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: Dr M Moghimi Ardekani and Professor N Theron

Last Revision: 16 January 2019

Contents

1. Educational Approach

Programming knowledge and skills are important for engineers. Complex and repetitive calculations and analyses that would take days if done by hand can take only seconds with a well-structured program. Although the program itself takes time to code, but once coded 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.

It is for these reasons that engineers obtain at least some proficiency in 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 therefore essential to be successful in this module. Similar to learning a natural language that requires dedicated time conversing, programming requires dedicated clock time in front of a computer solving engineering problems. *Ensure that you sit in front of a computer solving engineering problems using Python on a daily basis.*

Student-orientated and cooperative study methods will be applied during the lectures and tutorial sessions in order to establish the core concepts of the course. Lectures will strongly focus on solving problems in class and developing the programming concepts as required to solve mathematical and physics problems.

Students are expected to participate in discussions in class since this creates an opportunity to share experiences and solve problems in a team orientated environment. This will mimic what generally occurs in industry. Problem solving sessions during lectures will provide students with opportunities to assimilate and understand difficult concepts. Students are encouraged to bring their laptops to the lectures to code the covered material in class.

2. Departmental Study Guide

This study guide is a crucial part of the general study guide of the Department. 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, plagiarism, class representative duties, sick test and sick exam guidelines, vacation work, appeal process and adjustment of marks, university regulations, frequently asked questions), ECSA outcomes and ECSA exit level outcomes, ECSA knowledge areas, CDIO, new 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 in English and Afrikaans on the Department's website.

English:
https://www.up.ac.za/media/shared/120/ZP_Resources/Noticeboard/departmental-studyguide-eng-2019_version29-jan2019.zp167517.pdf

Afrikaans:
https://www.up.ac.za/media/shared/120/ZP_Resources/Noticeboard/departementele-studiegids-afr-2019_weergawe29-jan2019.zp167518.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

Lecturers	Offices	Telephone No.	Email
Dr Mohamamd Moghimi Ardekani	Eng III 6-81	012 4205367	mohammad.moghimiardekani@up.ac.za
Professor Nico Theron	Eng I 10-5	012 4203309	nico.theron@up.ac.za

For e-mail correspondence, please use the keyword MPR213 in the e-mail subject specification.

3.2. Teaching Assistants

The contact details and consulting hours for the teaching assistants will be added to 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 also be displayed on their office doors and on ClickUP. These hours indicate when the lecturers and teaching assistants are on campus and available for consultation. Students are requested to please make an appointment so as to ensure that the lecturers and teaching assistants are in their offices.

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

Students are also welcome to make use of the discussion board on the new ClickUP system. This will allow lecturers, teaching assistants, and fellow students to answer any queries regarding programming concepts, exercise problems, etc. during their free time.

4. Study Material and Software

4.1. Prescribed Textbook

There is no prescribed textbook for this course, however the following study notes have been developed at UP and will be made available on ClickUP:

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

Please bear in mind that this study note is not updated and some of the sections such as the sections related to Phyton 2.7, IPhyton (Qt), Spyder editor and etc. are no longer practiced in this module.

4.2. Complementary Sources

The following complementary source of notes will also be made available on ClickUP:

"Python for Computational Science and Engineering" by Hans Fangohr

Numerous other complementary sources for this course are mentioned in the study notes above. The discussion forum and wiki page on the ClickUP system will also be utilised for additional explanations and examples on various topics covered in this course.

4.3. Lecture Notes

Take note, we make the distinction here between study notes discussed above and lecture notes discussed here. These lecture notes will also form part of the syllabus and are available on GitHub:

https://github.com/mpr213/lecture-notes/releases

These lecture notes do not cover all the work discussed in class and students should work through the study notes and take down their own supplementary notes during lectures. Problem solutions covered in detail during the lectures will not be made available again at a later stage.

4.4. Required Software

In this course, we make use of open source software. This implies that students can freely copy and use the software legally without any restrictions.

The programming package used in this course is Python. Python is high level programming language used in many disciplines around the world. Python is a well suited programming language for engineers as it is easy to learn, well supported and documented, relatively fast, and well suited for numerical and scientific computing.

Microsoft Excel will be used for spreadsheets in this course. Microsoft Excel is the standard practice of this module. However, for those students that for any reason might not be able to install Microsoft Office, the alternative tool would be LibreOffice. LibreOffice is in many ways very similar to Microsoft Excel with the exception that can be downloaded and used without any restrictions. However, please bear in mind that Microsoft office is the only spreadsheet tool which is examined throughout the semester.

You can download Python via Anaconda from:

- https://www.anaconda.com/distribution

Note: Please download and install the Python 3.x version of Anaconda.

You can download Microsoft Office 365 and for installation please contact IT support. University of Pretoria has purchased the licence for Microsoft Office 365.

You can download LibreOffice from:

- http://www.libreoffice.org/download/libreoffice-fresh/

(See http://www.libreoffice.org/get-help/install-howto/ for installation instructions)

5. Learning Activities

5.1. Contact Time and Learning Hours

Number of lectures a week:	4 lectures, 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 to master the required skills (including time spent preparing for tests and examinations). This means that you should devote an average of 11 hours of study time per week to this module. The scheduled contact time is approximately 6 hours per week, which means that another 5 hours per week of own study time should be devoted to the module.

5.2. Lectures

Dr Moghimi Ardekani will present the English lectures and Professor Theron will present the Afrikaans lectures as shown in Table 1. The lectures are split as per the timetable listed in Table 1. Please refer to your individual timetable as to which English lecture group you should attend. Also refer to ClickUP for any timetable updates.

Lecture attendance and participation in discussions are compulsory. Since the contents of each lecture follows on those of previous lectures, it is in the student's own interest to study the material covered on a regular basis and not to miss a lecture. However, should a student not be able to attend a certain lecture for whatever reason, the onus is on him / her to obtain the study material and catch up on the work. No individual lectures will be presented. Previous material will rarely be repeated in a following lecture.

Students are expected to prepare for lectures. Since a large volume of work needs to be covered, it is not possible to cover every aspect in the finest detail in the lectures themselves. Students should therefore read the textbooks thoroughly and already know beforehand what the next lecture is about in order to identify anything that is unclear. The lecturer may also assign some sections of the study notes and lecture notes / handouts for self-study. These sections will be part of the syllabus, but will not be discussed in the class.

Table 1: Lecture Timetable

Time	ma/Mo	di/Tu	wo/We	do/Th	vr/Fr
7:30-8:20			[A] Eng III 6		
8:30-9:20			[A] Eng III 6		
9:30-10:20					
10:30-11:20					
11:30-12:20				[E2] Thuto 1-1	[E1] Centenary 6
12:30-13:20				[E2] Thuto 1-1	[E1] Centenary 6
13:30-14:20					
15:30-16:20	[E1] IT4-1	[E2] Thuto 1-2			
16:30-17:20	[E1] IT4-1	[E2] Thuto 1-2 [A] Eng III 6			
17:30-18:20		[A] Eng III 6			

5.3.Tutorials

Tutorial sessions will be used to 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. The tutorial sessions are tabulated in Table 2, and allocated according to discipline.

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 added to ClickUP.

It is vital that students do these tutorial problems on their own, prior to attending the tutorial session, and use the tutorial session for getting help with problems and / or concepts that the student is struggling with. It is important to understand that the
tutors are there to assist the student to grasp difficult concepts and not to solve the
problems on their behalf.

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.

Attendance of the **tutorial sessions** is **optional** but **handing in** of the t**utorial assignments** on ClickUP is **compulsory**, as they will be **graded and count towards the semester mark**.

Table 2: Tutorial Timetable

Discipline[1]	Day	Time	Venue
S2	Mon.	11:30-13:20	NWII Lab 1,2,3,4
c3 s2 C2	Mon.	13:30-15:20	NWII Lab 2,3,4
M2	Tues.	07:30-09:20	NWII Lab 2,3,4
n3 p3 N2 P2	Wed.	11:30-13:20	NWII Lab 1,2,3,4
b3 B2	Thurs.	15:30-17:20	NWII Lab 1,2,3,4
m3 M2	Fri.	13:30-15:20	NWII Lab 1,2

[1] Discipline Symbols: B - Industrial and Systems; C - Chemical; M - Mechanical and Aeronautical; N - Materials Science and Metallurgical; P - Mining; and S - Civil.
(uppercase letter - four year plan; lowercase letter -five year plan)

6. Rules of Assessment

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

To pass the subject a student must:

- Obtain a final mark of at least 50%; and
- Obtain a subminimum of 40% for the semester mark; 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), closed-book behind a computer

6.2.Determination of Semester Mark

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

Table 3: Determination of Semester Mark

Evaluation Method	No. of	Contribution of each	Total
Semester tests (written, closed book behind a computer)	2	40%	80%
Tutorial assignments	All		20%
Total			100%

6.3.Semester Tests

Two semester tests will be written during the scheduled Engineering Test weeks. The duration of each test will be 90 minutes. Syllabi of the tests will be announced during the lecture week preceding the test week. Both tests will be closed-book. The tests will be written in the computer labs where the students will have access to Python and Microsoft Excel.

Additional test instructions will be announced during the lecture week preceding the test week and uploaded to ClickUP.

Memoranda on the scheduled tests will be made available in electronic format on ClickUP and will not be discussed during lectures.

6.4. Tutorials

Tutorial assignment questions indicated for hand-in need to be uploaded onto ClickUP by the due date. **Please note that these assignments will be graded electronically with each question receiving a grade of 0% or 100% (binary grading).**
It is therefore extremely important that you follow the hand-in instructions on requested filenames, object names and function names. Failure to do so will result in a zero grade for the question.

Note: Tutorial assignments must be solved and programmed independently. All electronically submitted material is easily checked for plagiarism, which will not be tolerated. Refer to the Departmental Study Guide (see section 2) for more information on plagiarism.

6.5. Appeals and queries on marks

In accordance with the Departmental Study Guide, students have a 14 day period within which they can appeal for an adjustment of marks due to errors in grading or other interpretation related matters. Any adjustments will only be made at the discretion of the lecturer. After this 14 day period no marks will be altered. Refer to the Departmental Study Guide (see section 2) for more information.

6.5.1. Appeal Process

1. Download the memorandum, feedback, and query documents from ClickUP.
2. Go through your test paper carefully along with the memorandum and feedback documents.
3. Fill out the query form, describing your appeal or query, and attach it to your question paper.
4. Hand your question paper in to the lecturer at the end of the lecture within the 14 day window period.

No paper will be accepted without a query form attached. Only at the end of the 14 days window period (once all query forms have been received) will the lecturer re-grade / re-evaluate your paper.

8. Study Components

8.1.Purpose of the module

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 you wrote yourself 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 selfstudy, preparation of assignments, tests and the examination. The mode of instruction is via lectures, tutorial classes and, assignment.

Notional hours include contact time as well as the estimated time necessary for preparation for tests, and exams. 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		
2	Basic Programming • Using Python as a Calculator • Names, Objects and Assignment • Import and Use Modules • Python Namespace	12	4

3	Basic Functions • User-defined Functions • Python Namespace and Scoping • Basic Numpy Arrays	12	4
4	Control Statements • Iterator-Based Looping (For Loop) • Conditional Looping (While Loop) • Branching (If, Elif, Else)	34	10
5	Solving Problems • Breaking Down Problem Complexity • Use of Functions (Simplication) • Nested Statement	36	8
6	Plotting and Graphs • 2D Graphs • Graph Annotations • 3D Graphs	12	4
7	Reading and Writing Data • Data Sharing between Application	6	2
8	High Level Programming • Additional Modules • Advanced Built-In Functions	14	4
9	Graphical User Interfaces (GUI's) • TKinter Module • Forms and Form Objects • Mouse / Keyboard Events	5	2
10	Spreadsheets • Formulas and Calculations • Spreadsheet Detective • Plotting and Graphs • Linear Programming • Data Lookups and Pivots	41	8
Total		160	46

7.3. Lecture Plan

This outlines the lecture plan for the semester. Students should use this plan to make sure that they do not fall behind the class lectures and/or tutorials. There are a total of 48 lectures and 12 tutorial sessions over the 14 weeks of the semester.

Lecture plan is available on Click-up. The lecture plan lists study themes will be covered in which weeks of the semester.

Each week has a corresponding tutorial which should be completed during the specific week in the tutorial sessions. Please consider that the suggested lecture plan on click-up is tentative and may change depending on the progress during lectures.

Please note that some subsections in a chapter of the textbook will not be covered during the lectures, whereas other subsections may be given as self-study. The lecturer will provide information about the sub-sections that will not be covered for test- and exam purposes. Selected exercise problems, from the study notes, will be covered during the tutorial sessions. The selected chapters and problems (from the study notes) that will be covered in each tutorial session will be added to ClickUP.

7.4.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 (able 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

7.5. Study Theme Descriptions

7.5.1.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.

7.5.2.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 ?? 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, oat, str, bool as well as to know how to check the type of an object using type().

7.5.3.Theme 3: 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 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.

7.5.4.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 affects the program flow and control. The student also has to be able to determine the correct conditions in order to perform a given task.

7.5.5. 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.

7.5.6.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.

7.5.7.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.

7.5.8.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 use these additional modules and functions in Python in order to solve a specific problem.

7.5.9.Theme 9: Graphical User Interfaces (GUI's)

The student has to be familiar with the Tkinter module for creating simple GUI's. The student has to be familiar with creating simple form objects (e.g. buttons, input boxes, text boxes, etc.) and be able to create these form objects on a GUI. The student also has to be familiar with handling user events (keyboard or mouse events) and be able to link these events to functions in order to solve a given problem through using a GUI.

7.5.10. Theme 10: 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 are 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.