MATLAB

Programming key concepts

Programming?

- It may be (almost) impossible to solve a problem by executing commands at the command prompt.
- What is needed? A **sequence** of **precise instructions** that, once performed, will complete a **specific task**.
- Computer programs can't do that many things, they can:
  - Assign values to variables (memory locations).
  - Make decisions based on comparisons.
  - Repeat a sequence of instructions over and over.
  - Call subprograms.

Key concepts in programming

- Check Isaac Computer Science:
  https://isaaccomputerscience.org/topics/programming_concepts?examBoard=all&stage=all
- Instructions / Basic Syntax
- Data Types
  - Classification of the type of data being stored or manipulated within a program.
  - Data types are important because they determine the operations that can be performed on the data.
- Variables
  - Named container, held by the computer in a memory location.
  - Has a unique identifier (name) it lets you access it easily throughout the program.
- Input / Output

KU LEUVEN

Key concepts in programming

- Operators
  - Arithmetic
  - Comparison
  - Logical
- Sequence:
  statements are written one after another, will be executed one statement at a time in the order that the statements are written in.
- Selection:
  execute lines of code only if a certain condition is met.
- Iteration:
  repeat a group of statements .

KU LEUVEN

Top-down

Some thoughts

- http://www.walkingrandomly.com/?p=3586
- https://blogs.mathworks.com/loren/2014/01/29/coding-best-practices-a-good-read/
- *Correct and consistent is always better than faster* (Yair Altman – Accelerating Matlab Performance)

Program design?

- Turn a problem statement into a programmable solution
- Be as clear as possible about what your program should do, apply formal techniques to design an appropriate program that fulfills or answers the problem
- Work incrementally
- Write concise programs
- Write clear programs
- Many different styles of techniques of program design exist: we stick with Top-Down design
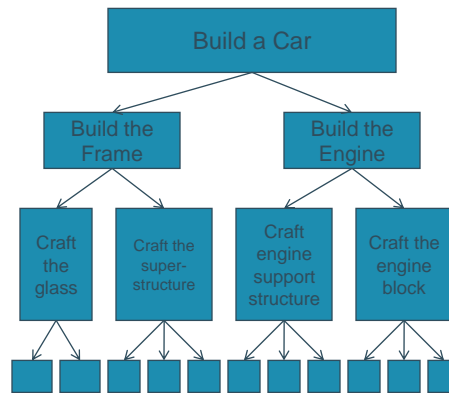
- http://ww[...]hp?title=844#Explanation
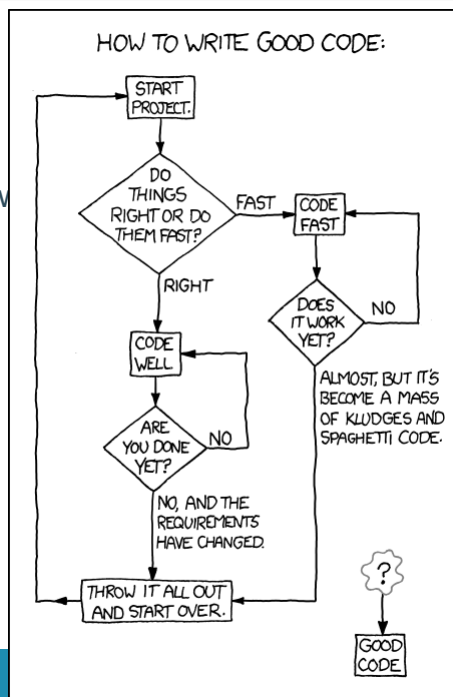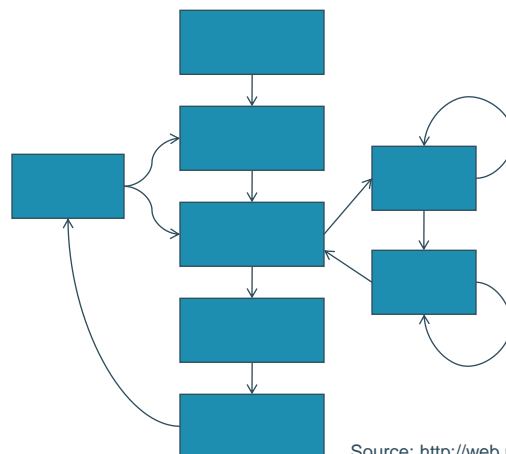
What is Top-down design?

- Process of starting with a large task and breaking it into smaller tasks
- Smaller tasks can be divided into yet smaller pieces if useful
- Continue in this manner until each small task is clearly understood and achievable
- Note
  - Many different ways to divide each problem
  - Some divisions are better than others
  - Experience and practice can lead to better designs

Build a Car

Build the Frame

Build the Engine

Craft the glass

Craft the super-structure

Craft engine support structure

Craft the engine block

Source: http://web.njit.edu/~djb38/cs101/

KU LEUVEN

Steps in Top-down design

1. Problem Statement

2. Input and output definition

3. Algorithm design
   1. Decomposition
   2. Stepwise refinement

4. Conversion of algorithm to programming language statements

5. Test final solution

Source: http://web.njit.edu/~djb38/cs101/

KU LEUVEN

Steps in Top-down design

1. Problem statement: understand the purpose of the problem.
2. Determine input/output you require.
   - What do you need to know?
   - Collect known data.
   - Some might later be found unnecessary
3. Develop conceptual model and draw a sketch.
   - Decomposition: simplify problem (enough) to enable it to be solved.
   - State any assumptions made.
   - Determine fundamental principles applicable. Set up mathematical model (logical plan if no mathematics).
   - Think generally about proposed solution approach and consider other approaches.

Before coding

4. Put it in a programming language
   - What are the subprograms?
   - Which variables are needed in which subprogram?
   - In which order and under which conditions are subprograms called? …
- Note:
   - Don't write subprograms of more than one page
   - First part of a subprogram should check the input arguments
   - Functions are black boxes: their behavior should be clear to a user:
     Clearly specify inputs and outputs using "Pre-conditions" and "Post-conditions".
   - Make a sketch before writing to save time
   - Wait 2 days before writing

Steps in Top-down design

5. Test!
- • Check dimensions and units.
- • Check solutions for a simple problem.
- • Perform a "reality check" on your answer.
  - • Does it make sense?
  - • Is the precision justified?
  - • What does the mathematics tell you?
  - • …

Test and test

- Unit testing is key
  - Test each subdivided task to ensure it works correctly
  - Each small task is tested individually
- Combine subtasks into their larger composition
  - Repeat unit testing on this new subtask
- Continue until all subtasks are combined into a single "task" or program
- Find and fix bugs
  - Bugs are unexpected or inconsistent program behavior
  - Also includes logic errors or outputs that do not match based on inputs received.

When coding

- Functions specification (comments)
  - Pre-conditions specify:
    - the meaning of each of the input parameters
    - what kinds of data the input parameters are allowed to contain
    - constraints regarding when the function can be called
  - Post-conditions specify:
    - what values are guaranteed to be in the output parameters
    - the format (what kinds of data) of the output parameters
    - constraints that are guaranteed to be true of the output parameters
  - any side-effects that the function might have, e.g. input/output interaction with the user, plotting data etc.