1 Expressions

1.1 Arithmetic operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Expression</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
<td>3 + 4</td>
<td>3 plus 4</td>
<td>7</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
<td>3 - 4</td>
<td>3 minus 4</td>
<td>-1</td>
</tr>
<tr>
<td>*</td>
<td>negation</td>
<td>-3 + 5</td>
<td>negative 3 plus 5</td>
<td>2</td>
</tr>
<tr>
<td>/</td>
<td>multiplication</td>
<td>3 * 5</td>
<td>3 times 5</td>
<td>15</td>
</tr>
<tr>
<td>/</td>
<td>division (of ints)</td>
<td>10 / 4</td>
<td>10 divided by 2, truncating remainder</td>
<td>2</td>
</tr>
<tr>
<td>/</td>
<td>division (of floats)</td>
<td>10.0 / 4.0</td>
<td>10 divided by 2</td>
<td>2.5</td>
</tr>
<tr>
<td>**</td>
<td>exponentiation</td>
<td>2 ** 4</td>
<td>2 to the power of 4</td>
<td>16</td>
</tr>
<tr>
<td>%</td>
<td>modulo (remainder)</td>
<td>10 % 3</td>
<td>10 mod 3</td>
<td>1</td>
</tr>
</tbody>
</table>

The inputs to an operator are called operands. For example, in this expression, 3 * 5, the operator is * and the operands are 3 and 5.

1.2 Types & Expressions

A type is a set of values and operations that can be performed on those values.

- **int**: the type representing integers. Operators include the arithmetic operators above.
- **float**: the type representing floating points numbers, which are approximations of real numbers. Operators include the arithmetic operators above.

An expression is a combination of operators and values (and/or variables) that python can evaluate, resulting in a single value. This definition will evolve as the course progresses.

1.3 Operator precedence

Higher precedence operators are evaluated first. To break ties, python evaluates from left to right.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>highest</td>
</tr>
<tr>
<td>- (negation)</td>
<td></td>
</tr>
<tr>
<td>*, /, %</td>
<td>lowest</td>
</tr>
<tr>
<td>+ (addition), - (subtraction)</td>
<td></td>
</tr>
</tbody>
</table>

To control what gets evaluated first, use parentheses. Example:

```python
>>> 3 + 2 * 10
23
>>> (3 + 2) * 10
50
```
2 Variables & Assignments

Data values are stored in memory. A variable is a named memory location. To make or update a variable, use an assignment statement.

An assignment statement has this form:

\[ \text{variable} = \text{expression} \]

How python executes assignment statement:

1. Evaluate the expression on the right-hand side. This produces a value. The value is stored in memory, at a particular address.
2. Store the memory address in the variable on left-hand side.

Example:

```python
>>> hourly_wage = 8.00
>>> hours_worked = 42
>>> total_pay = hourly_wage * hours_worked
>>> total_pay
336.0
```

You can use the python visualizer ([http://www.pythontutor.com/visualize.html](http://www.pythontutor.com/visualize.html)) to visualize of what happens when python executes. Be sure to adjust the settings so they look like this:

![Python Visualizer](http://www.pythontutor.com/visualize.html)

Variable names should be meaningful, like in the example above. Variable names must start with a letter or _ and contain only letters, digits, and _. Some names, like print, are special keywords in python and cannot be used.

3 Errors

There are three kinds of errors in programming:

1. **syntax error**: the statement does not match rules of python language. Ex: 3 + 5 * * 2
2. **runtime error**: the expression causes a crash. Ex: 4 / (8 - 2 ** 3)
3. **semantic error**: the program runs but produces the wrong answer. Ex:

```python
>>> base = 3
>>> height = 3
>>> triangle_area = base * height / 2
>>> triangle_area   # calculation is wrong: it should be 4.5, where is the bug??
4
```

2 of 2