Using Python as a Calculator

Basic Commands and Order of Operations

Execute commands in Python by typing them at the prompt (`>>>`) and pressing the return key. Let's start with some basic numerical operations (algebra).

How to write a numerical expression

`[number1] [operator] [number2]` (just like regular math)

Basic numeric operators

- Addition: `+`
- Subtraction: `-`
- Multiplication: `*`
- Division: `/`
- Exponentiation: `**`

Ex:

```
>>> 2 + 2
4

>>> 2 - 3
-1

>>> 3 * 2
6

>>> 5 / 2
2.5
```

– note that division can *promote* `int` to `float`

```
>>> 3 ** 2
9
```

Advanced operators

- Integer division: `//`
- Remainder (modulo): `%`

Ex:

```
>>> 5 // 2
2

>>> 5 % 2
1
```

Order of Operations

Python follows the PEMDAS order of operations:

1. **P**arentheses (evaluated from inside out)
2. **E**xponentiation (evaluated from "top most exponent" down, typically, right-to-left)
3. **M**ultiplication and **D**ivision (includes int division and remainder, evaluates left-to-right)
4. **A**ddition and **S**ubtraction (evaluated from left-to-right)

Ex:

```
>>> 2 + 3 * 2
8

>>> (2 + 3) * 2
10

>>> 2**3**2
512

>>> (2**3)**2
64

>>> 7 % 2 * 2
2

>>> 7 % (2 * 2)
3
```

Assigning variables

Use the `=` operator to *assign* a variable. The `=` operator always assigns from *right to left*.

Ex:

```
>>> x = 2
>>> x
2
>>> x + 3
5

>>> x = 5 / 6
>>> y = 3 * 2
>>> x * y
>>> 5.0

>>> x = 3
>>> y = 5
>>> x = y
>>> x
5
>>> y
5
```

Data types

When you assign a variable, it is given a type. For numbers, the default type is `int` for any integer number or `float` for any number that includes a decimal place. In Python, `int` types can be *promoted* to `float` if you perform any calculation that includes a `float` or if you perform a division (other than integer division).

Ex:

```
>>> x = 3
>>> x
3

>>> x = 3.0
>>> x
3.0

>>> x = 3
>>> x + 3
6
```

```
>>> x = 3
>>> x + 3.0
6.0

>>> x = 3
>>> x / 1
3.0

>>> x = 3
>>> x // 1
3
```

String types

A string (`str` type in Python) is just a sequence of letters or numbers. To let Python know that you are talking about a string, put an expression in either single or double quotes (it doesn't matter which). You can assign variables with the string type, just like you would assign other variables. Note that any expression in quotes is treated as a string, and not evaluated.

Ex:

```
>>> ‘‘hello world’’
‘hello world’

>>> ‘hello world’
‘hello world’

>>> x = ‘hello world’
>>> x
‘hello world’

>>> ‘‘3 + 2’’
‘3 + 2’

>>> ‘z = 4’
‘z = 4’
>>> z
NameError:  name ‘z’ is not defined
```

String operations

The + operator can be applied to 2 strings to join them together. You cannot perform mathematical operations on strings like numbers, even if the string contains numbers.

Ex:

```
>>> my_string = ‘‘hello ’’
>>> my_string + ‘‘world’’
‘hello world’’
```

```
>>> string5 = ‘5’
>>> string5 + 5
TypeError:  can only concatenate str (not ‘‘int’’) to str
```

Common errors

Python will report an error if you attempt to perform an operation that is not allowed. Examples include assigning a value to a literal number, trying to do math with strings, and trying to use a variable that does not exist.

Ex:

```
>>> 3 = x
SyntaxError:  cannot assign to literal
```

```
>>> ‘5’ + 5
TypeError:  can only concatenate str (not ‘‘int’’) to str
```

```
>>> 1 + n
NameError:  name ’n’ is not defined
```