Computer Science 1 — CSci 1100
Lecture 2 — Python as a Calculator

Overview

Most of this is covered in Chapter 2 of both *Practical Programming* and *Think Python*

- Basic operations and some surprises
- Types
- Variables and the assignment operator
- Variable names and Python keywords
- Expressions
- Precedence
- Interactive mode vs. running scripts

Throughout we will pay attention to problems and mistakes, both real and potental.

Aside: Strong Suggestion on Note Taking

- Lecture notes are outlines.
  - Therefore, you should hand-write details as we cover them in class.
- We will create and run examples in class.
  - You should write down the shorter ones
  - We will post the longer ones on-line, but you should write down as much as you can, especially about the results of running the examples.

Python As a Calculator

We will start class by using Python as an interactive calculator, working through a number of examples.

- The area of a square, rectangle or circle
- The number of minutes in a year.
- What about the volume of a sphere? We'll calculate the volume of the earth in cubic kilometers.
- Is this right? How do we know?
- Lesson: we need to be careful to check our calculations, doing so as much as possible through calculations that we can check by hand!

Python Types

- A type is a set of possible values — also called a "representation" — and a set of operations on those values, such as +, -, *, / and **
- Common Python types we are working with initially include **integer**, **float** and **string**

Integers

- Values include
$$\{\ldots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$$
- The operators on integers include `+`, `-`, `*`, `/`, `**`, `%`.
- The biggest surprise is division. The best way to think about integer `/` and `%` is in terms of the old "dividend" and "remainder" calculation from grade school:
$$17/6 \qquad \text{and} \qquad 17\%6$$
- This integer division rule, common to many programming languages, has been changed in Python 3.
- What happens with negative numbers? We'll try it out in class and then figure out what is happening.

Aside: Lots of Integers

- Unlike many other language such as C, C++ and Java, any integer you wish to represent can be represented in Python, provided you are patient enough.
  - Using the Python interpreter, we will look at the examples of `1**1**1`, `2**2**2`, `3**3**3`, etc.
- We will soon see that this is not true for real numbers.

Floats

- Floats are the Python types most like real numbers and most like what we see when we use a calculator.
- Playing around with calculations like
  - 5.0 / 9.0 and then subtracting off 0.55555555, and
  - 1.0**1.0**1.0, 2.0**2.0**2.0, 3.0**3.0**3.0, etc.

  will help illustrate float numbers and operators.
- The 'e' notation we will see in these examples is Python's version of scientific notation.
- Python can store a limited range of possible values, both in decimals and in the exponent.

Mixing Integer and Float

- In an expression that mixes integers and floats, the conversion is made to float, but this is not done until Python applies an operation that needs to combine an integer and a float
- Examples
  - `9.0 / 5.0` is the same as both `9 / 5.0` and `9.0 / 5`
- What is `9 / 5 * 5.0`?

Exercise

1. Fix our formula for computing the volume of the earth. What is the correct value?
2. Write a formula to convert from Celsisus to Fahrenheit. Use your intuition to figure out where to put parentheses
3. Write a version of this expression that uses as few floats as possible, but is still correct, and write a version that is not correct even though it might be on a calculator.

Variables and Assignment

- Most calculators have one or several memory keys. Python, and all other programming languages, use "variables" as their memory.
- We'll start with a simple example of the area of a circle, typed in class. You will notice as we go through this that there is no output until we use a `print` statement.
- Here is a more extensive example of computing the volume and surface area of a cylinder:

```
>>> pi = 3.14159
>>> radius = 2
>>> height = 10
>>> base_area = pi * radius ** 2
>>> volume = base_area * height
>>> surface_area = 2 * base_area + 2 * pi * radius * height
>>> print "volume is", volume, ", surface area is", surface_area
volume is 125.6636 , surface area is 150.79632
```

- A variable is a name that has a value associated with it.
  - There are six variables in the above code
- Far different from its use in mathematics, the operator `=` is an assignment of a value (calculated on the right side) to a variable (on the left). Thus, in the line

```
>>>  base_area = pi * radius ** 2
```

  - Python accesses the values associated with the variables `pi` and `radius`
  - Multiplies them, and multiplies the result by 2
  - Associates the resulting value with the variable `base_area`
  - Later, Python accesses the value of `base_area` when calculating the values to assign to `volume` and `surface_area`.
- The statement

```
>>>  base_area * height = volume
```

  is not legal Python code. Try it!
- It takes a while to get accustomed to the meaning of an assignment statement in Python.

Variable Names

- Notice that our example variable names include letters and the `_` (underscore) character.
- Legal variable names in Python must
  - Start with a letter or a `_`, and
  - Be followed by any number of letters, underscores or digits.

  Characters that are none of these, including spaces, signal the end of a variable name.
- Capital letters and small letters are different
- We will look at many examples in class.

Exercise

Create 2 invalid variable names and 4 valid variable names from the `_` character, the digit `0`, and the letter `a`.

Syntax Errors

- Python tells us about the errors we make in writing the names of variables and in reversing the left and right side of the `=` operator.
- These are examples of *syntax errors* — errors in the form of the code.
- Programs with syntax errors will not run; the Python interpreter inspects the code and tells us about these errors before it tries to execute them. We can then fix the errors and try again
- More difficult to find and fix are *semantic errors* — errors in the logical meaning of our programs.
  - We have already seen an example of a semantic error. Can you think where?
  - Throughout the semester we will discuss strategies for finding and fixing semantic errors.

Python Keywords

- Not all variable names that follow the above rules are legal Python names.
- The (only) exception is the set of "keywords" that have special meaning to Python and allow use to write more complicated operations — involving logic and repetition — than just calculating.
- `print` is the first keyword we used
  - What happens when you try to assign a value to `print`?
- You can get a list of Python keywords by typing

```
>>> import keyword
>>> print keyword.kwlist
```

- Over the next few lectures, we will soon understand the detailed meanining of .

Do Variables Exist Before They Are Assigned a Value?

- Suppose we forgot to assign `pi` a value? What would happen?
  - Try it out!
- Variables do not exist until they are assigned a value.
- This leads to a simple form of semantic error.

Exercise

- Which of these are legal variable names?

```
import
56abc
abc56
car-talk
car_talk
car talk
```

- Which of these lines of code contain syntax errors?

```
pi = 3.14159
area = pi * r * r
r = 6.5
r + 5 = r_new
```

Expressions

- In our volume and surface area calculation, the right hand side of the `=` in the line

```
>>> surface_area = 2 * base_area + 2 * pi * radius * height
```

  is what's known as an *expression*.

- Expressions are formed from syntactically-legal combinations of values (2), operators (`+` and `*`), and variables (`base_area`, `radius` and `height`)
- Expressions can be as simple as a single value, and can be arbitrarily complicated.

Precedence

- The order in which operations are applied is important.
- Among the basic operations in Python the order is
  1. Parentheses
  2. Unary + and -
  3. `**`, **right to left**
  4. `*` and `/`, left to right
  5. `+` and `-`, left to right
- We will play with many examples in class
- When in doubt
  - Try it out and see!
  - Insert parentheses to make sure

Mixed Operators

- Assignments of the form

```
>>> i = i + 1
```

  are commonly seen in Python

- Python contains a short-hand for these:

```
>>> i += 1
```

  These two statements are exactly equivalent.

- Other mixed operators include

```
-=      *=     /=
```

  but `+=` is the most commonly-used for reasons that will gradually become clear.

Summary — Python as a Calculator

- Values in Python are one of several different types — integers and floats for now.
- Variables are Python's form of memory
- Python keywords can not be used as variables.
- `=` is Python's means of assigning a value to a variable
- Variables do not exist in Python until they are given a value
- Expressions are formed from combinations values, variables and operators
- Make sure you have the precedence correct in your Python expressions.