Writing Simple Programs (part 2)

CMSC 104 Section 2
September 11, 2025

Administrative Notes

Classwork number 2 will be assigned today; it’s due next Wednesday (September 17) before midnight

Things to make sure I cover in this lecture:

Input statements

Print statements: variables; literals; + vs ,

Strings

Reals & integers

Strings, Reals and Integers

The computer only uses bits…

Internally, computers only understand collections of 1’s and 0’s

But high-level programmers (like us) need the computer to understand things like real numbers, integer numbers, letters, groups of letters, lists of things, …

- Because that’s what we understand

So the solution is for the system to understand the “type” of each data item

What does 1000001 mean?

Suppose a memory location in the computer stores the series of bits 1000001

What does that mean?

- By convention, that's how you represent an uppercase "A"
- But when you convert 1000001 from base 2 into base 10, you get the integer number 65

So which do you mean - "A" or 65?

Python resolves this by giving every value and variable a "type"

The simplest types are strings, integer numbers and real numbers

Types

We'll get to integer and real numbers mostly next week

Strings: a string is a set of 0 or more characters that is treated as a unitary item

- Characters can be anything that can be represented on your computer
  - Digits; upper & lower case characters; special characters; whitespace like blank spaces, tabs and newline characters - they're all valid

What can you do with a string?

Lots of things, but relevant to classwork 2: you can concatenate them

- string1+string2 yields string1string2
  - The two strings combined in the specified order with no spaces between them
  - If you want a space between them you have to explicitly add it
    - string1 + " " + string2

More on Print statements

Printing strings

To print a literal string, put it between quotes:

```
print(“Hello, world”)
```

To print a variable that holds a string, just use the variable name:

```
print(name)
```

Printing multiple values in a single statement

You can print multiple values in the same print statement

- If you separate the values with a + they will be concatenated - no space between them
- If you separate the values with a comma , there will be one blank space printed between them

print(“hello, world”+name)

print(“hello, world”,name)

Input Statements

Getting data from the user

At this point in the class, all data comes from the user via the keyboard.

You do that using the “input” statement

The syntax is:

```
var_name = input(“prompt string”)
```

At this point your program will stop and wait for input. It will continue waiting until the user types a newline (enter, carriage return, whatever you want to call it).

Everything the user types before the newline is stored in var_name as a string

Prompt strings

The prompt string in an input statement can only be a single string. You can't print out a paragraph telling the user what to do. You have to work around that.

WRONG:

```
name = input("The program will now ask for your name.", "Please type it in the format lastname, firstname, middle initial, generational suffix.")
```

RIGHT:

```
print("The program will now ask for your name.")
print("Please type it in the format lastname, firstname, middle initial, generational suffix.")
name = input("Enter your name now ")
```