KIT103/KMA155 Practical 02: Set operations

- Preparation
- 1 Working with sets
  - 1.1 Getting started with the student_db script
  - 1.2 Visualising the overlaps
  - 1.3 Getting some answers
  - 1.4 A little data analysis
- 2 (optional) Extension exercise: Bags
- Notes
  - Sources of documentation

This week we'll apply set operators to investigate the relationships between student enrolments in some fictitious units.

Preparation

In the Labs section of MyLO download the file **student_db.py**. (Note that MyLO makes this difficult and you will need to click the Download button that has a green circle containing an arrow; do this for any future Python script files we ask you to download from MyLO.) Save it to wherever you are keeping your programming materials for this unit.

Start Spyder and open student_db.py. It represents a simple 'database' of student and enrolment information, and has some helper functions to make using it simpler. You don't need to understand everything in this file right now; you can spend as much time as you want dissecting it later. For the moment note that it illustrates:

- importing functions from modules (.py script files that are already installed on the system);
- defining functions; and
- declaring some sets and a dictionary that we will use in today's exercises.

Press F5 or click the green 'play' arrow to run the script. Use the default settings in the window the pops up. The rest of your interaction will be in the IPython window.

1 Working with sets

The fictitious student database contains student details and enrolment information (sets of student IDs) for three units: TIK301, IKT130, KTI310. The sets for these are those names but all in lower case, as in `tik301`.

1.1 Getting started with the student_db script

The help function can be used to get information about functions and modules. Enter the following at the prompt to get an overview of the student_db module:

```
help('student_db')
```

Task: Try out `display_all()` and then `display_subset(kti310)`. Try out the other functions in student_db if you want.

The 'database' is stored in the `dict` object called `db`. To select a single student from the database, such as student 7907, use:

```
db[7907]
```

If you want the set of all student IDs then use:

```
set(db)
```

which will create a set from the keys in the db `dict`.

Task: How many students are there in total? How many in each of the units?

1.2 Visualising the overlaps

A first step in most data analysis is to visualise the data so that any obvious patterns can be identified for subsequent numerical analysis.

Task: Run the `visualise_overlaps()` function (one of the helper functions in student_db). It will display a Venn diagram of the number of students enrolled in each unit and the intersections between units.

1.3 Getting some answers

Task: Use set operators to answer the questions below. The result of each expression (i.e., combination of sets and set operators) will be a set of student IDs and you use the `display_subset()` function to display their details if you want. The expected cardinality of each result is shown in parentheses after the question. Use `len()` on each expression you write to check the size; don't count the results by hand.

Tip: Using I, K and T to represent the sets `ikt130`, `kti310` and `tik301`, respectively, write down the mathematical expression that answers each question *before* writing the code.

1. Who are the 10 students studying KTI310?
2. Who are the 20 students studying TIK301?
3. Who is studying both KTI310 and TIK301? (5)
4. Who is studying all three units? (2)
5. Who is the lone student studying both IKT130 and KTI310 only? (1)
6. Who is studying KTI310 and TIK301 but *not* IKT130? (3)
7. Who is studying *only* TIK301? (7)
8. Who is studying TIK301 or KTI310, but not both? (20)
9. (optional; with more than two sets this becomes messy) Who is studying exactly one of each of the units? (20; if you get 22 then you need to try a different approach)
10. Who is not studying any of the units? (25)

1.4 A little data analysis

The following expression will identify students with ages between 18 and 22:

```
{ s['sid'] for s in db.values() if 18 <= s['age'] <= 22 }
```

so by changing the values 18 and 22 you can select students in different age brackets. In contrast, this expression will return the set of ages for students enrolled in KTI310:

```
{ s['age'] for s in db.values() if s['sid'] in kti310 }
```

Task: Using any variation or combination of these two set comprehensions (or your own novel creations), answer the following:

1. How many students aged 18 to 20 are enrolled in IKT130?
2. What actual ages are represented in that group? (There should be two distinct values.)

2 (optional) Extension exercise: Bags

Your answer to question 2 above was most likely a set showing the ages represented, but not how many there were of each. Depending on the level of challenge you want, you can approach this next task either by writing your own functions to create a `dict` that represents a bag or by importing the `Counter` class with

```
from collections import Counter
```

This is Python's version of a bag and works mostly as you'd expect except that element counts can be <= 0.

Task: Adapt your code for answering question 2 in Section 1.4 above so that it produces a list instead of a set. Use that to create a bag either using your own code or by creating a `Counter`, as in:

```
age_counts = Counter(ages_as_list)
```

Notes

Sources of documentation

There are multiple ways of obtaining information about functions, data types and modules in Python. A recommended way when using the IPython terminal is to use the `help()` function.

`help(`*thing-you-want-help-with*`)` will display any documentation information written by the author of the *thing-you-want-help-with*. Some examples (using today's `student_db.py` script):

- on a function: `help(display_all)`

- on the `student_db.py` script after it's been run (note the quotes around the name): `help('student_db')`
- on a module that has been `import`ed: `help(matplotlib_venn)` (note that this will only work if you import the entire module, not just one function from it, as in `import matplolib_venn`)

And, of course, an internet search will readily find a number of suitable answers to your Python queries, as will a search of the documentation for the Python Standard Library.