Basics of Strings in Python

L555

Dept. of Linguistics, Indiana University
Fall 2010

Interactive Programs

- Strings can have placeholders, the values can follow:

```
print 'Hi %s' % 'Markus'
```


Interactive Programs

- Strings can have placeholders, the values can follow:

```
print 'Hi %s' % 'Markus'
```

- and we can do that with variables, too

```
greeting = 'Hi %s'
name = 'Markus'
print greeting % name
```


Interactive Programs

- Strings can have placeholders, the values can follow:

```
print 'Hi %s' % 'Markus'
```

- and we can do that with variables, too

```
greeting = 'Hi %s'
name = 'Markus'
print greeting % name
```

- and with more than one value

```
greeting = 'Hi %s, what a %s day today, it\'s
%i degrees outside.'
vals = ('Markus', 'cold', 36)
print greeting % vals
```


Interactive Programs

- Strings can have placeholders, the values can follow:

```
print 'Hi %s' % 'Markus'
```

- and we can do that with variables, too

```
greeting = 'Hi %s'
name = 'Markus'
print greeting % name
```

- and with more than one value

```
greeting = 'Hi %s, what a %s day today, it\'s
%i degrees outside.'
vals = ('Markus', 'cold', 36)
print greeting % vals
```

- Make sure to use a tuple, not a list, for the values!

Conversion Types

d,i	signed integer decimal
u	unsigned integer
f,F	floating point decimal
c	single character
r	string (converted with repr)
s	string (converted with str)

Width, Precision, and Zeroes

- in displaying a number, we define the width and precision:

```
from math import pi
print 'Pi: %10.3f' % pi
```


Width, Precision, and Zeroes

- in displaying a number, we define the width and precision:

```
from math import pi
print 'Pi: %10.3f' % pi
```

- and we can fill the whole thing with zeroes

```
from math import pi
print 'Pi: %010.3f' % pi
```

- a - creates left alignment, a + adds the plus sign to the number

```
from math import pi
print 'Pi: %-10.3f' % pi
```


String and Tuples

Example

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

```
'%s:  %4.2f' % ('pi', pi)
```


find

Example

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

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


find

Example

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')
```


find

Example

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')
```

- find does NOT return a Boolean value: if it does not find the substring, it returns -1

Join and Split

1. Split the haystack phrase into multiple words
   `words=phrase.split()`

Join and Split

1. Split the haystack phrase into multiple words
   `words=phrase.split()`
2. Reverse the order of the words
   `words.reverse()`

Join and Split

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

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

Changing Case

1. Make *ALLCAPS* all lowercase
   `'ALLCAPS'.lower()`
2. Make all but the first letter of *ALLCAPS* lowercase
   `'ALLCAPS'.title()`

Replace

1. Replace *needle* with *noodle* in the haystack phrase
   `phrase.replace('needle', 'noodle')`

Replace

1. Replace *needle* with *noodle* in the haystack phrase
   `phrase.replace('needle', 'noodle')`
2. Replace *e* with *o* in the haystack phrase
   `phrase=phrase.replace('e', 'o')`

Strip

1. Strip off newline characters from end of the haystack phrase
   `phrase=phrase.strip('\r\n')`

Strip

1. Strip off newline characters from end of the haystack phrase
   `phrase=phrase.strip('\r\n')`
2. Strip off any leading or trailing whitespace from the haystack phrase, and convert to upper case
   `phrase=phrase.strip().upper()`

Strip

1. Strip off newline characters from end of the haystack phrase
   `phrase=phrase.strip('\r\n')`
2. Strip off any leading or trailing whitespace from the haystack phrase, and convert to upper case
   `phrase=phrase.strip().upper()`
3. Strip off any leading or trailing whitespace from the haystack phrase, replace *needle* with *noodle* and convert to upper case
   `phrase=phrase.strip().replace('needle', 'noodle').upper()`

Translate

Definition

Translate can be used to map an entire character set to a different one, using a one-to-one mapping.

Example

I accidentally switch my keyboard to German mode, in which *y* and *z* are switched, and type my entire thesis this way without noticing.

```
thesis='verbositz in verbaliying is uglz'
from string import maketrans
germanToEnglish=maketrans('yz','zy')
thesis.translate(germanToEnglish)
```