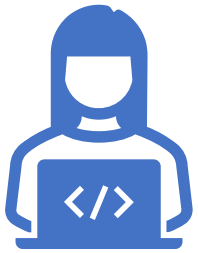
CO452
Programming Concepts

Lecture 0

Introduction to the Module

The Aim of CO452



CO452 is intended to be an introduction to programming

Theory: Concepts of programming

Practical: Solving problems and writing solutions

Learning Objectives

The learning objectives

1

Design, implement and test programs to solve simple problems

2

Correctly define fundamental programming concepts

3

Utilise best practice and professional conventions for quality code

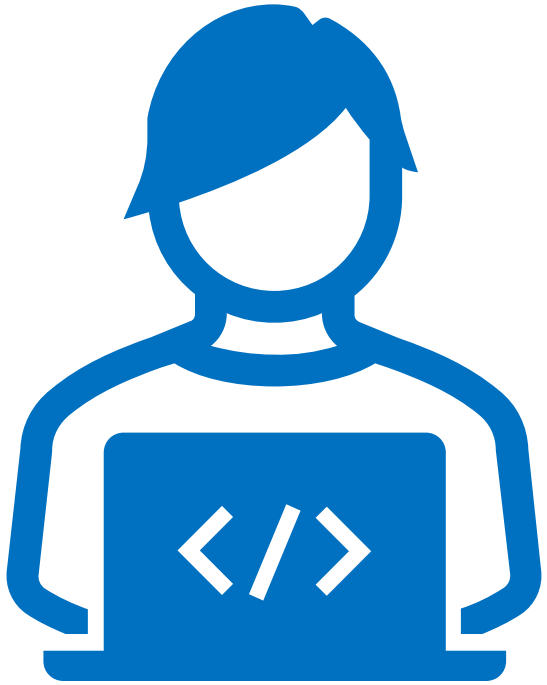
4

Apply a systematic approach to develop a solution for a project

The learning objectives

1. Design, implement and test programs to solve simple problems
2. Correctly define fundamental programming concepts
3. Utilise best practice and professional conventions for quality code
4. Apply a systematic approach to develop a solution for a project

Teaching format



Teaching format

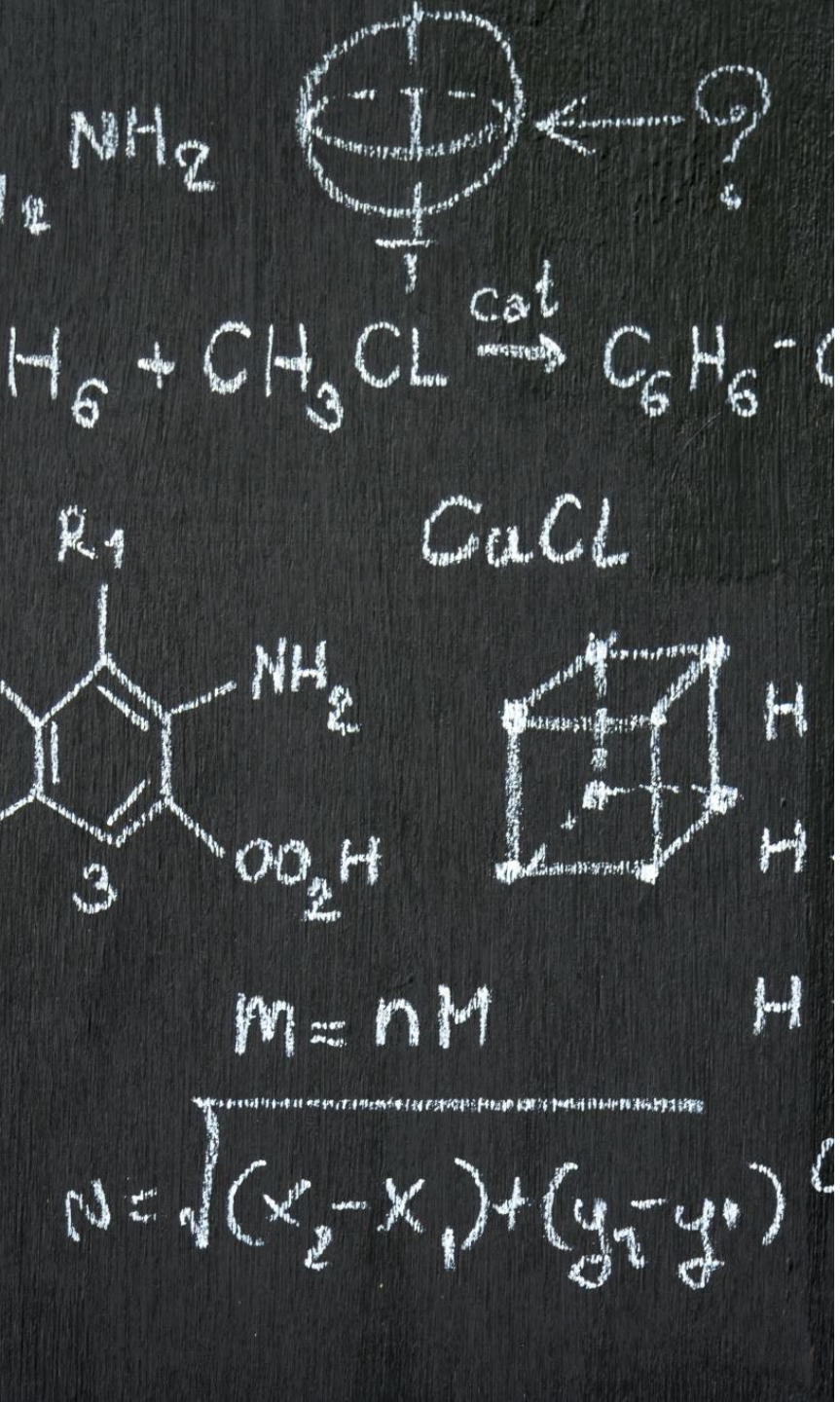
One-hour lecture

- Theory of programming
- Quizzes
- Q&A

Two-hour workshop session:

- Exercises
- Demo / discussion

Module Scheme



Module Scheme

Semesters are 15 weeks in length

CO452 is divided into three parts, each with a corresponding assessment

Module scheme is available on Blackboard and on the [CO452 GitHub wiki]

Module Scheme (1): **CW1 (20%)**

UW 1	Variables, Main, IO	W1 exercises
UW 2	Selection & Iteration	W2 exercises
UW 3	Classes & Objects	W3 exercises
UW 4	Collections: ArrayLists, Arrays & Lists	W4 exercises
UW 5	**Revision Quiz + CW1 Q&A**	W5 consolidation exercises
UW 6	Introduction to Greenfoot: Pyramid	**CW1 due (20%)**

Module Scheme (2): **TCA1 20%**

UW 7	Greenfoot demo:	Greenfoot workshop
UW 8	Greenfoot demo	Greenfoot workshop
UW 9	TCA revision quiz + Greenfoot demo	Greenfoot workshop
UW 10	(No lecture) : independent revision	**TCA1 (20%)**
UW 11	**TCA1 Feedback Session** + BlueJ Game Demo	PR1 Workshop

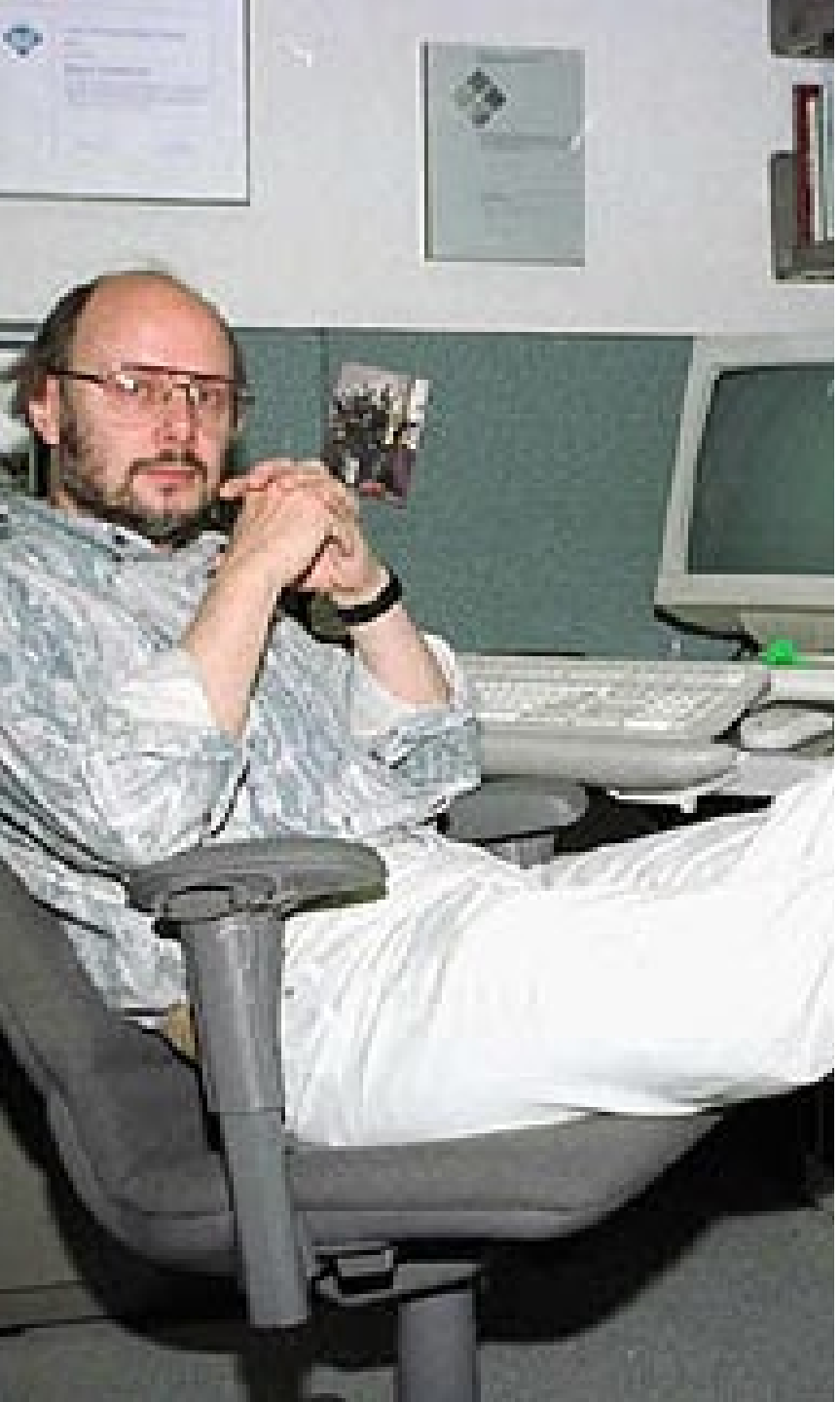
Module Scheme (3): **PR1 60%**

UW 12	Revision Quiz + PyGame Demo	PR1 Workshop
UW 13-15	Christmas Break	Christmas Break
UW 16	Presentation slot 1 for PR1	Workshop for PR1
UW 17	Presentation slot 2 for PR1	Workshop for PR1
UW 18	**Presentation slot 3 for PR1 (60%)**	N/A (module complete)

Language

The C Language

- C was first developed by Dennis Ritchie and Bell Labs in 1972
- Later formally published with the book “The C Programming Language” with Brian Kernighan in 1978.
- C contained primitive variables, functions, standard input and output stream, structures. The paradigm was procedural (before OO).



The C++ Language

- C++ was developed by Bjarne Stroustroup and his team at Bell Labs throughout the 1980s.
- The language was first referred to as ‘C with classes’, as it sought to apply the object oriented paradigm to the original C language.
- The increment operator (++) increases the integer value of a variable by one. This indicates that C++ is the next increment of the original C language.

The Java Language

- Java was published in 1995 by James Gosling of Sun Microsystems
- Java is strongly typed (data types required) like C and C++ but utilizes a virtual machine (JVM) to run on multiple devices (portability). Also has ‘automatic garbage collection’
- C# was later developed in the early 2000s to compete with Java!

Borland's Anders

The C# Language

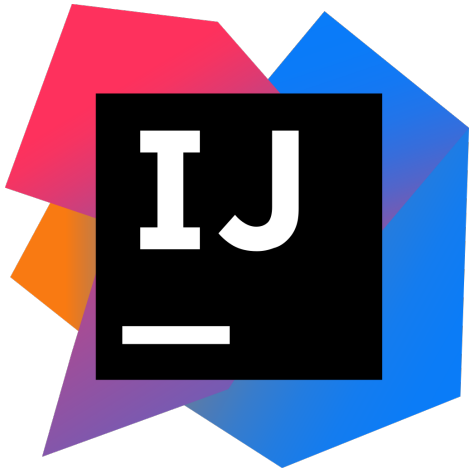
- C# was first published in 2001, developed by Anders Heljsberg from Microsoft, alongside .NET and Visual Studio.
- Close to Java syntax, and the .NET framework enables the language to run on multiple OS' (like Java's JVM).
- C# gets its name from music; in music theory, the note C# is one semitone higher in pitch than the note C.

The Python Language

- Python was published in 1991 by Guido Van Rossum, who was inspired by the UK show ‘Monty Python’.
- Python is a loosely typed (dynamic) language which doesn’t require types to be declared.
- Has grown in popularity due to data science, cyber security and AI applications.

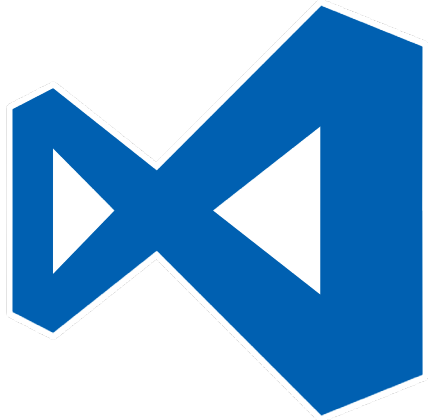
Software

Recommended IDEs

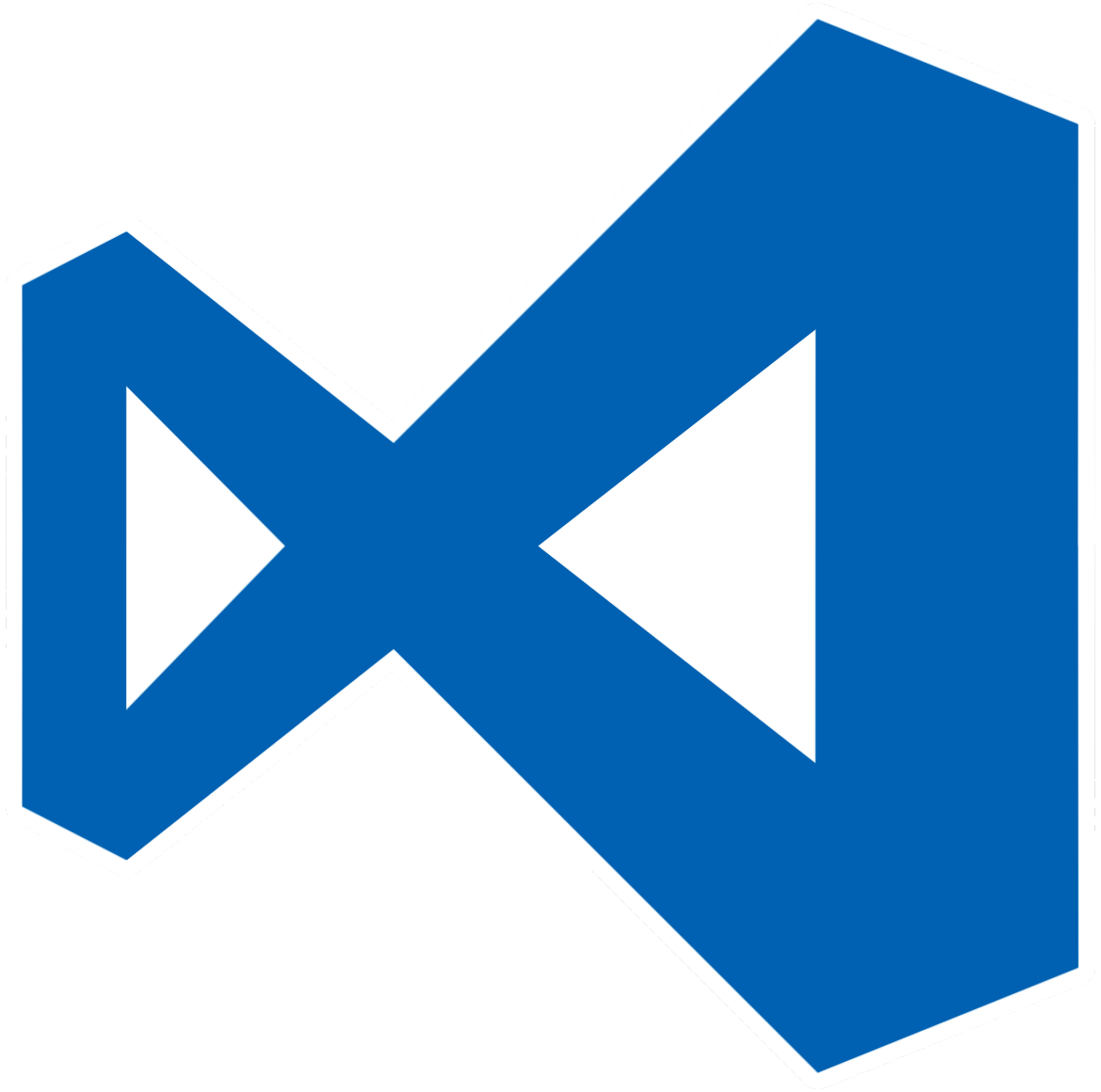


IntelliJ IDEA

GitHub Desktop



Visual Studio Code



Visual Studio Code

Smaller in size than the conventional Visual Studio and easier to get started.

You can install languages and templates as and when required.

Can code and compile projects in all the major languages.

IntelliJ

Developed by JetBrains

Professional development tool used in industry.

Structures projects so they are ready for production.

GitHub

GitHub is a version control system used in the industry

We'll use it to backup your code, nicely formatted.

Wiki pages can also be used for documentation

Create an account at GitHub.com and download GitHub Desktop

Greenfoot

We'll use Greenfoot for an introduction to game development.

It provides an easy way to get started with animations and collisions.

You may wish to continue developing your PR1 game in Greenfoot.

BlueJ

Same family of IDEs as Greenfoot.

Visualises the creation of objects from classes.

You may like to consider creating a text-based game in BlueJ for PR1. We'll demonstrate the World of Zuul.

PyGame

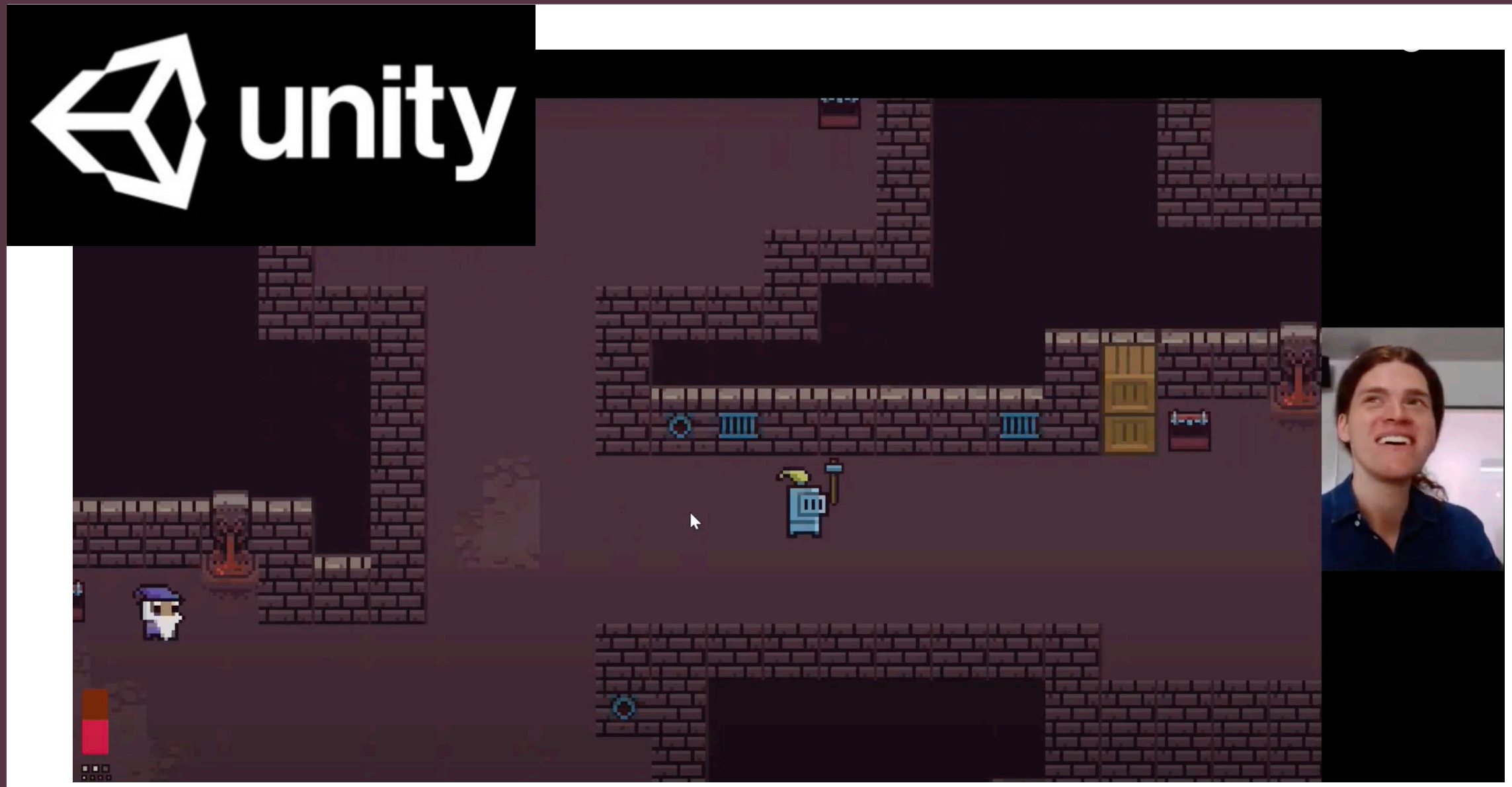
You may like to consider alternative languages to Java for the PR1 game.

PyGame is Python's games-based library. Python has less reliance on syntax.

Unity

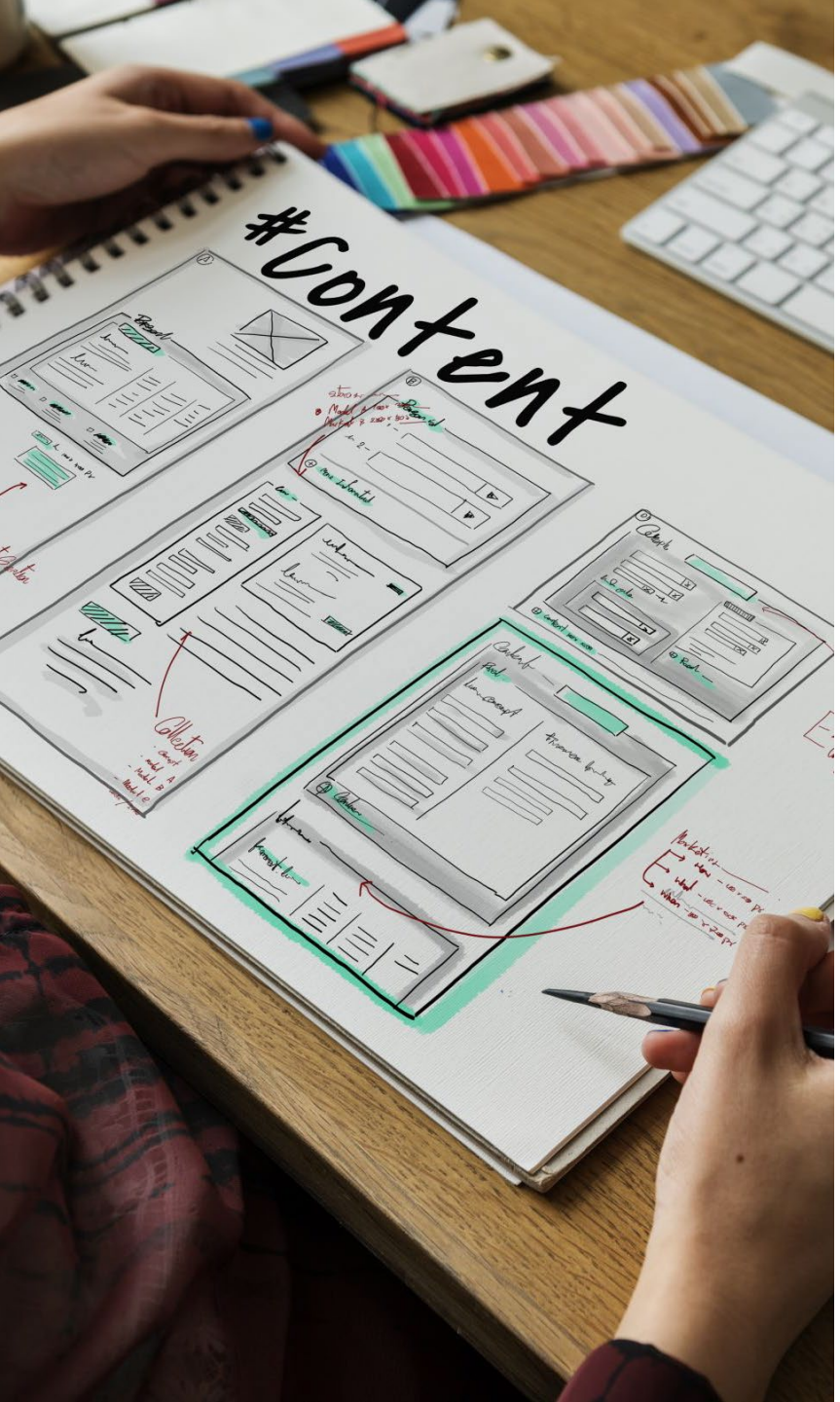
Unity is a professional game engine that is used to create 3D games.

Utilises scripts written in C# that enable objects to respond to events.


unity

pygame

Assessment



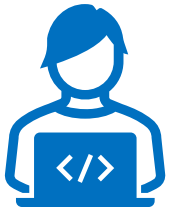
Formative assessment

Weekly exercises: these will help you practice and develop your problem-solving skills. If you engage with these, you will be able to solve the summative assessment tasks. “Practice makes perfect.”

Quiz questions: to measure your understanding of programming concepts. Remember that theory is the foundation you build upon to solve problems.

Programming is the application of conceptual understanding to problem-solving ability.

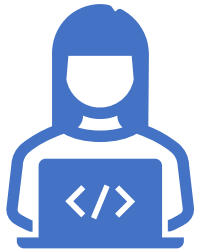
Summative assessment



CW1 (small independent project) 20

TCA1 (MCQ test on concepts) 20

PR1 (team project and presentation) 60



CW1

Small independent project based on the basic concepts taught in the initial weeks of the module.

This task is bigger in scope than the weekly formative exercises, but if you have attended each week and attempt the exercises, you'll be in a good position to solve this task.

Measures LO1 and is worth 20% of your module mark

1 Design, implement and test programs to solve simple problems

TCA1

TCA1 consists of 20 multiple-choice questions and is hosted on Blackboard. You are required to attend this TCA1 in your timetabled workshop session in W10.

The TCA will last one hour and you will have to select the correct definition of programming concepts.

A feedback session will be held the following week

Measures LO2 and is worth 20%

2 Correctly define fundamental programming concepts

PR1

This is a team-based project which requires you to deliver a game of your choice. You will need to work together to identify a suitable list of features which are then implemented.

We'll help you get started with Greenfoot, but also demo sample games in BlueJ and PyGame.

You will be required to give a small presentation on how you developed your game and also demonstrate gameplay live.

Measures LO3 and LO4 and is worth 60%

3 Utilise best practice and professional conventions for quality code

4 Apply a systematic approach to develop a solution for a project

Marking Criteria

CW1 (20%):

Testing of implemented features (80%)
Code quality (10%)
Documentation (10%)

TCA1 (20%):

Automated marking by MCQs (100%)

PR1 (60%):

Presentation (40%)
Demonstration of features (40%)
Documentation (10%)
Code quality (10%)

Marking Criteria (CW1)

	Fail 0-34	Marginal 35-39	Satisfactory 40-49	Good 50-59	Very Good 60-69	Excellent 70-79	Outstanding 80-100
Testing of features **(80%)**	No testing	Minimal testing	Basic features tested	Half of the required features tested	Most requirements tested	All requirements tested	An added feature that 'stands out'
Documentation **(10%)**	No wiki	Blank or template wiki	One or two elements included	Half of the elements included	One or two elements not included	All document--ation present but one or two issues	No issues and outstanding features documented
Code Quality **(10%)**	Quality absent	Minimal attempt to include quality	Six or more quality issues	Four to five quality issues	Two to three quality issues	One minor quality issue	No quality issues

Marking Criteria (PR1)

	Fail 0-34	Marginal 35-39	Satisfactory 40-49	Good 50-59	Very Good 60-69	Excellent 70-79	Outstanding 80-100
Presentation **(40%)**	No present--ation	Attempted but insufficient	Several technical or delivery issues	Three or four technical or delivery issues	One or two technical or delivery issues	Seamless delivery with no delivery or technical errors	A remarkable presentation that 'stands out'
Demonstration **(40%)**	No demo	Fails to run	Basic features demonstrated	Half features demonstrated	Most requirements demonstrated	All requirements demonstrated	An added feature that 'stands out'
Documentation **(10%)**	No wiki	Blank or template wiki	One or two elements included	Half of the elements included	One or two elements not included	All document--ation present but one or two issues	No issues and outstanding features documented
Code Quality **(10%)**	Quality absent	Minimal attempt at quality	Six or more quality issues	Four to five quality issues	Two to three quality issues	One minor quality issue	No quality issues

The Learning Objectives

1. DESIGN, IMPLEMENT AND TEST PROGRAMS TO SOLVE SIMPLE PROBLEMS

2. CORRECTLY DEFINE FUNDAMENTAL PROGRAMMING CONCEPTS

3. UTILISE BEST PRACTICE AND PROFESSIONAL CONVENTIONS FOR QUALITY CODE

4. APPLY A SYSTEMATIC APPROACH TO DEVELOP A SOLUTION FOR A PROJECT

Any questions?