AEP 4: Working with sets in Python

Overview

In this AEP, you will write Python functions that will compute the union, intersection, and symmetric difference of two sets.

Learning Targets associated with this AEP:

- SF.2: I can perform operations on sets (intersection, union, complement, Cartesian product) and determine the cardinality of a set.

Remember, AEPs do not have fixed deadlines; simply work on this item until you are ready to submit it. But remember the **Two Items Per Week Rule.**

Technology Background

This is a programming-focused AEP, so you will need all the skills you learned in the Python/Jupyter crash course.

AEP Description and Tasks

What this AEP is about

Sets are not naturally-occurring data structures in Python, and so we need to have a means of representing them if we are to work with them. The most natural way to represent a set is as a list of elements. For example the mathematical set $\{1, 2, 3, 4\}$ might be represented in Python as the list `[1,2,3,4]`. However, there are two major differences between mathematical sets and Python lists:

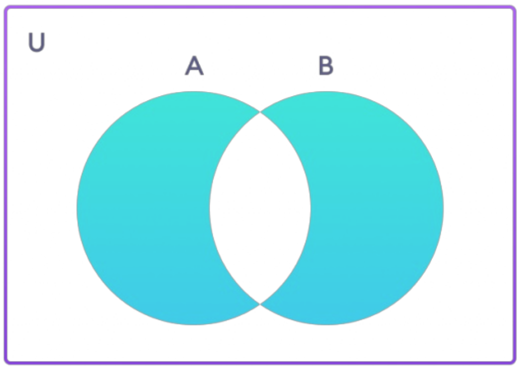
1. *Python lists can have duplicate elements, while mathematical sets can't*. For example the list `[10, 10, 20]` is a valid Python list, but $\{10, 10, 20\}$ is not a valid set. Or put differently, $\{10, 10, 20\}$ and $\{10, 20\}$ are the same set, but `[10,10,20]` and `[10,20]` are not the same list in Python.
2. *Python lists are ordered, while mathematical sets aren't*. For example, the two lists `[3,4,5]` and `[4,3,5]` are considered different, but the two sets $\{3, 4, 5\}$ and $\{4, 3, 5\}$ are equal.

It is possible to invoke some additional Python structure to treat lists like sets, using the command `set`. By running a list through this function, the list is changed into a different data type (namely type `set`) that can then be manipulated like a set. You can read about this here; but in this AEP, you're going to write some code that will let us manipulate lists like sets, without changing lists to the `set` data type.

Setup (do this first)

Before doing the tasks below, you'll need to review what you learned in the Python/Jupyter crash course, especially about lists. You can go back into CodeAcademy and review the lessons; or there are millions of websites and videos that also review the basics of Python lists.

Also, in this AEP you're going to learn a new set operation called the **symmetric difference**. The symmetric difference of the sets A and B is the set of all elements that are in A and in B, but not in both. The Venn diagram looks like this:



Tasks for this AEP

Write the following *four* Python functions:

- `to_set`: This function accepts a list and returns a modified version of that list that still is of the `list` data type, but which has no duplicate elements. Examples:

```
to_set([1,2,3,1,3,4])
> [1,2,3,4]

to_set(['John', 'Paul', 'John', 'George', 'Ringo']) # List could be any data type
> ['John', 'Paul', 'George', 'Ringo']

to_set([3,4]) # If a list already is a valid set, there are no changes.
> [3,4]

to_set([])  # The empty set is represented as the empty list.
> []
```

Please note the ">" character above merely represents a prompt, where the output happens. It is not part of the output itself.

- `union`: This function accepts two sets (represented as lists) and returns their set union. Example:

```
union([1,3,4], [2,3,1])
> [1,2,3,4]
```

For this AEP, we will just ignore any difference in ordering. For example, if your code produced the set `[3,4,2,1]` in the above example, that would be OK.

- `intersect`: This function accepts two sets (represented as lists) and returns their set intersection. Example:

  ```
  intersect([1,3,4], [2,3,1])
  > [1,3]

  intersect([10, 20], ['a', 'b'])
  > []
  ```

- `symmetric_diff`: This function takes two sets (again, represented as lists) and produces the symmetric difference. Example:

  ```
  symmetric_diff([1,2,3], [2,3,4])
  > [1,4]
  ```

Assignment Expectations and Grading Criteria

Assumptions you may make: Although the ordering matters in a Python list, for the purposes of this assignment, you may ignore the ordering. For example, `[1,2,3]` is considered unequal to `[2,3,1]` as lists, but you may consider them to be equal as sets.

Restrictions you need to follow:

- You are not allowed to use any external libraries on this problem. All your code should be "from scratch".
- You are not allowed to use the `set` function that converts lists to sets. All of your objects in this assignment should be lists and remain lists.

This AEP is unusual (for now) because there are no other problems to solve, just code to write. But it's still the case that **your grade is primarily based on your explanations and writing, and only secondarily on the precision and correctness of your computations.** What this means here is that **along with your code, you need to include verbal explanations of what your code is doing and why it works, as if you were explaining it in words to a human being**.

In order to have both executable code and verbal explanations in the same place, **your work must be submitted as a Jupyter notebook** that combines your code with the verbal explanations of what your code does and why it works. Put your code in code cells, and your verbal explanations in text cells. How you organize it is up to you; you can put verbal explanations before each function, or you can have one large verbal explanation before or after all four of the functions. However, please *do not put all your verbal* *explanations in with the code as comments*; this is OK programming practice but it's hard to read.

Here's an example of the kind of thing you should be creating:
https://colab.research.google.com/drive/1EwzR9kRDQFpQCyImasA-wx6F9eTYN1WH?usp=sharing
Notice how the text is in a text cell, and how it explains what is happening in the code in (relatively) plain English. (Comments are turned on in this notebook, so if you have a question about something you see, leave a comment and I'll address it.)

This assignment will be graded by running your code using a text file full of test cases I have prepared, and seeing if your code produces the right results; and by evaluating the quality of your verbal explanations.

- A grade of **E** is given if 100% of the test cases produce the correct results and if the verbal explanations provided are clear, correct, and well-written.
- A grade of **M** is given if 80% of the test cases produce the correct results verbal explanations provided are correct and reasonably well-written.
- A grade of **R** is given if fewer than 80% of the test cases produce the correct results; or if the verbal explanations provided are unclear, incorrect, or poorly written (either because of grammar/clarity issues or because they don't actually explain your code).
- A grade of **N** is given under any of the following conditions:
    - Your functions are named something other than the names stated in the problem. Because the grading is automated, they **must** be named `to_set`, `union`, `intersect` (not "intersectION"), and `symmetric_diff`. *It's your responsibility to double check that you have spelled the names of the functions correctly*.
    - There is a syntax error in your code; i.e. when the code is executed with control-enter in the Jupyter notebook, an error is produced. *It's your responsibility to turn in work that is free of syntax errors*.
    - The verbal explanations are missing, incomplete, irrelevant, or contain significant issues with clarity or correctness.
    - There are significant omissions in the work, for example missing verbal explanations or one or more of the functions is not submitted.
    - There are significant issues with the formatting or organization of your notebook.

Submitting your work

For this AEP, as mentioned above, your work needs to be packaged in a Jupyter notebook. Work not given as Jupyter notebooks will not be graded.

The **preferred** way to do this is as follows:

1. Write up your Jupyter notebook in Colab.
2. Change the sharing settings on your notebook so that **anyone with the link can edit** and then copy the link. Here is a short video on how to do this.
3. In the AEP assignment, paste the link as "Submission text". There will be no file to upload this time; all I need is the link. I'll then open the link and check your code against my test cases.

Note: This process is different than sharing the notebook with just me, as many of you did with the Python/Jupyter crash course. It turns out that sharing the notebook with just me, the email to me that contains the link ended up in my spam folder about 90% of the time. So, **please do it this way or I will probably not see your submission.**

If you cannot carry out these steps or have some other process you prefer, please let me know first.

Getting Help

Please note that according to the syllabus, for AEP's **"no interactions at all with another person or with unauthorized sources on the internet is allowed."** Violations of this rule include *any* consultation with other students or former students, including Math Center tutors; using work from another student or former student; submitting the problem set to an online help site such as Chegg or Coursehero; or asking for help in an online forum. All such violations will be treated as academic dishonesty and will result in a grade of "N" and being banned from revising the work.

You **may** ask me (Talbert) for help on this assignment in the form of **specific mathematical or technical questions**. If I cannot answer a question because it would give too much away, I'll tell you so. **However please note: I will not "look over your work" before you submit it to give you feedback on the overall submission**; the expectations are clearly laid out above, so just follow those directions and submit your best work, and you'll be allowed to revise it if needed.

You can ask technology related questions to anyone at any time. For example if you need help with Desmos, or with figuring out how to type up your work, there are no restrictions on that. I recommend the `#tech` channel on Campuswire so that you'll reach a large audience.