DSA Exam Notes

The exam will be similar in spirit to the mock exam available on the Hub. It will have questions on both core Python concepts and the algorithm topics we have seen.

The exam will cover all the material we have gone through, with the exception of Python libraries. You will not be asked to import any libraries in the exam, and you conversely should not import any libraries to solve a problem unless otherwise instructed. So, for example, there won't be questions about `numpy` or `pandas`; while these are important tools, I don't want you to spend time memorising library commands for the exam.

The exam will be in **Jupyter Notebook** format. You will answer questions in the notebook file and submit it electronically following instructions detailed in the exam. You will not submit anything besides the notebook file.

Frequently asked questions

Q: May I use the module materials during the exam?

A: Yes.

Q: May I do online searches or communicate with classmates or friends during the exam?

A: No.

Q: Will the exam cover topic X?

A: If we covered the topic in one of the sessions, then yes - unless we covered it through a library (web scraping, for example).

Q: Will the exam cover optional exercise Y?

A: Rather than specific exercises, the exam will test your understanding of the *concepts* we've learned. Both mandatory and optional exercises are useful for practising these concepts. Again, the exception is libraries - if an optional exercise requires an imported library, it will not be examined.

Q: Will I need to memorise Python commands from the module?

A: No. You are expected to be comfortable with the core tools we have learned - for example, looping through built-in Python data structures and accessing/adding/removing data. If you would need to use a less common method, say `str.isupper()`, hints will be provided.

Q: Will we be asked to implement the merge sort, BFS, Dijkstra's algorithms from scratch in the exam?

A: It is unlikely that you would be asked to recreate the exact same algorithms we have implemented in class, but understanding how these algorithms work and how we moved from the algorithm descriptions to Python implementations can prove very useful. The final questions in the exam may be similar to those in the mock exam: you may be either asked to implement a more involved algorithm based on a description of how it works (Q8 in the mock exam or say, a graph algorithm) or to also come up with an algorithm to solve a problem (Q9-Q10).

Q: How do I open the mock exam file?

Before looking at the mock exam, you should first go through Session 7 to familiarise yourself with Jupyter Notebooks. You can open the Jupyter Notebook application from Anaconda Navigator on your system. The app will launch in your default browser.

Q: Will you post solutions to the mock exam?

A: Yes, check the exam folder on the Hub.

Q: The mock exam looks tough and/or long! Are we expected to solve everything in just two hours?

A: Yes and no. In the Imperial College grading scale, the average exam mark is expected to be **B**, or between 60-70%. This means that very few people will solve every question in the exam. The vast majority of exam scores are typically between 50% and 85% (grades A, B, or C). If the average exam mark falls outside the B range, the marks may be scaled.

Q: What is the passing grade for the exam?

A: You need at least 40% in the exam and 50% in the module overall to pass. The exam makes up 65% of your overall module mark.

Q: Will the exam be machine-graded? Can I get partial credit?

A: The exam will not be machine-graded and you may get partial credit for a good attempt at a problem. If your code does not quite work, it is worth writing a comment as specific as possible about how you would solve the problem.

Q: How can I practice working on Jupyter Notebooks?

A: Some of our exercises in the second half of the module will be in this format. The first such exercises are in Sessions 5-6. The mock exam is also a Jupyter Notebook.

Q: Where can I find more practice problems?

A: For core Python, I recommend Edabit, Snakify, CheckIO and Exercism.

For more advanced Python/algorithms problems, try Leetcode or HackerRank.