Basics of Programming: Python

Status: Version 1

Teaching Period	Fall 2021
LUT Course Code	CT60A0203
LUT Course Title	Introduction to Programming with Python
Campus	LUT University – Lahti
Primary Learning Mode	**Face-to-Face and Online**
Credit Points	**6**
Course hours	**162**
Lecturer	Ashok Kumar Veerasamy
Lecturer Email	**Ashok Veerasamy.lut.fi**
Lecturer Location	Lahti
Course instructor(s)	**Xiaboo Bi**

Course Description

This course introduces fundamentals of programming using the Python programming language. This course covers writing simple programs using standard control structures, design subroutines such as subprograms, the use of standard Python libraries, data structures, data, and text file processing, introduces techniques for code reuse and basic strategies for software testing.

Objectives/Learning Outcomes/Capability Development

This course contributes to the development of the following graduate capabilities:

Enabling Knowledge: Syntax and basic features of programming language Python; good programming style, standards, and practices in programming; the use of standard Python libraries and basic techniques for code reuse and testing.

Critical Analysis: Ability to analyse and define requirements for solving programming tasks.

Problem Solving: Ability to design and implement computer programs to solve simple programming problems, based on analysis and modelling of requirements.

Communication: Ability to explain key concepts of programming in Python, code reuse and strategies for software testing at fundamental levels.

On completion of this course, you should:

- be able to use standard Python.
- be able to develop simple algorithms and implement them using the standard control structures.
- be able to use existing libraries and user defined functions when writing programs
- be able to write programs that promote code reuse.
- be able to write programs that correctly manipulates standard data and text files
- be able to handle exceptions thrown and writing own exception classes.
- be able to develop python programs that can read and update CSV files, for data analytics-based tasks at basic level.
- follow good coding guidelines and devise strategies to test the programs developed.

All resources and assessment submission procedures can be accessed **via Moodle** for this course.

Overview of Learning Activities

Lecture sessions per week: Week 1 – Week 14

90 minutes of lectures per week [2 X 45 minutes]

In each session, the lecturer will present new concepts and demonstrate how these concepts can be applied using Python as the vehicle. To make the lectures more interactive students are encouraged to read the notes before attending the lectures. The lectures will consist of slide presentation, writing and testing of simple Python programs. Students are expected to write simple programs and complete simple exercises as part of the lecture. Attending lectures is not obligatory for students.

Tutorial sessions per week: Week 1 – Week 14

1 X 90 -120 minutes

Integrated into each session is a tutorial which will help clarify the concepts and build up the problem-solving ability. Students will be encouraged to form small groups to promote greater interaction. Individual weekly exercises will be given via Moodle and students are expected to write and test programs using **CodeGrade** on computers. *Marks will be awarded* each week for completing the specified tasks (refer assessment tasks section).Students are expected to spend at least 2-4 hours a week writing and testing programs. Attending tutorial sessions is compulsory. *Marks will be awarded* for attending tutorial sessions.

Moodle Quizzes
The quizzes are online quizzes deigned to allow students to self-assess themselves to test their understanding of the concepts. The feedback will refer students to the necessary sections of the course. Quizzes will be announced in the lecture/tutorial sessions. There will be 5 quizzes across the whole course period. A timed Moodle quizzes may require students to work 10-15 minutes on it.

Project work
The Project work is carefully designed to demonstrate the concepts taught in the course. Students are expected to work regularly on their programming assignment tasks spending at least 10-20 hours or more outside contact hours.

The project work will help to develop the problem-solving ability of students by writing simple algorithms. The students will be expected to use all the control structures, variables input/output statements etc. We have attempted to make the project work – tasks interesting and provide visual feedback wherever possible. In addition, this project work expects students to incorporate Python concepts including data structures, Files, and creation of own exception objects.

Teaching Schedule

Lecture Week 1: INTRODUCTION

- Class introduction, course schedule, assessment tasks and course grade assessment
- Introduction to programming: History of programming
- Python Vs other programming languages
- Applications of Python
- To write a simple Python program: Code Editor (IDE)
- To create, interpret, and run Python programs: Installing Python
- To know the basic syntax of a Python program: The first Python program
- To display output on the console or IDE
- To become familiar with Python IDLE code Editor

Lecture Week 2: VARIABLES AND EXPRESSIONS

- To create, assign, and use variables to store data in Python programs.
- To know variable naming conventions and in line commenting
- To use arithmetic operators to write expressions
- Concatenation of string and numbers
- User input in Python: input ()
- Type conversions str (), int () and float ()
- String operations

Lecture Week 3: SELECTION / CONDITIONAL STATEMENTS + QUIZ 1

- To know selection statements in Python programs: if-else
- To use comparison operators for selection control statements
- To write expressions using the comparison and logical operators
- To implement selection control using if and nested if statements: if-elif-else
- To learn menu-based coding using if-elif-else

Lecture Week 4: PYTHON LOOPS / ITERATION AND ITERATIVE STATEMENTS

- To use while, for loop statements to control the repetition of statements
- To understand the flow of control in loop statements: range () and infinite loops
- To use selection control statements (if-else) in loops
- To implement program control with break and continue keywords
- To write nested loops and using selection

Lecture week 5: PYTHON PROCEDURE, FUNCTION, CALLING FUNCTIONS + QUIZ 2

- To learn how to declare functions (subprograms), invoke functions, and pass arguments to functions
- To know Python namespaces and scope of the variables

Lecture Week 6: FILE PROCESSING

- To create, read, write, update/append and close Files in Python
- To explore file properties (modes) in Python
- To learn how to seek and print contents of a file in Python
- To define member functions in Python

Lecture Week 7: DATA STRUCTURES + QUIZ 3

- To describe why data structures is essential
- To learn the steps involved in using Python Lists
- To learn combining and multiplying Lists and more.
- To learn the steps involved in using Python Tuples

Lecture Week 8: DATA STRUCTURES AND MODULES

• To learn the steps involved in using Python Set
• To learn the steps involved in using Python Dictionary
• To explore Python standard module library
• To learn creating and using modules in Python
• To learn importing user created modules in Python

Lecture Week 9: EXCEPTION HANDLING + QUIZ 4

• To know what exception is and what is exception handling
• To learn understanding the error messages and to know when to use exceptions
• To declare exceptions and distinguish exception types
• To know Python Error classes
• To write a try-except-finally block to handle exceptions
• To declare custom exception classes
• To know testing and debugging computer programs

Lecture Week 10: DATA ANALYTICS AND PYTHON PROGRAMMING

• To learn basics of data analytics
• Data analytics from the programmer's perspectives
• To know programming styles
• Data analytics and Python programming

Lecture Week 11: ALGORITHMS, PSEUDOCODE, AND LANGUAGE PROCESSORS + QUIZ 5

• To know algorithms and Pseudocode
• To know Recursion in Python programming
• To explore programming language processors: Compiler and Interpreter

Lecture Week 12: DATA/INFORMATION STORAGE FORMATS

• To know ASCII and other characters from languages around the world

Lecture Week 13: INTERFACES AND DOCUMENTATION

• To learn software documentation at various stages of program development
• To know visualization techniques
• To explore GUI and command line interface

Lecture Week 14: COURSE SUMMARY
• To recap, course summary and conclusion

Learning Resources

Prescribed Texts / References

Python programming manual 2nd Edition	Moodle course tab: Learning resources

Other Resources and useful materials
Copies of PowerPoint slides used in the lectures, lecture videos and additional Python programs Videos and supporting files can be accessed through the **Moodle** for this course. This system will also contain any special announcements, changes to lecture and assignments etc. The Moodle for this course will also contain a link to recommended online materials which can beused for revision.

Overview of Assessment / Course performance assessment

Assessment tasks for this course will include periodic tests including weekly programming assignments, online quizzes, project work and a formal online final examination.

Assessment Tasks

Assessment Tasks and Feedback
The assessment for this course uses several methods and is designed to provide regular feedback on student progress. The results for Web learn tests (Quiz for example) will be automatically marked on submission. The feedback for selected assignments will be provided during the tutorial and or prearranged student-lecturer consultation sessions.

Course grade assessment: ***0 - 5****. Weekly assignments 40%. Project work 30% and Final exam 30%*

CodeGrade based weekly Programming Exercises (PE): These are weekly formative assessment tasks distributed via Moodle for students to practice and submit their answers electronically over a total of 12 weeks. The submitted answers for PE are automatically graded by CodeGrade. The possible total score for PE is 800. It *worth 25%* of the total grade.

Moodle Quiz (MQ): These are bi-weekly continuous summative assessment tasks conducted via Moodle for students to measure learning outcomes. The submitted answers for MQ is automatically graded by Moodle. As you log in to Moodle, you can access (online) quizzes. Quizzes will be conducted online during the lecture/tutorial sessions. The possible total score for MQ is 500. It *worth 10%* of the total grade.

Project work (PJ): This is an individual based assignment (PJ). The minimum *penalty for plagiarism is failure* for this assignment. If this means that a hurdle requirement is not met, the student fails the course. Students submit solutions for PJ electronically in Moodle for grading. It *worth 30%* of the total grade.

Weekly lecture survey, Tutorial attendance and LUT course end survey (TA_SUR): As noted, *attending tutorial is mandatory* and student *get points* for tutorial attendance. In addition, students are expected to complete a weekly lecture survey (in Moodle) and LUT's course end survey (TA_SUR). The possible total score for the weekly lecture survey, tutorial attendance and LUT course end survey are 120,120 and 60 respectively (300 in total). It *worth 5%* of the total grade.

Final Exam (FE): This is an online summative assessment and will be conducted in Moodle at the end of the course. FE is mandatory and to be eligible to sit for the FE students must previously have secured at least *50% in PE, 50% in MQ and 50%* in PJ. In addition, student is expected to submit the project work before FE. It *worth 30%* of the total grade.

Table 1: Weightage of assessment tasks for CT60A0203

Assessment tasks	Points / scores	Weight
Weekly + biweekly assignments i. Weekly programming exercises (PE)* ii. Biweekly Quizzes (MQ)* iii. Weekly lecture survey + Tutorial attendance* + LUT course end survey (TA-SUR)	 800 500 120 + 120 + 60 = 300	 25% 10% 5% = 40%
Project work (PJ)*	100	30%
Online Final exam (FE)*	100	30%
****Marked are hurdle assessment***	Total	100%

To pass the course:

To attain a pass in the course, students are required to pass both the continuous assessment and the final exam components. Each selected assessment component should therefore be viewed as a *hurdle*.

Example: To be eligible to sit for the FE* and to secure a course grade**

- Possible total score for PE is 800*. Student should secure at least 400 points (50% of the PE
- Possible total score for MQ is 500*. Student should secure at least 250 points (50% of MQ).
- Possible total score for PJ is 100*. Student should secure at least 50 points (50% of the PJ).
- Possible total score for tutorial attendance is 120. Student should secure at least 60 points (50% of the tutorial attendance).

- Possible total score for FE is 100**. Student should secure at least 50 points (50% of the FE).

**Grade calculation: (25% of PE + 10% of MQ + 5% of TA_SUR + 30% PJ + 30% of FE = 100%)

The final mark is determined by totaling the weighted marks of each assessment component. If the weighed total is less than 50%, it will be the final numeric mark, and the final grade will be ZERO.

****Table 2: Grade calculation for CT60A0203**

Scores / Points in %	Grade
0 to 49	0
50 to 59	1
60 to 69	2
70 to 79	3
80 to 92	4
93 to 100	5

Assessment Timeline, Submission Dates & Submission Procedure

Week 1 – Week 14

- Programming exercises will be released via Moodle and submitted back via CodeGrade enabled Moodle system on or before scheduled due dates for each assignment.
- Weekly lecture survey will be conducted from Week 1 – Week 12 in Moodle
- Tutorial attendance will be taken from Week 1 – Week 12 although it lasts till Week 14
- LUT course end survey will be conducted at the end of the course period.

Weeks 3, 5, 7, 9, 11

- Moodle Quizzes over 14 weeks.

Week 6-7

- Project work will be released via Moodle and submitted back via CodeGrade for evaluation.

Week 12 or before

- Final exam dates will be released for registration

Week 12

- Project work (should be done individually) must be submitted by *Week 12 Friday (revised)*.