Software Design

Layers of a Software System

- Statement
  - x = 10
- Method/Function
  - def average(number_list):
- Class/File
  - class WebScraper:
- Namespace/Directory
  - from Crypto.Hash import SHA256
- Subsystem
  - PDF Export
- System

Software Design can occur at all levels!

What is Software Design?

- Software design is the process of defining software methods, functions, objects, and the overall structure and interaction of your code so that the resulting functionality will satisfy user requirements [source].

- Usually Occurs in two levels
  - High-Level Design (HLD)
  - Low-Level Design (LLD)

High-Level Design (HLD)

- Close to Analysis
- Overall System Design
- Includes Architecture
  - Determining what exactly is important
  - Having a shared understanding of the system design
- Represents solution to requirements

Low-Level Design (LLD)

- Close to Code
- Detailed descriptions of every module
- Expressed in the design of the classes and methods

Types of Design

- Structured design
    - From structured programming
    - More linear in nature
    - Concerned with individual modules of functionality
        - like you might in the C language
    - *What are the functions?*

- Object-oriented design
    - From object-oriented programming
    - More interested with abstractions and their interactions
    - *What are the classes?*

Classes

- A way to bundle data and functionality together
- Define a new *type* of object and *instances* of that object can be created
- Each class instance can have attributes attached to it for maintaining state
- Class instances can also have methods (defined by its class) for modifying its state

Class Design

- What classes should exist?
- What should they be named? (way harder than you'd think…)
- What are the methods of the class?
  - Names
  - Parameters
  - Return types
  - Method specifiers
    - const, static, virtual, friend, etc.
  - Access
    - public, private, protected
    - Technically Python doesn't have these like C/C++ does
- Relationship to other classes (more on this later)

Target Audience for Design Decisions

- You as a developer
- Other developers
- You again in a few months
- The other developers again in a few months
- Future developers

Informally, what indicates a good design?

- Easy to add features
- Easy to determine source of bugs
- Easy to fix bugs
- Has the required efficiency
- Has the required security
- Handles errors safely

Why does bad design occur?

- Design primarily involves making choices between tradeoffs
- Design decisions are often made before the problem is fully understood
- Incomplete knowledge by current and previous software engineers
- Requirements changes since design was made
- Security requirement changes since the design was made

Features of Good Design

- Consistent, shared vocabulary
- Simplicity
- Clear roles
- High *cohesion*
- Low *coupling*

Cohesion and Coupling

- Cohesion:
  - The degree to which the elements inside a module belong together
  - Represents the clarity of the responsibilities of a module

- Coupling:
  - The dependence two (or more) classes/modules have each other

Cohesion is *within* a class/module while coupling is *between* modules/classes.