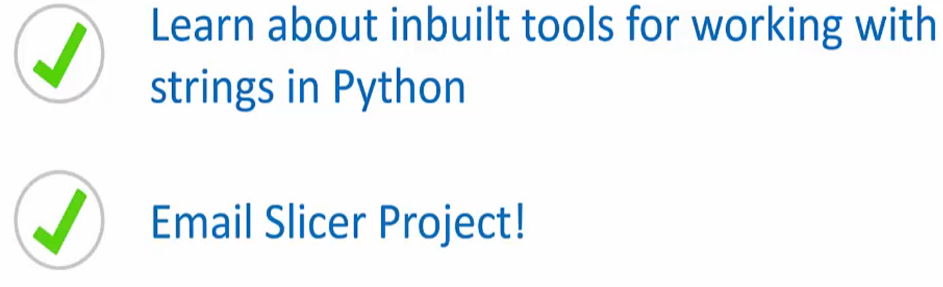
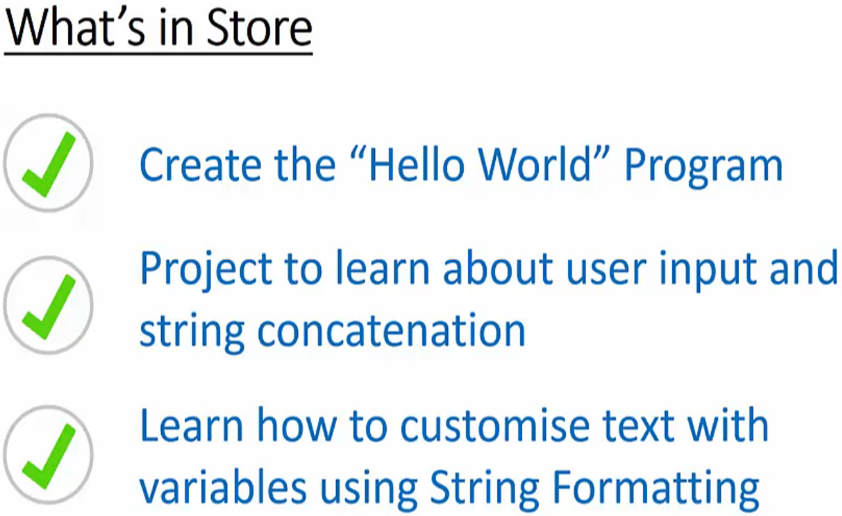
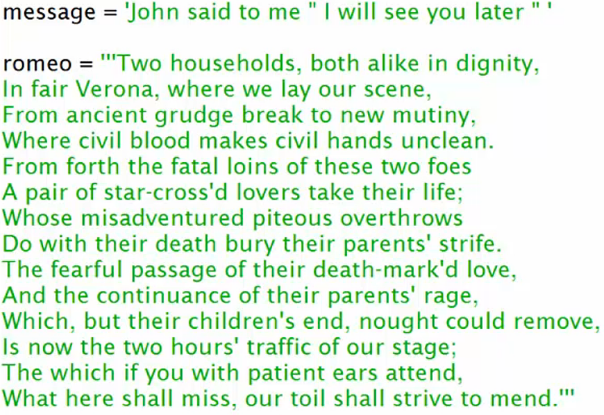
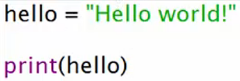
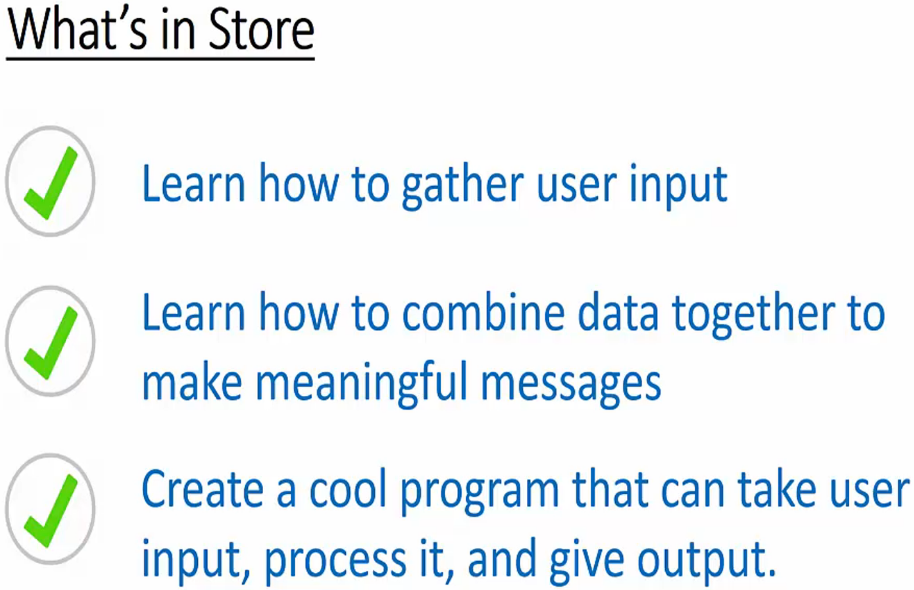
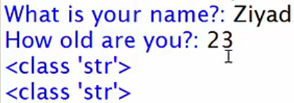
**Section Overview**  


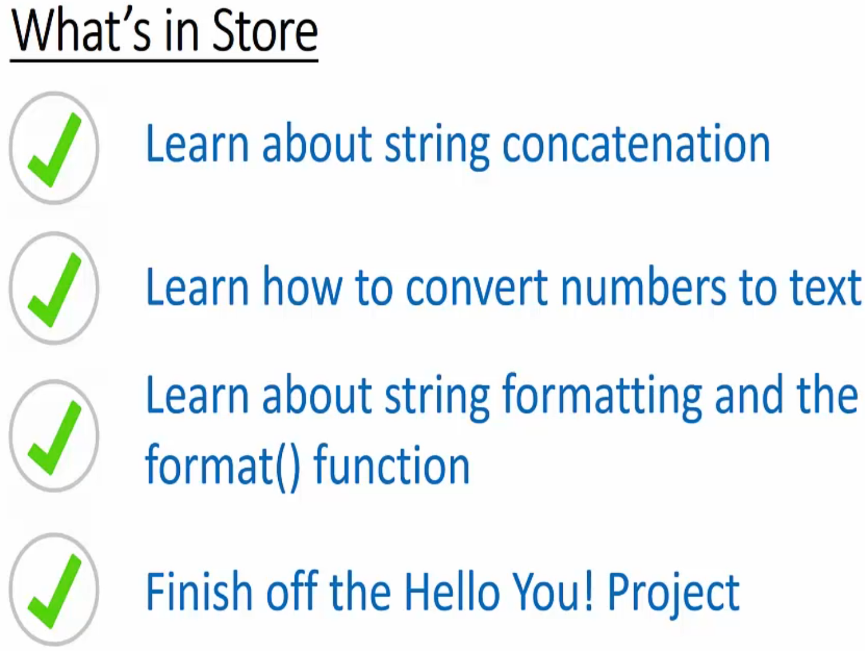
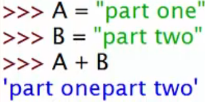
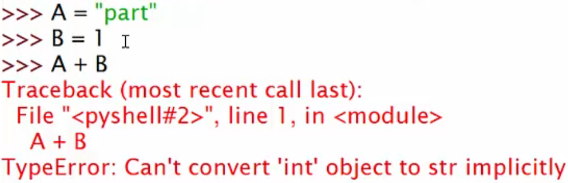
**Hello World!**  
  
\* **Broken String**  
**1. “”**  
**2. ‘’**  
**3a.** **“”” “”” => multi-line string**  
**3b.** **‘’’ ‘’’ => multi-line string**  
  


**Coding Exercise 4: Hello World!**

**Coding Exercise 5: Fixing Broken Strings**

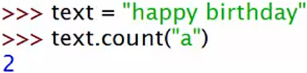
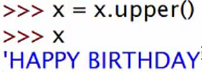
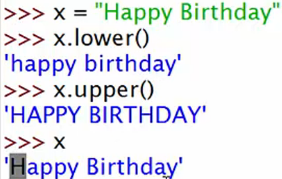
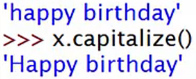
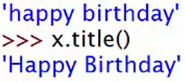
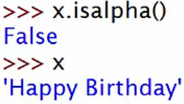
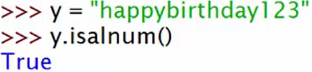
**PROJECT 2: Hello You! - Part 1 - Project Overview**\* **One of Python’s biggest strengths is how easy it makes it to combine and convert between various datatypes in order to make meaningful outputs**.  
\* **By learning this you will have more flexibility in your programs and more flexibility means more options and more options mean better programs**.  


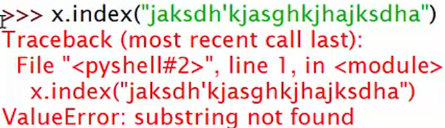
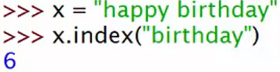
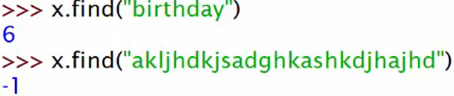
**PROJECT 2: Hello You! - Part 2 - Collecting Data**  
**#** => **comment**  
**input()** => **saved as a string**  


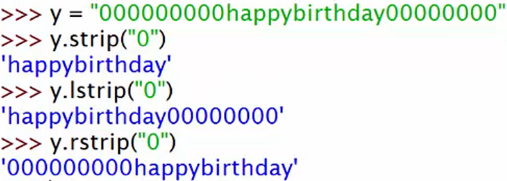
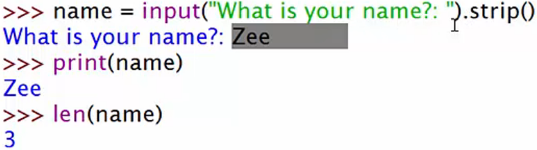
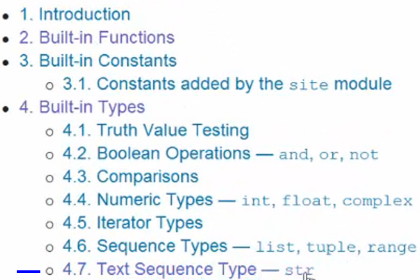
**PROJECT 2: Hello You! - Part 3 - String Formatting**  
  
\* **string + string** => **String Concatenation** - **combines the 2 strings**  
\* **Addition Operator +** => **add together 2 pieces of compatible data**  
  
\* **One particularly useful use for this is when you’re creating menus or something like that**:  
  
  
**str()**  
  
\* **Strings and Numbers are not implicitly compatible.**  
  
**“{}”.format() => formats whatever data you put in there into a string.**  
  

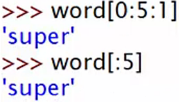
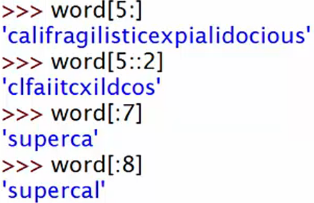

**Coding Exerciise 6: How old are you?**

**Coding Exerciise 7: Tell us about yourself!**

**Cool String Methods - Part 1**  
\* **String Methods**  
**Method** => **a function which is specific to a certain datatype**.  
**“”.count()**  
  
**“”.lower() => not modifying the actual string.  
“”.upper() => not modifying the actual string.  
\* Strings are Immutable => they can’t be changed. But they can be ovewritten.** **“”.capitalize()** **“”.title() => every word capitalized.** **“”.islower()  
“”.usupper()  
“”.istitle ()  
“”.isalpha()** **=> Space is not a letter.  
“”.isdigit()** **“”.isalnum()**

**Cool String Methods - Part 2**  
**“”.index() => when you want something to hard stop when not found.** **“”.find()**

**“”.strip() => if you don’t put anything in it, it’s going to strip spaces.  
“”.lstrip()  
“”.rstrip()**  **“”.len()**

**PROJECT 3: Email Slicer - Part 1 - What are Slices?**  
\* String is basically a bunch of characters and letters that are strung together.  
\* In Python, strings are an **ITERABLE** datatype.  
=> It means you can go step by step along it until you get to the end.  
\* **Each character** of a string is an **Element, each element has a number which refers to its position**.  
=> By using this number, you can get to the element.  
  
\* **String** = **Immutable** **Iterable** **datatype  
“”[start:end:step]   
=> start is included  
=> end is excluded**  
  
 **“”[start:] => start to the end of the string.  
“”[start::step] => start to the end of the string by step.  
“”[:end] => from the start to the end but not including it.**  
**“”[::-1] => reverse order, whole string.**  
  
\* **We have a main issue - the way we’re doing it so far is error prone, we’re counting up the index from the beginning and that can be time consuming and we can make errors**.

**PROJECT 3: Email Slicer - Part 2 - Automated Slices**  
\* **Negative index to count from the end**.  
  
  
  
  
\* **Bear in mind that index() only returns the 1st instance of what you’re searching for**.  
 

**Quiz 3: Slices Quiz**

**Coding Exercise 8: Slicer Exercise**

**PROJECT 3: Email Slicer - Part 3 - Making the Slicer**  
**\* You can use numbers to change the index.**  
