Computer Programming for Non-majors

Lecture #5 - Stringing Along – Using Character and String Data

How Do Computer Handle Character Data?

- Like all other data that a computer handles, characters are stored in numeric form. A particular code represents a particular character. The most commonly used code is ***ASCII*** (***A***merican ***S***tandard ***C***ode for ***I***nformation ***I***nterchange). Java uses a code called ***Unicode***.

Example: Comparing Characters

```
char1 = 'a'
char2 = 'b'
char3 = 'A'

if  char1 > char2 :
  print("Very good")
else :
  print("Very bad")

if char1 > char3 :
  print("Very good")
else :
  print("Very bad")
```

What are Strings?

- A collection of characters that are read and written together to form words, numbers and so on are called ***strings***.
- Strings have certain methods that can be used to manipulate them. At the same time, they can be used in some ways that are like the basic data type in Python, such as `int` and `float`.
- Individual characters in Python are considered string of length 1.

Assigning a Value to String

- A value can be assigned to a string by putting the characters inside single or double quotes:

```
>>> s = "This is the first"
>>> print (s)
This is the first
>>> t = 'This is the second'
>>> print(t)
This is the second
>>>
```

How does Python handle input and output?

- In Python, it is very easy to read in data is as a string.
- All input read using the input functions are initially strings; if they are some other data type, we use the appropirate function to convert the data to that type:

```
s = input("Enter the next string")
x = int(input("Enter another number")
```

Python String Input/Output - An Example

```
s = input(("Enter your string"))
print("Your string is \"", s, "\".")

>>>
Enter your stringThis is the first
Your string is " This is the first ".
>>>
```

Concatenation and Repetition

- Concatenation is the operation where we join two strings together into one longer string.

  ```
  s = "The " + "Second"
  print(s)
  ```

  will print **"The Second"**
- Repetition is the operation where we create a string that contains the same sequence of characters multiple times.

  ```
  s = "my" * 3
  print(s)
  ```

  will print **"mymymy"**

The Python String Functions

- The way in which you specify a string's function is to write:

```
objectName.methodName();
```

- **`len(s)`** - Returns the number of characters in s
- **`s.strip()`** - Returns s with loading and trailing white space characters removed.
- **`s[]`** - Returns a substring of s
- **`s.indexOf()`** - Returns the starting position of the first occurrence of a substring within s.

`len(s)`

- **`len(s)`** returns the number of characters that **`s`** contains:

```
s = "This Is The First"
print(len(s))
```

- This program prints **`17`**

s.strip()

- Returns s with leading and trailing white space characters removed.

```
s = "   This Is The First   "
s = s.strip();
print("My String is \'", s, "\'")
print ("It has ",  len(s), " character.")
```

The output is:

```
My String is 'This Is The First'
It has 17 character.
```

s[]

- **s[]** returns a substring of **s**.
- **s[i]** will return the ith character in the string.
- **s[i:j]** will return characters i through j.

`s[12]` - An Example

```
s =
  "The quick brown fox jumped over the lazy dogs"
t = s[12]

print("My String is \'", t, "\'")
```

- Output

```
>>>
My String is ' o '
>>>
```

`s[12, 17]` - An Example

```
s =
  "The quick brown fox jumped over the lazy dogs"
t = s[12:17]

print("My String is \'", t, "\'")
```

- Output

```
>>>
My String is ' own f'
>>>
```

`s.find()`

- `s.find()` can be used to find where a substring appears within `s`.
- Example

```
s = "John Francis Xavier Smith"
i = s.find("Fran");
t = s[i:i+7]

print(t, "\'begins at position", i)
```

Output

```
Francis 'begins at position 5
```

Comparing Strings

- We can compare strings in the same way that we compare numbers; using the standard operators **==** , **!=**, **>**, **>=**, **<**, **<=**

Collating Sequence

- The order in which characters are assumed to appear is called the collating sequence.
- For now, we are most concerned with the following facts about the collating sequence:
  - Digits (0-9) come before letters.
  - All 26 upper case letters come before the lower case letters.
  - Within upper case and within lower case, the letters all fall within alphabetical order.

CompareStrings.py

```
s = "First"
t = "first"
u = "Second"

if s == t :
  print("\'", s, "\' and \'",
                  t, "\' are the same.")
else :
  print("\'",s, "\' and \'",
                  t, "\' are different.")

if s > t :
  print("\'", s + "\' goes after \'",
                  t, "\'.")
else :
  print("\'", s + "\' comes before \'",
                  t, "\'.")
```

```
if s == u :
  print("\'", s, "\' and \'",
                  t, "\' are the same.")
else :
  print("\'",s, "\' and \'",
                  t, "\' are different.")

if s > u :
  print("\'", s + "\' goes after \'",
                  t, "\'.")
else :
  print("\'", s + "\' comes before \'",
                  t, "\'.")
```

Other String Functions

- There are other string functions that can be useful:
  - **s.replace(t_1,t_2)**
  - **s.upper()**
  - **s.lower()**
  - **s.capitalize()**
  - **s.count()**

s.replace(t_1,t_2)

- **s.replace(t1,t2)** replaces each occurrence of **t1** in the string s with **t2**
- Example

```
s = "The quick brown fox"
s.replace("brown", "blue")
print("\"", s, "\"")
```

- Output

```
" The quick brown fox "
```

s.upper()

- s.upper() changes all the characters in the string to upper case.
- Example

```
>>> s = "This is a test"
>>> t = s.upper()
>>> print(t)
THIS IS A TEST
>>>
```

`s.lower()`

- **`s.lower()`** changes all the characters in the string to lower case.
- Example

```
>>> s = "Barry yelled, \"OVER HERE!!!\""
>>> t = s.lower()
>>> print(t)
barry yelled, "over here!!!"
>>>
```

`s.capitalize()`

- s.capitalize() places the first character of the string in upper case and the remainder of the string in lower case.
- Example

```
>>> s = "i like SoHo, TriBeCa and iPhones"
>>> t = s.capitalize()
>>> print(t)
I like soho, tribeca and iphones
>>>
```

`s.count()`

- **`s.count(t)`** returns the number of occurences of the string t in s that do not overlap.
- Example

```
>>> s = "Aiiieie!"
>>> i = s.count("ii")
>>> print(i)
1
>>>
```

Example: Writing Changing a Form Letter

- Let's write a program to read a file and change every occurrence of the name "John" to "Robert"
- Initial algorithm:
  1. Instruct the user
  2. Change every occurrence on each line of "John" to "Robert

Refining the Form Letter Algorithm

1. Instruct the user
2. Change every occurrence on each line of "John" to "Robert

2.1 Get the first line
2.2 As long as it isn't "The End", replace every occurrence of John with Robert

Refining the Form Letter Algorithm

1. Instruct the user
2.1 Get the first line
2.2 As long as it isn't "The End", replace every occurrence of John with Robert

2.2 WHILE the line ≠ "The End"
2.2.1 Replace each occurrence of "John" with Robert
2.2.2 Print the new line
2.2.3 Get the next line

Refining the Form Letter Algorithm

1. Instruct the user
2.1 Get the first line
2.2 WHILE the line ≠ "The End"
2.2.1 Replace each occurrence of "John" with Robert
2.2.2 Print the new line
2.2.3 Get the next line

```
outString = inString.replace("John", "Robert")
```

Refining the Form Letter Algorithm

1. Instruct the user
2.1 Get the first line
2.2 WHILE the line ≠ "The End"

```
outString = inString.replace("John", "Robert")
```

2.2.2 Print the new line
2.2.3 Get the next line

```
inString = input("Please begin typing. "
        + "End by typing \'The End\'\n")
```

Refining the Form Letter Algorithm

```
inString = input("Please begin typing. "
        + "End by typing \'The End\'\n")
```

2.2 WHILE the line ≠ "The End"

```
outString = inString.replace("John", "Robert")
```

2.2.2 Print the new line

2.2.3 Get the next line

```
while inString != "The End" :
```

Refining the Form Letter Algorithm

```
inString = input("Please begin typing. "
        + "End by typing \'The End\'\n")
while inString != "The End" :
outString = inString.replace("John", "Robert")
```

2.2.2 Print the new line

2.2.3 Get the next line

```
print(outString)
```

Refining the Form Letter Algorithm

```
inString = input("Please begin typing. "
        + "End by typing \'The End\'\n")
while inString != "The End" :
outString = inString.replace("John", "Robert")
print(outString)
```

2.2.3 Get the next line

```
inString = input("Next line?\n")
```

Example: **ChangeLetter.py**

```
#  Change every occurrence of "John" in the
#  text of a form letter to "Robert"

#  Prompt the user and instruct him.her how
#  to indicate the end of the letter
inString = input("Please begin typing. "
        + "End by typing \'The End\'\n")

#  Keep changing as long as (s)he didn't
#  type "the end"
while inString != "The End" :
   outString = inString.replace("John", "Robert")
   print(outString)

   inString = input("Next line?\n")
```