Kenneth Page

CIS104 – Intro to programming – Malleis

Winter 2022 Final Exam

#1

Git is what you’d call a ‘Version Control System.’ It allows one to keep static updates to a particular set of files – a ‘repository’ – and maintains all updates that have been made to all files so that any changes can be reversed if that is ever desired. It also has the ability to allow different ‘branches’ of code or files to be made off of the original, which would allow for different people or teams to work on updating, improving, or modifying different sections of code simultaneously, which could be merged back together later. This would be useful if you were making changes to software that is actively used, so that if you somehow encounter a bug or some kind of system-breaking flaw in the code, those changes could be reverted back to the previous working version. This would also be useful with larger programs that have team dedicated to different functionalities within the code; to be able to work on the different branches and test merging them back together to make an updated version and be able to see what changes caused what problems in a way provides feedback on what parts of the code need to be fixed. To clone a repository, you’ll need a link to the location of that repository online, and you’d either go to that repository’s main page on github.com and click the ‘code’ button above the files list, and choose the option best-suited for you, depending on the security in place. Alternatively, you could open a git program on your local computer, select the clone repository option, enter the url and create a local directory to clone it to your machine.

<https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

<https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository>

#2

‘Wire protocol’ is a set of rules for how data is to be transferred from one machine, user, or application to another. This is so that despite how much traffic might exist over any given ‘wire,’ clients and servers will not ‘bump into’ one another, or otherwise disrupt the data. One example of this is HTTP – ‘Hypertext transfer protocol’ – which is a mainstay of the internet as we know it. In general, the protocol works in a request-response cycle; the client-side program will transmit a request call to the server, and the server will make a response, often in the form of a stream of the data requested. Accompanying this will usually be a simple header structure which will confirm success of the operation, how much data was transferred, when it occurred, and the type of content transferred. Assuming all went well there, the rest of the work is on the client machine to interpret the data it’s just received and parse and restructure the data for the user. There is an organization that standardizes how these protocols work, called the IETF – the Internet Engineering Task Force. The IETF designates different ‘working groups’ with charters to work on either a new specification standard or a guideline, and will discuss things usually via an email list. In general, protocols are important for the same reason traffic laws are important, as there are many of us active in the same space at the same time, but there is limited room to accomplish our task (lanes on the road vs bandwidth), so we all need to operate on the same basic set of principles so that we don’t create problems for one another.

<https://en.wikipedia.org/wiki/Wire_protocol>

<http://rcbj.net/blog01/2012/02/18/wire-protocols/>

the third source I used for this question was the powerpoint for module 11

#3

The important parts of a loop are its length and its break condition(s). The length of the loop would also be described as the type of loop; these are usually called definite – or a ‘for’ loop – or indefinite/infinite – a ‘while’ loop. A definite loop operates off of a known number of values. For every ‘iteration’ through the loop, the machine will follow the given instructions – code. Once that set of instruction is complete, it will then repeat those same instructions on the next value, and will repeat this process until it has worked its way through all of the given values. An indefinite loop operates a given set of instructions until some overall condition that exists, changes. The code will operate and continue to repeat the code within the ‘while’ loop until the condition upon which the loop was constructed – usually a comparison operation – is no longer true. This, however, brings us to the second important part of the loop, the break conditions. Depending on what it is the loop is constructed to do, there may be reason to disrupt the code in some way so that you don’t create any errors within your overall goal. A ‘break’ condition will exit out of the loop upon some condition so that the code does not continue on forever. Otherwise, a ’continue’ statement can be added in to avoid running certain parts of the code with information that you don’t want, and will simply restart at the beginning of the loop with either the next iteration value or the current variable changes. In general, these two types of loops will be used in different ways: a ‘for’ loop will be used when you have a given set of data to work with and want to do something with it or extract some information from it, and a ‘while’ loop will be used both for an unknown set of data, but also when you want to work the data *until* some condition is met.

<https://towardsdatascience.com/understand-loops-in-python-with-one-article-bace2ddba789>

<https://www.w3schools.com/python/python_for_loops.asp>

<https://betterprogramming.pub/how-to-pick-between-a-while-and-for-loop-14ef217c3776>

#4

Programming is an interesting thing, to be sure. While I only know the one language at the moment – and a small sample of it, I’m sure – the thing that obviously makes style and structure important, to me, is the human condition. We’re all unique, we all have different ways of thinking and interpreting things. This goes very specifically into the code we write. When writing code, we are often defining our own variables, spacing and indenting in our own way, and maybe leaving cryptic comments that make sense to us and our own unique way of thinking. This may arise as an issue when working with others. I’m sure it's easy to forget that others also have their own unique way of looking at or doing things, and when we are looking through their code, either for errors or to try to help improve it, or they are looking at ours, if it isn’t structured all that similarly it may be difficult to parse through it and make sense of it. If cognizant of this while writing code, adopting a set of best practices will hopefully make working within a team all that much easier. Using variables with names that clearly allude to their purpose, separating blocks of code with comments about what the block does and why it exists, and other practices like this will make it easier for others to both read and work within our code. If we all wrote flawless code on the first try, every time, that would be one thing, but when one needs to debug, it would be quite difficult if you couldn’t even read the code. I definitely learned this the hard way with at least one of our exercises, as I constructed a block of code with my first understanding of the problem given, but when it didn’t provide the output desired, I had a hard time debugging *my own* code in order to solve the problem at hand. So even if not for others, style and structure are important for the inevitable event that you’ll need to go back through your own code and either figure out problems or improvements.

<https://owlcation.com/stem/Coding-Style-How-to-write-code>

<https://www.smashingmagazine.com/2012/10/why-coding-style-matters/>