index

January 7, 2022

1 Introduction to Sets - Lab

1.1 Introduction

Probability theory is all around. A common example is in the game of poker or related card games, where players try to calculate the probability of winning a round given the cards they have in their hands. Also, in a business context, probabilities play an important role. Operating in a volatile economy, companies need to take uncertainty into account and this is exactly where probability theory plays a role.

As mentioned in the lesson before, a good understanding of probability starts with understanding of sets and set operations. That's exactly what you'll learn in this lab!

1.2 Objectives

You will be able to:

- Use Python to perform set operations
- Use Python to demonstrate the inclusion/exclusion principle

1.3 Exploring Set Operations Using a Venn Diagram

Let's start with a pretty conceptual example. Let's consider the following sets:

- Ω = positive integers between [1, 12]
- A= even numbers between [1, 10]
- $B = \{3, 8, 11, 12\}$
- $C = \{2, 3, 6, 8, 9, 11\}$

a. Illustrate all the sets in a Venn Diagram like the one below. The rectangular shape represents the universal set.

b. Using your Venn Diagram, list the elements in each of the following sets: Do this work by hand (writing in the values of each set), then you will check your answers using Python code later!

For example, if the question was just asking for the values of B, you would replace `None` with `{3, 8, 11, 12}` typed out.

$ A B$

[2]:
```
ans1 = {8}
ans1
```

[2]:
```
{8}
```

$ A C$

[3]:
```
ans2 = {2,3,4,6,8,9,10, 11}
ans2
```

[3]:
```
{2, 3, 4, 6, 8, 9, 10, 11}
```

A^c

[5]:
```
ans3 = {1, 3, 5, 7, 9, 11, 12}
ans3
```

[5]:
```
{1, 3, 5, 7, 9, 11, 12}
```

The absolute complement of B

[6]:
```
ans4 = {1, 2, 4, 5, 6, 7, 9, 10}
ans4
```

[6]:
```
{1, 2, 4, 5, 6, 7, 9, 10}
```

$(A \cup B)^c$

[13]:
```
ans5 = {1, 5, 7, 9}
ans5
```

[13]:
```
{1, 5, 7, 9}
```

$B \cap C'$

[14]:
```
ans6 = {12}
ans6
```

[14]:
```
{12}
```

$A \backslash B$

[15]:
```
ans7 = {2, 4, 6, 10}
ans7
```

[15]:
```
{2, 4, 6, 10}
```

$C \backslash (B \backslash A)$

[16]:
```
ans8 = {2, 6, 8, 9}
ans8
```

[16]:
```
{2, 6, 8, 9}
```

$(C \cap A) \cup (C \backslash B)$

[17]:
```
ans9 = {2, 6, 8, 9}
ans9
```

[17]:
```
{2, 6, 8, 9}
```

c. For the remainder of this exercise, let's create sets A, B and C and universal set U in Python and test out the results you came up with. Sets are easy to create in Python. For a guide to the syntax, follow some of the documentation here

[18]:
```
# Create set A
A = {2, 4, 6, 8, 10}
'Type A: {}, A: {}'.format(type(A), A) # "Type A: <class 'set'>, A: {2, 4, 6,␣
↪8, 10}"
```

[18]:
```
"Type A: <class 'set'>, A: {2, 4, 6, 8, 10}"
```

[19]:
```
# Create set B
B = {3,8,11,12}
'Type B: {}, B: {}'.format(type(B), B) # "Type B: <class 'set'>, B: {8, 11, 3,␣
↪12}"
```

[19]:
```
"Type B: <class 'set'>, B: {8, 11, 3, 12}"
```

[20]:
```
# Create set C
C = {2, 3, 6, 8, 9, 11}
'Type C: {}, C: {}'.format(type(C), C) # "Type C: <class 'set'>, C: {2, 3, 6,␣
↪8, 9, 11}"
```

[20]:
```
"Type C: <class 'set'>, C: {2, 3, 6, 8, 9, 11}"
```

[21]:
```
# Create universal set U
U = set(range(1, 13))
'Type U: {}, U: {}'.format(type(U), U) # "Type U: <class 'set'>, U: {1, 2, 3,␣
↪4, 5, 6, 7, 8, 9, 10, 11, 12}"
```

[21]:
```
"Type U: <class 'set'>, U: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}"
```

Now, verify your answers in section 1 by using the correct methods in Python. For example, if the question was just asking for the values of B, you would replace `None` with `B`.

To provide a little bit of help, you can find a table with common operations on sets below.

Method	Equivalent	Result
s.issubset(t)	s <= t	test whether every element in s is in t
s.issuperset(t)	s >= t	test whether every element in t is in s
s.union(t)	s | t	new set with elements from both s and t
s.intersection(t)	s & t	new set with elements common to s and t
s.difference(t)	s - t	new set with elements in s but not in t
s.symmetric_difference(t)	s ^ t	new set with elements in either s or t but not both

Every cell should display `True` if your original answer matches the answer you calculated with Python. If it displays `False`, that means either your original answer or your Python code is incorrect.

1. $ A B$

[12]:
```
A_inters_B = A.intersection(B)
A_inters_B == ans1
```

[12]:
```
True
```

2. $ A C $

[23]:
```
A_union_C = A.union(C)
A_union_C == ans2
```

[23]:
```
True
```

3. A^c

[24]:
```
A_comp = U.difference(A)
A_comp == ans3
```

[24]:
```
True
```

4. The absolute complement of B

[25]:
```
B_comp = U.difference(B)
B_comp == ans4
```

[25]:
```
True
```

5. $(A B)^c $

```
[26]: A_union_B_comp = U.difference(A.union(B))
      A_union_B_comp == ans5
```

```
[26]: True
```

6. $B C' $

```
[27]: B_inters_C_comp = B.intersection(U.difference(C))
      B_inters_C_comp == ans6
```

```
[27]: True
```

7. $A\backslash B$

```
[28]: compl_of_B = A.difference(B)
      compl_of_B == ans7
```

```
[28]: True
```

8. $C \(B \A) $

```
[29]: C_compl_B_compl_A = C.difference(B.difference(A))
      C_compl_B_compl_A == ans8
```

```
[29]: True
```

9. $(C\cap A)\cup(C\backslash B)$

```
[30]: C_inters_A_union_C_min_B = (C.intersection(A)).union(C.difference(B))
      C_inters_A_union_C_min_B == ans9
```

```
[30]: True
```

1.4 The Inclusion Exclusion Principle

Use A, B and C from exercise one to verify the inclusion exclusion principle in Python. You can use the sets A, B and C as used in the previous exercise.

Recall from the previous lesson that:

$$|A\cup B\cup C| = |A| + |B| + |C| - |A\cap B| - |A\cap C| - |B\cap C| + |A\cap B\cap C|$$

Combining these main commands:

Method	Equivalent	Result
a.union(b)	A | B	new set with elements from both a and b
a.intersection(b)	A & B	new set with elements common to a and b

along with the `len(x)` function to get to the cardinality of a given x (“|x|”).

What you’ll do is translate the left hand side of the equation for the inclusion principle in the object `left_hand_eq`, and the right hand side in the object `right_hand_eq` and see if the results are the same.

```
[35]: left_hand_eq = len(A.union(B.union(C)))
      print(left_hand_eq)  # 9 elements in the set

      # From Github

      # left_hand_eq = len(A | B | C)
      # print(left_hand_eq)  # 9
```

```
9
```

```
[36]: right_hand_eq = (len(A) + len(B) + len(C) - len(A.intersection(B))
                       - len(A.intersection(C)) - len(B.intersection(C))
                        + len(A.intersection(B.intersection(C))))
      print(right_hand_eq) # 9 elements in the set

      # From Github
      # right_hand_eq = len(A) + len(B) + len(C) - len(A&B) - len(A&C) - len(B&C) +␣
       ↪len(A&B&C)
      # print(right_hand_eq) # 9
```

```
9
```

```
[37]: left_hand_eq == right_hand_eq
      # None # Use a comparison operator to compare `left_hand_eq` and␣
       ↪`right_hand_eq`. Needs to say "True".
```

```
[37]: True
```

1.5 Set Operations in Python

Mary is preparing for a road trip from her hometown, Boston, to Chicago. She has quite a few pets, yet luckily, so do her friends. They try to make sure that they take care of each other’s pets while someone is away on a trip. A month ago, each respective person’s pet collection was given by the following three sets:

```
[41]: Nina = set(["Cat","Dog","Rabbit","Donkey","Parrot", "Goldfish"])
      Mary = set(["Dog","Chinchilla","Horse", "Chicken"])
      Eve = set(["Rabbit", "Turtle", "Goldfish"])
```

In this exercise, you’ll be able to use the following operations:

Operation	Equivalent	Result
s.update(t)	$s \mid= t$	return set s with elements added from t
s.intersection_update(t)	s &= t	return set s keeping only elements also found in t
s.difference_update(t)	s -= t	return set s after removing elements found in t
s.symmetric_difference_update(t)	s ^= t	return set s with elements from s or t but not both
s.add(x)		add element x to set s
s.remove(x)		remove x from set s
s.discard(x)		removes x from set s if present
s.pop()		remove and return an arbitrary element from s
s.clear()		remove all elements from set s

Sadly, Eve's turtle passed away last week. Let's update her pet list accordingly.

```
[43]: Eve.remove("Turtle")
      Eve # should be {'Rabbit', 'Goldfish'}
```

```
[43]: {'Goldfish', 'Rabbit'}
```

This time around, Nina promised to take care of Mary's pets while she's away. But she also wants to make sure her pets are well taken care of. As Nina is already spending a considerable amount of time taking care of her own pets, adding a few more won't make that much of a difference. Nina does want to update her list while Mary is away.

```
[44]: Nina.update(Mary)
      Nina # {'Chicken', 'Horse', 'Chinchilla', 'Parrot', 'Rabbit', 'Donkey', 'Dog',␣
       ↪'Cat', 'Goldfish'}
```

```
[44]: {'Cat',
       'Chicken',
       'Chinchilla',
       'Dog',
       'Donkey',
       'Goldfish',
       'Horse',
       'Parrot',
       'Rabbit'}
```

Mary, on the other hand, wants to clear her list altogether while away:

[45]:
```
Mary.clear()
Mary   # set()
```

[45]:
```
set()
```

Look at how many species Nina is taking care of right now.

[46]:
```
n_species_Nina = len(Nina)
n_species_Nina # 9
```

[46]:
```
9
```

Taking care of this many pets is weighing heavily on Nina. She remembered Eve had a smaller collection of pets lately, and that's why she asks Eve to take care of the common species. This way, the extra pets are not a huge effort on Eve's behalf. Let's update Nina's pet collection.

[47]:
```
Nina.difference_update(Eve)
Nina # 7
```

[47]:
```
{'Cat', 'Chicken', 'Chinchilla', 'Dog', 'Donkey', 'Horse', 'Parrot'}
```

Taking care of 7 species is something Nina feels comfortable doing!

1.6 Writing Down the Elements in a Set

Mary dropped off her pets at Nina's house and finally made her way to the highway. Awesome, her vacation has begun! She's approaching an exit. At the end of this particular highway exit, cars can either turn left (L), go straight (S) or turn right (R). It's pretty busy and there are two cars driving close to her. What you'll do now is create several sets. You won't be using Python here, it's sufficient to write the sets down on paper. A good notion of sets and subsets will help you calculate probabilities in the next lab!

Note: each set of action is what *all three cars* are doing at any given time

a. Create a set A of all possible outcomes assuming that all three cars drive in the same direction.

b. Create a set B of all possible outcomes assuming that all three cars drive in a different direction.

c. Create a set C of all possible outcomes assuming that exactly 2 cars turn right.

d. Create a set D of all possible outcomes assuming that exactly 2 cars drive in the same direction.

e. Write down the interpretation and give all possible outcomes for the sets denoted by:

- I. D'
- II. $C \cap D$,
- III. $C \cup D$.

1.7 Optional Exercise: European Countries

Use set operations to determine which European countries are not in the European Union. Use the `Country` column. You just might have to clean the data first with pandas.

Note that this data is from 2018, so EU membership may have changed.

[48]:
```
import pandas as pd

# Load Europe and EU
europe = pd.read_excel('Europe_and_EU.xlsx', sheet_name = 'Europe')
eu = pd.read_excel('Europe_and_EU.xlsx', sheet_name = 'EU')

# Your code here to remove any whitespace from names
```

Preview data:

[49]:
```
europe.head(3)
```

[49]:
```
   Rank    Country  Population  % of population  \
0   1.0     Russia   143964709            17.15
1   2.0    Germany    82521653             9.80
2   3.0     Turkey    80810000             9.60

   Average relative annual growth (%)  Average absolute annual growth  \
0                                0.19                          294285
1                                1.20                          600000
2                                1.34                         1035000

  Estimated doubling time (Years)  Official figure (where available)  \
0                             368                          146839993
1                              90                           82800000
2                              52                           77695904

   Date of last figure Regional grouping             Source
0  2017-01-01 00:00:00              EAEU                [1]
1  2016-12-31 00:00:00                EU  Official estimate
2  2016-12-31 00:00:00               NaN                [2]
```

[50]:
```
eu.head(3)
```

[50]:
```
   Rank          Country  2017 population  % of pop.  \
0     1          Germany         82800000      16.18
1     2           France         67210459      13.10
2     3   United Kingdom         65808573      12.86

   Average relative annual growth  Average absolute annual growth  \
0                            0.76                          628876
1                            0.40                          265557
```

```
2                             0.65                          428793

   Official figure Date of last figure                       Source
0         82576900          2017-03-31            Official estimate
1         67174000          2018-01-01   Monthly official estimate
2         65648100          2017-06-30            Official estimate
```

Your code comes here:

```
[56]: # Check to confirm that the EU countries are a subset of countries in Europe
      set(eu["Country"]).issubset(set(europe["Country"]))

      # From GitHub
      # set(eu.Country) < set(europe.Country)
```

```
[56]: True
```

```
[58]: # Find the set of countries that are in Europe but not the EU
      set(europe["Country"]).difference(set(eu["Country"]))

      # From GitHub
      # set(europe.Country) - set(eu.Country)
```

```
[58]: {'  Switzerland ',
       '  Vatican City ',
       ' Albania ',
       ' Andorra ',
       ' Armenia ',
       ' Azerbaijan ',
       ' Belarus ',
       ' Bosnia and Herzegovina ',
       ' Faroe Islands (Denmark) ',
       ' Georgia ',
       ' Gibraltar (UK)',
       ' Guernsey (UK) ',
       ' Iceland ',
       ' Isle of Man (UK) ',
       ' Jersey (UK) ',
       ' Kosovo ',
       ' Liechtenstein ',
       ' Macedonia ',
       ' Moldova ',
       ' Monaco ',
       ' Montenegro ',
       ' Norway ',
       ' Russia',
       ' San Marino ',
       ' Serbia ',
       ' Svalbard (Norway) ',
       ' Turkey ',
       ' Ukraine',
       ' Åland Islands (Finland) '}
```

1.8 Summary

In this lab, you practiced your knowledge on sets, such as common set operations, the use of Venn Diagrams, the inclusion exclusion principle, and how to use sets in Python!