Practical 8
Exploring Structured Data

Learning Objectives

1. Understand and implement structured data processing in Python using the pandas library
2. Understand and critique the value of reproducible research
3. Apply and create Python notebooks to support exploratory research

Overview

We will be using python notebooks for this practical to keep track of our steps and get use to how they work. The activities will surround the use of pandas to work with structured data.

Tasks

1. Running a Jupyter notebook

Create a Prac08 directory for this practical and change into it.

Type: **jupyter notebook** at the command line to start the jupyter notebook in your browser.

Once in the dashboard for jupyter, create a new notebook with the default kernel and call it "tuple" (creates the file "tuple.ipynb")

Create a markdown cell to give a short description: "Testing out jupyter notebook with Tuple task"

Create a code cell and type in the following:

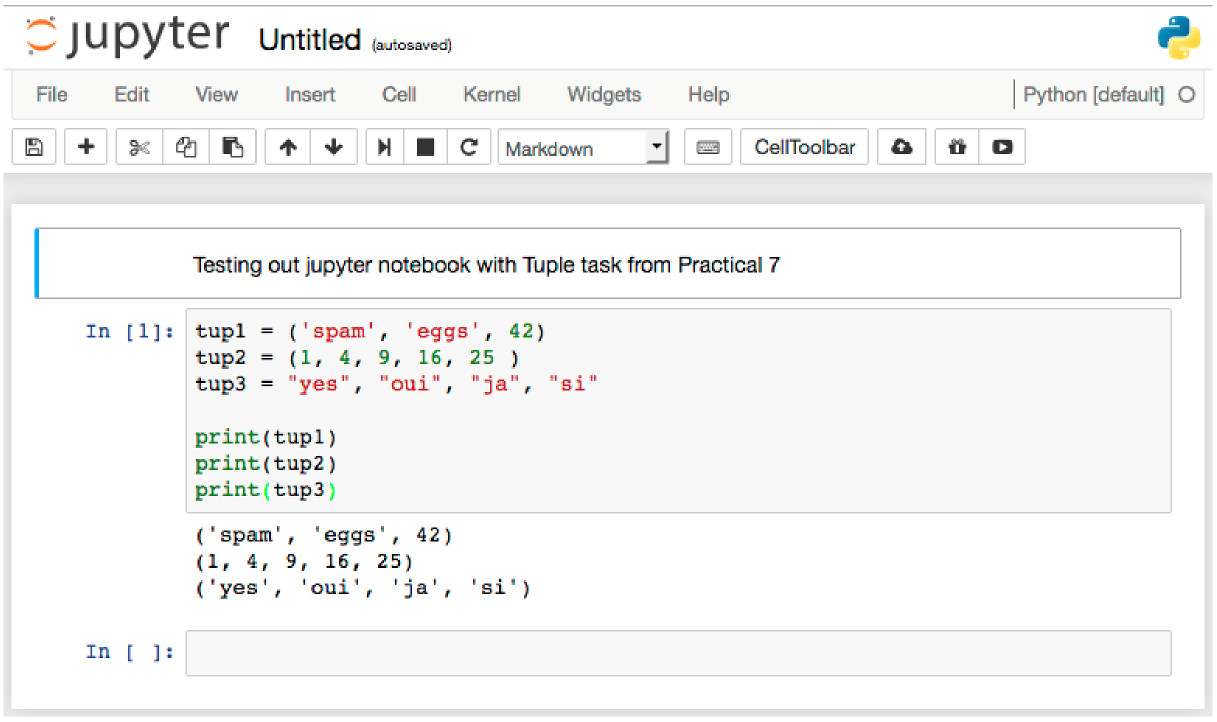
```
tup1 = ('spam', 'eggs', 42)
tup2 = (1, 4, 9, 16, 25 )
tup3 = "yes", "oui", "ja", "si"
print(tup1)
print(tup2)
print(tup3)
```

Modify the code to:

- Create tup4 as tup2 and tup3 added together
- Print out the length of tup4
- Print out the values in tup4, omitting the first and last values
- Create a new tuple, tup5. It should be similar to tup2 – this time holding the squares of numbers from 1-10 inclusive.

Hints: look at the lecture slides for similar operations. For tup5, you can create a list and convert to a tuple with tup5 = tuple(tuplelist)

Your window should look a bit like this:



Testing out jupyter notebook with Tuple task from Practical 7

```
In [1]: tup1 = ('spam', 'eggs', 42)
        tup2 = (1, 4, 9, 16, 25 )
        tup3 = "yes", "oui", "ja", "si"

        print(tup1)
        print(tup2)
        print(tup3)

        ('spam', 'eggs', 42)
        (1, 4, 9, 16, 25)
        ('yes', 'oui', 'ja', 'si')

In [ ]:
```

Save the notebook using the save button, then close and halt the notebook (File menu).

Go to the command line and do an ls in the directory to see that the notebook file is there – tuple.ipynb.

2. Movie Sets

Extending on the sets example in the lecture notes, we will make sets of actors to compare using set operations.

Pythons: Eric, John, Terry, Michael, Terry, Graham
Goodies: Bill, Tim, Graham
Wandas: John, Jamie, Kevin, Michael
Yellowbeards: Graham, Peter, Marty, Eric, Martin, Madeline, John
Yorkshiremen: Tim, John, Graham, Marty
Yorkshiremen2: Terry, Michael, Eric, Graham, Graham

In Jupyter notebook create six sets based on the movies and actors above. Then do the following, checking the output matches your understanding:

- Print each of the sets

- Print the **intersection** of (Pythons and Yellowbeards) & (Goodies and Yorkshiremen)
- Print the **union** of (Goodies and Yorkshiremen) & (Pythons and Wandas)
- Print the **difference** between (Yellowbeards and Wandas) & (Yorkshiremen and Yorkshiremen2)

The original Four Yorkshiremen Sketch: https://www.youtube.com/watch?v=VKHFZBUTA4k

3. Rolling Dice

Go to the jupyter dashboard and create a new notebook called "dice". Type the program for rolling dice from the Lecture 8 slides into your notebook.

When you have executed the code, check that your plot of the dice rolls looks correct. Next, enter the commands to do the following (put a comment before each one to describe what you are doing):

- add a plot title "Dice Rolls (1000)"
- add an x-axis label "Number"
- add a y-axis label "Count"
- change the plot colour to green
- use plt.savefig() to save your plot
- plot the file in the notebook again

Notice how the notebook keeps the status so far so you don't need to repeat the imports and early code to redo the plotting parts.

4. Dictionaries

Dictionaries or maps let us connect a key to a value, using key-value pairs. We can then manipulate these dictionaries to gain additional information. Type in the code for state populations in the lecture notes, calling the program **pops.py.**

Extend pops.py to do the following:
- Print the populations of each state
- Print the total population across all states
- Print the states and populations where the population is less than 3,000,000

Test if "New Zealand" is a state and print the result (remember that a dictionary is a set – so use set operations)

5. Running a Python Script in Jupyter

In your notebook (dice.ipynb should still be open) run pops.py by typing:

```
%run pops.py
```

You can also run it in a different directory:

```
%run ../Prac09/pops.py
```

Type "pops" into a code cell and execute it. Notice that you can get the values of any variables that have previously been used in your notebook – they're all still there. Try typing "dicecount" into a code cell – it's still there.

Make sure that your notebook is saved, then switch to the terminal window and type "ctrl-C" and then "y" to clode down the notebook.

6. Storytime

Project Gutenberg offers over 50,000 books in electronic format. They are available as text files – which means we can use Python scripts to process them. Go to the Project Gutenberg website (http://www.gutenberg.org/) and then scroll down to the Site Map and click through on Most Downloaded Books. You can access any of these books through these links.

Click on Grimms' Fairy Tales and see the different file types available. Download the text file version to your Prac8 directory. We just want to work with Rumpelstiltskin, so follow these steps:

1. Copy the grimm text file to rumple.txt
2. Open rumple.txt in vim
3. If you don't have line numbers showing, type ":set number" (and "set nonumber" to turn them off)
4. Find the start of Rumpelstiltskin by typing "/RUMPEL" (type n to go to the next occurrence if you're at the contents page)
5. Take note of the line number before the opening title
6. Go to the first line of the file using "1G"
7. Delete the lines up to the start of the story by typing "<lineNo>dd" – so if the line before the story was 3634, type "3634dd"
8. Find the end of the story by typing "/RUMPEL" again
9. Go to the next line after the end of the story and type "dG"
10. You should have a file about 113 lines long
11. Save and exit the file

The lecture notes included code to count words in a text file. Type in that code and analyse the rumpel.txt file. See if your answers match the slides.

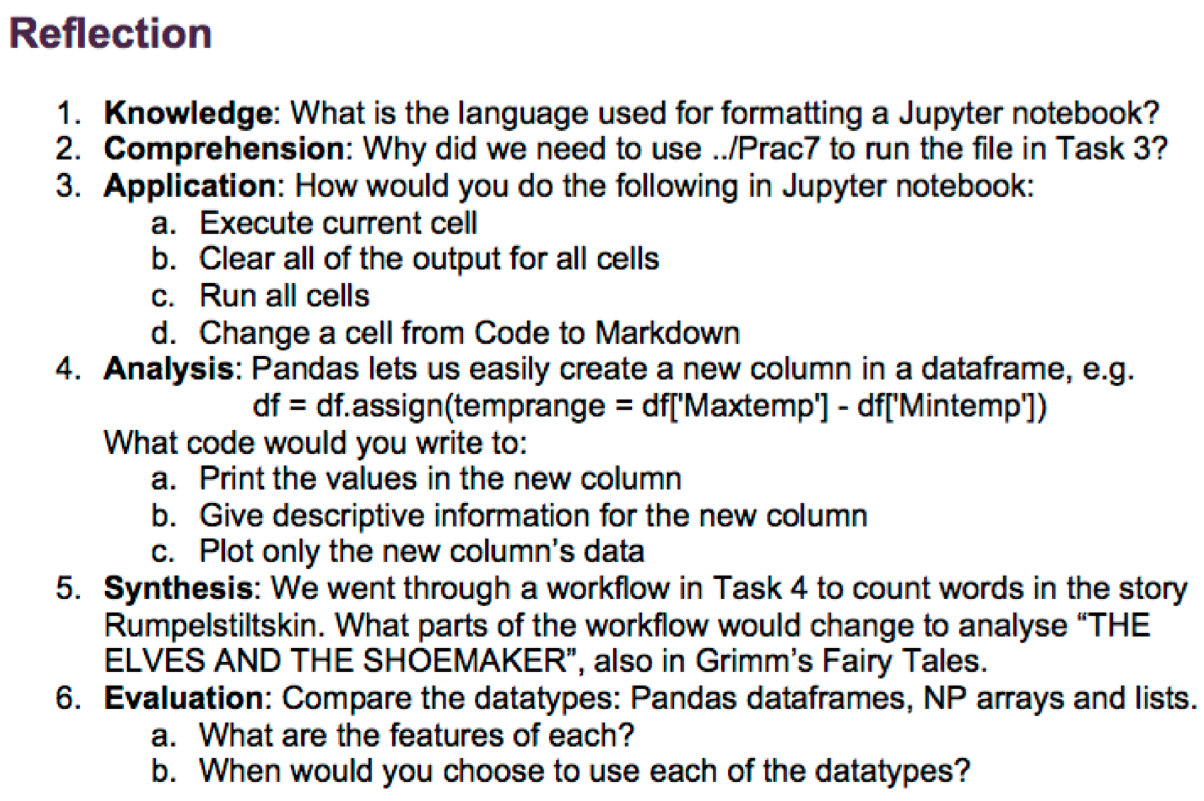
7. Hello Pandas

Download surveys.csv from the unit website into the Prac08 directory. This file is from the Data Carpentry tutorial, which is highly recommended - http://www.datacarpentry.org/python-ecology-lesson/ .

Start jupyter again from the Prac08 directory. Create a new notebook called "pandasurvey". Go through the slides from the lecture and run all the commands in your notebook. Check that your results match those in the lecture slides.

Submission

Create a README file for Prac08. Include the names and descriptions of all of your Python programs from today. All of your work for this week's practical should be submitted via Blackboard using the link in the assessments area. This should be done as a single "zipped" file.



Reflection

1. **Knowledge**: What is the language used for formatting a Jupyter notebook?
2. **Comprehension**: Why did we need to use ../Prac7 to run the file in Task 3?
3. **Application**: How would you do the following in Jupyter notebook:
   a. Execute current cell
   b. Clear all of the output for all cells
   c. Run all cells
   d. Change a cell from Code to Markdown
4. **Analysis**: Pandas lets us easily create a new column in a dataframe, e.g.
   ```
   df = df.assign(temprange = df['Maxtemp'] - df['Mintemp'])
   ```
   What code would you write to:
   a. Print the values in the new column
   b. Give descriptive information for the new column
   c. Plot only the new column's data
5. **Synthesis**: We went through a workflow in Task 4 to count words in the story Rumpelstiltskin. What parts of the workflow would change to analyse "THE ELVES AND THE SHOEMAKER", also in Grimm's Fairy Tales.
6. **Evaluation**: Compare the datatypes: Pandas dataframes, NP arrays and lists.
   a. What are the features of each?
   b. When would you choose to use each of the datatypes?