**Module 8: Portfolio Project**

Cameron McIntyre

CSU Global

24SD-ITS320-1

Professor Mortoza Abdullah

August 3, 2024

**ITS320 Retrospective**

As the ITS320 class concludes in week 8, a student becomes reflective: how has our understanding of programming changed throughout this course? Gratifyingly, the response is both positive and foundational. An analysis of conceptual understanding pre and post class informs this response.

To this student, programming concepts can be divided into two groups: elements that are fundamentally intuitive, and elements that are not yet fundamentally intuitive as understanding has not been reached. Although the initial weeks of ITS320 operated on the first group, the latter half of course focused on the second group, and this is where the course was notably productive.

Elements of programming that are fundamentally intuitive are best described as elements found in children’s logic games, spreadsheets, and the programming language Scratch. These teaching tools focus on understanding the flow of programs and the computing operations that take place within the computer. A human being can intuitively understand a loop, for instance, because repetitive tasks in the natural world are, on a basic level, for or while loops. Conditional statements are also similarly natural to humans – the concept of choice between a dichotomy or a polytomy is encountered every day by a human being. Lists are similarly unconfusing. These programming concepts, although different in their syntax or operation when used programmatically, are completely natural to people as they learn to code.

There are, however, elements of programming that seem alien to an aspiring programmer; that is, they are alien until the connection between these elements and the natural world is cleanly established. The idea of data types seems arbitrary to a new programmer, until the realization that data types are necessary to structure items in a program solidifies in the student’s mind. Similarly, the concept of methods and classes are opaque until one realizes that these structures exist to mirror object relationships in the real world, segregate disparate functions, and maintain data integrity. It is a critical function of the computer science textbook, then, to inspire these connections for the student.

The author has a personal example: initially, in week one of ITS320, this author created a program called “AnythingProgram.py” that was a collection of small tasks used to practice programming. The tasks included actions such as: guess a random number, sort a list of items, print into a file, calculate numbers, and so on. This program contained a variety of conditionals, loops, and programming structures familiar to a beginner. At a high level, the program consisted of a main section with a choice list with the actions embedded in the choice list. At first this program was simple and easy to read, but as new actions were added to the program the code became difficult to modify and unattractive. Heavy scrolling was required to analyze lines of code near the bottom of the program. Unexpected data entry crashed the program in its entirety. Code for simple checks was often copy-pasted and reused. Variables became difficult to track and organize.

After concluding ITS320, version two of this program was released. The new version features all the actions as small discreet modules separate from the main function. A Python package was used to bundle all the modules into a common folder, and troubleshooting a single item in the code can be done on a single small laptop screen due to the lightweight nature of these modules. Additionally, a utility module with numerous small functions is used to sanitize variables and create exceptions when desirable, an example of the use of polymorphism to create a more durable and feature-rich user experience. To benefit the programmer, the use of classes and methods simplifies the task of adding new program actions. The understanding of object oriented programming that the ITS320 course imparted can be seen most tangibly in the improvements between version 1.0 and 2.0 of AnythingProgram.py.

All these changes in version 2.0 of this program were not immediately intuitive to the programmer when initially designing the program, but are sensible and obvious after exposure, education, and example. Retrospectively, this is the greatest benefit ITS320 has brought the author – an improved understanding of a better way to write a computer program. With excitement we look ahead to future classes – if eight weeks of ITS320 has changed our programs so foundationally, what will they look like after the next class?