Strings

Darin Brezeale

The University of Texas at Arlington

Strings

We have many occasions to store words as values. For this we can create variables of the string type.

Example:

```
first = ’a string’

second = "another string"

third = """a very long string that
stretches across multiple lines"""
```

ASCII

Strings are collections of characters. Each character has a numeric value. The numeric values corresponding to characters are defined in the ASCII set.

One thing to keep in mind is that numeric characters are not equivalent to numbers as we have been working with them. For example, "123" $\neq$ 123.

Comparing Strings

We can compare strings just as we would numeric data types.

Example:

```
if "abc" == "xyz" :
    print "these strings are equal"
```

See `cmpstrings.py` on the course website.

Concatenation

We can concatenate (i.e., add) strings using a plus sign. Example:

```
first = "Darin"
second = " "
third = "Brezeale"

total = first + second + third

print total
```

produces

```
Darin Brezeale
```

Manipulating Strings

Strings are sequences of characters and as such can be manipulated in ways similar to lists.

This includes indexing and slicing characters.

Accessing String Elements

We can access characters in strings using their subscripts in the same way we did for lists.

```
word = "ABCXYZ"

print word[0], word[1], word[2]

print word[-3], word[-2], word[-1]
```

produces

```
A B C
X Y Z
```

Accessing String Elements cont.

We can also slice a string to get a range of values:

```
#                   1         2
#         01234567890123456789012345
letters = "abcdefghijklmnopqrstuvwxyz"

print letters[3:10]   # indices 3 to 9
print letters[:8]     # beginning to 7
print letters[10:]    # 10 to end
print letters[4:15:2] # 4 to 14, increment by 2
```

produces

```
defghij
abcdefgh
klmnopqrstuvwxyz
egikmo
```

Iterating Through a String

We can easily iterate through a string using the `for` statement:

```
for letter in "Zebra" :
    print letter
```

produces

```
Z
e
b
r
a
```

String Methods

We can use the `len()` function to get the length of a string.

There are many, many methods for strings as well (too many to list here).