COMP 143—301 Notes

Alex Dasneves

Monday, January 30th, 2023

Basic Mathematical Operators

\+ Addition
Ex. $2 + 3 =$

\- Subtraction
Ex. $2 - 3 =$

* Multiplication
Ex. $2 * 3 =$

/ Division
Ex. $2/3 =$

// Integer Division
Ex. $2//3 =$

** Exponentiation
Ex. $2 * *3 =$

Order of Operations

Python obsrerves basic order of operations. They are:

- Groupings — In python, the only valid grouping is Parenthesis.
- Exponentiation — Evaluated from left to right. Roots are under this category, and they are written as: $x^{\frac{a}{b}} \equiv \sqrt[b]{x^a}$
- Multiplication and Division — Evaluated from left to right.
- Addition and Subtraction — Evaluated from left to right.

Variables

Variables in math are used to represent a value. In python, variables are used to store a value. This value can be any type. We can declare any variable with the following syntax:

```
variable_name = value
```

We can display the value of a variable with the 'print' command.
Ex.

```
print(variable_name)
```

Question? Why is it called print?

Variable Types

Variable Types are used to determine what type of value a variable can store. In python, there are 4 variable types:

1. Integer — Stores a whole number value. Can be positive or negative. **Ex.** $2, -3, 0$,etc
2. Float — Stores a decimal number. Can be positive or negative. **Ex.** $2.5, -3.2, 0.001, \pi$, etc.
3. String — Stores text, such as letters, numbers, symbols, and other useful characters, such as spaces and 'Enter Key' characters. **Ex.** 'Hello World!', $'2 + 3 = 5'$,$'2.5'$, etc.
4. Boolean — Stores a logical value. Can be either 'True' or 'False'.

Styling

Styling is used to make your code more readable. It is not required, but it is recommended.
Python has it's own style guide, which can be found here: Style Guide.

Basic Overview

1. Variable names should be all lowercase, with underscores seperating words. **Ex.**

   ```
   user_name, point_total, grade_average, etc...
   ```

2. Spaces around mathematical operations are not required, but are recommended. **Ex.**

   ```
   2 + 3 is Correct, but 2+3 is not.
   ```

As with all rules, there are exceptions to these, but we will handle those when we get to them.
There are more rules, however, at this point, they do not apply to us, since we do not have the knowledge to use them. We will cover them as we learn more.

Common Pitfalls

Common Pitfalls are mistakes that are commonly made by beginners (And sometimes experienced users). They are not required to know, but it is recommended to know them, so you can avoid them.
Consider the following code:

```
num = 2
Num = Num + 1
```

This code will result in an error. This is because python is case sensitive with variable names, and as such, num and Num are not the same variables.
Another common pitfall is the following:

```
#This code will work, and print out the result of 2 + 3, which is 5.
number_one = 2
number_two = 3
print(number_one + number_two)

#This code will not work, and while it will not cause an error, it will cause unexpected results.
string_one = '2'
string_two = '3'
print(string_one + string_two) #What do we think this will print out?
```

Finally, the last one in this section is:

```
string_one = '2'
number_one = 4
print(string_one + number_one)
```

What will the code above print out?

- The result of $2 + 4$, which is 6.
- The text '24'
- An error.

How can we avoid these pitfalls?

There are a few ways we can avoid pitfalls like the 3rd example above.

- We can use the 'str()' function to convert a number to a string. **Ex.**

  ```
  num_as_str = str(3) #This will result in the variable containing '3'
  ```

- We can use the 'int()' function to convert a string to a number. **Ex.**

  ```
  num_as_int = int('3') #This will result in the variable containing 3
  ```

  However, with this example, as if the string cannot be converted into a number (without a decimal) it will result in an error. **Ex.**

  ```
  num_as_int = int('3.5') #This will result in an error
  #So will:
  num_as_int = int('Hello')
  ```

- The last method we can use is a formatted string. Formatted strings are strings that can have variables and other code snippets inserted into them. **Ex.**

  ```
  formatted_string = f'{string_one} + {number_one} = {string_one + number_one}'
  #This will result in the variable containing '2 + 4 = 24'
  ```