# Abstraction: The First Principle

## What is Abstraction?

Abstraction is the process of turning complex ideas into simple ones. It is removing characteristics from something so that only the essential ones remain. As programmers, we create and use abstractions all the time. Consider the following line of Python code.

```
print("hello world")
```

Whether or not you're familiar with Python, you probably recognize this as the first program we all learn to write. You also probably understand this statement will show the words, "hello world", on a computer display. However, relatively few programmers know the details of how this is accomplished. As it turns out, it takes over 3000 lines of C to implement this function.

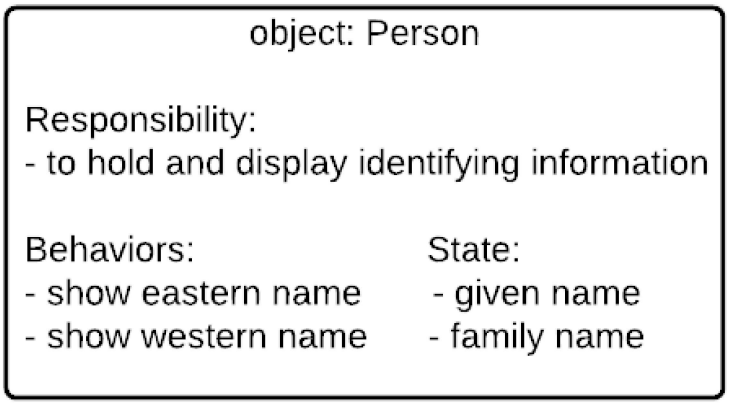
The print function in Python is an abstraction. It is the simplification of something that is actually quite complex. In order to use it, all we need to know is the function name itself, or "print", and the required arguments, some literal text or a variable that can be transformed into literal text.

The print function might be used within another function we write to create yet another abstraction. That is, a different function that simplifies a higher order idea or task. This layering of abstractions is exactly how all of the software used in the world today, from text editors like Notepad to social media platforms like Facebook, is written.

## Objects and Classes

Programming with classes is another way of creating abstractions in software. We begin, however, by thinking about objects. An object is a conceptual model for a category of things, real or imagined, that has a specific responsibility within our program. For example, we might think of an object that holds and provides identifying information about a person.

Objects have state and behavior that allow them fulfill their responsibility. The person object may have state like "given name" and "family name". It may also have related behaviors like "show western name" and "show eastern name". Following is a graphical depiction of our person object.




Thinking about a person this way is an abstraction. It is a simpflication of something that is more complex. It might seem trivial at first but not having to worry about differences in lexical name form anywhere else in our software is significant. We can just rely on the person object to take care of it.

There aren't any unrelated behaviors either. There are certainly many other ones we could think of as belonging to a person. However, our conceptualization only contains those that help fulfill its specific responsibility.

With an object in mind we are ready to translate it to a code template called a class. The object's state is translated to variables called attributes. The object's behaviors are translated to functions called methods. Following is an example of the person object translated to a class. Be sure to read all of the comments and code carefully.

Python C#

```
class Person:
    '''A code template for the category of things known as Person. The
    responsibility of a Person is to hold and display personal information.
    '''

    def __init__(self):
        '''A special method, called a constructor, that initializes two
        attributes. It is invoked using the class name followed by parentheses.
        '''
        self.given_name = ""
        self.family_name = ""

    def show_eastern_name(self):
        '''A method that displays the person's full name as used in eastern
        countries or <family name, given name>.
        '''
        print(f"{self.family_name}, {self.given_name}")

    def show_western_name(self):
        '''A method that displays the person's full name as used in western
        countries or <given name family name>.
        '''
        print(f"{self.given_name} {self.family_name}")
```

## Classes and Instances

By itself, a class is just a template for something. It only becomes useful when an instance is created and assigned to a variable in your program. An instance is the realization of attributes and methods in the computer's memory.

Another way to think about creating an instance of a class is to imagine baking a cake. In this metaphor, a class is like the recipe. It is a template for a cake but not the actual baked good. In contrast, an instance is what comes out of the oven. It is a realization of the recipe details in a sweet tasting crumb! Consider the following code.

Python C#

```
person = Person()
person.given_name = "Joseph"
person.family_name = "Smith"
person.show_western_name()
person.show_eastern_name()
```

Output:

```
Joseph Smith
Smith, Joseph
```

In this example, an instance of the Person class is created and assigned to the variable called "person". It is created by invoking a special method, called the constructor, which is the name of the class followed by parentheses. Some programming languages, like Java, C# and others, require the "new" keyword when calling a constructor.

One of the most important aspects of programming with classes is that multiple instances can be created and used in the same program. The following example shows the creation of two Person instances. Notice how the "given name" attributes are assigned different values, varying the behavior of the "show western name" method from one instance to the other.

Python C#

```
person1 = Person()
person1.given_name = "Emma"
person1.family_name = "Smith"
person1.show_western_name()

person2 = Person()
```

```
person2.given_name = "Joseph"
person2.family_name = "Smith"
person2.show_western_name()
```

Output:

```
Emma Smith
Joseph Smith
```

## Instances and Objects

Many people use the word "object" to refer to an instance of a class as well. It happens so often that "object" really seems to have a dual meaning. If you're new to programming with classes our advice is to be strict with yourself and avoid this situation while giving your colleagues some latitude. After all, you can usually tell what someone means by listening carefully to the rest of what they're saying.

## Closing Thoughts

Think back to the beginning paragraphs in this article. One of the more interesting aspects of abstraction is that we can layer them. We can do a similar thing with objects, classes and instances. Imagine creating a person object that is composed of a name object, address object, etc. Each would have their own responsibility, behaviors and state.

Abstraction is the first principle of programming with classes. As you get better at applying it, you'll find you're able to think and talk about your software, with programmers and non-programmers alike, in a very natural way. Please don't underestimate the value of this ability. Being able to communicate clearly about what is needed and what you're doing is at the very heart of creating software that is ready for change.