

MATH0011 project instructions and marking criteria 2024

Project groups

You must work in a team of 3 or 4 students, but it is up to you who to work with. If you're having trouble finding people to work with, email me for help: m.towers@ucl.ac.uk.

Project notebooks

Project notebooks can be found in the Projects folder of your CoCalc account. Your group should choose **one** of the project notebooks. Read them all carefully before deciding: they differ considerably in style and content.

If you need a fresh copy of the project file you can get a copy from one of your team members or the Moodle page for MATH0011.

Each project has its own Jupyter Notebook file containing several exercises and space for you to write code to solve these exercises. It's OK to create new code cells or text cells in the notebook.

You must put the student numbers of all team members in the space given at the top of the notebook. Don't put their names - marking is supposed to be anonymous. You do not need to indicate who contributed what to the project.

Imports

Any modules you import must be available on the CoCalc website by default.

Submitting your project

The deadline for submitting the project is **6pm Friday February 23rd 2024**. Only one file should be submitted per minigroup. Download the notebook file following the instructions at the end of the project notebook and submit it using the appropriate link on the MATH0011 Moodle page.

You must submit a Jupyter notebook file (the extension will be `.ipynb`) so that I can run your code. Don't save the project in any other format; don't convert it to a pdf file or to anything else.

Plagiarism and use of AI

If you got help from resources other than the course notebooks and lectures, you must clearly indicate this in the project notebook.

You can ask chatbots for help, but you must not get them to write code for you.

Anything you submit must be the work of your project team, except where you've indicated clearly that it isn't. Please see the [UCL Plagiarism page](https://www.ucl.ac.uk/students/exams-and-assessments/plagiarism) (<https://www.ucl.ac.uk/students/exams-and-assessments/plagiarism>) for details, and contact me before the deadline if you have any questions about what is acceptable.

Advice

- **Read the marking criteria below.**
- **Back up your work** especially if you are working on the project notebook outside CoCalc.
- Most exercises can be solved in around 10 lines of Python. It is OK to write more than this if you need to, but you may be missing an easier way to solve the problem.
- None of the projects require writing code that runs for more than a few seconds on a typical laptop computer. If your code takes longer then you are doing something wrong. Try to find the problem and improve it, and ask me for advice if you are stuck.

Marking criteria

The project is worth 15% of your overall MATH0011 mark. It is marked out of 10.

- 5 marks for the mathematical content,
- 5 for the quality of the code you write.

You get one mark out of 10 for the whole project. Below there is a description of what your project needs to do to get full marks.

Mathematical content: 5 marks

- Output produced is mathematically correct and answers all parts of all exercises fully.
- Mathematical formulas and requirements are correctly interpreted and translated into Python code.

Code quality: 5 marks

- Code runs without errors.
- Code is efficient (the amount of time it takes is reasonable for what it is trying to compute).
- Code is concise and not unnecessarily complicated.
- Any diagrams or plots are clear and easy to understand.
- Variables and functions have descriptive names where appropriate, and explanatory comments are included where necessary.

Here are some examples of how to comment your code

```
def mean(l):  
    """  
    Input: a list of numbers  
    Output: the arithmetic mean of those numbers  
    """  
    return sum(l) / len(l)
```

The docstring (the part in triple quotes) adequately explains what the function does, so there is no need to add more explanation. The next function contains comments (using the # symbol) that help to make it easier for humans to understand what it does:

```
def fibonacci(n):  
    """  
    Input: a non-negative integer n  
    Output: the nth Fibonacci number  
    """  
    if n < 2:  
        return n # The 0th Fibonacci number is 0 and the 1st  
                 # Fibonacci number is 1, so fibonacci(n) = n  
                 # for n < 2  
    else:  
        return fibonacci(n-1) + fibonacci(n-2) # in the else  
        clause  
        # we can assume n >= 2, so that  
        # fibonacci(n) = fibonacci(n-1) + fibonacci(n-  
        2)
```