Strings
Chapter 6

String Data Type

- A string is a sequence of characters
- A string literal uses quotes ‘Hello’ or “Hello”
- For strings, + means “concatenate”
- When a string contains numbers, it is still a string
- We can convert numbers in a string into a number using int()

```python
>>> str1 = "Hello"
>>> str2 = 'there'
>>> bob = str1 + str2
>>> print bob
Hellothere
```

```python
>>> str3 = '123'
>>> str3 = str3 + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str' and 'int' objects
```

```python
>>> x = int(str3) + 1
>>> print x
124
```

Reading and Converting

- We prefer to read data in using strings and then parse and convert the data as we need
- This gives us more control over error situations and/or bad user input
- Raw input numbers must be converted from strings

```python
>>> name = raw_input('Enter:')
Enter:Chuck
>>> print name
Chuck
>>> apple = raw_input('Enter:')
Enter:100
>>> x = apple - 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```

```python
>>> x = int(apple) - 10
>>> print x
90
```
Looking Inside Strings

• We can get at any single character in a string using an index specified in square brackets.
• The index value must be an integer and starts at zero.
• The index value can be an expression that is computed.

```
fruit = 'banana'
letter = fruit[1]
print letter
```

A Character Too Far

• You will get a python error if you attempt to index beyond the end of a string.
• So be careful when constructing index values and slices.

```
zot = 'abc'
print zot[5]
```

Strings Have Length

• There is a built-in function `len` that gives us the length of a string.

```
fruit = 'banana'
x = len(fruit)
print x
```

Len Function

A function is some stored code that we use. A function takes some input and produces an output.

```
'banana'  len()  6
```
Guido wrote this code
Len Function

A function is some stored code that we use. A function takes some input and produces an output.

```python
def len(inp):
    blah
    for x in y:
        blah
        y

'banana'
(a string)
6
(a number)
```

Looping Through Strings

• Using a while statement and an iteration variable, and the `len` function, we can construct a loop to look at each of the letters in a string individually.

```python
fruit = 'banana'
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(index, letter)
    index = index + 1
```

Looping Through Strings

• A definite loop using a `for` statement is much more elegant.

```python
for letter in fruit:
    print(letter)
```

Looping Through Strings

• The iteration variable is completely taken care of by the `for` loop.

```python
fruit = 'banana'
for letter in fruit:
    print(letter)
```

Looping Through Strings

• The iteration variable is completely taken care of by the `for` loop.

```python
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(letter)
    index = index + 1
```
Looping and Counting

- This is a simple loop that loops through each letter in a string and counts the number of times the loop encounters the 'a' character.

```python
word = 'banana'
count = 0
for letter in word:
    if letter == 'a':
        count = count + 1
print count
```

Looking deeper into in

- The iteration variable "iterates" though the sequence (ordered set)
- The block (body) of code is executed once for each value in the sequence
- The iteration variable moves through all of the values in the sequence

```python
for letter in 'banana':
    print letter
```

The iteration variable "iterates" though the string and the block (body) of code is executed once for each value in the sequence

Slicing Strings

- We can also look at any continuous section of a string using a colon operator
- The second number is one beyond the end of the slice - "up to but not including"
- If the second number is beyond the end of the string, it stops at the end

```python
s = 'Monty Python'
print s[0:4]  # Mont
print s[6:7]  # P
print s[6:20]  # Python
```
Slicing Strings

- If we leave off the first number or the last number of the slice, it is assumed to be the beginning or end of the string respectively.

```
>>> s = 'Monty Python'
>>> print s[2]
Mo
>>> print s[8:]
thon
>>> print s[:]
Monty Python
```

String Concatenation

- When the + operator is applied to strings, it means "concatenation".

```
>>> a = 'Hello'
>>> b = a + 'There'
>>> print b
HelloThere
>>> c = a + ' ' + 'There'
>>> print c
Hello There
>>> 
```

Using in as an Operator

- The in keyword can also be used to check to see if one string is "in" another string.
- The in expression is a logical expression and returns True or False and can be used in an if statement.

```
>>> fruit = 'banana'
>>> 'n' in fruit
True
>>> 'm' in fruit
False
>>> 'nan' in fruit
True
>>> if 'a' in fruit:
...     print 'Found it!
...
Found it!
>>> 
```

String Comparison

```
if word == 'banana':
    print 'All right, bananas.'

if word < 'banana':
    print 'Your word,' + word + ', comes before banana.'
elif word > 'banana':
    print 'Your word,' + word + ', comes after banana.'
else:
    print 'All right, bananas.'
```
String Library

- Python has a number of string functions which are in the string library

- These functions are already built into every string - we invoke them by appending the function to the string variable

- These functions do not modify the original string, instead they return a new string that has been altered

```python
>>> greet = 'Hello Bob'
>>> zap = greet.lower()
>>> print zap
hello bob
```

```python
>>> print greet
Hello Bob
```

```python
>>> print 'Hi There'.lower()
hi there
```

http://docs.python.org/lib/string-methods.html

---

```python
>>> stuff = 'Hello world'
>>> type(stuff)
<type 'str'>
>>> dir(stuff)
```

http://docs.python.org/lib/string-methods.html

---

```python
str.capitalize()   str.replace(old, new[, count])
str.center(width[, fillchar])   str.lower()
str.endswith(suffix[, start[, end]])   str.rfind([chars])
str.find(sub[, start[, end]])   str.rindex([chars])
str.lstrip([chars])   str.rjust(width[, fillchar])
str.lower()   str.replace(old, new[, count])
str.rfind([chars])   str.rindex([chars])
str.rstrip([chars])   str.rjust(width[, fillchar])
str.strip([chars])   str.replace(old, new[, count])
str.upper()   str.replace(old, new[, count])
```

http://docs.python.org/lib/string-methods.html
Searching a String

- We use the `find()` function to search for a substring within another string.
- `find()` finds the first occurrence of the substring.
- If the substring is not found, `find()` returns -1.
- Remember that string position starts at zero.

<table>
<thead>
<tr>
<th>banana</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

```python
>>> fruit = 'banana'
>>> pos = fruit.find('na')
>>> print pos
2
>>> aa = fruit.find('z')
>>> print aa
-1
```

Making everything UPPER CASE

- You can make a copy of a string in lower case or upper case.
- Often when we are searching for a string using `find()` - we first convert the string to lower case so we can search a string regardless of case.

```python
>>> greet = 'Hello Bob'
>>> nnn = greet.upper()
>>> print nnn
HELLO BOB
>>> www = greet.lower()
>>> print www
hello bob
```

Search and Replace

- The `replace()` function is like a “search and replace” operation in a word processor.
- It replaces all occurrences of the search string with the replacement string.

```python
>>> greet = 'Hello Bob'
>>> nstr = greet.replace('Bob','Jane')
>>> print nstr
Hello Jane
>>> greet = 'Hello Bob'
>>> nstr = greet.replace('o','X')
>>> print nstr
HellX BXb
```

Stripping Whitespace

- Sometimes we want to take a string and remove whitespace at the beginning and/or end.
- `lstrip()` and `rstrip()` to the left and right only.
- `strip()` Removes both begin and ending whitespace.

```python
>>> greet = ' Hello Bob '  
>>> greet.lstrip() 
'Hello Bob'
>>> greet.rstrip() 
' Hello Bob'
>>> greet.strip() 
'Hello Bob'
```
Prefixes

```python
>>> line = 'Please have a nice day'
>>> line.startswith('Please')
True
>>> line.startswith('p')
False
```

```python
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

>>> data = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print atpos
21
>>> sppos = data.find('.',atpos)
>>> print sppos
31
>>> host = data[atpos+1 : sppos]
>>> print host
uct.ac.za
```

Summary

- String type
- Read/Convert
- Indexing strings []
- Slicing strings [2:4]
- Looping through strings with for and while
- Concatenating strings with +
- in as an operator
- String comparison
- String library
- Searching in strings
- Replacing text
- Stripping white space