Lab 1 – Welcome to Data 6!

[Solutions]

Data 6, Summer 2022

Welcome to the first lab section of Data 6!

First and foremost, labs are designed to complement homework assignments. While labs are submitted or graded, we encourage you to work through all questions carefully to familiarize yourself with key programming concepts that you will use later on in your homeworks. Remember, you learn data science by ***doing*** data science!

The goals of this lab section:

- Access all of the technology we'll be using in this class;
- Familiarize yourself with the Jupyter Notebook environment;
- Use arithmetic operators in Python; and
- Work with Python names.

Part 1: Jupyter Notebook

Cells

Not these! →

Jupyter Notebooks are made up of cells. There are two types of cells in a Jupyter Notebook: code cells and Markdown cells.

- Code cells are where we write our Python code.
- Markdown cells allow us to write text, like the text you're reading right now. In Homework 1, you'll get a chance to learn a bit of Markdown.

Code cells

Running a code cell will execute all of the code it contains. Any output produced by the code will appear below the cell. Notice the `In:` and `Out:` statements to the left of the cells, followed by a number. That number designates the order in which the cell was run. For example, the number `[3]` next to a cell would indicate that this cell is the third cell that has been run in the notebook so far, or that this is the third time you've run this cell if you've re-run the cell multiple times.

To run the code in a code cell, first click on that cell to activate it. It'll be highlighted with a little green or blue rectangle. Next, do one of the following:

- **Keyboard shortcut**: Hold down the `Shift` key and press `Return` or `Enter`.
- **Button click**: Click the "Run" button in the cell toolbar above.

While the above two options give no difference in computer runtime performance, it's useful to internalize some common keyboard shortcuts early in your Jupyter experience (think of it like learning "Ctrl-C" or "Command-C" to copy instead of "Right click -> Copy to Clipboard"). While they're not required for this course, keyboard shortuts will greatly improve your quality of life when programming Python (just trust us on this).

We can use code cells to do arithmetic in Python. Try running the code cells below:

```
In [1]: 3 + 4 ** 2
```

```
Out[1]: 19
```

The order of evaluation is what you might expect: parentheses, then exponentiation, then multiplication and division, then addition and subtraction; from left to right.

```
In [2]: 2 *(3 + 10)
```

```
Out[2]: 26
```

You can use parentheses to modify the typical order of evaluation. **How would the above output change, if you removed the parentheses?**

```
In [3]: 4 ** 2
        1 + 2 + 4 + 8 + 16 + 32 + 64
```

```
Out[3]: 127
```

Note that only the value from evaluating the last expression is displayed; the output from `4 **2` isn't shown at all. This is expected behavior.

```
In [4]: 10000
```

```
Out[4]: 10000
```

Also notice that this cell contains a very simple expression that evaluates to `1000`.

```
In [5]: 10 + 20 + 30 - max(10, 20, 30) - min(10, 20, 30)
```

```
Out[5]: 20
```

The output you see from running these five code cells are the values `19`, `16`, `127`, `1000` and `20` displayed, respectively.

Comments in code cells are denoted using a hashtag (`#`). Comments are used to explain what code does when necessary (or, occasionally in our case, to provide instructions).

```
In [6]: # This is a comment!
        19 - 18

Out[6]: 1
```

If you try and use syntax Python doesn't know about, it will give you an error.

```
In [7]: 3 + * 4

  File "<ipython-input-7-9a0a44f9beb0>", line 1
    3 + * 4
        ^
SyntaxError: invalid syntax
```

```
In [8]: # Why does this work while the above cell causes an error?
        3 * - 4

Out[8]: -12
```

Question 1: Markdown Cells

To edit an existing Markdown cell, double click it and change whatever you want. To *render* (display) it, you can "run" it like you'd run a code cell.

Edit the following cell to contain your favorite color.

Note: This is how questions will be formatted in your homework assignments.

My favorite color is Berkeley blue :)

Question 2: Converting cells

To convert a cell from code to Markdown (or vice versa), you should use the dropdown menu in the cell toolbar. For instance, to convert the following cell from Markdown to code, you should:

1. Click the cell.
2. Click "Markdown" in the cell toolbar, and change it to "Code."

Your cell toolbar should then look like this:

Change the following cell from a Markdown cell to a code cell, and run it by pressing the "Run" button or using the keyboard shortcut `Shift + Enter` .

```
In [9]:  3 + 4 + 5  # This is now a code cell

Out[9]:  12
```

Adding and deleting cells

To add cells, you can go to "Insert" in the toolbar at the top, then click "Insert cell Above" or "Insert cell Below." To delete cells, you can go to "Edit" in the toolbar at the top and click "Delete cells". (Be very careful when deleting a cell.)

For your convenience, we provide this screenshot that details how to switch between "Edit mode" and "Command mode", which is relevant if you want to learn keyboard shortcuts. Again, keyboard shortcuts are not required for this course, but learning some now will greatly improve your programming quality of life.

Some useful keyboard shortcuts:

Action	Mode	Keyboard shortcut
Run cell + jump to next cell	Either (puts you in edit mode)	`SHIFT + ENTER`
Save notebook	Either	`CTRL/CMD + S`
Switch to command mode	Either (puts you in command mode)	`ESCAPE`
Switch to edit mode	Command	`ENTER`
Comment out the current line	Edit	`CTRL/CMD + /`
Create new cell above/below	Command	`A/B`
Delete cell	Command	`DD`
Convert cell to Markdown	Command	`M`
Convert cell to code	Command	`Y`
Show all shortcuts	Command	`H`

Question 3: Insert Cell

Add a Markdown cell below this one, and in it write your middle name, or write "I don't have a middle name".

James' middle name is Patrick

In [10]:
```
# The cell you create should be above this one!
```

Question 4: Delete Cell

Delete the cell below.

Part 2: Arithmetic Expressions and Operators

We have covered several arithmetic operators so far:

In [11]:
```
(7 - 5) * 2**10 - 3 * 8 + 6 - 8
```

Out[11]:
```
2022
```

Combining several `int` s without using "regular" division always results in another `int` . Below, we use the `type()` function to find the type of the evaluated expression from the previous cell.

In [12]:
```
type((7 - 5) * 2**10 - 3 * 8 + 6 - 8)
```

Out[12]:
```
int
```

As soon as division (`/`) or other `float` s are involved, the result is a `float` :

In [13]:
```
15 / 3
```

Out[13]:
```
5.0
```

In [14]:
```
type(15 / 3)
```

Out[14]:
```
float
```

We can get `int` results from division by using the **integer division** operator `//` . When the dividend (the number we're dividing) isn't evenly divisible by the divisor (the number we're dividing by), the *result is rounded down*.

In [15]:
```
15 // 3
```

Out[15]:
```
5
```

In [16]:
```
16 // 3
```

Out[16]:
```
5
```

In [17]:
```
type(15 // 3)
```

Out[17]:
```
int
```

We can **cast** `int` s to `float` s and vice versa by applying the `int()` or `float()` functiosn to expressions.

In [18]:
```
int(15 / 3)
```

Out[18]:
```
5
```

In [19]:
```
float(5)
```

Out[19]:
```
5.0
```

If our `float` is not a round integer, we lose information:

In [20]:
```
int(4.1)
```

Out[20]:
```
4
```

Question 5

In the cell below, **write an arithmetic expression that evaluates to your birth year *as an `int`* that uses at least four of these operators**:

- `+`
- `-`
- `*`
- `/`
- `**`

Use can also use the `int()` function.

In [21]:
```
int(4003 / 2 - (2 / 4) ** 2 + (-1.25))
```

Out[21]:
```
2000
```

Note: On homeworks, questions that ask you to write code will usually be followed with a cell that says something like `otter.grade('q5')` that you'll need to run in order for your work to be graded. We'll bring this up when Homework 1 goes out, but we wanted to give you a heads up now.

Question 6: Pythagorean Theorem

So far we have discussed one critical facet of data science: **data validation**, where, given existing data, we verify another study's claims or findings. Another important data science skill is **feature engineering**, where we augment our data with an additional variable. While we leave the formal definition of "feature" to future lectures, let's use the following example to explore the benefit of using existing data to generate more information.

Suppose the campus IT department is exploring different computer monitors for their computer lab. Computer monitor sizes are defined as diagonal measurements from corner to corner, but the IT department is concerned about the overall *monitor width* (since the monitor has to fit on their desk). They are considering monitors from the following dataset:

Aspect Ratio	Monitor Size (inches)	Monitor Height (inches)
16:9	20	9.8
16:9	22	10.8
16:9	24	11.8

The data above include aspect ratio (all widescreen, 16:9), monitor size in inches, and monitor height in inches–but no monitor width. As a data scientist, you decide to generate another feature for each listed monitor in the data by computing the associated monitor width.

In the cell below, **write an arithmetic expression that, when evaluated, computes the monitor width for the 22" monitor size** listed in the table above.

Hints:

- Recall the Pythagorean theorem for a right triangle: $a^2 + b^2 = c^2$, where a, b, and c are the lengths of the first leg, second leg, and hypotenuse, respectively.
- We didn't explicitly cover how to take the square root of a number, but recall that in mathematics, the square root function $\sqrt{x}$ is equivalent to exponentiating with the exponent $\frac{1}{2}$ – that is, $\sqrt{x} = x^{\frac{1}{2}}$. That means we can use the exponentiation operator `**` to compute square roots, too. For example, `121 ** 0.5` evaluates to `11.0`.
- Like in math, may need to use parentheses `(` and `)` for order of operations.
- There are two valid ways of computing the monitor width. Each method uses values from two out of the three available columns.
- Feel free to validate your result with a screen size calculator, e.g., http://screen-size.info/.

In [22]:
```
# (c^2 (monitor size) - b^2 (monitor height)) ** 0.5 = a (monitor width)
```

```
(22 ** 2 - 10.8 ** 2) ** 0.5
```

```
Out[22]: 19.1666376811375
```

```
In [23]: # You can also multiply the monitor heigh by the aspect ratio (16/9) to get the
         10.8 * 16/9
```

```
Out[23]: 19.200000000000003
```

Part 3: Names

In this part, we will continue our exploration of the Tuberculosis dataset discussed in lecture. Here's that data again:

Tuberculosis data. CDC MMWR source

U.S. jurisdiction	2019 # cases	2020 # cases	2021 # cases	2019 incidence	2020 incidence	2021 incidence
Total	**8,900**	**7,173**	**7,860**	**2.71**	**2.16**	**2.37**
Alabama	87	72	92	1.77	1.43	1.83
...	...	...	...	...	...	...
California	2,111	1,706	1,750	5.35	4.32	4.46
...	...	...	...	...	...	...

U.S. Census population estimates (2019 source, 2021-2021 source)

	2019 population	2020 population	2021 population
Total	**328,239,523**	**331,501,080**	**331,893,745**
Alabama	4,903,185	5,024,803	5,039,877
California	39,512,223	39,499,738	39,237,836

Recall that the Tuberculosis (TB) incidence is defined as the number of TB cases per 100,000 persons in a particular population:

$$\text{TB incidence} = \frac{\#\text{ TB cases}}{\text{population}} \times 100000$$

We computed the tuberculosis (TB) incidence of 2020 Using Python's arithmetic expressions, this value can be written as:

```
In [24]: 7173/(331501080/100000)
```

```
Out[24]: 2.163793855513231
```

We'll go one step further in the cell below by binding (i.e., assigning) a name to the value returned from evaluating this arithmetic expression. Below, note the name `incidence_2020` clearly indicates what year's incidence we are computing.

In [25]:
```
# just run this cell
incidence_2020 = 7173/(331501080/100000)
incidence_2020
```

Out[25]:
```
2.163793855513231
```

Notice! You'll often see the above *syntax* (i.e., program code structure) in Jupyter notebooks. It is effectively evaluating **two** Python statements:

1. `incidence_2020 = 7173/(331501080/100000)`

   Assign the name `incidence_2020` to the result of evaluating the arithmetic expression `7173/(331501080/100000)` .

2. `incidence_2020`

   Evaluate the expression `incidence_2020` . Since `incidence_2020` is a name, the result of this arithmetic expression is the value bound to the name `incidence_2020` .

Remember! Jupyter code cells output only the value of the *last* evaluated expression. So this two-step syntax is useful to bind a name to a value and output the value itself.

Question 7: Changes in Percentage

Consider this line in the abstract of the CDC MMWR report:

> Reported TB incidence (cases per 100,000 person) increased 9.4%, from 2.2 during 2020 to 2.4 during 2021...

The 2.2 TB incidence in the quote above is the same incidence we computed above, *rounded* to the first decimal place (i.e., to the nearest tenth). For the purposes of this exercise, we will not work with any rounded numbers, and then qualitatively compare our computations to the quote above.

Question 7a

Assign the name `incidence_2021` to the 2021 tuberculosis incidence. Note we use the same structure as explained above to bind the name and output its value in a single code cell.

Hint: Use the tables above.

```
In [26]: incidence_2021 = 7860/(331893745/100000)
         incidence_2021
```

```
Out[26]: 2.3682278194185313
```

Validate: How close is your result (rounded to the nearest tenth) to the 2.4 TB incidence reported in the quote from the CDC report?

Question 7b

The "percent change from measured value A to measured value B" is computed as follows:

$$\text{percentage change} = \frac{A - B}{A} \times 100$$

A useful framework for understanding the values in this formula (particularly the denominator) are that A is the "starting point" of the measurement, and all changes are with respect to this starting point.

Assign the name `percent_change` to the percent increase of TB incidences from 2020 to 2021.

```
In [27]: percent_change = 100 * (incidence_2020 - incidence_2021)/incidence_2020
         percent_change
```

```
Out[27]: -9.447940864810828
```

Validate: How close is your result to the 9.4% percentage increase as reported in the CDC report? Why is your result negative?

The result is negative because the formula for percentage change subtracts the "final" value (B) from the initial value (A), resulting in a negative number when the B is bigger than A. We should just note that this negative number indicates an *increase* in the incidence between 2020 and 2021.

Done! 😇

That's it! There's nowhere for you to submit this, because again labs are not assignments. However, please ask any questions you have with this notebook in lab or on Ed.