COMP 141

Strings

Announcements

• Program 6 has been assigned
  – Due Thursday, November 2nd by 11:55pm via Moodle

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

Strings are built from characters

The string "Computer" is represented internally like this:

```
"C" "o" "m" "p" "u" "t" "e" "r"
```

• Each piece of a string is called a character.
• A character is a special kind of string that is made up of exactly one letter, number, or symbol.
Accessing Characters

Each character in a string is numbered by its position:

```
0 1 2 3 4 5 6 7
C o m p u t e r
```

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

**Figure 9.2**: String indexes

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

String are *immutable* (unchangeable)
- Once they are created, they cannot be changed.

```
• These individual variables can be used just like regular variables, except you cannot assign to them.
my_string = "Computer"
my_string[0] = "B"    # illegal!
```

```
• You can print them, assign them to variables, pass them to functions, etc.
my_string = "Computer"
first = my_string[0]
third = my_string[2]
print(first, third, my_string[4])
```
Another Example

name = input(“What is your name?”)
initial = name[0]
print(“The first initial of your name is”, initial)

Sample Output:
What is your name?  Catie
The first initial of your name is C

Getting the Length of a String

- **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

myString = “Hello World”
n = len(myString)
print(myString[n+1])  # This will cause an IndexError
print(myString[n])    # This will also cause an IndexError

Getting the Length of a String

- Assume `s` is a string variable
- `len (s)` returns the length of `s`
- `len(“Computer”)` returns 8
- `len(“A B C”)` returns ??? 5
- `len(“”)` returns ??? 0
- `len` uses return, meaning if you want to capture the length, you should save the return value in a variable
Loops over Strings

- Wanting to be able to access characters one at a time naturally leads to using a loop to process strings.

- 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.

```python
# This program counts the number of times
# the letter 't' (upper 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('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.

```python
# This program counts the number of times
# the letter 't' (upper 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 ind in range(0, len(my_string)):
        ch = my_string[ind]
        if ch == 't' or ch == 'T':
            count += 1

    # Print the result.
    print('The letter t appears', count, 'times.')

# Call the main function.
main()
```
Practice

- Write a loop to count the number of capital letter A's in a string.
- Write a loop to count capital or lowercase A's.
- Write a loop to print all the letters in a string in reverse order.
- Write a loop to print every other character in a string, starting with the first.

Example using isupper()

```python
# This program counts the number of times
# an uppercase letter appears in a string.

def main():
    # Create a variable to use to hold the count.
    count = 0
    # Get a string from the user.
    my_string = input('Enter a sentence: ')  
    # Count the uppercase letters
    for ch in my_string:
        if ch.isupper():
            count += 1
    # Print the result.
    print(count, 'of the letters were uppercase.')
    # Call the main function.
    main()
```

String Testing Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isalnum()</td>
<td>Returns true if the string contains only alphanumeric letters or digits and is at least one character in length. Returns false otherwise.</td>
</tr>
<tr>
<td>isalpha()</td>
<td>Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.</td>
</tr>
<tr>
<td>isdigit()</td>
<td>Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.</td>
</tr>
<tr>
<td>islower()</td>
<td>Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.</td>
</tr>
<tr>
<td>isspace()</td>
<td>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 (\n), and tabs (\t).)</td>
</tr>
<tr>
<td>isupper()</td>
<td>Returns true if all of the alphanumeric letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.</td>
</tr>
</tbody>
</table>

String Modification Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower()</td>
<td>Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.</td>
</tr>
<tr>
<td>upper()</td>
<td>Returns a copy of the string with all alphabetic characters converted to uppercase. Any character that is already uppercase, or is not an alphabetic letter, is unchanged.</td>
</tr>
<tr>
<td>ljust()</td>
<td>Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines (\n), and tabs (\t) that appear at the beginning of the string.</td>
</tr>
<tr>
<td>rjust()</td>
<td>Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (\n), and tabs (\t) that appear at the end of the string.</td>
</tr>
<tr>
<td>center()</td>
<td>Returns a copy of the string with both leading and trailing whitespace characters removed. The character argument is a string containing a character. Returns a copy of the string with all instances of that character that appear at the beginning or the end of the string removed.</td>
</tr>
<tr>
<td>replace()</td>
<td>Returns a copy of the string with all instances of a character that appear at the beginning of the string removed. The character arguments are string containing a character. The method returns a copy of the string with all instances of that character that appear at the end of the string removed.</td>
</tr>
<tr>
<td>strip()</td>
<td>Returns a copy of the string with all leading and trailing whitespace characters removed.</td>
</tr>
<tr>
<td>lstrip()</td>
<td>Returns a copy of the string with all leading whitespace characters removed.</td>
</tr>
<tr>
<td>rstrip()</td>
<td>Returns a copy of the string with all trailing whitespace characters removed.</td>
</tr>
<tr>
<td>split()</td>
<td>Returns a list of the substrings separated by the specified separator. If the separator is not specified, returns a list of the substrings divided by whitespace.</td>
</tr>
<tr>
<td>join()</td>
<td>Returns a new string, which is the concatenation of the elements in the list. The elements are separated by the specified separator. If the separator is not specified, the list items are concatenated without separators.</td>
</tr>
</tbody>
</table>

Elizabeth College
Example

```python
weapon = input("Enter rock(R), paper(P), or scissors(S)")
weapon = weapon.lower()
if weapon == 'r' or weapon == 'p' or weapon == 's':
    validMove = True
else:
    validMove = False
```