Introduction to Python Programming Project Ideas

The grading for this course is based entirely on the final project. While everyone is studying biology in one form or another, the goal of this course is to learn how to program in Python. In this respect, you should not feel as though you have to do a bioinformatics-related project. In this document I have listed three project ideas, but only one of them involves bioinformatics. I actively encourage you to invent your own project (particularly relevant for PhD students) as it will make it easier to stay motivated.

The deadline is Friday 29nd May 2015

Project ideas

High throughput sequencing quality control

A single run from a high throughput sequencer will produce FASTQ files that are several gigabytes in size [1]. Many bioinformatics programs operate on the assumption that the input data is mostly error-free, which means low quality data must be filtered out first. A quality control program for high-throughput data should:

1. filter out sequences where the length is below a threshold
2. filter out sequences where the average quality is below a threshold
3. filter out or trim sequences with ambiguous base calls (e.g. 'N')
4. filter out or trim sequences with quality scores less than 3 (a quality score of 2 means that no quality score could be given)
5. trim sequences from the 3' end where the average quality of a sliding window is below a threshold
6. trim sequences from the 3' end in the style of BWA where we trim everything after position x where:

   $$argmax_x \sum_{i=x+1}^{l} (INT - q_i)$$

   if $q_l < INT$ where l is the original read length.
7. allow for paired-end data where the pairs are either interleaved in a single file or in two separate FASTQ files
8. output statistics of the dataset pre- and post-quality control, for example, average quality at each base position, overabundant subsequences, etc
9. trim adaptor sequences (this is actually very difficult)
10. all of the above, but using the getopt module to produce a nice command line interface

Text adventure game

With what you have learned on the course so far, a text-based computer game is well within your reach. This could take the form of a menu-based system similar to "choose your own adventure" books [2] or a more free-form adventure game like Zork [3].

Drawing/paint program

Python includes a library called Tkinter that can be used to build a graphical user interface and could be used to build a simple drawing program like MS Paint. This project is slightly more difficult than the others in that we did not cover how to use the Tkinter module in any of the lectures. You would need to utilise tutorials and documentation on the web to get started and fill in any gaps in your knowledge. As a starting point you should look at the Canvas object [4] and how to implement mouse events [5].

This project is intended to be more of a challenge, but it is not as difficult as it sounds; excluding empty lines I wrote a small program to draw rectangles using mouse events on a canvas in 20 lines of Python. Possible features to implement could include:

1. multiple colours
2. multiple tools (line, circle, rectangle, etc)
3. selecting colours and tools using widgets (buttons, menus, etc)
4. undo/redo

Your own idea

You are welcome to come up with your own project idea. If you are a PhD student, then by all means do something related to your own research.

Good project ideas are ones that involve multiple small parts and therefore get progressively harder as more features are added. If you choose to do a project of your own, then **you must come and see me during one of the practical sessions to agree to it**. While a project may be important or interesting to you, it does need to be at least difficult enough for me to assess your abilities and give you a grade.

Assessment

Assessment is based on the following criteria:

Challenge
: Was the project selected of sufficient challenge? All of the projects suggested here fulfill this criteria given that multiple features have been implemented.

Correctness
: Does the program do what it is intended to do and not crash? Instead of crashing, your program should print out an explanatory error message and state that it is exiting.

Readability
: Your code should be readable to another programmer. Your variable and function names should reflect what it is they contain or do. Your code should be commented appropriately.

Documentation
: Your project should come with usage information (I intend to attempt to run every program) and give a brief explanation of the design choices made. Please do not spend too much time on this part, the code is more important.

What/how to hand in

To hand in your work for marking, email me two files: a zip file of your code and a PDF containing the documentation. If you have some reason why any of this is not possible, please email me as soon as possible and we will work something out.

The zip file should include:

1. Your .py files
2. Any supplementary files needed to run your program
3. Example output from your program

The PDF documentation should include:

1. Your name and email address (just in case)
2. A short description of your project, what it does, why that is important (if it is not obvious)
3. A list of libraries necessary to run your program (e.g. BioPython)
4. Instructions to run your program
5. A short description of the design decisions you made in your program and why you wrote the program the way you did. This is quite important as it will allow us to understand your motivation behind the approach you took

References

[1] `https://en.wikipedia.org/wiki/FASTQ_format`

[2] `https://en.wikipedia.org/wiki/Fighting_Fantasy`

[3] `https://en.wikipedia.org/wiki/Zork`

[4] `http://effbot.org/tkinterbook/canvas.htm`

[5] `http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm`