python

Introduction To Python

Christopher Cox

Interpreted or Compiled?

- Python is an interpreted language. This means that instead of having to be compiled before running, the Python interpreter reads the files in their plain text form.
- One major advantage of being an interpreted language is the portability. You can program it in once place, and run it on any computer that supports Python.
- One slight disadvantage is performance. Since the interpreter has to actively read through your file and interpret it, it's not as fast as a language like C.

This isn't Python

```
// Begin looping through the questions array
foreach ($questions as $qn => $qData) {

    // Print question
    echo $counter++ . ": " . $qData['text'] . "<br />\n";

    // Print out the options
    foreach ($qData['choices'] as $letter => $choice) {

        $fChoice = htmlspecialchars("$letter. $choice", ENT_SUBSTITUTE);

        echo "<input type=\"radio\" name=\"$qn\" value=\"$letter\">";
        echo "$fChoice</input><br />\n";
```

Installing Python and Getting Practice Material

- You can download the Python installer from Python's website at: https://www.python.org/downloads/ (I highly recommend v3.6.5 or later)
- You can download a custom Python script I made for this lecture here: https://goo.gl/qS7n3X
- For the sake of the lecture, I recommend using the stripped version.
- The normal version has a lot of notes in the code to explain what each section is doing.

Variables

- Variables are imperative to almost every Python program out there.
- They are used to store data temporarily in your program.
- An example of how to define a basic variable named 'variableName' holding an integer value of 200 is done via this syntax:

```
# Variable definition example
variableName = 200;
```

- Variable should have meaningful names that represent what they are being used for.
- Python uses a method called “Duck Typing” which means Python is not a strongly typed language like C or Java. The name duck typing comes from the phrase: "If it looks like a duck and quacks like a duck, it's a duck".

Basic Usage of Functions

- Python comes with many built-in functions to make a coder's life easier.
- The first built-in function we encounter is called print(). (if you see parenthesis after a name in Python, it is usually a function call/definition)
- The print function does exactly what its name implies, it outputs the data you pass to it.

```
>>> print("Hello world!")
Hello world!
```

- The print function will be unable to understand what you are passing to it unless you surround the text in single or double quotes. By doing this, you are defining that text as a 'String', which Python can interpret.
- Any variables/literals you put inside the parenthesis of a function call are called arguments. Not every function takes arguments, however.

Passing Variables to Functions

- In the previous slide, I passed what is called a String Literal to the function. That works fine, but sometimes its better to store that string in a variable, then pass it to the function.

```
myMessage = "This is a message stored in a variable!";
print(myMessage)
```

- This is a much cleaner way to pass larger strings to the print function.
- A few important notes about variable definitions: The equals sign is called an assignment operator. When the interpreter sees the assignment operator, it executes everything on the RIGHT side first. This applies to function calls and math operations.

```
>>> result = 5 + 3
>>> print(result)
8
```

Getting User Input

- Python has another useful built-in function called input() which allows us to accept user input. When Python sees this function call, it pauses program execution and waits for the user to type someone and hit enter.

```
userInput = input("Please enter something: ")
```

- Once the input function has finished its job, it will return whatever the user typed. If we didn't store the input function's returned value into a variable, we wouldn't be able to use it anywhere else in the program.
- Now we can use that variable however we please. In this case I am printing it out using the print function. (The comma is used to pass more than one argument to a single function call. The print function combines these arguments together automatically and even adds a space between them.)

```
print("You typed:", userInput)
```

Lists

- Lists allow a programmer to store more than one value in a single variable.
- With basic lists, you can access each of these elements using an integer based index system.

```
someNumbers = [10, 20, 4, 12]
```

- This list has 4 elements in it. With Python, you can mix and match different data types in the same list, but that can cause headaches unless it is properly organized and maintained.

```
print(someNumbers[0])
```

- The syntax is simple to select an item in the list. You can also use this syntax to redefine parts of the list:

```
someNumbers[1] = 15
```

Booleans and Boolean Expressions

- Whenever you see the word boolean, you should immediately think of true or false.
- In Python, you can define Boolean literals as such (they are case sensitive)

```
pythonIsFun = True
```

- Boolean Expressions are used in any program that has decision making capabilities.
- Boolean Expressions are basically statements that the Python interpreter evaluates and spits out a True or False value.
- These values can then be used to control decision structures, loops etc.

Operator	What it means
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Operator	What it means	What it looks like
and	True if both are true	`x and y`
or	True if at least one is true	`x or y`
not	True only if false	`not x`

```
myExpression = (10 / 2) + 3 == (4 ** 2) / 2
```

```
myExpression = True and True or False
```

Basic Looping

- Python has a few different types of loops to work with. We will be focusing only on the while loop
- The while loop is controlled by a boolean expression which it checks before each loop execution. If the expression evaluates to True, then it loops again. If it evaluates to false, then the skips the loop body and continues executing the code below.
- Both of these code snippets produce the same result

```
print("I used CTRL + V here")
print("I used CTRL + V here")
print("I used CTRL + V here")
print("I used CTRL + V here")
print("I used CTRL + V here")
```

```
sentinalValue = 1

while (sentinalValue <= 5):
    print("I used a while loop here")
    sentinalValue += 1
```

(for more information on this, follow the link to my github at the beginning of the PP and select main.py)

Decision Structures

- The if statement is something you'll be using quite frequently in any language. As you can see, it takes a Boolean expression as an argument. If that expression evaluates as true, then the block of code immediately following the if statement runs. If it isn't true, then the else block runs. The else block is optional.

```
uInput = int(uInput)

if (not (uInput > 1 and uInput < 10)):

    print("You didn't enter a number between 1 and 10!")

else:

    specialNumber = 3

    if (specialNumber == uInput):
        print("You guessed the correct number!")
    else:
        print("Sorry! You guessed incorrectly. The number was:", specialNumber)
```

Conclusion

- That was a lot of information to take in! If you're feeling like you didn't understand something, try viewing my note-heavy version of all this code using this link: https://goo.gl/qS7n3X (select main.py)
- I will also have a download of this presentation at that link, too.
- Coding is a lot like drawing, so don't be afraid to color outside the lines a bit to test things out. Have fun with it!
- Python has a lot more functionality than what I have showed off in this powerpoint, so start exploring!