# CSE231 - Lab 10

Problem Solving, Intro to Classes

# Program Development

As we get closer and closer to knowing everything about our first programming language, the more we want to look towards applying our knowledge to real-world situations.

Classes are extremely useful to organize and abstract away code. When we use programming to tackle a problem, we generally want to dedicate some portions of our code to one thing, and leave other portions to something else. Like functions, breaking our code into sections with classes can provide a better mental structure of what’s happening during the breakdown of a problem, and can eliminate time spent debugging, among other things.

We want you to begin thinking about how classes work. For right now, we’re going to get some practice *using another, custom class* and later transition to making our own next week.

# Problem Solving Principles

1. Understand the problem; Can you think of a picture/diagram that might help you understand the problem?
2. Devise a plan; Try creating a pseudocode algorithm for how you want to tackle the problem. Perhaps try to solve a simpler version.
3. Carry out the plan
4. Review/Extend

You might think that all of this is fairly unnecessary. But, as you deal with problems that scale in complexity *outside of this class*, these principles can be extremely helpful. Creating classes gives you the opportunity to simulate a diagrammatic way of thinking in code.

# What is a class?

We’ve secretly been using classes this entire time. Dictionaries, tuples, strings, ints, floats -- all of these are classes.

```
print(type(“Hello World!”))    # <class ‘str’>
```

It’s important to note here that “types”, as we’ve been referring to them traditionally, are synonymous with “classes”. The string we made above is an *instance of a class, called ‘str’, that holds a value: “Hello World!”*

Classes act as “templates” for the kind of data you want to store. They're helpful to us because they work in *general* situations. You can store *anything* inside a list or tuple, as an example -- we want to apply that same principle to make classes that can manage anything related to a particular problem.

# Class Behavior

User-defined classes can have custom operator definitions and method functions. Let’s take the list and dictionary for example:

Lists have `.sort()`, `.pop()`, `.append()`...

Dictionaries have `.keys()`, `.values()`, `.items()`...

Method functions are one of the ways we can dictate how our class operates. The concept of a list is fundamentally different from a dictionary, and so the Python developers created method functions catered to the kind of problems lists can be used for.

# The `Card` and `Deck` Classes

Soon, you'll be dealing with the `Card` and `Deck` classes. We won't be making them, we'll be *using* them.

These are two classes that were made by Dr. Dillon a while back, she's a computer science professor here at MSU. The classes serve as both an example of what you can do with a class, and as an example on how to *use* a class.

You'll have a lab and a project that use these, so make sure you're comfortable with them.

L10-1, L10-2