UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Faculty of Engineering, Built Environment and Information Technology

Fakulteit Ingenieurswese, Bou-omgewing en Inligtingtegnologie

Programming and Information Technology - MPR213

Department of Mechanical and Aeronautical Engineering

Lecturers: Mr II Setshedi, Ms D Ramatlo, Dr AJ Oberholster and Dr M Mahdavi

Last Revision: 16 February 2022 (Revision 1)

Contents

1. Educational Approach

Programming knowledge and skills are essential for engineers. Complex and repetitive calculations and analyses that would take days if done by hand can take only seconds with a well-structured computer program. The program itself may take significant time to code, but once coded it allows the engineer to ask different questions and conduct various investigations that would otherwise not be feasible or possible. Programming combined with mathematics has become a prerequisite to basic engineering literacy and engineering problem-solving. In addition, the life cycle of a computer program is similar to the design of a product or the design of a process.

For these reasons, engineering students must obtain at least some proficiency in computer programming. The general objectives of this module are:

- To enable the students to solve engineering problems by developing, debugging, and running basic computer programs.
- To enable the students to process computer data and output (text and visual) information or knowledge.

Elementary mathematical and physics concepts, which the students should already be familiar with, will be used to illustrate basic computer logic and programming principles. The student will be required to "translate" mathematical formulas and/or principles into working computer programs to solve some problems.

The **educational approach** is **problem-driven**; in other words, programming skills are acquired by solving problems. The mastering of tutorial and homework problems is essential to succeed in this module. Like learning a natural language that requires dedicated time conversing, programming requires dedicated clock time in front of a computer to solve engineering problems. *Ensure that you sit in front of a computer solving engineering problems using Python or Excel daily.*

Student-orientated and cooperative study methods will be applied during the lectures and tutorial sessions to establish the course's core concepts. Lecturers will intensely focus on

solving problems during classes and tutorials to develop the programming concepts required to solve mathematical and physics problems.

Lecture classes will be held in two sessions: Students will be allowed to watch pre-recorded lecture videos prescribed for the respective lecture session. During the second session, students are allowed to interact with lecturers on the content covered in the pre-recorded lecture videos. Students are expected to participate in discussions during the second session as this creates an opportunity to share experiences and solve problems in a team-oriented environment. This will mimic what generally occurs in industries. Interaction during lectures will allow students to assimilate and understand complex concepts. Students are encouraged to experiment with their computers while watching recorded lecture videos and then interact with lecturers and the rest of the class, on-campus or online.

2. Departmental Study Guide

The MPR 213 Study Guide is a crucial part of the general study guide of the Department of Mechanical and Aeronautical Engineering, which is the host department for this module. In the study guide of the Department, information is given on the mission and vision of the department, general administration and regulations (professionalism and integrity, course-related information and formal communication, workshop use and safety, grievances, support services, plagiarism, class representative duties, sick test and sick exam guidelines, vacation work, appeal process and adjustment of marks, university regulations, frequently asked questions), ECSA Graduate Attributes, ECSA knowledge areas, CDIO, curriculum and assessment of cognitive levels. It is expected that you are very familiar with the content of the Departmental Study Guide. It is available on the Department's website.

https://www.up.ac.za/media/shared/120/Noticeboard/2021/departmental-studyguide-eng-2021_version26feb2021.zp199803.pdf

Department Website:
http://www.up.ac.za/en/mechanical-and-aeronautical-engineering/article/21692/noticeboard

Take note of the specific instructions in the above study guide on:

- Safety
- Plagiarism
- What to do if you were sick (very important)?
- Appeal process on the adjustment of marks

3. Lecturers and Consulting Hours

3.1. Lecturers

The lecturer contact details are given in table 1.

Table 1: Lecturer contact details			
Module Manager	**Offices**	**Telephone No.**	**Email**
Mr Isaac Setshedi	Eng I 10-19	012 420 2014	isaac.setshedi@up.ac.za
Lecturers			
Mr Isaac Setshedi	Eng I 10-19	012 420 2014	isaac.setshedi@up.ac.za
Ms Dineo Ramatlo	Eng I 10-16	012 420 2462	dineo.ramatlo@up.ac.za
Dr Abrie Oberholster	Eng I 10-18	012 420 3288	abrie.oberholster@up.ac.za
Dr Mostafa Mahdavi	Eng I 6-99	012 420 2935	mostafa.mahdavi@up.ac.za

Please use the keyword MPR213 in the email subject specification for email correspondence. Not doing this may leave the email unanswered or cause a significant delay in the answer. All administrative queries should be sent to the module manager.

Depending on the lockdown level and the arrangements that the University has made regarding the COVID-19 pandemic, the lecturers may not be available in their offices or at the telephone numbers given in the table above. In that case, please use an email message to contact a lecturer.

3.2. Teaching Assistants

The contact details and consulting hours for the teaching assistants will be added to MPR 213 S1 2022 clickUP as soon as the teaching assistants have been allocated.

3.3. Consulting Hours

Consultation hours of the lecturers and teaching assistants will be announced at the start of the semester. These hours will be shown on MPR 213 S1 2022 clickUP. These hours indicate when the lecturers and teaching assistants are available for consultation.

There will be no extraordinary consulting hours during test and exam time. Students may make special arrangements to talk to the lecturers outside of their consulting hours if necessary. It is important not to wait until before a test or exam to clarify any problems.

Students are also welcome to use the discussion board on MPR 213 S1 2022 clickUP. This will allow lecturers, teaching assistants, and fellow students (during their free time) to answer any queries regarding programming concepts, exercise problems, etc.

4. Study Material and Software

4.1. Lecture Notes

We distinguish between the lecture notes discussed here and the study notes discussed below. The lecture notes form the primary set of study material made available in the module presentation and consist of two parts:

- Groupings around various topics: Python computer code with explanations, most of which are demonstrated during contact sessions or on video recordings.
- Several Excel spreadsheets and related files are explained and demonstrated on video recordings.

Both these parts are available on MPR 213 S1 2022 clickUP, in the "Study Material" content area, in the "Lecture Notes" content folder.

Students will be notified with a clickUP announcement whenever any of the Lecture Notes files on clickUP is updated.

4.2.Study Notes

There is no prescribed textbook for this module. However, a set of study notes has been developed at UP and is available on MPR 213 S1 2022 clickUP, in the "Study Material" content area, in the "Additional Material" content folder. These study notes is a pdf document:

"Introduction to programming for engineers using Python" by Logan G. Page, Daniel N. Wilke, and Schalk Kok.

Please note that this publication is not updated and some of the sections related to Python 2.7, IPython (Qt) and Spyder editor are no longer used in this module.

Students should primarily work through the lecture notes mentioned in section 4.1 above but may in parallel use the study notes as an additional source.

4.3. Complementary Sources

A link to the following complementary source is available on MPR 213 S1 2022 clickUP, in the "Study Material" content area, in the "Additional Material" content folder:

"Python for Computational Science and Engineering" by Hans Fangohr

Numerous other complementary sources for this module are mentioned in the study notes. The discussion forum and wiki page on MPR 213 S1 2022 clickUP may also be utilised for additional explanations and examples on various topics covered in this module.

4.4. Software used in this module

As implied in section 4.1 above, in this module, we will make use of

- Python, which is open-source software. This means that students can freely copy and use this software legally without any restrictions.
- Microsoft Excel, which is part of the Office package, and the University has a campus-wide licence. This licence allows registered students to download the package to their computers without cost. Excel is used for spreadsheet programming.

Python is a high-level programming language used in many disciplines around the world. It is a well-suited programming language for engineers as it is easy to learn, well supported and documented, relatively fast, and very good for numerical and scientific computing. In this module, we specifically use the Jupyter Notebook implementation of Python by Anaconda. Students must be familiar with using this specific implementation of Python, as its autograding facility (where a computer grades students' work) is used exclusively in the weekly tutorial assignments, semester tests and exams.

You can download Anaconda, including Jupyter Notebook and Python, depending on your operating system, from the links shown in table 2 below or use it on on-campus computer labs. Download links are available on MPR 213 S1 2022 clickUP. Please carefully follow the installation instructions given in writing and video recordings in the "Software" content area of MPR 213 S1 2022 clickUP.

Table 2: Anaconda 3 download links

Operating system	Link	Size	Release date	Notes
Linux	Anaconda3-2020.07-Linux-ppc64le.sh	290.4 M	2020-07-23 12:16:47	64-Bit (Power8 and Power9) Installer
	Anaconda3-2020.07-Linux-x86_64.sh	550.1 M	2020-07-23 12:16:50	64-Bit (x86) Installer
Mac	Anaconda3-2020.07-MacOSX-x86_64.pkg	462.3 M	2020-07-23 12:16:42	64-Bit Graphical Installer
	Anaconda3-2020.07-MacOSX-x86_64.sh	454.1 M	2020-07-23 12:16:44	64-Bit Command Line Installer
Windows	Anaconda3-2020.07-Windows-x86.exe	397.3 M	2020-07-23 12:16:51	32-Bit Graphical Installer
	Anaconda3-2020.07-Windows-x86_64.exe	467.5 M	2020-07-23 12:16:46	64-Bit Graphical Installer

The version available at the links above is not the newest, but it is the official version used in this module in 2022. Do not use a more recent version in answering weekly tutorial assignments. Using the more recent version could result in nbgrader crashing and losing marks.

Download instructions for Microsoft Office 365 (including Excel) are available at the Mining Industry Study Centre. Students can use the following link for assistance:

studycentre.helpdesk@up.ac.za

5. Learning Activities

Disclaimer: Sections 5 and 6 of this study guide have been written under the assumption that on-campus learning will partly take place and **assessment activities such as tests and exams will take place on campus** unless the country's alert levels are adjusted. Some activities are on campus and others online. If this changes in the semester, this study guide will be revised.

5.1. Contact Time and Learning Hours

Number of lectures a week:	4 lectures (two double period sessions), 50 minutes per lecture
Tutorial sessions a week:	1 session, 100 minutes per session

This module carries a weight of 16 credits, indicating that a student should spend an average of 160 hours mastering the required skills (including time spent preparing for and writing tests and examinations). This means that you should devote an average of 11 hours of study time per week to this module. The scheduled lecture and tutorial time is approximately 6 hours per week, which means that the other 5 hours per week of study time should be devoted to the module.

5.2. Lectures

Classes are split according to study discipline into three groups for the lectures. Ms Dineo Ramatlo will be responsible for group 1 (Tuesdays 15:30 and Thursdays 11:30), Dr Abrie Oberholster group 2 (Tuesdays 15:30 and Thursdays 12:30) and Mr Setshedi group 3 (Mondays 13:30 and Fridays 10:30).

Annotated PowerPoint (video) recordings of formal lectures covering the whole module (recorded in 2020) are available on YouTube or Google Drive, with links on MPR 213 S1 2022

clickUP. Because of this, students will be expected to **watch the pre-recorded lecture videos** and **attend question and answer sessions** that will be **held 30 minutes before the end of each lecture**. The Q&A sessions may be used for learning activities other than formal lectures, like discussions, answering students' questions, problem-solving, etc. Students are expected to view the recorded material prior to or during the corresponding lecture sessions (see the lecture plan in section 8 of this study guide and daily lecture timetable MPR 213 S1 2022 clickUP) and to then participate in discussions and pose questions to be answered during the session. Furthermore, students can expect test and exam questions on material covered in the video recordings but not specifically covered during the 2022 Q&A sessions.

5.3. Tutorials

Tutorial sessions will further develop the students' understanding and knowledge of the topics covered in lectures by solving numerous engineering and mathematical problems with the tools introduced in this module.

Each week's tutorial session will be used to review the previous week's lecture content and concepts. Selected exercise problems, from the study notes, will be covered during the tutorial sessions. The selected chapters and problems that will be covered in each tutorial session will be announced on MPR 213 S1 2022 clickUP.

Students must do these tutorial problems independently before attending the tutorial session and use the tutorial session to get help with issues or concepts that the student is finding difficult. It is important to understand that the teaching assistants participate in assisting the student in grasping difficult concepts and not solving the problems on their behalf. **Students have the option of attending tutorials on-campus or online**.

It will be to the benefit of the student to complete these problems, as solving these problems will develop the student's thinking and programming ability. The tests and exam will test the ability of the student to solve new problems of similar complexity.

Each week, to conclude the tutorial, a tutorial assignment needs to be submitted by uploading to MPR 213 S1 2022 clickUP, usually before midnight on a Friday evening. **These assignments will be graded and count towards the semester mark.** Because these are tutorial assignments, there is absolutely no way that an assignment not submitted before the due date, for whatever reason: including network problems, apparent overload of clickUP when submitting, and not having been able to register for the module in time. In the case of a valid reason for not submitting an assignment, like sickness or death in the family, an excuse can be registered with the module manager if proper substantiating documentation is supplied. If, in the case of a single student, too many of these assignments are not submitted for valid reasons, the percentage contribution of the assignments to the semester mark may be reduced, and the contribution of the two semester tests may be increased correspondingly.

6. Rules of Assessment

Refer to the exam regulations in the Faculty regulations and information, Faculty of Engineering, Built Environment and Information Technology.

To be admitted to the exam, a student needs to have a semester mark of at least 40 %.

To pass the module, a student must:

- Obtain a final mark of at least 50%; and
- Obtain a subminimum of 40% for the final examination

6.1.Determination of Final Mark

The final mark is compiled as follows:

- Semester mark: 50%
- Final exam mark: 50% (3-hour exam)

6.2. Determination of Semester Mark

The semester mark will be determined as shown in table 3.

Table 3: Determination of Semester Mark

Evaluation Method	No. of	Contribution of each	Total
Semester tests	2	40%	80%
Tutorial assignments	All		20%
Total			100%

6.3. Semester Tests

Two semester tests will be written during the official School of Engineering semester test weeks. The duration of each test will be 90 minutes. The scope of the tests will be announced on MPR 213 S1 2022 clickUP during the lecture week preceding the test week.

Additional test instructions will be announced on MPR 213 S1 2022 clickUP during the lecture week preceding the test week.

After the tests, model answers will be made available in electronic format on MPR 213 S1 2022 clickUP after the tests and will not be formally discussed during lectures. Students may, however, ask questions on the model answers in class.

6.4. Autograding of weekly tutorial assignments, semester tests and the exam

In the weekly tutorial assignments, semester tests and the exam a student needs to answer by filling in code or information in a number of Jupyter Notebooks and/or Excel spreadsheets. The resulting Jupyter Notebook and Excel files are then submitted electronically to clickUP as the student's answer. These files are marked (or graded) by a computer in a process called autograding.

The grading is set up in such a way that various parts of the answer are assigned certain marks. If a part is correct, that mark is awarded by the autograder. If there is one error in the part in question, a zero mark is awarded (i.e., a binary marking is used). In setting up an assignment, test or exam, the lecturers carefully decide in advance how many marks are allocated to a certain part. In doing this, the lecturers try to be reasonable. Usually, and especially in the tests and exams, the maximum mark awarded to a certain part is relatively small compared to full marks for the assignment or paper. This means that there is no possibility for being awarded partial marks for something the student deems to be correct, even though the whole part is not correct unless the question was set up beforehand to provide for partial marking. The marking system is fair towards all students: everybody's

work is graded exactly in the same way (totally impartially, by a computer) and according to the same rules. Even though the lecturers are most willing to discuss the technical aspects of questions with students, they will not entertain questions on partial marking of graded binary units or the fairness thereof.

Suppose a student uses Python or Excel functionality that is not formally taught as part of the module, and the approach and coding produced by the student are deemed correct but causes the autograding process to penalise the student. In that case, they will not be compensated for this. Students are therefore advised to only use functionality taught in the module. In this regard, students should especially be careful in the case of semester tests or the exam, if this is answered on an off-campus computer like in times of lockdown, as different versions of software may produce different results. It is the student's responsibility to ensure that the correct version is used.

6.5. Appeals and queries on marks

Note the description of the rules that apply to the appeals process in the Departmental Study guide (see section 2). Due to the use of the autograding process, appeals will be limited to administrative errors with the processing of marks.

6.6. Penalty for errors in stating identifying information during assessments

During all tutorial assignments, semester tests and the exam the student is instructed to insert his/her student number in a very specific way, as a string variable, in a Python code cell. In on-campus semester tests and exams, the student must also supply the identifying number of the computer at which they write the test/exam in the same cell. This is very important because it is the only way in which the student can be identified during grading. Therefore, the following rules apply if the student number is reported incorrectly and not according to the instructions in any way:

- **Tutorial assignments:** a zero mark will be awarded, as the additional administrative load in trying to sort out these errors is simply too large.
- **Semester tests and exams:** the lecturers will try to identify the student in whatever alternative ways are available, but for this additional effort, the student will be penalised by up to 10 % of the mark that they would otherwise have been awarded for the paper. This penalty will be subtracted from the student's mark, i.e., a form of negative marking will be employed.

6.7. Penalty for errors causing the autograder to crash or where submitted files were named incorrectly

Sometimes students make errors, usually by not following instructions, that cause the autograder to crash. When this happens, the whole of the student's assignment or test is not graded. Also, if a student does not follow the rules for naming the files they submit, considerable additional work needs to be done to fix these errors before autograding can commence. The following rules apply if a student makes these errors:

- **Tutorial assignments:** a zero mark will be awarded, as the additional administrative load in, first of all, identifying the reason for the error and then trying to sort it out, or to rename files, is simply too large.
- **Semester tests and exams:** the lecturers will try to identify the reason for the crash. If this can be found in a reasonable time, the lecturers will try to fix the error. If the error was caused by the student not following instructions or if the files were not named correctly, he/she will be penalised by up to 10 % of the mark that he/she would otherwise have been awarded for the paper. This penalty will be subtracted from the student's mark, i.e., a form of negative marking will be employed.

It is therefore important to meticulously follow all instructions on autograded assignments, tests and exams.

7. Study Components

7.1. Purpose of the module

The purpose of the module is to introduce students to the following:

Advanced spreadsheet applications:
Named ranges, referencing, linear programming, solving non-linear equations, fitting regression lines, interpolation, manipulating large data sets, extracting information from data sets.

Basic structured programming:
Data container types, looping, branching, subroutines, iteration, reading and writing data files. Development, coding, and debugging of simple programs in a high-level programming language. Programming principles are illustrated via mathematical and physics concepts such as limits, differentiation, integration, linear algebra and simple motion. Structured programming by coding and using functions written by the student as well as using functions available in packages. Basic graphical output (plotting) is also covered.

7.2. Module Structure

The structure of this module is shown in table 4. The total module hours (notional hours) include the contact time, as well as the estimated time to be allocated for self-study, preparation and writing of assignments, tests and the examination. The mode of instruction is via lectures, tutorial classes and assignments.

Contact sessions indicate the regular lectures. The number of contact sessions per chapter is tentative and may change depending on the progress during lectures.

Table 4: Module Structure			
Theme No.	Topic	Notional Hours	Contact Sessions
1	Introduction	1	1
2	Basic Programming ● Using Python as a Calculator ● Names, Objects and Assignment ● Import and Use Modules ● Python Namespace	13	4
3	Basic Functions ● User-defined Functions ● Python Namespace and Scoping ● Basic **numpy** Arrays	13	4
4	Control Statements ● Iterator-Based Looping (For Loop) ● Conditional Looping (While Loop) ● Branching (**if**, **elif**, **else**)	30	10
5	Solving Problems ● Breaking Down Problem Complexity ● Use of Functions (Simplification) ● Nested Statement	30	10
6	Plotting and Graphs ● 2D Graphs ● Graph Annotations ● 3D Graphs	14	4
7	Reading and Writing Data ● Data Sharing between Applications	7	3
8	High Level Programming ● Additional Modules ● Advanced Built-In Functions	12	4
9	Spreadsheets ● Formulas and Calculations ● Spreadsheet Detective ● Plotting and Graphs ● Linear Programming ● Data Lookups and Pivots	40	8
Total		160	48

7.3. Fundamental Concepts

The following concepts need to be mastered in order to pass the course. If any of the following concepts are not mastered, the student will fail the course:

- Basic objects (e.g. **int** vs **float**)
- Importing modules and proper usage
- Functions (ability to properly create a new function, understand how objects are passed to and from functions, be able to properly use any created function)
- For loop (proper usage and understanding when and why to use a for loop)
- Iterating through lists and growing lists (single lists, not nested lists)
- While loop (proper usage, understanding termination conditions and understanding when and why to use a while loop)
- If statements (proper usage, understanding program flow and control and understanding when and why to use an **if**, **elif** or **else** statement)
- Reading and writing of CSV files
- Plotting (be able to create a simple 2D plot with proper annotations)
- Data processing using spreadsheets

7.4. Study Theme Descriptions

7.4.1. Theme 1: Introduction

The introduction allows the student to obtain a general overview on the roles and responsibilities of both the student and the lecturer as well as an overview on what computer programming is. This section will not be explicitly tested. It is important to note that an understanding of computer architecture and flow diagrams will make the following sections more accessible.

7.4.2. Theme 2: Basic Programming

Using Python as a Calculator: The student has to ensure that he/she is comfortable with the Jupyter environment and must be able to use Python to do simple mathematical calculations. It is the student's responsibility to spend enough time using these environments throughout the semester. The student must be able to use Python's built-in math module **numpy** and access the functions stored in it. The student must be able to use Python's built-in help to obtain information on the various functions.

Names, Objects and Assignment: The student has to be familiar with the guidelines in the class notes on names and objects e.g. objects are created in memory and a name is assigned to that object. The student must also be familiar with the memory model of Python. The student must also know the difference between the object types e.g. **int**, **float**, **str**, **bool** as well as to know how to check the type of an object using **type()**.

7.4.3. Theme 3: Basic Functions

The student must be familiar with functions and code structure. The student must be able to properly create and use functions with multiple inputs and outputs. The student must be able to rewrite an existing computer program such that it makes use of functions. The student has to be familiar with the local namespace and understand “name scope" within a function. The student must understand any object can be passed into a function or returned from a function. This includes another function and is particularly important for some of the built-in functions.

7.4.4. Theme 4: Control Statements

For Loop: The student must understand that the *for loop* iterates through iterators (e.g. the elements of a list). The student has to be able to identify when and why to use *for loop(s)* from a given problem statement. The student has to be familiar with Python's syntax and indentation and be comfortable to implement *for loop(s)* in a program to perform a given task.

While Loop: The student must understand conditional statements and how to combine them using **and** and **or** operators. The *while* continues to iterate as long as the condition evaluated is **true** or **1**. The student must understand that should the condition not update during a while loop iteration, the program is stuck in an infinite loop. The student has to be able to identify when and why to use *while loop(s)* from a given problem statement. The student has to be familiar with Python's syntax and indentation and be able to implement *while loop(s)* in a program to perform a given task. The student also has to be able to determine the correct *while loop* termination conditions in order to perform a given task.

Basic List Generation: The student must be able to generate simple list objects, both manually and by using the range function. The student also has to be familiar with the various list operators (***+, ****), functions (**sorted**) and in-place functions (methods) (**append***,* **insert**, **index**, **sort**, **reverse**). The student must be able to use, read and understand Python's built-in help to obtain information on the various functions and in-place functions.

Branching: The student must understand conditional statements and how to combine them using **and** and **or** operators. The student has to be able to identify the usage of branches (when and why to use an **if**, **elif** or **else** statement) from a problem statement. The student has to be familiar with Python's syntax and indentation and be able to implement branches in a program to perform a given task. The student needs to understand how each of the **if**, **elif** and **else** statements affect the program flow and control. The student also has to be able to determine the correct conditions in order to perform a given task.

7.4.5. Theme 5: Solving Problems

The student must be able to evaluate and break down a complex problem into logical control statements (**for**, **while**, **if**). The student must be able to identify which nested statements are needed from the problem statement and breakdown and then be able to implement the nested statements in a program to solve the given problem. The student has to be familiar with Python's syntax and indentation for nested statements. The student must also be familiar with nested lists (list of lists) and the memory model thereof. The student

must be able to identify when nested lists are needed from the problem statement and breakdown and be able to implement nested lists.

7.4.6. Theme 6: Plotting and Graphs

The student must be able to create 2D plots i.e. the student must be familiar with different presentations of data, presenting multiple graphs on the same figure, displaying grids, labelling of the axis, naming of figures, using legends, and scaling axis systems. The student must take note of arrays which are an object type of the **numpy** module and understand their associated operators (**+**, -, *****, ******). The student must be familiar with creating 3D figures. The student has to be familiar with creating surface and mesh plots.

7.4.7. Theme 7: Reading and Writing Data

The student must be familiar with reading and writing to and from CSV files. The student must be able to store the data in a given format. The student must also understand how to access data, stored by Python, in Microsoft Excel and vice versa.

7.4.8. Theme 8: High-Level Programming

The student must be able to navigate Python's documentation as well as online documentation and be able to find, read and understand the documentation of the required functions in order to solve specific problems. The student should therefore be able to independently increase his/her knowledge of the Python programming language. The student must then be able to use these additional modules and functions in Python in order to solve a specific problem.

7.4.9. Theme 9: Spreadsheets

Formulas and Calculations: The student must be able to solve problems using formulas. The student must be able to solve problems using functions that he/she is familiar with, as well

as use the function wizard for unknown functions. The student must be able to name ranges of cells and use them in calculations.

The student must be able to use filters to filter data and perform calculations on the filtered dataset. The student must be able to perform calculations on data that is split over multiple spreadsheets.

Spreadsheet Detective: The student must be able to use the spreadsheet detective to familiarise themselves with the dependencies in an unknown spreadsheet. The student must be able to switch between formula and value view.

Plotting and Graphs: The student must be able to create and customise charts. The student must be able to find trends in data.

Non-Linear Solver: The student must be able to solve non-linear one-dimensional problems using goal seek in Microsoft Excel.

Linear Programming: The student must be able to solve linear programming problems using Microsoft Excel. The student must be able to solve problems with mixed-integer, real and, Boolean variables. The student must be able to accommodate multiple constraints.

Data Lookups and Pivots: The student must be able to extract relational data using the Pivot Tables in Microsoft Excel.

Visual Formatting: The student must be able to improve the appearance of a worksheet by using colours, highlighting, borders, and font properties. The student must be able to perform conditional formatting of cells.

8. Lecture plan

The UP class timetable divides the class into three groups according to the engineering discipline. Two lecture plans are used for these three groups, as follows:

- Group 1, lecture plan in Table 5: Industrial and Chemical 2nd years; Industrial, Chemical and Mechanical Engage (3rd years); lectures Tuesdays 15:30 to 17:20 and Thursdays 11:30 to 13:20.
- Group 2, lecture plan in Table 5: Metallurgical, Mining and Civil 2nd years; Metallurgical and Mining Engage (3rd years); lectures Tuesdays 15:30 to 17:20 and Thursdays 12:30 to 14:20.
- Group 3, lecture plan in Table 6: Mechanical 2nd years; Civil Engage (3rd years); lectures Mondays 13:30 to 15:20 and Fridays 10:30 to 12:20.

Note that Excel Part I covers the worksheets in the Excel lecture notes "Excel Lectures_2022.xlsx" (from left to right) "Excel Overview" to "Extracting data". Excel Part II covers "Pi approximation", "Plotting Pi approximation", "Motion Example" and "Pivot tables".

Table 5: Presentation roster for Groups 1 and 2					
Week	Number of lectures	Dates	Study Theme (see Table 4)	Lecture Notes (Python: Section_)	Remarks
1	2	22/02	1,2	Section_01 Section_02	On-campus attendance
1	2	24/02	2	Section_02	
2	2	01/03	3	Section_03	
2	2	03/03	3	Section_04	
3	2	08/03	4	Section_05	
3	2	10/03	4	Section_06	
4	2	15/03	4	Section_06	
4	2	17/03	4	Section_07	
5	2	22/03	4	Section_07	
5	2	24/03	5	Section_08	
1st Test Week					
6	4	05 & 07/04	5	Section_08	
7	2	21/04	5	Section_08	
8	2	26/04	5	Section_08	28/04 Wednesday timetable
9	4	03 & 05/05	9	Excel Part I	
10	4	10 & 12/05	6	Section_09	
11	4	17 & 19/05	9	Excel Part II	
2nd Test Week					
12	2	31/05	7	Section 10	
12	2	02/06	8	Section_11	
13	2	07/06	8	Section_12	
13	2	09/06	8	Section_12	
Exam					

Table 6: Presentation roster for Group 3					
Week	Number of lectures	Dates	Study Theme (see Table 4)	Lecture Notes (Python: Section_)	Remarks
1	1	21/02	1	Section_01 Section_02	On-campus attendance
1	2	25/02	2	Section_02	
2	2	28/02	3	Section_03	
2	2	04/03	3	Section_04	
3	2	07/03	4	Section_05	
3	2	11/03	4	Section_06	
4	2	14/03	4	Section_06	
4	2	18/03	4	Section_07	
5	2	25/03	4	Section_07	
1st Test Week					
6	4	04 & 08/04	4	Section_08	
7	2	11/ 04	5	Section_08	
8	2	22/04	5	Section_08	
9	2	25/04	5	Section_08	
09	2	29/04	9	Excel Part I	
10	2	06/05	9	Excel Part I	06/05 Monday timetable
11	4	09 & 13/05	6	Section_09	
12	4	16 & 20/05	9	Excel Part II	
2nd Test Week					
13	2	30/05	7	Section 10	
13	2	03/06	8	Section_11	
14	2	06/06	8	Section_12	
14	2	10/06	8	Section_12	
Exam					