* **Why would you want to create your own function? Give an example.**

Of the many reasons that one may want to create their function, there are 3 that, to me, immediately come to mind: to prevent writing the same code more than once, to add application-specific functionality, and for the sake of increased understanding of the code that is being used. For me, the most common reason for creating my function tends to be my desire to understand my code – or rather a preference to write my code as opposed to interpreting someone else’s. Although there are certainly times that a specific function must be written to accomplish a unique task within an application, in many cases, there is likely to be an open-source solution that would accomplish the same task while requiring only a fraction of the effort. Therefore, my reason for creating my own functions is usually for the sake of understanding the processes that are taking place when they are called rather than finding someone else’s solution. Granted, this is not the most scalable approach, but in my experience thus I believe that creating my own functions has not only helped me to understand the functions themselves, but also the various programming languages I have worked with and general programming logic in general. Nevertheless, I would not consider this question adequately answered without also mentioning the related benefits of creating functions for understanding, such as having consistent styling/syntax, more relevant comments, and potential performance gains based on having lean, optimized code that only does exactly what it needs to do.

* **Describe how an application would use a database. Give an example of an application you use that has a database as a critical component.**

While an application provides the means to apply logic to data, databases are used by applications to maintain the consistency of data between applications, by providing a set of static rules in which languages can query and manipulate the data that they store. There are countless applications that utilize one or more databases as critical components. In fact, one could argue that a more difficult question to answer would be to name an application that does *not* utilize a database for any reason. One example of an application that I use which relies on a database is Brightspace. Although I am not certain of the processes/queries that are happening in the back end, there is no doubt that Brightspace has established links between several databases that it uses to organize the relevant information. Aside from the obvious Student Information System (SIS) or Student Management System database that it pulls student and class information from, it also seemingly queries information from external learning services, such as this Python course, and TestOut. For that matter, TestOut’s database serves as a perfect example of a database providing consistency between applications by providing grade details to Brightspace as referenceable records for each student, while consistently storing grade values that are updated by TestOut application. Another database that is used by Brightspace is the one that contains the records that are updated by Starfish, though I have to assume that Starfish also queries data from the central SIS / SMS, as much of the information is related to both applications.

* **What are the important parts of a loop? Why are they important? What are the two general categories of loops?**

The important parts of a loop are the initialization, the condition, and the incrementor. Each of these parts is equally important, as they work in conjunction with each other to accomplish a specific task as many times as needed. The initialization serves as a starting point for a loop by establishing a base value prior to beginning the loop, and is usually defined either prior to, as part of, or nested within the loop control structure, depending on the language and type of loop. The ‘condition’ statement is what is used as the ‘kill switch’ of a loop. It defines when the loop should cease repetition, and is similar at the beginning, or end of the control statement, depending on the type of loop being used. Lastly, the incrementor of a loop is the value that is checked against the condition either before or after each loop iteration. The types of loops that can be used are: ‘for loops’ and ‘while loops’, though it should be said that some languages also utilize a ‘do… while’ control. Although they are generally used to accomplish similar tasks, ‘for’ loops tend to be preferred over while loops, especially when the number of iterations is known prior to its beginning, such as when iterating through an array, or list. In ‘for’ loops, initialization typically takes place prior to, or within the control, while the condition and iterator are defined as part of the control structure. In ‘while’ and ‘do while’ loops, initializers are not always necessary, as the condition is usually based on another variable elsewhere in the program, and are checked either before beginning an iteration, while ‘do… while’ loops check the condition after completing one. Iterators can also vary in ‘while’ type loops and, depending on the purpose of the loop, may not iterate at all.

* **Why are dictionaries useful? Give an example of a problem you would solve with dictionaries.**

Dictionaries are useful in programming whenever you want to establish a relationship between several different pairs/sets of values (or objects) of the same type, particularly when you want to be able to reference said sets of values, which may be otherwise unrelated as one ‘pseudo-object’, so to speak. Dictionaries allow these relationships to be made using key-value pairs, which, in practice, link values to their associated keys. An example of when this may be useful would be an application that is used to query, and print information from a .csv file. For instance, if I was to write a program that was going to keep a running total of how many times, I press each key on my keyboard while typing these answers, it could do so by storing each key, as a ‘key’ in my dictionary (coincidental pun). As such, every time I press a key, the program would be able to check that key for an existing entry, add it as a unique key if necessary, and add one to the associated ‘value’.