

# CMSC 201

## Computer Science I for Majors

### Lecture 02 – Intro to Python

# Last Class We Covered

- Syllabus
  - Grading scheme
  - Academic Integrity Policy
    - (Collaboration Policy)
- Getting Help
  - Office hours
- Programming Mindset
  - “Failure” (isn’t really failure)

# Any Questions from Last Time?

# Today's Objectives

- To start learning Python
- To learn about variables
  - How to use them
  - Different types
- To learn how to use input and output
  - To do interesting things with our program
- Written programs vs Python interpreter

# Variables

# Python

- Python is a widely used language
  - General purpose
  - High-level language
- Emphasizes code readability
  - More streamlined than some other languages

# “Hello World!”

- In Python:

```
print("Hello World!")
```

- In the C++ programming language:

```
#include <iostream>
int main() {
    std::cout << "Hello World!\n";
}
```

# Elements of a Program

- Identifiers
  - Variables
  - Functions (later in the semester)
- Expressions
  - Code that manipulates or evaluates identifiers
- Literals
- Operators



# What Is a Variable?

- Something that holds a value
  - Can change (unlimited number of times)
- Similar to variables in math
- In simple terms, a variable is a “box” that you can put stuff in



# Rules for Naming Variables

- Variable names can contain:
  - Uppercase letters (**A–Z**)
  - Lowercase letters (**a–z**)
  - Numbers (**0–9**)
  - Underscores (**\_**)
- Variables can't contain:
  - Special characters like **\$, #, &, ^, ), (, @**



# More Rules for Naming Variables

- Variables can be any length
  - **x**
  - **IsKanyeRunningForPresidentIn2020**
  - **myName**
- Variables cannot start with a digit
  - **2cool4school** is not a valid variable
  - **cool4school** is a valid variable

# Variables and Keywords

- Keywords are “reserved” words in Python

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

- Variables cannot be keywords
  - `or` is not a valid variable name
  - `orange` is an acceptable variable name

# Exercise: Variables

- Are the following legal or illegal in Python?

`1spam`

`raise1`

`Spam_and_Eggs`

`EXIT_CODE`

# Exercise: Variables

- Are the following legal or illegal in Python?

`1spam`

**No – Illegal!**

`raise1`

**Yes – legal!**

`Spam_and_Eggs`

**Yes – legal!**

`EXIT_CODE`

**Yes – legal!**

# Exercise: Variables

- Are the following legal or illegal in Python?

`Spam_and_Eggs`

**Yes – legal!**

**But it doesn't follow  
our coding standards!**

`spamAndEggs` or  
`spam_and_eggs`

# Using Variables in Python

- You create a variable as soon as you declare it
- You also need to initialize it before using it
  - Use the assignment operator (equal sign)

assignment operator

`mascotUMBC` **=** `"dog"`

`newStudents` **=** `1772`

`dogsAreGood` **=** `True`

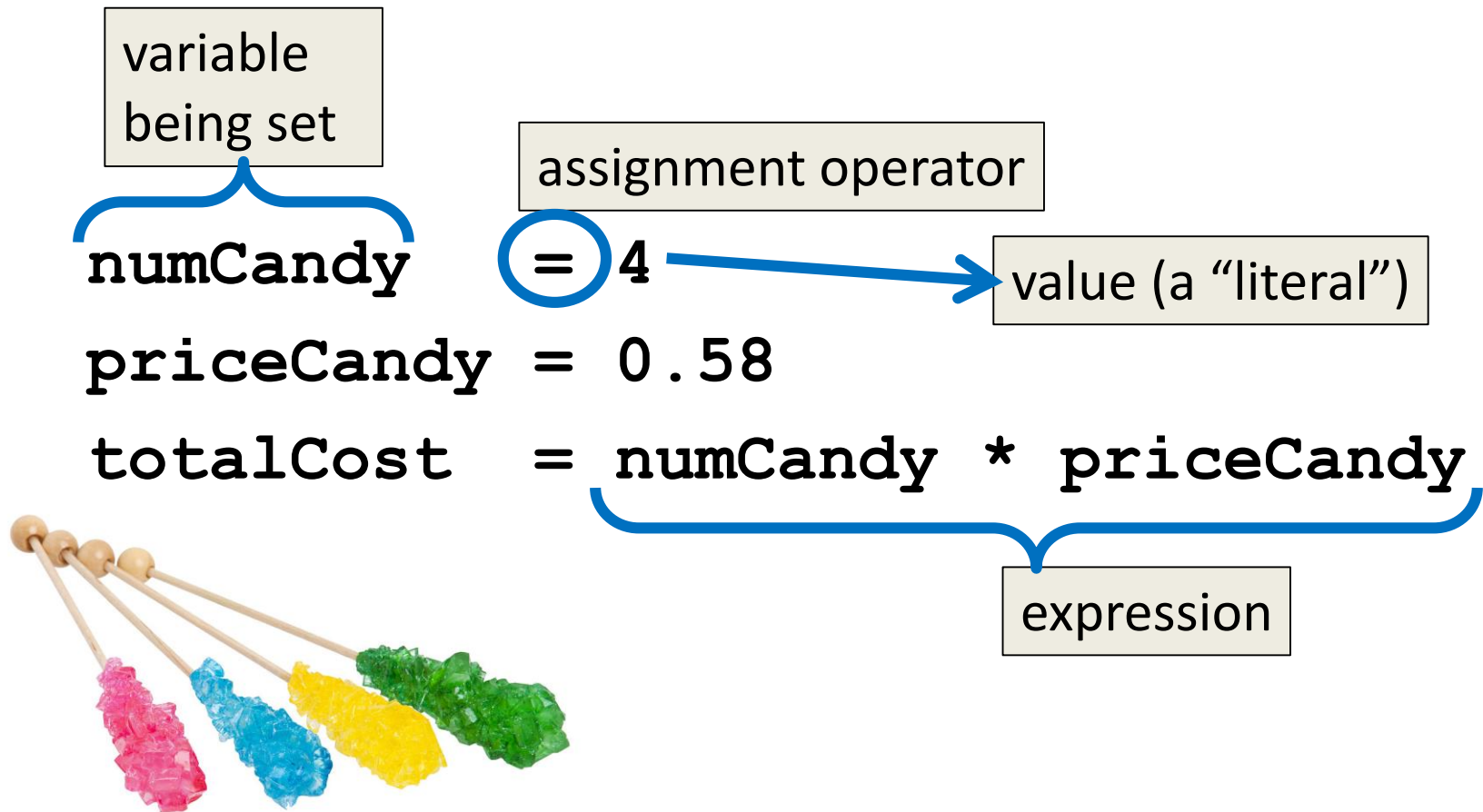


# Expressions

# Expressions

- Programs manipulate data
  - Allows us to do interesting things
- Expressions calculate new data values
- Use assignment operator to set new value

## Expressions Example



# Common Mistake

- Many new programmers mix up the left and right hand sides of the assignment operator
  - Variable being set must be on the *left*
  - Expression is on the *right*
  - Evaluate the expression first, then assign the value

`numCandy = 4 + 1`



`4 + 1 = numCandy`



# Variable Types

- There are many different kinds of variables!
  - Numbers
    - Whole numbers (Integers)
    - Decimals (Floats)
  - Booleans (**T**ru**e** and **F**als**e**)
  - Strings (collections of characters)

# Variables Types: Examples

```
aString      = "Hello class"
```

```
float_1     = 1.12
```

```
myBool      = True
```

```
anInteger   = 7
```

```
dogName     = "Ms. Wuffington"
```

```
classCode   = 201
```

# Variable Usage

- Variables are designed for storing information
- Any piece of information your program uses or records must be stored in a variable
  - Python doesn't have a “short term memory,” so everything needs to be written down for it

# Literals and Operators



# Literals

- Literals in Python are values you use “literally”
  - Can be assigned to a variable or not
- For example:
  - 2 is an integer literal
  - “Hello” is a string literal
  - 4.0 is a float literal
  - False is a Boolean literal

# Using Literals

- The expression below assigns the string literal “CMSC” to a variable called major  
`major = "CMSC"`
- The expression below prints the integer literal 50 without assigning it to a variable  
`print (50)`

# Operators

- Operators are special symbols that allow Python to perform different operations
- There are many types of operators
  - Mathematical
  - Comparison
  - Assignment
  - Logical

# Operator Types

- We won't cover all the types in detail today, but here are some simple examples

- Mathematical

+          -          \*          /          %

- Comparison

<          <=          !=          >=          ==

- Assignment

=          +=          \*=

we'll cover the  
"weird" ones later

# Practice Exercises

- Print the value of the variable `myDog`
  - Remember to assign a value to `myDog` first!
- Set a value for a variable called `bill`, and calculate and print the 15% tip for that `bill`
- Create your own expression using at least two variables, and print out the result

# Input and Output

# Output

- Output is text that is printed to the screen
  - So the user can see it
- The command for this is **print**
  - Use the keyword “**print**” and put what you want to be displayed in parentheses after it

# Output Example

```
print (3 + 4)
print (3, 4, "3 + 4", 3 + 4)
print ()
print ("The answer is", 3 + 4)
```

7

3 4 3 + 4 7

What does this  
output to the screen?

The answer is 7



# Output Exercise 1

- What will the following code snippet print?

```
a = 10
```

```
b = a * 5
```

```
c = "Your result is:"
```

```
print(c, b)
```

```
Your result is: 50
```

## Output Exercise 2

- What will the following code snippet print?

```
a = 10
```

```
b = a
```

```
a = 3
```

```
print(b)
```

There are a few possible options for what this could do! Any guesses?

10

# Output Exercise 2 Explanation

- Why does it print out 10?
- When you set one variable equal to another, they don't become linked!
  - They are separate copies of a value
- After **b** is set to 10, it no longer has anything else to do with **a**



# Output Exercise 2 Explanation

→ `a = 10`

`b = a`

`a = 3`

`print(b)`



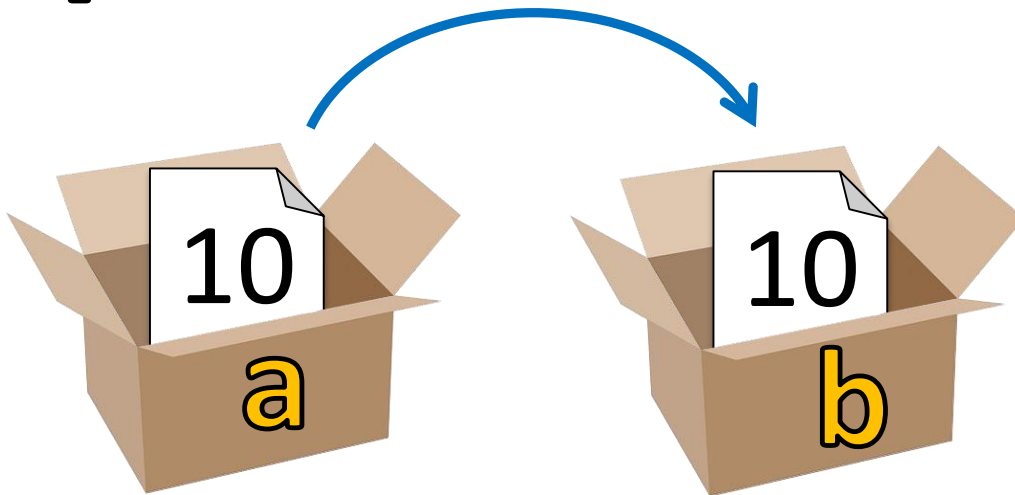
# Output Exercise 2 Explanation

```
a = 10
```

```
→ b = a
```

```
a = 3
```

```
print(b)
```



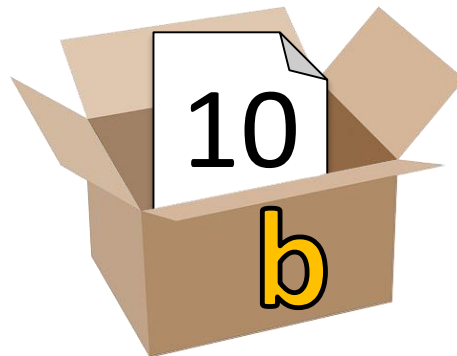
# Output Exercise 2 Explanation

`a = 10`

`b = a`

 `a = 3`

`print(b)`



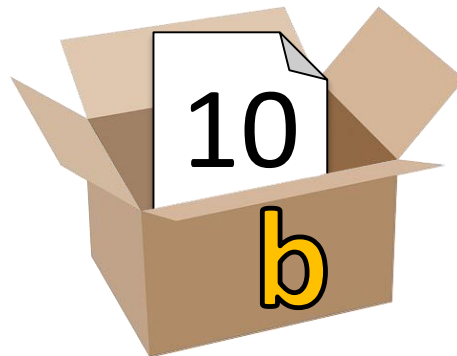
# Output Exercise 2 Explanation

`a = 10`

`b = a`

`a = 3`

 `print(b)`



 **output: 10**

# Input

- Input is information we get from the user
  - We must tell them what we want first

```
userNum = input("Please enter a number: ")  
print(userNum)
```

- The input and output will look like this:

```
Please enter a number: 22  
22
```



# How Input Works

```
userNum = input("Please enter a number: ")
```

- Takes the text the user entered and stores it
  - In the variable named `userNum`

- You can do this as many times as you like!

```
userNum = input("Enter another number: ")
```

```
userNum2 = input("Enter a new number: ")
```

```
userAge = input("Please enter your age: ")
```

# Input as a String

- Everything that is stored via `input()` will come through in the form of a string
- There is a difference between `"10"` and `10`
  - `"10"` is a string containing two characters
  - `10` is understood by Python as a number

# Converting from String

- To turn an input string into a number, you can do the following:

```
aNum = input("Enter a number: ")
```

```
aNum = int(aNum)
```

- “int” stands for “integer” (a whole number)
- You can also do it in one line:

```
aNum = int(input("Enter a number: "))
```

# Converting from String

- We can cast to other data types as well

```
gpa = float(input("Enter GPA: "))
```

- Do you think the string `"1,024"` will work if we try to cast it as an integer? Why?
- It won't work
  - The comma character isn't a number

## Written Programs vs Python Interpreter

# We Started Python Today!

- Two ways to use Python

We will write programs for assignments

– You can write a program as a series of instructions in a file and then execute it

Use the interpreter to help you test things

– You can also test simple Python commands in the Python interpreter

# Written Programs

- Create, write, and save a Python file (.py)
- File is run via the command line  
`python3 myProgram.py`
- File must be complete to run correctly
- Program cannot be edited on the fly
  - Must be exited, file re-opened, changes made, file saved and closed, and then re-run the program

# Python Interpreter

- The “interactive” interpreter evaluates each individual line of code as it’s typed in
- Type “**python3**” to launch the interpreter

>>> is where the user types their code

```
>>> print("Hello")
```

```
Hello
```

```
>>> 4 + 7
```

```
11
```

```
>>>
```

lines without a “>>>” are Python’s response



Time For...

**LIVECODING!!!**

# Daily emacs Shortcut

- **CTRL+X, CTRL+S**
  - Saves the file and stays in emacs
  - Allows you to keep editing the file
  
- **CTRL+X, CTRL+C**
  - Closes emacs, does not automatically save the file
  - Will prompt you to save if changes were made

# Announcements

- Your discussions (Labs) start next week!
  - Go to your scheduled location and time
  - Pre Lab quiz will be posted and announced on BB
- Lab 1 is due Sunday, Sept 9th at 8:59:59 PM
  - In-person labs start the week of September 10th
- HW0 is also due on Sept 9th at 8:59:59 PM

# Image Sources

- Cardboard box:
  - <https://pixabay.com/p-220256/>
- No cursing sign (adapted from):
  - <https://www.flickr.com/photos/rtgregory/1332596877>
- Rock candy:
  - <https://commons.wikimedia.org/wiki/File:Rock-Candy-Sticks.jpg>
- Broken chain:
  - <https://pixabay.com/p-297842/>