Intro to Computing—CSCI-1310

Email David

Wed, Jan 21, 2015

Operators & algorithms

In our last lecture, we looked at operations, primarily on numbers and strings. The `+` operator could work on both numbers and strings, however, some of the other operators you've seen, such as `*`, `/`, and `-` are possible only on numbers. If you try to divide a string by another string, or even by a number, what does that mean. It doesn't really make sense to do so, which is why you'll get an error. For example, `c = s / a` in the following equations won't work.

```
a = 5
s = "this is a string"
# c = s / a
```

Another operator for numbers that we haven't discussed yet, but it very useful in programming is the modulus operator. The modulus will produce the remainder of a division. To understand this, it's helpful to think back to when you learned long-division in 5th grade. This was before you were introduced to decimals, most likely. In 5th grade division, if you divide 75/50, you get 1 R 25, also known as 1 remainder 25. This is because 50 goes into 75 one time with 25 left over. Since 25 is less than 50, there isn't another chunk of 50 in the 75. We can see this in code in the following example

```
a = 75
b = 50
c = int(a / b)
print(c)
c = a % b # the % is the mod operator
print("mod ", c)
```

The mod operator is really useful for finding out how many discrete elements are left over after a division.

Exercise

Convert 200 seconds to minutes and seconds.

If you just do division only, you'll have a fractional number of minutes, but you won't have minutes and seconds. To get seconds,

you need to use the mod operator.

Algorithms

In the couple of programs that we've written so far, the one where we calculated area and circumference and the seconds to minutes and seconds conversion that we just did, there is what's called an algorithm that was implemented. The concept of an algorithm is key in computer programming because ultimately every program that you write will be implementing some sort of algorithm. Some algorithms will be simple and some will be very complex.

An algorithm is a defined set of steps, or a process, that is followed to solve a problem. We implement algorithms in computer code. One important concept to note here is that algorithms generalize to all programming languages, they are not specific to Python. All languages are used to implement algorithms, and we have selected Python as our vehicle for implementing algorithms.

You can think of an algorithm as being similar to a recipe. There is an objective to accomplish (problem to solve) and the order of the steps that you take to solve that problem matters. For example, seconds converter (shown in Lecture4Mins.py), we declare a variable called seconds and then use it in our calculations. If we tried to use seconds before declaring it, the program would fail, we missed a step in implementing our algorithm.

One of the features of a good algorithm is that it is general purpose. This can mean many different things depending on the algorithm, but for our purposes right here, it will mean that the algorithm works for many different inputs. In our previous examples, we've calculated area and circumference for a fixed radius and we've calculated minutes and seconds from a fixed number of seconds. But, what is much more common and useful in a program is to have the program calculate area and circumference for any radius given by a user, for example. We want the user to be able to input a value for one or more variables, and then our program will use those inputs in the calculations.

There are a few ways to get user input in Python. There is a built-in function called input that will prompt the user when the program runs. There are also command-line arguments, which are values that you enter when you run the program. We will discuss both approaches.

The Python input command will display a prompt on screen, and wait for a response typed on the keyboard by the user. Whatever the user types in then stored in a variable, that can then be used in the code.

For example:

```
d = input("Please enter a number:")
print(d)
```

The `input` command waits for the user to enter something when the program runs in this example, the value they enter is stored in variable `d`, which is a string by default doing any math operations on d requires converting it to a number, such as an int or float.

```
# for example, this expression will break
# e = d + 5

# however, this is legal
e = int(d) + 5
print(d," plus 5 equals ", e)

# for example, we could modify the code to work for
# any number of seconds
# Python has an input command to get user input
seconds = input("Enter a number of seconds to convert: ")

# convert seconds to an integer, it's treated as
# a string by default
seconds = int(seconds)

# do the calculations
mins = int(seconds / 60)

# to get seconds, we want the modulus of dividing by 60
# this is the fractional seconds in a minute
s = seconds % 60
print("200 seconds is ",mins, " mins and ", s, "seconds")
```

Exercise

Burning one gallon of gasoline in your car produces 19.4 pounds of CO2. First, Write a program to answer the following question: If the Subaru WRX gets 26 mpg, how much CO2 does this vehicle produce if it travels 12,000 miles a year? Print the result in a nice print statement.

Next, modify the code to ask the user how many miles per year they travel and the mpg that their car gets, use these inputs in your calculation instead of the 12,000 miles and the 26mpg

The value of having miles and mpg be inputs to the calculation for CO2PerYear is we can calculate this value for any yearly mileage and any mpg rating.