Wiki - Level 0 - Expressions

In Python, a program consists of a sequence of statements. When executing a program, Python executes each statement in the program one after the other (sequentially). Executing different kinds of Python statement causes Python to perform different actions. For language level zero, we only consider one kind of Python statement, the print statement. To begin, load the first example at language level zero, “hello_world”, into *Pystep*.

```
print "Hello world"
```

For level zero, most of our example programs will consist of a single print statement. At this point, we suggest that you experiment with the left and right arrow keys for a moment. Note that when the entire statement is selected (the entire line is red), the selection pane shows that the selected component is a statement whose particular instance is print. Now, pressing the right arrow key selects the expression "Hello world". The selection pane notes that this expression is a “string_value” with a particular instance of "Hello world".

To evaluate this Python statement, next press the “Step” button in the left-hand control area. When evaluating a print statement, Python repeatedly reduces the expression associated with the print to a increasingly simple equivalent expressions until no further simplification is possible. At this point, the expression is a value. In this example, "Hello world" is already a value so pressing the "Step" button causes Python to execute the print statement and print the value "Hello world" in the console of the underlying CodeSkulptor window, just as executing this Python program in CodeSkulptor would. In this example, "Hello world" is a particular type of value known as string. We will return to strings later.

As part of this level, we will focus on understanding the most common kind of expressions in Python, arithmetic values or numbers, and the arithmetic expressions built from numbers. In Python, there are two types of numbers, int — corresponding to integers, and float — corresponding to real numbers. The second example “print_nunbers” has expressions of both types.

```
print 42
print 3.14
```

Stepping through this example prints out the numbers 4 and 3.14 in the console.

As in a calculator, we can construct more interesting arithmetic expressions in Python using arithmetic operators. *Pystep* includes several examples that illustrate the basic strategy for creating arithmetic expressions in Python. A arithmetic expression is either a number or a binary arithmetic operator applied to two arithmetic sub-expressions. We suggest that you use the left and right arrow keys to select the various arithmetic expressions (and sub-expressions) in these examples and note some of the possible operators in basic Python. These basic arithmetic operations include plus +, minus -, times *, division /, power **, integer division // and remainder %. The minus operator - can also be unary (i.e., take one argument), but we'll just represent that operation in binary form as multiplication by -1 for now. The functions int and float convert other kinds of data (such as strings) into ints and floats.

Python computes the value of arithmetic expression by repeatedly reducing the expression to a simpler expression until the expression is number. Python selects an arithmetic operator whose associated arithmetic expressions are numbers and then applies the operator to the numbers to compute the numerical value of the expression. When choosing between several options for which expression to evaluate, Python typically choose the leftmost unevaluated expression. We suggest that you experiment with “Step” and

“Unstep” on these examples to better understand this process of evaluation for arithmetic examples.