Python Literals, Variables, CONSTANTS, Conditionals & I/O

CMSC 201 Section 60
September 9, 2024

Administrative Notes

Homework 1 is out

- Due for our section next Monday night, September 16 before midnight
- Please don’t wait until next Monday to work on it

Submitting homeworks:

- The submit strings are case-sensitive
- Submit early, submit often:
  - You can submit an assignment multiple times; only the last is graded
  - This ensures you don’t get a 0 on an assignment even if something prevents you from submitting it at the last minute

Stuff you need to understand after today

- Type: int, float, boolean, string
- Literal: exactly this
  - String literal: either single quotes or double quotes, but they have to match
  - Int or float literal: no “magic numbers” in your code!
- Constant: a value that doesn’t change in your program
  - Python doesn’t really support them; use a variable name in ALL_CAPS
- Variable: a symbolic name for a location in memory that will hold a value of a specific type
  - Names: can contain letters (upper and lower case), digits, and underscores
  - Cannot start with a digit
  - CMSC 201 standard: use snake_case for long names
  - Remember how the symbol table works

Input and Output (I/O) - *you need this for HW 1*

- For now, all input is from the keyboard, and all output goes to the screen
- Input: use the “input” statement
  - Var_name = input(“Print your prompt here”)
- Some notes on input:
  - When an input() statement is executed, the program stops until the user provides requested input and hits “enter” (types a newline character)
    - What if the user enters the wrong data? That’s what error checking is for
  - Input always returns a single string. If you want another type, you have to cast it.
    - Var_name = int(input(“Enter your age”))
    - Gpa = float(input(“Enter your GPA”))
    - Warning: if the user enters something that can’t be cast, it’s an error and your program may crash
  - The prompt can ONLY be a single string
    - Trying to have the prompt be more complex causes errors

Output: the print statement

- Print output to the screen using the print() statement
- You can print:
  - Literals
  - Variables
  - Constants
  - Of any type!!!
  - Results formed by combining multiple values
    - print(“Your GPA is”, gpa)
    - print(“Your name is “, x, “ and your GPA is “, gpa)

After a print() statement is executed

By default, all print statements end with a newline being printed

The cursor moves to the beginning of a new line

If you don’t want that, you tell Python how to end the print() statement

```
print(“Here is an example”, end =” “)

print(“of printing a single line with three”, end = “ “)

print(“print statements”)
```

The end value can be any valid string

Conditionals

Code execution:

1. Sequential - execute each line of code, exactly once - no more, no less - then move on to the next line of code
2. Conditional - execute a line of code, once, IF and ONLY IF some condition (or set of conditions) is true. Otherwise, skip this line and do not execute it - all conditions are Boolean - True or False
3. Iterative - execute a line of code, or group of lines of code, multiple times

Conditionals

if, else, and elif

elif is short for “else if” It requires typing four characters instead of 7, and programmers tend to like shortcuts and optimization

Note: Python 3.10 introduces/makes robust the “match…case…” structure to handle conditionals. We will NOT use that in this class. Thus, don’t use Python 3.10.

A look-ahead to Wednesday:

The examples here make use of ==, >, >=, <, <=, and !=

These are called *operators* in Python. Specifically, they are *comparison operators*.

There will be a lot more about them on Wednesday. For now, just understand them in the context of standard arithmetic.

Three cases:

1. One option: If a condition is true, do something. Otherwise, do nothing. “If” statement
2. Two options: If a condition is true, do something, Otherwise, do something else. “If….else…” statement
3. More than two options: If a condition is true, do something. Otherwise, check to see if another condition is true; do something else. Otherwise, keep checking conditions until we find one that’s true or we just give up. “If...elif...elif...else…” statement

“If” statement

if {some boolean condition is true}:

    print(“Yay it is true”)

Notes on this:

- Typically boolean conditions are True or False.
  - A trick: any integer value other than 0 is regarded as “True” and any value that’s equivalent to 0 is regarded as False
  - An empty string is False; any non-empty string is True
-
```
if 5:

    print (“5 is true”)
```

Further notes

The boolean condition is everything between “if” and the colon : A colon terminates the condition. It can be as simple or as complex as you want

Indentation matters!! To the python, white space - either tabs or spaces - indicates what’s in the code to be executed. You only have to indent one space, but I’m a believer that you should indent with tabs.

```
If x == 6:
    print (“X is 6”)
    print (“and we’re done”)
```

If x is really equal to 5, this doesn’t print out anything.
This still prints “and we’re done” because that’s not part of the conditional

```
if x == 6:
  print (“X is 6”)
print (“and we’re done”)
```

“if...else …” statement

Ask a student for her major. If she’s a CMSC major, print out “smart choice.” Otherwise print out “there is still time”

```
major = input(“please enter your current major”)

if major == “CMSC”:

    print(“smart choice”)

else:

    print(“there is still time”)
```

Remember that input returns a string, so this comparison is valid

Colon after else, as well!

Indent the code that’s part of the “else” block

“if...else…”

- There must always be at least one line of code under the “if” statement
- You don’t have to have an “else” part, but if you do have an “else” there must be at least one line of code under it.

“if...elif…else”

Input returns a string. This turns it into an integer

Ask a student her age. If it’s less than 18, tell her she’s a minor and faces some restrictions. If it’s between 18 and 21, tell her she’s an adult but still has some limitations. If she’s over 21, tell her she’s legally an adult.

This is called ‘pseudocode’ - an English-language description of what you want to do

```
age = int(input(“Please enter your current age in years.”))
if age < 18:
    print(“Sorry but you are a minor”)
    print(“there are a lot of things you cannot do on your own”)
elif age < 21:
    print(“you are an adult but there are still some things you can’t do”)
else:
    print(“congratulations you are legally an adult”)
```

Notes on “if...elif...else”

- Only one case will have its code executed when the program runs; the rest of the code will be skipped
- There doesn’t have to be an ending “else” condition. If there’s not, we just do nothing and skip all the code
- There must be at least one statement - one line of code - in each case
- Keep your conditions simple. In the previous example, we only got to the ‘elif’ when the age was >= 18. So we didn’t have to put that in the condition - that is, no need to say `elif age >= 18 and age < 21:` We already know that first part is true.