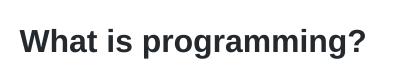
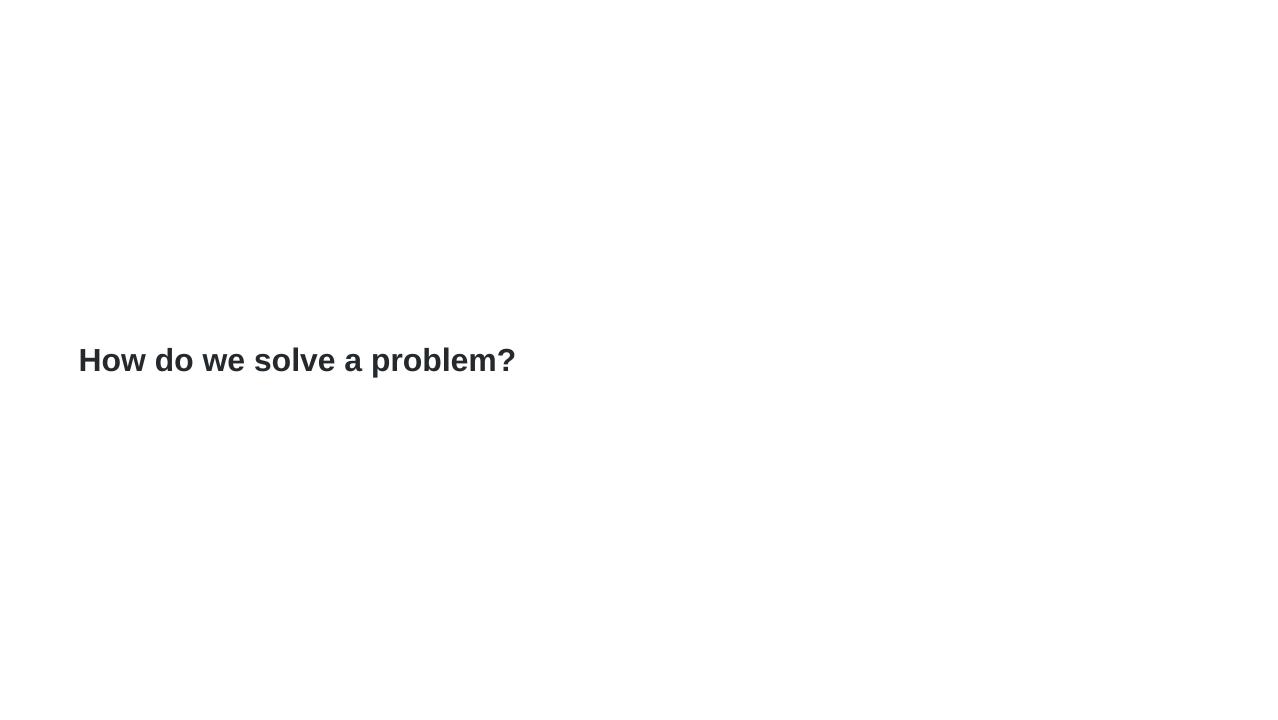
# Week 4 - Factoring & Debugging



#### What is programming?

- Problem solving
  - Using a specific toolkit (computer code)
  - Combined with logic

We write a series of logical steps that can be taken, given assumed inputs, in order to realize a proposed outcome



#### How do we solve a problem?

- In programming, we utilize a method called **Functional Decomposition** 
  - Also called Factoring
  - Break a problem down (decompose it) into its smallest functional elements
  - Construct those elements
  - Combine elements to achieve the end goal

## **Factoring Recent Assignments**

1. Homework 3 - StudentRecord and Course classes

2. In-Class Exercise - Recursive Functions

Let's walk through factoring these problems

#### **Advantages of Factoring**

- Your code will be easier to read
- You will know what you need to do
- It is clear what the next step is
- Your code will be **reusable** to a greater extent
  - Other programmers will have an easier time following your code
- It will be easier to debug and run unit tests

What is Debugging?

#### What is Debugging?

**Debugging** is, like the name suggests, the process of removing bugs from a program or script.

- Why do we get the error that we get?
- How is data moving through our code?
- What needs to be fixed?

#### What is Unit Testing?

**Unit Testing** is the process of feeding many different (and possibly wrong) types of information to our code in order to determine how the code will work under less-than-ideal circumstances.

- What happens if our input is incorrectly formatted?
- What if the data is the wrong type?
- What if ...

### Why Should I Debug and Unit Test?

- **Debugging** is critical, since our code will not work if it contains bugs. At the very least, it will not work as we expect it to
- **Unit Testing** is how we understand where our code fails to prepare for any possible case that could occur
  - We need this if we want to prevent "Garbage In, Garbage Out" problems in the future

### **Moving from Mimir to Spyder**

In order to be better able to use these functions, we need to leave Mimir behind (although it is all possible there, just harder to work with).

Let's work through some code, in order to learn how to debug it.

Here is the file

**Doing Debugging** 

**Doing Unit Tests** 

#### **Using Try, Except**

```
try:
   myCode()
except:
   raise RuntimeError("This is what went wrong...")
   # We could also use any other kind of error
   # TypeError, KeyError, etc.
```

For more types of errors, see this list

This kind of code block allows us to create code that **might actually fail**, but that we want to run wherever possible, while being notified when it does not succeed.

#### **Bonus Work**

Choose one of your lab/homework assignments from earlier in the semester.

- 1. Factor the code (if you didn't do so before)
- 2. Debug any problems that you may not have resolved during that assignment
- 3. Leave yourself comments explaining your changes.