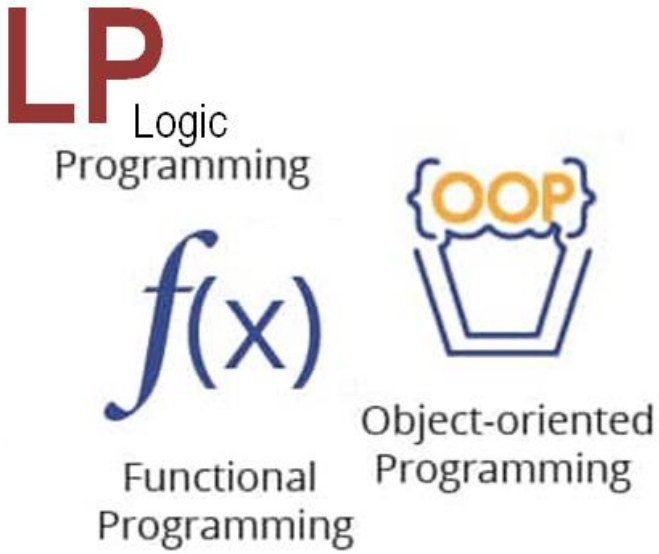
COMP1811
Paradigms of Programming

COMP1811 - Introduction

programming basics



COMP1811
Overview and Learning Outcomes

- Understand the fundamental commonalities and distinctive features of different programming languages.
- Learn computational modes of thinking and think like a programmer.
- Master the art of computational problem solving.
- Write professional style programs.
- On successfully completing this module, you will be able to:
  A. Understand the programming paradigms introduced and their applicability to real problems.
  B. Apply appropriate programming constructs in each programming paradigm.
  C. Design, implement and test small-scale applications in each programming paradigm.
  D. Use appropriate tools to design, edit and debug programs in each paradigm.

What can a computer do?
fundamentally, it

- Performs calculations
  - billions of calculations per second!
  - and can even perform them concurrently.
- Remembers results
  - 100s of gigabytes of storage!
  - or even terabytes.
- What kinds of calculations?
  - built-into the language, *called primitives*, and
  - ones that you define as a programmer.
  - How? By Computer Programming.

What is Computer Programming?

- Computer programming is the science (and art) of writing computer programs.
- Notice the confusing and *illogical* spelling.
  - you are studying a degree in a computing related **programme** (e.g. BSc Computer Science, BEng Software Engineering, etc., ...),
  - but you will be writing a computer **program**, and
  - when you do this, you will be **programming** (and not programing!).

What is a Computer Program?

- A computer program is a set of instructions that tell a computer what to do
  - everything a computer does is controlled by a program.
- Some well-known examples of programs:
  - web browsers (e.g. Firefox or Chrome) is a computer program used to view web pages,
  - an office suite (e.g. Microsoft Office) is a collection of computer programs for documentation,
  - video games are computer programs,
  - a cash machine (ATM) is controlled by a computer program.
- Computer programs are often referred to as **code** and programming is called **coding.**
- A computer program is written in a programming language.

Elements of a Programming Language
commonality to natural languages

- Programming languages have many similarities with natural languages
  - their purpose is communicating with the computer (vs human),
  - they conform to rules for syntax and semantics (grammar and meaning), and
  - there are many paradigms (dialects).

Elements of a Programming Language
Syntax - What is Syntax?

- Definition:
  - the arrangement of words and phrases to create well-formed sentences in a language,
  - the structure of statements in a computer programming language.
- It answers the question: how do I construct a valid sentence?
  - example: `x + 1` is syntactically correct in Python, but `x 1` is not.
- In programming, syntax is very strict
  - if you get it wrong, the computer won't know what you mean! and
  - sometimes in Natural Languages too: "computer get if it know mean the what won't wrong you!?
- This is why beginners sometimes find programming hard
  - they in the wrong order get the commands
  - and sum times they miss spelt the wurdz

Adapted from COMP1753 Foundations of Programming, C. Walshaw, 2018

Elements of a Programming Language
Semantics - What is Semantics?

- Semantics is concerned with the meaning of (programming) languages
  - i.e. the logic of the code,
  - usually much more difficult to define than syntax.
- It answers the questions: is this sentence valid? and if so, what does the sentence mean?
  - example: `x + 2` is valid and means add 2 to x
- A programmer should be able to anticipate what will happen by understanding the code semantics before actually running a program.

What is an IDE?

- IDE: Integrated Development Environment.
- An IDE is a software application for writing code
  that also provides comprehensive facilities for software development.
- IDEs normally consists of at least:
  - a source code editor,
  - build automation tools,
  - and a debugger.

How to think like a programmer?

- Don't start by writing code in a programming language!! (there is no typo here!)
- Start by thinking about a logical /algorithmic solution to the given problem:
  - break that problem down into smaller problems/parts,
  - find solution to each one of the smaller problems, then
  - put the solutions together to solve the given problem.

  } divide-and-conquer approach
- Write out the concepts in English (or any other natural language). } pseudocode
- Then, convert the English into code.
- Now you'll have a better program!

Why study more than one programming language?
knowing one language helps learn others...

- Makes it easier to learn new languages.
- By looking at more than one language in depth, you will begin to see the *similarities* and *differences* between the languages
  - Once you have been exposed to your first language there is no need to struggle to understand the different types of iteration or conditionals, etc.
- Concepts have even more similarity; if you
  - think in terms of abstraction – data structures, iteration, recursion, function (for example),
  - it is easier to assimilate the syntax and semantic details of a new language than if you try to pick it up in a vacuum.

Why study more than one programming language? *Contd.*

- Different tools for different jobs
- Don't Get left behind: become a more versatile developer
- Change the World!!
  - Think of what the world must have been like before 2 university students created Google!
  - Can you imagine a world without mobile phones, iPods or cars?
  - Almost everything we interact with now has a computer in it – and a computer scientist / IT professional dreamed it up!
  - A programmers developed it...