Computer Science Principles

Class 3

Today's Overview

- Homework review
- Github Sync
- Functions and Modules
- If – then – else decisions
- Project Work

Functions

- A Function is a small, standalone program that is created when a set of code is used multiple times.
- Functions may/may not have “inputs” to the function.
- Functions may/may not have “output” from the function

Example Function

- Sum (a, b, c, d)
- return (a+b+c+d)

Example Function

- Sum (a, b, c, d)
- return (a+b+c+d)

- Inputs are what?

Example Function

- Sum (a, b, c, d)
- return (a+b+c+d)

- Inputs are what? a,b,c,d.
- Output is what?

Example Function

- Sum (a, b, c, d)
- return (a+b+c+d)

- Inputs are what? a,b,c,d.
- Output is what? The Sum, or adding up the values of a, b, c, d.
- What type of variables are inputs and output?

Example Function

- Sum (a, b, c, d)
- return (a+b+c+d)
- What type of variables are inputs and output?
- The function doesn't expressly say!
- Some languages FORCE you to specify what type of variables so it knows how the function should be used.

Example, in ‘C’

- Int Sum(int a, int b, int c, int d)
- {
- return(a+b+c+d) //output will be an integer
- }

Example in Ruby

- Def Sum (a, b, c, d)
- return (a+b+c+d)
- End

- Ruby doesn't care about type as much. You could use this to add strings (or whatever) together

Change to Code Guidelines

- We are going to change code guidelines to be more like real code.
- Before:
- SUM
- INPUT : a, b, c, d
- OUTPUT: sum (addition) of a,b,c,d

Change to code guidelines

- New way:
- *Data type* FunctionName (input 1, input 2 ….)
- Data Type is what is going to be “returned”.
- Inputs are surrounded by parenthesis ()
- If no specific output, type is “void”
- If no specific inputs, type is “void”

Data Types

- Void = nothing, no data expected
- Integer, string, character, float, etc.
- Or
- You can create your own data type!
- Class Project example:
- HamburgerType Hamburger(……..)

Customization of inputs

- You can either “call” (i.e. use) a function with no inputs, or specify all inputs.
- Example: Function to make a hamburger, you want it “as is” or you want to customize it.
- You must put a “Default” state to the inputs, and if you do not specify any inputs, then they take the default state. If you need to change inputs, then you have to set all of them.

Hamburger Example

- HamburgerType GetHamburger (ketchup = TRUE, mustard = TRUE, pickles = TRUE, onion = TRUE)
- In our program, we will create a variable, whose “type” is HamburgerType. The call to GetHamburger will then output into our variable, so we can use it for other things we need to do.

Hamburger example

- HamburgerType newOne;
- As-is: newOne = GetHamburger() #no inputs
- Custom: newOne = GetHamburger(TRUE, FALSE, FALSE, FALSE) #hamburger with only ketchup

Modules

- A code Module is a computer file/files that contain related functions.
- Example: Hamburger module. Would contain hamburger function, Big Mac function, ¼ lber function, double cheeseburger function, etc.
- Makes code easier to understand, as your code is organized.

Class Project Modules

- Will only be done if time permits.
- For now, every function has it's own file.

Code Decision Making

- All code, and really everything, has decisions that need to be made, and what you do changes based on the decision.
- If I take the programming class ….. Then I'll need to be at Brad's house every Friday.
- If I obey my parents, then life will be easier, etc.

If - else

- Most common code decision is the “if” statement.
- If (something is TRUE)
- Do this step #note indent
- Do this step
- Do this step
- End #end of if statement

If - else

- Each “if” block stands on it’s own.
- What about the “fork in the road” type of problem. Where there are multiple things that need to happen?
- There is an associated “else” statement that provides another path

If - else

- If (something is TRUE)
- Do this step #this gets executed if true
- else
- Do this step #this gets executed if false
- end

Else if

- Maybe there are multiple paths
- For that we use “else if”
- If(this is TRUE)
- Do this #if first item is true
- Else if (something else is true)
- Do this #if first item is false, and second item is true
- Else
- Do this. #if both tests are false
- end