ECE12: Introduction to Programming
Lecture 21

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 21: Overview

- String Manipulation
  - String operations
  - String methods
  - Example
    - **`name.py`**

String Manipulation

- String Operations (revisited)
  - **`len(s)`**
    - returns the length of the string **`s`**
  - **`s[index]`**
    - returns a character of the string at **`index`**
  - **`s[left:right]`**
    - returns a slice of the string from index **`left`** to **`right`**
  - **`s1 + s2`**
    - returns the concatenation of **`s1`** and **`s2`**
  - **`format % tuple`**
    - returns the string **`format`** with **`%`**-sequences replaced by formatted arguments from **`tuple`**

String Manipulation

- String Methods
  - **`upper()`**
    - returns the string converted to upper-case
      - **`"Test String".upper()`** returns **`"TEST STRING"`**
  - **`lower()`**
    - returns the string converted to lower-case
      - **`"Test String".lower()`** returns **`"test string“`**
  - **`swapcase()`**
    - returns the string with upper- and lower-case exchanged
      - **`"Test String".swapcase()`** returns **`"tEST sTRING"`**

String Manipulation

- String Methods
  - **`capitalize()`**
    - returns the string with capitalized beginning
      - **`"test string".capitalize()`** returns **`"Test string"`**
  - **`title()`**
    - returns the string with every word capitalized
      - **`"test string".title()`** returns **`"Test String"`**

String Manipulation

- String Methods
  - **`ljust(width)`**
    - returns a string left-justified in **`width`** characters
      - **`"test".ljust(10)`** returns
        `"test      "`
  - **`rjust(width)`**
    - returns a string right-justified in **`width`** characters
      - **`"test".rjust(10)`** returns
        `"      test"`
  - **`center(width)`**
    - returns a string centered in **`width`** characters
      - **`"test".center(10)`** returns
        `"   test   "`

String Manipulation

- String Methods
  - **`lstrip()`**
    - returns the string with any leading white space removed
      - **`" \tTest string \t".lstrip()`** returns
        `"Test string \t“`
  - **`rstrip()`**
    - returns the string with any trailing white space removed
      - **`" \tTest string \t".rstrip()`** returns
        `" \tTest string"`
  - **`strip()`**
    - returns the string with any leading and trailing white space removed
      - **`" \tTest string \t".strip()`** returns
        `"Test string"`

String Manipulation

- String Methods
  - **`isalpha()`**
    - returns true if the string consists of only alphabetic characters
      - **`"Test".isalpha()`** returns **`1`**
      - **`"test string".isalpha()`** returns **`0`**
  - **`isdigit()`**
    - returns true if the string consists of only digits
      - **`“123".isdigit()`** returns **`1`**
      - **`“1.23".isdigit()`** returns **`0`**
  - **`isalnum()`**
    - returns true if the string consists of only alphanumeric characters
      - **`"Test123".isalnum()`** returns **`1`**
      - **`"test string".isalnum()`** returns **`0`**
  - **`isspace()`**
    - returns true if the string consists of only white space characters
      - **`" \t \t".isspace()`** returns **`1`**
      - **`"Test string".isspace()`** returns **`0`**

String Manipulation

- String Methods
  - **`islower()`**
    - returns true if the string consists of only lower-case characters
      - **`"test".islower()`** returns **`1`**
      - **`"Test".islower()`** returns **`0`**
  - **`isupper()`**
    - returns true if the string consists of only upper-case characters
      - **`"TEST".isupper()`** returns **`1`**
      - **`"Test".isupper()`** returns **`0`**
  - **`istitle()`**
    - returns true if each word in the string is capitalized
      - **`"Test String".istitle()`** returns **`1`**
      - **`"Test string".istitle()`** returns **`0`**

String Manipulation

- String Methods
  - **`find(substring, start, end)`**
    - returns position where **`substring`** is found in the string, or -1, if **`substring`** is not found
    - **`start`** and **`end`** are optional search indices
      - **`"test string".find("st")`** returns **`2`**
      - **`"test string".find("ST")`** returns **`-1`**
  - **`rfind(substring, start, end)`**
    - returns position from the end where **`substring`** is found in the string, or -1, if **`substring`** is not found
    - **`start`** and **`end`** are optional search indices
      - **`"test string".rfind("st")`** returns **`5`**
      - **`"test string".rfind("ST")`** returns **`-1`**

String Manipulation

- String Methods
  - **`index(substring, start, end)`**
    - returns position where **`substring`** is found in the string, or raises **`ValueError`** if **`substring`** is not found
    - **`start`** and **`end`** are optional search indices
      - **`"test string".index("st")`** returns **`2`**
      - **`"test string".index("st", 4,-2)`** returns **`5`**
      - **`"test string".index("x")`** raises **`ValueError`**
  - **`rindex(substring, start, end)`**
    - returns position from the end where **`substring`** is found in the string, or raises **`ValueError`** if **`substring`** is not found
    - **`start`** and **`end`** are optional search indices
      - **`"test string".rindex("st")`** returns **`5`**
      - **`"test string".rindex("st", 4,-2)`** returns **`5`**
      - **`"test string".rindex("x")`** raises **`ValueError`**

String Manipulation

- String Methods
  - **`count(substring, start, end)`**
    - returns number of times **`substring`** is found in a string
    - **`start`** and **`end`** are optional search indices
      - **`"test string".count("st")`** returns **`2`**
      - **`"test string".count("st", 4,-2)`** returns **`1`**
  - **`startswith(substring, start, end)`**
    - returns if string starts with **`substring`**
    - **`start`** and **`end`** are optional search indices
      - **`"test string".startswith(“te")`** returns **`1`**
      - **`"test string".startswith("st")`** returns **`0`**
  - **`endswith(substring, start, end)`**
    - returns if string ends with **`substring`**
    - **`start`** and **`end`** are optional search indices
      - **`"test string".endswith("ing")`** returns **`1`**
      - **`"test string".endswith("st")`** returns **`0`**

String Manipulation

- String Methods
  - **`replace(old, new, max)`**
    - returns the string with all occurrences of **`old`** replaced by **`new`**
    - **`max`** optionally indicates maximum number of replacements
      - **`"test string".replace("st","X")`** returns **`"teX Xring"`**
      - **`"test string".replace("st","X",1)`** returns **`"teX string"`**
  - **`expandtabs(t)`**
    - returns a string where tabs are expanded to **`t`** spaces
    - tabulator size **`t`** is optional (defaults to **`8`**)
      - **`"\ttest string".expandtabs()`** returns **`"        test string"`**

String Manipulation

- String Methods
  - **`split(separator)`**
    - returns a list of strings created by splitting the string at **`separator`** strings found
    - **`separator`** is optional and defaults to white space
      - **`"a, b, c".split(", ")`** returns **`["a","b","c"]`**
      - **`"a, b, c".split()`** returns **`["a,","b,","c"]`**
  - **`splitlines()`**
    - returns a list of strings created by splitting the string at new line characters
      - **`"a\nb\nc".splitlines()`** returns **`["a","b","c"]`**
  - **`join(sequence)`**
    - returns a string concatenated of strings in the **`sequence`** using the original string as separator
      - **`", ".join(["a","b","c"])`** returns **`"a, b, c"`**

String Manipulation

- Example: **`name.py`**
  - Given a string with a full name, such as
    - **`"George W Bush"`**, or
    - **`"  george W     BUSH  "`**,
  - write the name in different common formats, as follows:

```
Please enter your full name:
  george W   BUSH
Your first name is        George
Your middle name is       W
Your last name is         Bush
Your full name is         George W Bush
Your initials are         GWB
Your email address is     george_bush@uci.edu
Suggested user IDs are    bush or gbush or georgeb
```

String Manipulation

- Example: **`name.py`** (part 1/3)

```
# name.py: example for string manipulation
# author: Rainer Doemer
# 03/10/04 RD  initial version

# input
name = raw_input("Please enter your full name:\n")

# compute
name = name.strip()
list = name.split()
if len(list) == 2:
    first = list[0].capitalize()
    middle = ""
    last = list[1].capitalize()
else:
    first = list[0].capitalize()
    middle = list[1].capitalize()
    last = list[-1].capitalize()
...
```

String Manipulation

- Example: **`name.py`** (part 2/3)

```
...

if middle:
    full = "%s %s %s" % (first,middle,last)
    initial = first[0] + middle[0] + last[0]
else:
    full = "%s %s" % (first,last)
    initial = first[0] + last[0]
email = first.lower() + "_" + last.lower() + "@uci.edu"
id1 = last.lower()
id2 = first.lower()[0] + last.lower()
id3 = first.lower() + last.lower()[0]
ids = [id1,id2,id3]
for i in range(len(ids)):
    if len(ids[i]) > 8:
        ids[i] = ids[i][:8]

...
```

String Manipulation

- Example: **`name.py`** (part 3/3)

```
...

# output
print "Your first name is".ljust(25), first
print "Your middle name is".ljust(25), middle
print "Your last name is".ljust(25), last
print "Your full name is".ljust(25), full
print "Your initials are".ljust(25), initial
print "Your email address is".ljust(25), email
print "Suggested user IDs are".ljust(25), " or ".join(ids)
```