Institute for Personal Robots in Education (IPRE)

Class Notes for Computer Science 1 with Robots

Title: Variables, Types, and Math

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.

New Concept: A Variable

- In Python, a variable is a name (identifier) that points to a value.
- The name can be made up of letters, numbers and the underscore (_) character (A letter must be the first character) and can have arbitrary length.
- Variable names are case sensitive. myName and myname are different variables.
- To create a variable, simply name it and assign it a value to point to with the assignment operator (single equal sign)
- Some identifiers are already in use by python (the python keywords).
- If you try to use a keyword, or an invalid name, you will get a syntax error.

```
>>> myName = "Jay"
>>> number1 = 3
>>> number2 = 2

>>>3rdnumber = 5
Syntax Error: invalid syntax

>>>import = 7
Syntax Error: invalid syntax
```

- Explain why each syntax error occurred (number starting the variable name, use of a python keyword) [If your editor has syntax highlighting, the import python keyword will be a different color, which is a good hint to students that they shouldn't use it as a variable name)
- *At this point, you might want to draw a diagram on the board of the three variable names, with each name followed by an arrow that points to the value it was assigned. When you re-assign a new value to the variable, you can move the arrow to point to a new value. Leaving the old value (with no arrow pointing to it) may help student's mental model later on when we discuss variables that point to lists.*

- Using Variables
  - When python sees a variable name, it evaluates the variable (sees what it points to) and uses the value that the variable points to instead of the variable name.
  - If the python interpreter finds an identifier or variable name that has not yet been defined, it will return an error.

```
>>> myName
Jay
>>> number1
3
>>>aRandomName
NameError: name 'aRandomName' is not defined
```

- Note that you can do math with variables that point to numbers just like you can do math with numbers. This is because the python interpreter evaluates the variables by checking to see what number (value) they are pointing to, and then uses that value for the math.
- You can even store the result of your calculation in a new variable.

```
>>> 3+2
5
>>>number1 + number2
5
>>> answer = number1+number 2
>>> print answer
5
>>> answer
5
```

- Note that when you do this, the result is now being pointed to by the "answer" variable, even though it was not displayed on the screen.
- You must use the print command to view the value that the answer variable points to.
- You could also simply evaluate the variable directly, by typing answer.
  (Many students have difficulty understanding the difference between "print answer" and "answer" at the IDLE interpretive window. You may want to mention the slight difference and explain that the IDLE interpreter will automatically print out the result of each line that is evaluated (typed) into the window. When the students are running a program from a file, if they want output to appear on the screen they need to use the print command.

Variable can be re-assigned:

- Variables can only point to one value at a time, but the value they point to can be changed by using the assignment operator (single equal sign).
- *Remember, if you have drawn a representation of the variables on the board, update the values that they point to!*

```
>>> print myName
Jay
>>>myName = "Robot"
>>>print myName
Robot
>>>print answer
5
>>>answer = 7
>>> print answer
7
```

Different Types of Data (Strings & Integers)

- When stored in a computer, values have different types. So far, we have seen two types of values, Strings and Integers.
- Strings are made up of a series of characters.

- We indicate that something is a string by putting it in quotes.
- Integers are numbers that do not have a fractional component, such as -2 or 7.
- Python includes a special function that will return the type of any value, called type(). Note that String is abbreviated "str" and integer is abbreviated "int".
- When you use type() on a variable, python looks up the value that the variable is currently pointing at, and gives that variable to the type() function, which returns the type of that variable.

```
>>> type(7)
<type 'int'>
>>> type("Jay")
<type 'str'>
>>> type(answer)
<type 'int'>
>>> type(myName)
<type 'str'>
```

Math with Integers

- You can do math with integers. Python includes the typical mathematical operators, such as addition, subtraction, multiplication and division (represented with the following single character operators: +, -, *, /.

```
>>> 3 - 2
1
>>> 7 * 5
35
>>>  100 / 5
20
>>>  10 / 3
3
```

- Wait! What just happened there? Why is the answer 3 instead of 3.33333?
- The answer is the difference between integer division and floating point division.
- Because 10 and 3 are both integers (try typing type(3), and type(10)), Python uses integer division when calculating 10 / 3. Integer division always drops the fractional part of the answer, so you get an answer of 3.
- If you want your answer to include fractional parts, you should use a floating point number!

Floating Point Numbers

- Floating point numbers represent numbers that have a fractional component. They do this with a decimal point (hence the "floating point" part of the name). For example, 3.333333 is really 3 and 1/3. (The 1/3 is the fractional part)
- If you type a number that has a decimal point (even if the fractional part is zero) Python knows that it is a floating point number (abbreviated "float") and when it does math with that number it will produce floating point results.

```
>>> type(10.0)
<type 'float>
>>>10.0 / 3.0
3.3333333333333335
>> 10.0 / 3
3.3333333333333335
>> 10 / 3.0
3.3333333333333335
```

- Notice that only one of the operands to the division operator needs to be a floating point number for the answer to be returned as a floating point number.
- Math with floating point numbers is basically the same as with integers, but Python will not truncate to the nearest integer.

Order of Operations (operator precedence)

```
>>> answer = 5 * 10 + 2
```

- What value does the answer variable point to? 52 (5 * 10) + 2 or 60 (5* (10 + 2) ?
- Python follows standard mathematical rules for order of precedence. Multiplications (and divisions) are done before additions and subtractions.
- If you want to change the order of operations, you can use parenthesis. Anything inside of parenthesis is evaluated before things outside of parenthesis:

```
>>> print answer
52
>>> answer = 5 * (10 + 2)
>>> print answer
60
```

- One subtle point to make is that the assignment operator (=) happens last, after everything else. This only matters if a variable appears on both sides of the assignment operator.

```
>>> answer = 10
```

```
>>> answer = answer + 5
>>> print answer
15
```

- In this example, answer is assigned (points to) the (integer) number 10. When python evaluates this line, it evaluates answer (looks up the value answer is pointing to), and gets the number 10. Then, it adds the number 10 to the number 5 and gets the number 15. Finally, it assigns the variable answer to point to the number 15. The variable answer does not point to 15 until the very end of the evaluation, because the assignment operator happens last.

Math with Strings!

- In standard math, the plus sign only works on numbers. However, in Python, the plus sign also works on strings. (although it works differently on strings than it does on numbers.) When you "add" two strings together in python, you are really concatenating them.

```
>>> "Hello" + "There"
'HelloThere'
>>> "Hello" + " " + "There"
'Hello There'
>>> 3 + 5
8
>>> "3" + "5"
'35'
```

- Notice the difference between adding the numbers 3 and 5 and adding the strings "3" and "5"!
- We can also multiply a string by a number! (But you can't multiply a string by a string.)

```
>>> "Boo!" * 4
'Boo!Boo!Boo!Boo!"
```

Variables as function parameters

- Functions will be covered in more depth later, but you should know that anyplace where you can pass a value to a function, you can substitute a variable that points to a value of the correct type. Here are some examples with the robot: forward(1,0.5) is equivalent to:

```
>>> speed = 1
>>> time = 0.5
>>> forward( speed, time)
```

- You can also assign the return value of a function to a variable with the assignment operator!

```
>>> getName()
'Scribby'
>>> robotsName = getName()
>>>print robotsName
```

```
'Scribby'
>>> answer = "My Robots name is: " + robotsName
>>> print answer
'My Robots name is: Scribby'
```