CS 157 - Introduction to Programming

Using Numbers in Python

Dr. Stephen P. Carl

Quote of the Day

Not everything that can be counted counts,
and not everything that counts can be
counted.

- Albert Einstein

Objects

In Python, all values are represented by *objects* and relationships between them. Three things define an object:

- its unique *identity,* such as an address in computer memory
- its *data type*
- its *value*

Data Types

A *data type* describes a set of values and the operations that can be performed on those values.

For example, the **`int`** data type includes all whole numbers, positive, negative, and 0; and its legal operations which include (but not limited to) addition (+), subtraction (-), multiplication (*), and the different kinds of division (/, //, %).

Numbers

Python provides several data types for working with numbers. The most often used of these are **`int`** representing integers and **`float`** representing real numbers.

Another type, **`complex,`** deals with complex numbers of the form `a + ib` where `i` is the square root of -1.

SEWANEE
The University of the South

Order of operations

In algebra we have rules governing the order of operations:

- multiplication/division/modulus are done first (left to right)
- addition/subtraction are done next (again, left to right)
- order can be changed by using grouping symbols ()

Evaluating expressions in Python works the same way, except there are more operators.

```
x = (12 + 28) // 5 * 3.5 - 10

x = 40 // 5 * 3.5 - 10

x = 8 * 3.5 - 10

x = 28 - 10

x = 18
```

SEWANEE
The University of the South

Mathematics

Along with the built-in operations on numbers, Python provides a set of the most commonly used mathematical functions and constants. These are defined in the math module. Some useful operations provided by this class include:

- Special Numbers: *e, pi*
- Trigonometric functions: sin, cos, tan
- Exponentials, powers, square roots: exp, pow, sqrt
- Absolute value, rounding and truncation: abs, floor, ceil

SEWANEE
The University of the South

Using the math module

To use functions or values defined in math you must first import it:

```
import math
```

Any module we use must be imported in this way. An import may be done at any time in the Python shell, but in a program, all import statements typically come first, before any statements or function definitions.

Once a module is imported, we access its functions (or values) using the *dot operator:*

```
math.cos(0)  # compute the cosine of 0 radians
```

SEWANEE
The University of the South

Using the math module

A few more examples:

```
math.log(10)       # compute the natural logarithm of 10
math.sin(math.pi)  # compute sine(3.14159...)
```

The trigonometric functions take angles in radians, so their arguments are of type **`float`**. Use math.radians to convert from degrees to radians.

SEWANEE
The University of the South

Getting Output from Programs

When running a program on the command line we won't see the results of each expression, so we need the **`print`** function to output any results that are computed. This function is extremely flexible and has many bells and whistles, so we'll start with just the basics.

To print results of some math functions to the Terminal window (also called the *console*) we could do:

```
print(math.pow(2, 10))   # outputs 1024 (2 to the 10th power)
print(math.cos(math.pi)) # outputs -1.0 (cosine of 3.14159...)
```

Calling **`print`** with no arguments prints a blank line.

SEWANEE
The University of the South

Using Output Methods

It's useful to be able to combine text and numbers so we can more easily interpret a program's output. Say I wanted a program to output something like this:

The time is 10:15 am

and there is a variable hours that holds 10, and a variable minutes that holds 15. Then we can do this:

```
print('The time is ', hours, ':', minutes, ' am')
```

SEWANEE
The University of the South

Combining strings and numbers

Characters between single- or double-quotes are called *string literals.*

When the **+** operator is given string literals as operands, it's meaning changes from addition to *concatenation.* The concatenation operator combines two strings into a new object:

If only one operand is a string, as in:

```
"The time is " + hours
```

the operation fails. The non-string operand must first be *converted* to a string (can you guess how?).

SEWANEE
The University of the South

Generating Random Numbers

The module random provides functions to generate a stream of pseudorandom numbers in various ranges. Random numbers are useful in many games and simulations.

To generate a random (float) value in the range [0.0,1.0) do this:

```
rand_val = random.random()
```

To generate a random (float) value in any desired range, use `uniform` like so:

```
rand_val = random.uniform(1.0, 6.0)  # generates a value in range [1,6]
rand_val = random.uniform(2.5, 10.0)  # generates a value in range [2.5,10.0]
```

SEWANEE
The University of the South

Color

Another use of numbers is to define colors. Most displays generate colors by combining red, green, and blue (RGB) components. The function color_rgb provided by the `graphics` module is used to create a new color by specifying each RGB value in the range 0..255 like this:

```
import graphics
c = graphics.color_rgb(20, 20, 166)  # a nice deep blue
```

Many online resources provide *color wheels* to help choose RGB values for a desired color. One such resource is at `https://www.colorspire.com/rgb-color-wheel/`

SEWANEE
The University of the South

Provisos and Quid Pro Quos

The familiar number systems **integer** and **real numbers** are present in Python and in general work as expected. However, computers impose limitations on numbers that can surprise us in exceptional cases.

- Computers only represent a finite subset of the integers, though the `int` type can represent arbitrarily-large integers
- Computers represent most real numbers only as approximations
- Computing with very large or very small numbers can be tricky

SEWANEE
The University of the South

Python Math Bloopers

Try these expressions in the Python interpreter:

```
3 * ( 1 / 3 )
3 * ( 1 // 3 )
math.pow(math.sqrt(2), 2)
2 - math.pow(math.sqrt(2), 2)
(2 - math.pow(math.sqrt(2), 2)) * 10e20
'0' + '4'
'0' + '4' - '3'
'0' + 4
```

SEWANEE
The University of the South