed onto another variable x. Two different fits are generated, called Fit 1 and Fit 2 respectively; the residual plots are displayed below. Which model is performing "better" (i.e. fitting the data better)?





A. Fit 1

B. Fit 2

Solution: Recall that a "good" residual plot displays *no* pattern. This is because the residuals are our attempt at modeling the noise component of our statistical model; if there is a pattern remaining, then our model fit has not captured all aspects of the true underlying relationship between the two variables.