Ling 555 — Programming for Linguists

Python - Strings

Robert Albert Felty

Speech Research Laboratory
Indiana University

Sep. 22, 2008

Outline

1. String basics
   - templates
   - width and precision
   - alignment
   - Strings and tuples
2. String methods
   - find
   - join / split
   - lower
   - replace
   - strip

Templates

Definition

The string module provides templating functionality. Templates can be useful for internationalization and markup formats.

Example

The first heading in your “hello, world” webpage.

```
from string import Template
s=Template('<h1>$hello</h1>')
German=True
if German==True:
    print(s.substitute(hello='Servus, welt')
else:
    print(s.substitute(hello='Hello, world')
```

Width and precision

Definition

You can define how you want numbers to be printed out. Pythons string formatting uses the same conventions as C.

practice

1. Print out pi to the 3rd decimal place, with a width of 7
2. Print pi times 100 in the same fashion

Width and precision

Definition

You can define how you want numbers to be printed out. Pythons string formatting uses the same conventions as C.

practice

1. Print out pi to the 3rd decimal place, with a width of 7

   ```
   print('%7.3f' % pi )
   ```

2. Print pi times 100 in the same fashion

Width and precision

Definition

You can define how you want numbers to be printed out. Pythons string formatting uses the same conventions as C.

practice

1. Print out pi to the 3rd decimal place, with a width of 7

   ```
   print('%7.3f' % pi )
   ```

2. Print pi times 100 in the same fashion

   ```
   print('%7.3f' % (100 * pi) )
   ```

Alignment

Practice

1. Print out 23 padded with zeros to make it 4 wide
2. Print out 456.7 left aligned

Alignment

Practice

1. Print out 23 padded with zeros to make it 4 wide
   `print('%04.0f' % 23)`
2. Print out 456.7 left aligned

Alignment

Practice

1. Print out 23 padded with zeros to make it 4 wide
   `print('%04.0f' % 23)`
2. Print out 456.7 left aligned
   `print('%-.1f' % 456.7)`

String and tuples

You can format tuples all at once, e.g.

practice

1. Create a new list *lol* which contains the lists *foo* and *bar*

String and tuples

You can format tuples all at once, e.g.

practice

1. Create a new list *lol* which contains the lists *foo* and *bar*

find

practice

1. Find where *in* starts in the phrase *needle in a haystack*
2. If *needle in a haystack* contains *hay*, print *hey*

find

practice

1. Find where *in* starts in the phrase *needle in a haystack*

```
phrase='needle in a haystack'
phrase.find('in')
```

2. If *needle in a haystack* contains *hay*, print *hey*

find

practice

1. Find where *in* starts in the phrase *needle in a haystack*

```
phrase='needle in a haystack'
phrase.find('in')
```

2. If *needle in a haystack* contains *hay*, print *hey*

```
if phrase.find('hay')>=0:
    print('hey')
```

Join and split

practice

1. Split the haystack phrase into multiple words
2. Reverse the order of the words
3. Join the words back together with commas

Join and split

practice

1. Split the haystack phrase into multiple words
   words=phrase.split()
2. Reverse the order of the words
3. Join the words back together with commas

Join and split

practice

1. Split the haystack phrase into multiple words
   words=phrase.split()
2. Reverse the order of the words
   words.reverse()
3. Join the words back together with commas

Join and split

practice

1. Split the haystack phrase into multiple words
   `words=phrase.split()`
2. Reverse the order of the words
   `words.reverse()`
3. Join the words back together with commas
   `','.join(words)`

Changing case

practice

1. Make *ALLCAPS* all lowercase

Changing case

practice

1. Make *ALLCAPS* all lowercase `'ALLCAPS'.lower()`

Replace

practice

1. Replace *needle* with *noodle* in the haystack phrase
   What is the value of phrase now?

Replace

practice

1. Replace *needle* with *noodle* in the haystack phrase
   `phrase.replace('needle', 'noodle ')`
   What is the value of phrase now?

Strip

practice

1.