School of Computer Science

CS5901 Programming in Python

Practical 2

Credits: 30% of the module

Aim / Learning objectives

The objectives of this practical are:

- Learn how to investigate, clean and prepare data
- Analyse the complexity of algorithms

Set-up

You are **only allowed** to use the following imported packages for this practical. No off-the-shelf machine learning packages such as *scikit-learn* are allowed.

In [2]:
```python
import pandas as pd
import numpy as np
import time
import gc
import psutil
import os
```

Data

- You will use the data in **P2data????.csv**, where **????** denotes the last four digits of your matriculation number. This is obtained by emailing **cs5901.staff**
- An existing dataset will be sampled and sent you you in CSV format

Instructions

Stage 0. Code repository

- Code repos are an essential resource for the modern programmer
- Normally you'd push all your commited changes to a server, but we'll skip this step
- Everything below assumes use of *git*, but you can use any other suitable repository tool

1. Install *git* locally (or the repo of your choice)
   - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
2. Create a repo
   - https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository

3. Add your initial code to the repo, and commit changes as you update the code
    - https://www.atlassian.com/git/tutorials/saving-changes/git-commit
4. Include your git log in the notebook part of the submission
    - https://git-scm.com/docs/git-log

Again, don't *git push* anything if you are working locally

Stage 1. Data cleaning and profiling

1. Import the data as a pandas dataframe
2. Identify and remove any data rows that make no sense
3. Replace any missing values in T3 and T4 with average values **for their specific level**
4. Write code - without using *pandas.describe()* - that presents a table of descriptive statistics for each column, then compare you results to those obtained using *pandas.describe()*
5. Write code that identifies any repeated rows or confirms that there are none

Stage 2. Time and space complexity

- The code below is an undocumented example of one way to find the CPU time and RAM space needed to perform calculations on randomly generated data, and investigate how the time and space needed grows as the size of the problem increases

1. Using adaptations of the code below, or the method of your choice, investigate the time and space needed for:
    - standard matrix multiplication (for time this should be cn^3 where c is a non-negative constant)
    - sorting an unordered list of integers inefficiently by searching for the smallest element, then the next smallest,...
    - a comparison of Python's *find()* method and your loop-based implementation of a method that checks if one string is a substring of another

In [3]:
```
def myInv(size):
    intA = np.random.randint(-500,50000,  (size,size))
    flA = np.random.rand(size,size)
    A = intA + flA
    b = np.random.randint(-10,100,size)
    return(np.linalg.inv(A))

process = psutil.Process(os.getpid())
baseRam = process.memory_info().rss

resTime = np.zeros(10)
resSpace = np.zeros(10)

for i in range(1,11):
    gc.collect()
    start = time.time()
    myInv(i*1000)
    end = time.time()
    ram = process.memory_info().rss
    resTime[i-1] = end - start
    resSpace[i-1] = ram - baseRam

print(resTime)
print(resSpace)
```

```
[ 0.06470418  0.28054786  0.68449807  1.31404376  2.17583394  4.08201385
  5.82783508  8.15327191 11.1856277  14.79041791]
```

```
[5.94944000e+07 2.53435904e+08 6.87443968e+08 1.32944691e+09
 2.33149235e+09 3.19752192e+09 4.37561344e+09 6.42568192e+09
 4.86090342e+09 3.79873690e+09]
```

Key points

- All your code should be in fully documented *.py* files
- Your notebook should import and demonstrate the code
- For task two, you need to understand what *myInv()* does
- For task two, a scatter plot of *(time,size)* and/or *(space,size)* is often a good way to visualise the complexity
- For most tasks, there is no single correct answer - the idea is to show that you can write code that provides a better understanding of both data and the complexity of methods used in Data Science
- As for the first practical, the aim is to use the smallest number of external libraries in order to develop core Python skills
    - If you have problems generating inline plots in Jupyter without using *matpotlib*, then it is acceptable to also import this package.

Sumbission

Upload two things via Moodle:

1. A single Python file containing all your documented code

2. A Jupyter notebook that imports your code, demonstrates its use, and presents any results or insights obtained. You should also discuss any design decisions you made in the notebook

Assessment Criteria

Marking will follow the guidelines given in the school student handbook (see link in next section).

Policies and Guidelines

Marking

See the standard mark descriptors in the School Student Handbook: http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Good academic practice

The University policy on Good Academic Practice applies: https://www.st-andrews.ac.uk/students/rules/academicpractice/