Writing Useful Python Programs

September 9, 2020

Today’s topics

Getting a Python program running:

- Variables
- Literals
- Input and output
- Conditionals

SCP: moving code between your laptop and the GL system

Types

Variables and constants have defined types that determine what operations you can perform on them.

For now, we'll deal with int, float, string, and boolean

int - integer - whole numbers. You can do math on them

float - floating point numbers - integer part and decimal part - you can do slightly different types of math on them

Hello, world

string - zero or more characters treated as a whole

Boolean - have the value True or False (note case sensitivity)

Variables

Assignment is done using the equals sign =

```
Number_of_students = 20

Grade_point = 3.862

Error_message = “Sorry you must enter an integer between 1 and 4 inclusive”

Is_integer = True
```

Using variables

Declaring variables: unlike some other languages, you do not pre-declare a variable in Python. When you use a variable, that declares it

- The python interpreter recognizes that you have just declared a new variable and allocates a memory location to store its value

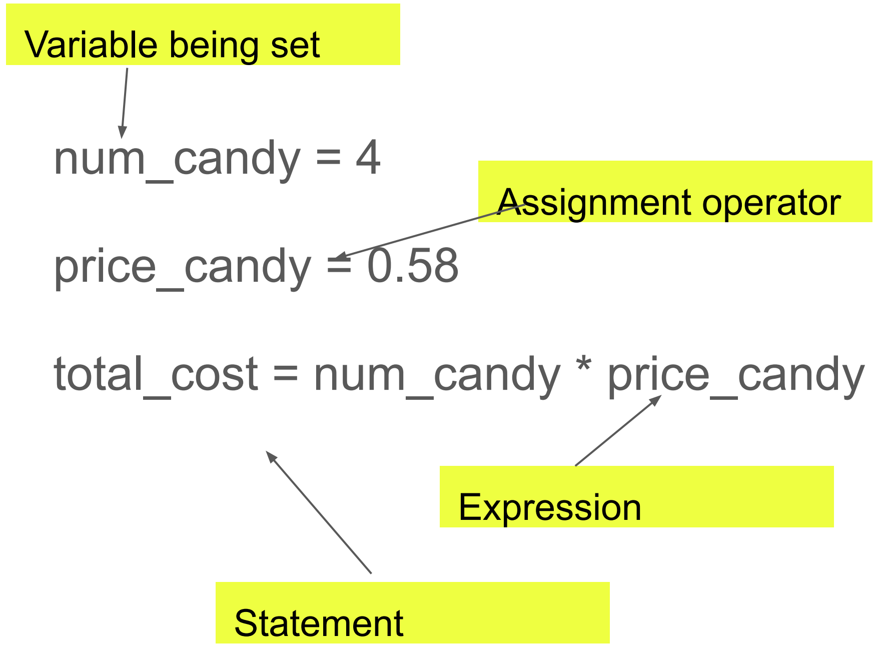
But you do have to initialize a variable before trying to use it

- Initialize means assign it a value

A very common error is to try to use an uninitialized variable - use it before it has been assigned a value

Expressions

A way of calculating a new value to assign to a variable



```
num_candy = 4

price_candy = 0.58

total_cost = num_candy * price_candy
```

“Sides” of the assignment operator

“Left hand side” and “right hand side”

- Left hand side is before - to the left of - the equals sign. This is where the value of the expression will be store
- Right hand side is after the equal sign. Evaluate everything to the right of the equal sign, and the store that value in the variable on the left

```
num_candy = 4 * 12

4 * 12 = num_candy  X not legal
```

Operators

Special symbols that perform defined operations:

- Mathematical
- Comparison/Relational
- Assignment
- Boolean/Logical/Conditional

Mathematical Operators

+ - * / // % **

+ - addition; works as you would expect

- Subtraction; works as you would expect

* multiplication

/ floating point division - results in a float number

// Integer division. Only valid if you have two integers; produces an integer.

% modulo

** exponentiation

Some examples:

5/3

5//3

5%3

Comparison Operators

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Be careful that you don’t confuse = and ==

= is the assignment operator; it sets the value on the right to the variable on the left

== is a test to see if what’s on the left is equal to what’s on the right

Assignment operators

=

+=

* =

-=

/=

Some examples

num_candy = num_candy + 1

num_candy += 1

num_candy *= 2

num_candy -= 2

num_candy /= 2

Boolean operators

and

or

not

A boolean is either True or Fals

Boolean operators take Boolean values, combine them and yield a single Boolean value

```
 9 > 8 and 5 < 9

num_candy != 0 or choice = ‘yes’
```

Practice

Set the variable meal_bill to 30 dollars and 51 cents. Then calculate a 20 percent tip on meal_bill, and print out the total amount due

Calculate a GPA. Assign values to the number of hours earned and total quality points. Calculate the GPA as number of quality points divided by number of hours earned. Print the GPA.

Input and Output

Initially, we will read in all input from the keyboard and print all results to the screen.

- We’ll cover reading from files and printing to files later

Input is done with the “input” statement; output is done with the “print” statement

```
print (3 + 4)
```

```
print (3, 4, 3+ 4)
```

```
print()
```

```
print (“the answer is”, 3 + 4)
```

Examples

```
a = 10

b = a * 5

c = “your result is:’

print(c, b)
```

```
a = 10

b = a

a = 3

print(b)
```

Input from the keyboard

If you expect the user to input a meaningful value, you have to tell him or her what you want. The input statement lets you enter a string to be printed out as a prompt

```
user_num = input(“please enter your student number:”)

print(user_num)
```

When the input statement is executed by the interpreter, the program stops until the user has entered the required data

In python 3, input is always entered as a string. Even if it’s supposed to be an integer or a floating point number

If the user types 10, you get the string “10” NOT the integer 10.

If the user types 42.75, you get the string “42.75” NOT the floating point number 42.75

You can’t do math on a string!!!

Changing the type of a value

If a variable’s value is the wrong type, you can change its type by casting it to the type you want

To change a string to an integer, use int(the value)

```
  age = int(input(“enter your age in years as a whole number:”))
```

To change a string to a floating point number, use float()

```
  gpa = float(input(“enter your GPA to 3 decimal places: “)
```

Pseudocode

A natural language description of what a section of a program is supposed to be doing

Used during program design

- Write the pseudocode first
- Then translate it into the code

An Example

Pseudocode

```
To calculate a GPA:  Assign values to the number of hours earned and total quality points.
Calculate the GPA as number of quality points divided by number of hours earned. Print the GPA.
```

Code:

```
hours = int(input(“Enter the number of credit hours you have”))

q_p = float(input(“Enter the number of quality points you have earned”))

gpa = q_p/hours
```

So, what do you need to know how to do?

1. Write pseudocode. It will help you explain what you are trying to accomplish with your program.
2. Take someone’s pseudocode, and produce Python code that accurately implements the pseudocode

Moving code between your laptop and gl.umbc.edu

Your code for labs, homeworks and projects has to run on gl.umbc.edu

- Testing is done on that system

You can choose to write your code directly on gl.umbc.edu if you want to

- Use emacs

It might be easier to write and debug it on your own computer using pycharm

Then you move it to gl using the Secure Copy (SCP) protocol

Make sure you run it on gl to make sure it works properly!!

Secure Copy (SCP)

On a mac: command line in the terminal window

For Windows boxes: use WinSCP

    https://winscp.net/eng/index.php

Some examples of how it works

Homework 1

Now let’s spend some time talking about homework #1