Object-Oriented Programming

1. Model the problem in terms of objects (nouns)
2. Consider the actions that go with each object (verbs)
3. Identify the data that belongs to each object
4. Design classes that combine data and actions

Object-Oriented Programming

1. Model the problem in terms of objects (nouns)
2. Consider the actions that go with each object (verbs)
3. Identify the data that belongs to each object
4. Design classes that combine data and actions

- Imagine that you are a calendar program.
  - What objects do you work with?
  - What actions do you need to perform?
    - (Think about different levels of abstraction)
  - What specific pieces of information make up the objects you listed?

Object-Oriented Programming

1. Model the problem in terms of objects (nouns)
2. Consider the actions that go with each object (verbs)
3. Identify the data that belongs to each object
4. Design classes that combine data and actions

- Classes combine data and functionality
  - Data members hold information (like we've already seen)
  - Members can also be functions!
    - (Like the constructor we keep making)
    - Functions that are part of a class are called *methods*

Object-Oriented Programming

1. Model the problem in terms of objects (nouns)
2. Consider the actions that go with each object (verbs)
3. Identify the data that belongs to each object

- What specific pieces of information make up the objects you listed?

Using a class to combine data and actions

- The string class has private data members to store the characters that make up a string
  - We don't know how it stores them – and we don't care!
    - (They're private)

- The string class has public methods to do stuff
  - Things like split() and format()
  - Methods are *called on* an object of the class
  - `string` provides lots of built-in functionality for your use
  - Good classes are simple, flexible and general
  - https://docs.python.org/3/library/stdtypes.html#string-methods