*Little Kidogo Apprenticeship*

*Week 1.*

Object Oriented Programming.

OOP being a programming paradigm is mainly used when referring to a collection of modules or objects as designed by a programmer. These objects have attributes that mainly contain the data to be manipulated, they also have instructions depicting how these attributes or data is to be modified or used to modify other parts of the object i.e other attributes. In some cases the instructions/functions/methods defined in a class can modify data outside its scope, this it’s said to have ‘side-effects’. This becomes a useful tool when processes have to interact with other process not within the same scope of operation.

This programming paradigm is mostly useful when dealing with large applications that can be broken down to small manageable chunks called modules. Another benefit of OOP; through this modules it’s much easier to debug and modify individual parts of the program. It also provides reusability of code instead of rewriting same procedures whenever they are needed.

OOP brings about Abstraction as one of its major tools to programming, where certain levels of abstraction can be maintained allowing for a programmer to hide away certain levels of complexity of a system from the current user or even a fellow programmer working on the same program. A good example is the library that comes with most languages where common procedures are tailor-made for use by a programmer thus tacking away the nitty gritty details of let’s say; how printf displays information on a console window.

OOP is especially useful when designing GUI programs where certain actions e.g. mouse click, or mouse hover can change the state i.e. colour, position, size etc. this can also be used to call other methods that also change state of the program like when it opens a new window or something of that sort.

Functional Programming.

Unlike the Object Oriented Programming paradigm, that mostly deals as to how we get to a desired output from a given input; functional programming mostly deals with how the data is transformed from input to output. In this case, it doesn’t delve into the data as much but rather looks at how the data is transformed, and thus it doesn’t deal with assigning values thus making no changes to the data. This is referred to as immutability of the data, for this reason, it doesn’t support side effects as in OOP.

Another advantage of functional programming is that it allows concurrency through piping commands together. Meaning you can have multiple processes running at the same time. This is especially useful dealing with the current CPUs that have more cores/threads rather than before where the speed was the only factor defining a processor.

This type of programming is mostly applicable in the command line interfaces such as the Unix systems. This can be seen in most server management consoles, that mostly deal with the procedures.

*Prepared by :*

*Erickson Muchiri.*

*Ref:*

*wikipedia,org*

*stackoverflow.com*

*softwareengineering.stackexchange.com*