Subject: ECE 1111: writing flexible code
From: Joseph Picone <picone@temple.edu>
Date: 1/11/23, 12:52 PM
To: ECE 1111 <temple_engineering_ece1111@googlegroups.com>

This note is a bit advanced for this class, but it gives you an idea of where we are headed this semester during your journey to become good programmers. Programming is as much about philosophy as it is about engineering. Design is the most critical part because if you get the design wrong, the implementation becomes a nightmare.

-Joe

-------- Forwarded Message --------
Subject: writing flexible code
Date: Wed, 11 Jan 2023 12:49:56 -0500
From: Joseph Picone <joseph.picone@gmail.com>
To: nedc_research <nedc_research@googlegroups.com>

Phuykong showed me a very nice trick that solves a long-standing problem in programming. I wanted to take a moment to explain this.

Very often in research, or GUI tool development, you want a user to select an algorithm from a set of alternatives. The absolute worst way to do this is to write code like:

```
 if alg is A:
  run algorithm A
 elif alg is B:
  run algorithm B
 ...
```

This is awful because every time you add a new algorithm you have to update this if/elif construction. Usually this code appears in many places, so you create a support nightmare. The code is not sustainable.

Imagine if every time you had to add an application to your desktop, you had to recompile the desktop program. That just wouldn't work. Also, imagine you want to read data from a file and act on that data without knowing what is in the file.

In C/C++, we can build these kinds of capabilities, but it is really hard, and the code you write is not pretty (e.g., vectors of void function pointers).

Python, however, provides a very interesting capability:

```
# define variables to configure the machine learning algorithms
#
ALGS = {"PCA": PCA(), "LDA":LDA(), "QDA":QDA()}
```

This is a dictionary that contains a key and a function pointer to a constructor for a class. The syntax is pretty clean and yet very powerful.

This allows me to do something like this:

```
 if alg_name in ALGS:
  self.alg = ALGS[alg_name]
 else:
  ..."unknown algorithm name"...
```

I only need to update the above dictionary definition (and add the corresponding class of course). I don't need to update any other code. It is a very sustainable approach.

The variable self.alg is an object of whatever type you want it to be based on the key. For example, I can execute a command line like this:

```
 nedc_vclass -name PCA
```

and have the program take the string "PCA" and create an object of type PCA.

This is pretty amazing stuff and it is something Python does really well.

-Joe