OIM3640 - Problem Solving and Software Design

Functions

How do we write code?

- So far...
  - covered language mechanisms
  - know how to write different files for each computation
  - each file is some piece of code
  - each code is a sequence of instructions
- problems with this approach
  - easy for small-scale problems
  - messy for larger problems
  - hard to keep track of details
  - how do you know the right info is supplied to the right part of code

Good Programming

- More code not necessarily a good thing
- Measure good programmers by the amount of functionality
- Introduce **functions**
- Mechanism to achieve **decomposition** and **abstraction**

Example - Projector

- A projector is a **black box**
  - You don’t know how it works
  - You know the **interface**: input/output
  - You connect any electronics to it that can communicate with that input
  - It somehow converts image from input source to a wall, magnifying it
- **ABSTRACTION IDEA:** do not need to know how projector works to use it

Example - Projector(s)

- Projecting large image for Rio 2016 Opening Ceremony
  - decomposed into separate tasks for separate projectors
  - each projector takes input and produces separate output
  - all projectors work together to produce larger image
- **DECOMPOSITION IDEA:** different devices work together to achieve an end goal

Apply these ideas to programming

- **DECOMPOSITION**
  - Break problem into different self-contained pieces
- **ABSTRACTION**
  - Suppress details of method to compute something from use of that computation

Create Structure with DECOMPOSITION

- In example, separate devices
- In **programming**, divide code into modules
  - are **self-contained**
  - used to **break up** code
  - intended to be **reusable**
  - keep code **organized**
  - keep code **coherent**
- In this lecture, we achieve decomposition with **functions**
- In a few weeks, we achieve decomposition with **classes**

Suppress Details with ABSTRACTION

- In example, no need to know how to build a projector
- In **programming**, think of a piece of code as a **black box**
  - cannot see details
  - do not need to see details
  - do not want to see details
  - hide tedious coding details
- We achieve abstraction with **function specifications** or **docstrings**

Functions

- write reusable piece/chunks of code, called **functions**
- functions are not run in a program until they are "**called**" or "**invoked**" in a program
- function characteristics:
  - has a **name**
  - has **parameters** (0 or more)
  - has a **docstring** (optional but recommended)
  - has a **body**

> Civilization advances by extending the number of operations we can perform without thinking about them.
>
> - Alfred North Whitehead