Input & Output

February 3, 2021

Administrative Notes

Make sure you do your labs & homeworks

Labs are typically due Friday nights

In general, for our section, future homeworks and projects will be due Monday nights at midnight:

- HW1 Monday 2/15
- HW2 Monday 2/22
- HW3 Monday 3/1
- HW4 Monday 3 / 8
- HW5 Monday 3/29
- HW6 Monday 4/19

- Project 1 Monday 4/12
- Project 2 Monday 5/3
- Project 3 Wednesday 5/12 (last class day)

These are tentative dates!

Quick Review from Monday:

Types: int, string, float, bool

Variables: symbolic name; has a type; can be changed

- Names can consist of upper and lower case letters, digits, and underscores
- Cannot start with a digit
- CMSC 201 convention: use snake_case for long names, NOT camelCase

Constants: Python doesn’t support; we fake by using ALL CAPS variable names

Literals: use this value, exactly

Input and Output (I/O)

Python’s default is to get all input from the keyboard, and put all output onto the screen

“Print” puts output onto the screen

- You can print variables, literals, or expressions that need to be evaluated
- You can print any type we’ve covered so far

Formatting output:

Formatting Output

Python has a whole *bunch* of different ways to print out results

- There’s always an “improvement” - I’ve taught different things every semester I’ve taught this course

We don’t make it a huge part of this course because it’s not fundamental “computer science” but it is a nice skill to have

This won’t be on the exam!!!

The basic “Print” statement

Variables and literals

Separate values with commas or plus signs

- Commas means a space is inserted between values
- Plus signs means run them together with no space

Print statement ends with a newline (\n) unless you explicitly tell it otherwise

Inserting line breaks

The Python “newline” character is \n. When Python encounters “\n” it prints a new line.

```
print(“This will result in one line”, “ being printed”)
print(“This will result in two lines”, “\n”,  “ being printed”)
```

By default, Python prints a new line at the end of every print statement

```
print(“This will result in two lines”)
print(“being printed”)
```

If you don’t want a new line, you can suppress it by using an “end” value

```
print(“This will result in one line”, end=” “)
print(“being printed”)
```

How do you print out a newline character?

Escaping - using \

print(“\\n”)

The tab character is \t. How do you print that out? print(“\\t”)

To print a single quote, print(‘\’’). A double quote is print(‘\”’)

Formatting printed output

Default field lengths when printing:

String: Python takes exactly as many spaces as there are characters in the string

Int: Python takes exactly as many spaces as digits in the integer

Float: Python prints everything to the left of the decimal point, and up to 16 digits after the decimal point

Boolean: Python takes four spaces for True and five spaces for False

f-strings

Introduced in Python version 3.6

- The “hot new toy” of Python programming
- Similar to str.format, but you can put any Python expression into your statement
- Start with ‘f’ or ‘F’
    - Then include anything you want in a string
    - Variable names get put in curly braces. They will be evaluated at runtime (when the statement is executed) and the actual value will be printed

Multiline f-strings are permitted

```
>>> profession = "comedian"
>>> affiliation = "Monty Python"
>>> message = (
...     f"Hi {name}. "
...     f"You are a {profession}. "
...     f"You were in {affiliation}."
... )
>>> message
  'Hi Eric. You are a comedian. You were in Monty Python.'
```

Setting field sizes

After the variable name or literal value, put a colon (:) and then the minimum number of spaces you want to take up

- Python will ALWAYS expand the field to include the entire string, the entire integer, the entire Boolean, and all the numbers to the left of the decimal point in a float
- For floating point values, the specification is a.b where a is the minimum total number of spaces and b is the maximum number of places to the right of the decimal that will be printed
    - The a value is optional

Let’s spend a lot of time on examples

F-strings: pay attention to

- Justification: if the field size is bigger than the value to be printed:
    - Strings are left-justified
    - Floats and ints are right-justified
    -

Input: the ‘Input’ Statement

The input statement has the syntax:

```
  var_name = input(“Prompt to be printed”)
```

Whatever the prompt is will be printed; then the program will stop until the user finishes typing in the input

Note: the result of an input statement is ***always*** a string

- If you want it to be something else, you need to cast it (convert it) to a different type

A few notes on the input statement

- The prompt can be a literal or a variable
- But it must be a single string
    - You can’t print multiple strings in a single “input” statement
- You can only bring in one value per input statement
    - If you need multiple values, use multiple input statements
    - Later on in the semester we’ll cover how to bring in multiple values in a single statement

Examples