Python Programming and Using the UMBC GL system

September 2, 2020

Overview

Programming in Python

- Using GL
- On your computer

Some programming concepts

Some programming examples

Using gl.umbc.edu to do your homework

Windows users:

- You will use a tool called putty to remotely log in to gl.umbc.edu
- Download putty
- https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html (going to www.putty.org will redirect you to this site)
- Then see https://www.youtube.com/watch?v=DH3of3tuBxA
-

Mac users:

- You can directly SSH into gl.umbc.edu from a terminal window
- See https://www.youtube.com/watch?v=UTU-CTH_xLw
-

About gl.umbc.edu

It's a Linux system

It uses the bash shell for command-line input

- A "shell" is a command line interpreter
  - There are a number of them - c shell; sh; korn shell; bourne shell
  - bash - the "bourne-again shell" - is pretty much the standard for Linux & Mac these days

Emacs is the file editor provided

Python3 is the command used to execute python programs

When you're logged in to gl, everything is executing on that remote server. The only thing happening on your system is the input & output through your Terminal or PuTTy window

Watch (https://www.youtube.com/watch?v=r0otsJZ1ry0) before lab if you have a chance; it covers the basics

- It's less than 4 minutes long; you can watch it during lab if necessary

Computer Languages (*aka, “Why Python?”*)

A computer language defines the syntax and semantics you use to tell the computer what you want it to do

- Semantics: what you mean
- Syntax: how you say it

Consider all languages to be equivalent - if you can write it in one language, you can write it in another

- It may be easier or harder

Languages you might have seen if you’ve programmed before

Java - general purpose language

Javascript - used for Web programming; no relation to Java

C++ - general purpose, object-oriented - CMSC 202 is taught using this language - very fast

Python - general purpose language; high level; easy to understand

- Commonly used with Artificial Intelligence & Machine Learning
- Commonly used with Security tools
- Easy to extend; tons of free software to import and use

Python

Interpreted language; NOT compiled

- Executed line-by-line
- Not the fastest; but these days most people don’t worry about speed
    - Just get more hardware or use a cloud

Whitespace is important - indentation matters!

Running Python on your computer

You’ll do your assignments on a UMBC server, but you should know how to use Python on your own computer if you have one

First: go to www.python.org

Mouse over “Downloads” and select the type of operating system you have - Windows, Mac, or other type

Make sure you get the latest version of Python - should be 3.8.5

Download it and install it

To run it from a terminal/command shell

Bring up a command shell on Windows or a terminal on Mac

When you see the prompt, type “python3”

- If it doesn’t run, check to see where you installed it. If you didn’t install it in the default location you may have to add a directory to your path
- Mac users: Mac comes with an old version of Python pre-installed. Typing “Python3” instead of just “python” means you’ll be running the correct version.

Okay, now you’re in a python console. You can type in a line of Python code and it will be executed

Your first Python program

By tradition, whenever you learn a new language, you start by running a program that prints out “Hello, world” to prove that this thing works.

Let’s do that.

```
# Our first python program
# Print “hello, world” on the screen
print(“hello, world”)
```

When you’re finished, type exit() to leave the python3 interpreter.

When you do exit, whatever you’ve done is gone, unless you saved it somewhere. What if you want to write a program, save it as a file, and then run the entire program?

- By tradition, all python programs are stored with a .py extension. So let’s create a file, save it, and then run it

Just for comparison: the same program in C++

```
#include <iostream>

cout << “Hello, World”
```

There’s gotta be an easier way!

You have to create/edit/save a file some way, then run the file.

If you want to change something - fix an error - you have to go back to the editor

There’s no easy way to fix problems

There IS an easier way - an Integrated Development Environment (IDE)

- Contains all the tools in one package - editor, python interpreter, debugger, file manager
- NOT a mandatory part of this class, but if you’re going to do much programming on your own laptop, it’s good to have

Pycharm

In this class, I will do many of the coding examples using the Pycharm IDE

NOTE: the TAs are NOT responsible for teaching you Pycharm or debugging problems with it, but it’s pretty intuitive

Get the current free (“Community”) version from
https://www.jetbrains.com/pycharm/download/other.html

Do NOT get the professional version; it’s not worth the money for what you’re going to do in this class

Okay, now to write some Python

The first program you write in any language is “hello, world”

The program does just what you think: it prints hello, world on the screen

So let’s go.

A few terms you should know

Comment

Variable

Literal

Constant

Reserved Word

Type

Comments

Notes you write for yourself and other humans to explain what it is you’re trying to do

Mark a comment in Python with a #

Anything after the # on a line is a comment and is not executed by the python interpreter

#this is a comment

print (“hello, world”) # this is also a comment

Variables

A symbolic name associated with a value

The value may change during the program’s execution

When you use a variable in python, the python interpreter reserves a location in memory and associates the variable’s name with that location. Then, any values assigned to that variable are stored to and read from that location

We’ll go over a few examples:

The syntax of variable names

In python, variable names MUST start with either a letter or the underscore _ character

- You cannot start a variable name with a digit

Variable names can only contain uppercase letters, lower case letters, digits, and underscores. NO OTHER special characters

Variable names are case sensitive. Height is not the same variable as HEIGHT

Course coding standards enforce additional rules

Exercise: Variable names

Is each of the following a legal variable name in Python? Why or why not?

1spam

raise1

Spam_and_Eggs

EXIT_CODE!!

Literals

A literal is an explicit value that is to be taken exactly as it is written in the program.

“Hello, world” is a literal (it’s a string literal, to foreshadow)

3   is a literal

3.14159 is a literal

2.71828 is a literal

Constants

A constant is a symbolic name associated with a value that *WILL NOT CHANGE* during the execution of a program

You use a constant to represent a “magic number” to make code more readable

```
print(3.14159)   # or
PI = 3.14159
print(PI)
```

Python does NOT provide built in support for constants
We use ALL CAPS to represent a constant

Reserved Words

`False`	`await`	`else`	`import`	`pass`
`None`	`break`	`except`	`in`	`raise`
`True`	`class`	`finally`	`is`	`return`
`and`	`continue`	`for`	`lambda`	`try`
`as`	`def`	`from`	`nonlocal`	`while`
`assert`	`del`	`global`	`not`	`with`
`async`	`elif`	`if`	`or`	`yield`

These words have special defined semantics in Python - you can't use them as variable names or for any other purpose

These are Case-Sensitive.

False is not the same as false

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

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

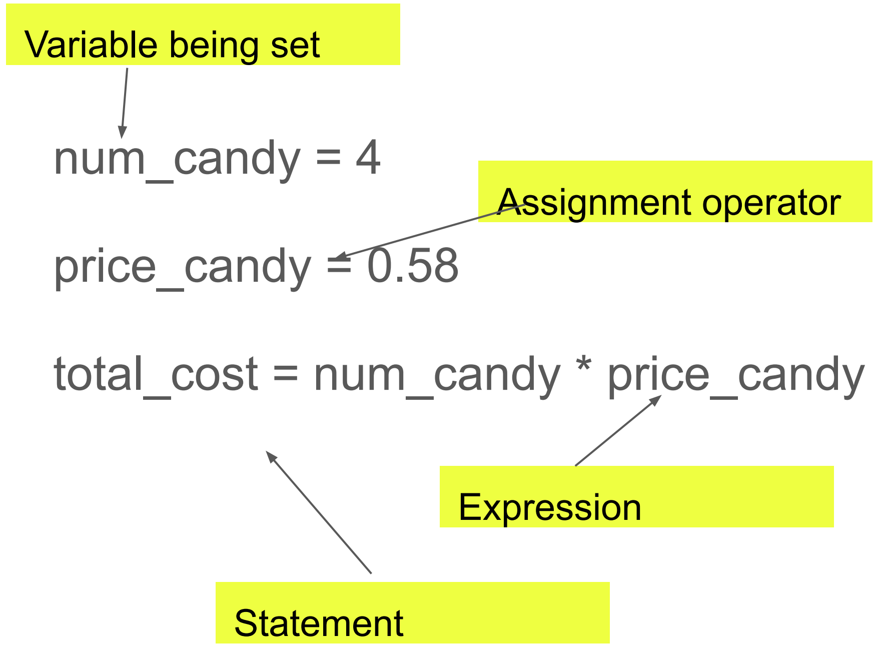
But you do have to initialize a variable before trying to use it

- 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: “)
```