Institute for Personal Robots in Education (IPRE)

Class Notes for Computer Science 1 with Robots

Title: Introduction to Functions

Contents:

IPRE Class Notes are designed to act as a reference, reminder, and prompt to an instructor lecturing. You may use PowerPoint slides for some or all of your lecture, but ideally you will have a projected display from a computer with Python/Myro installed and an example robot to demonstrate the following concepts live in class. We believe that programming in front of your students is the best way to teach programming. Corrections and additions to these class notes and other educator resource material is appreciated. Please contact Jay Summet (contact info on the www.roboteducation.org website) with corrections or additions.

Functions:

- What is a function, and why do you create and use them?
    - A named piece of code that you can use multiple times.
    - A function encapsulates behavior, allowing you to make your code smaller, easier to read and easier to maintain.
        - Encapsulation of behavior allows you to repeat the same behavior multiple times without having to copy and paste a section of code.
        - If you make a change to code inside of a function, the change only needs to be made in one place (at the function) and not in every place that the function is called.

The Simplest Function

An example program:

```
forward(1,1)
backward(1,1)
forward(1,1)
backward(1,1)
```

This program has repeated behavior! Let's define a function to encapsulate this repeated behavior.

```
def yoyo():
    forward(1,1)
    backward(1,1)
```

Here we have created a simple function that takes no input and returns no (explicit) output. By using it, our program becomes:

```
def yoyo():
    forward(1,1)
    backward(1,1)


yoyo()
yoyo()
```

Notice that even including our function definition, our program takes up a few less characters now than it did originally (although it takes up a few more lines). If our original program had a few more forward/backward repeats, the new program would definitely be shorter.

Things to point out about creating and using a function:

- Indentation is used to group the code inside the function into a block.
- When the indentation ends, so does the function! (the yoyo() calls are OUTSIDE of the function)
- The code inside of the function does NOT get executed until the function is called.

- A function call will interrupt the normal flow-of-execution. When a function call is reached, the flow of execution jumps to the code inside the function, and when that code ends, the flow of execution jumps back to after the function call.

A Function with Input

You can create a function that accepts input in the form of parameters by including the name of the parameters (a type of variable that is local to the function) inside the parenthesis after the function name. Our program becomes:

```
def yoyo( time ):
        forward(1, time )
        backward(1, time )

yoyo(1)
yoyo(1)
```

Points to make:

- When we call the yoyo() function, we are passing in a one. When it executes, the time parameter (local variable) points to the value one, so the forward and backward function calls execute for 1 second each.
- Our function is now more general. We can choose how many seconds to yoyo at the time we call the function, not when we write the function.
- In this case, the behavior of the program has not changed. BUT, we could easily modify the program like this:

```
def yoyo( time ):
        forward(1, time )
        backward(1, time )

yoyo(1)
yoyo(2)
```

Points to make:

- We have changed the value we pass to the second call of the yoyo function from one to two.
- Now, the behavior of the program has changed, but we only needed to change one number. Without a function, we would have had to change the duration in two places (forward and backward) to make the same change.
- This is a trivial example, but as functions get more complicated, the ability to customize their behavior by changing parameters grows more powerful.

We could also use a second parameter to specify the speed, as follows:

```
def yoyo( speed, time ):
        forward(speed, time )
        backward(speed, time )

yoyo(1,1)
yoyo(1,2)
```

Now we can control both the speed and time of a yoyo by changing the parameters we give the function when we call it. Challenge your class to create the same behavior as the following code with only calls to forward() and backward():

```
yoyo(1,1)
yoyo(0.5,2)
yoyo(1,2)
```

Returning a Value:

If you have not already, demonstrate the getLight("left") and getLight("right") function calls to your class and explain to them that it gets the light sensor values from the left and right light sensors of the scribbler robot. *For pedagogical reasons, we are omitting the getLight("center") call for now.*

Make sure they understand that the numbers get larger as the sensor gets darker.

Here is a function that will return the total value of the left and right light sensors:

```
def getTotalLight():
        totalLight = getLight("left") + getLight("right")
        return totalLight
```

Points to make:

- This function takes no input!
- This function calls other functions!
- This function creates a local variable (totalLight), and points it at the value of the two light sensors added together.
- This function uses a return statement to return the value pointed at by the local variable.

Demonstrate how to call this function and place the value into a variable:

```
totalValue =  getTotalLight()
print totalValue
```

- Demonstrate that the value returned is approximately the sum of that returned by the

getLight("left") and getLight("right") function calls.

```
lValue = getLight("left")
rValue = getLight("right")
totalValue = getTotalLight()
print lValue
print rValue
print totalValue
```

- Why do you think that the values may not exactly add up? (Timing of light samples and variability in sensor and lighting.)
- Is the name of the function accurate? Should we re-name it to be getLeftAndRightLight (because it does not add up the getLight("center") value)? Should we at least include this detail in a comment above the function? What else should we include in a comment above the function?

Input and Output

Here is a function that takes input (number of seconds) and returns output (light sensor values) while also having a side effect (making a beep for the number of seconds provided at the frequency of the light level of the center light sensor.)

```
def beepByLight( seconds):
    lightValue = getLight("center")
    beep(seconds,lightValue)
    return lightValue
```

Run this function several times with different times and pointing the robot towards and away from a light source. We know the frequency of the beep because of the return value.

```
frequency = beepByLight(2)
print frequency
```