Announcements

- MPL #5 assigned – Due on Tuesday, October 28th by 11:59pm
- Program 5 has been assigned
  - Due Thursday, October 30th by 11:55pm via Moodle
- Solutions for Review Activity from Friday in Public directory
- Extra Credit Opportunity today
  - CS Research Talk by Professor Kirlin and David Thomas
    - 4pm in Ohlendorf 225

Basic String Operations

- Many types of programs perform operations on strings
  - So far we’ve only really seen strings as input/output
- In Python, many tools for examining and manipulating strings
  - Strings are sequences, so many of the tools that work with sequences work with strings

Accessing the Individual Character in a String

- To access an individual character in a string:
  - Use a for loop
    - Format: for character in string:
  - Useful when need to iterate over the whole string, such as to count the occurrences of a specific character
  - Use indexing
    - Each character has an index specifying its position in the string, starting at 0
    - Format: character = my_string[i]
Accessing Characters

Each character in a string is numbered by its position:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>'C'</td>
<td>'o'</td>
<td>'m'</td>
<td>'p'</td>
<td>'u'</td>
<td>'t'</td>
<td>'e'</td>
<td>'r'</td>
</tr>
</tbody>
</table>

The numbers shown here above the characters are called **indices** (singular: index) or **positions**.

Accessing Characters

- **IndexError** exception will occur if:
  - You try to use an index that is out of range for the string
  - Likely to happen when loop iterates beyond the end of the string
- **len(string)** function can be used to obtain the length of a string
  - Useful to prevent loops from iterating beyond the end of a string

```python
myString = "Roses are red"
ch = myString[6]  # ch is now equal to 'a'
```

```python
# This program counts the number of times
# the letter T (uppercase or lowercase)
# appears in a string.

def main():
    # Create a variable to use to hold the count.
    # The variable must start with 0.
    count = 0

    # Get a string from the user.
    my_string = input('Enter a sentence: ')

    # Count the Ts.
    for ch in my_string:
        if ch == 'T' or ch == 't':
            count += 1

    # Print the result.
    print(f'The letter T appears {count} times.')

# Call the main function.
main()
```
You can also access individual characters by index and loop over the range of all possible indices.

### String Concatenation

- **Concatenation**: appending one string to the end of another string
  - Use the `+` operator to produce a string that is a combination of its operands
  - The augmented assignment operator `+=` can also be used to concatenate strings
    - The operand on the left side of the `+=` operator must be an existing variable; otherwise, an exception is raised

### Strings Are Immutable

- **Strings are immutable (unchangeable)**
  - Once they are created, they cannot be changed
    - Concatenation doesn’t actually change the existing string, but rather creates a new string and assigns the new string to the previously used variable
    - Cannot use an expression of the form `string[index] = new_character`
      - Statement of this type will raise an exception

String Methods

- Strings in Python have many types of methods, divided into different types of operations
  - General format: \texttt{mystring.method(arguments)}
- Some methods test a string for specific characteristics
  - Generally Boolean methods, that return \texttt{True} if a condition exists, and \texttt{False} otherwise

<table>
<thead>
<tr>
<th>Table 9.1 Some string testing methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>isalnum()</td>
</tr>
<tr>
<td>isalpha()</td>
</tr>
<tr>
<td>isdigit()</td>
</tr>
<tr>
<td>islower()</td>
</tr>
</tbody>
</table>
|isspace() | Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. Whitespace characters are spaces, newlines (\texttt{\r
}), and tabs (\texttt{\t}). |
| isupper() | Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise. |

Practice

Write a program that asks the user to input a sentence and counts the number of spaces that are in the inputted string.

\textbf{Example:}

Please enter a sentence: Fall break was so fun!
The number of spaces in your sentence is: 4

Next Time

More Strings
Read Section 8.2