

Strings: the Basics

L435/L555
Dept. of Linguistics, Indiana University
Fall 2016

Strings: What we already know

- ▶ Strings are sequences: order is important
 - ▶ indexing, slicing
 - ▶ looping over characters in a string
 - ▶ concatenation, len(), etc.
- ▶ Strings are immutable: they do not change
 - ▶ no use of append, etc.
 - ▶ cannot change values via index re-assignment, etc.
- ▶ Strings can occur in boolean statements
 - ▶ in test
 - ▶ alphabetic checks (<, >, etc.)

String formatting

Basic placeholders

- ▶ Strings can have placeholders, the values placed in `.format()`:

```
s="you're eating {0}".format("crazy cheese")  
# s = "you're eating crazy cheese"
```
- ▶ We can do that with variables, too:

```
location='paris'  
s="you would think I'm from {0}".format(location)  
# s = "you would think I'm from paris"
```
- ▶ And with more than one value:

```
lyric = "you know I get {0}, you think I get {1}"  
adjs = ("fly", "high")  
print(lyric.format(*adjs))
```

See: <https://docs.python.org/3.4/library/string.html>, sec. 6.1.3.2

String formatting

Alignment & width

- Using a colon (:), we can do left (<), right (>), & center (^) alignment
- ```
s="{:<10} know that I'm gone".format("you")
s="you know that I'm gone"
```
- ```
s="I'm-a tell {:>10} all why".format("you")  
# s="I'm-a tell          you all why"
```
- ```
s="who are you {:^20}".format("dissing")
s="who are you dissing "
```
- ```
s="maybe I'm {:*^20}".format("missing")  
# s="maybe I'm *****missing*****"
```
- In the last case, we use `*` as a fill character

String formatting

Different variable types

- This is the real "win" in string formatting: easy integration of different kinds of information
- ```
s="we have { :4d } MCs and { :4d } DJ".format(3,1)
s='we have 3 MCs and 1 DJ'
```
- ```
s="we have { :4.1f } MCs and { :4.2f } DJ".format(3,1)  
# s='we have   3.0 MCs and  1.00 DJ'
```
- ```
s="we have { 1:4.1f } MCs and { 0:4.2f } DJ".format(3,1)
s='we have 1.0 MCs and 3.00 DJ'
```
- ```
s="we have { :4.1f } MCs and { :*^6.2f } DJ".format(3,1)  
# s='we have   3.0 MCs and *1.00* DJ'
```

Conversion types

- s string (converted with str)
- r string (converted with repr)
- c single character
- d,n decimal integer
- f,F floating point decimal

find

Example

- Find *where* a string starts (cf. `index()` for lists)


```
phrase="the_reason_that_you're_smilin'"
phrase.find('son')      # 7
phrase.find('smile')   # -1
if phrase.find('you')>=0:
    print("me!")
```
- not: `find` does NOT return a Boolean value: if it does not find the substring, it returns -1

Strings: the Basics

String basics

- Review
- Formatting
- Conversion types
- String methods
- find**
- join & split
- lower & upper
- replace
- strip

7/11

join & split

- Split the haystack phrase into multiple words


```
words=phrase.split()
```

 - `.split()` can take an argument, namely the thing you want to split on (default=whitespace)
- Reverse the order of the words


```
words[::-1]
```
- Join the words back together with commas


```
','.join(words)
```

Strings: the Basics

String basics

- Review
- Formatting
- Conversion types
- String methods
- find
- join & split**
- lower & upper
- replace
- strip

8/11

Changing case

- Make a string all lowercase


```
'SMILIN'.lower()
```
- Make a string all uppercase


```
'wildin'.upper()
```
- Make all but the first letter of a string lowercase


```
'LISTEN'.title()
```

Strings: the Basics

String basics

- Review
- Formatting
- Conversion types
- String methods
- find
- join & split
- lower & upper**
- replace
- strip

9/11

replace

- Replace *smilin* with *frownin* in the phrase


```
phrase.replace('smilin', 'frownin')
```
- Replace *e* with *o* in the phrase


```
phrase=phrase.replace('e', 'o')
```

Strings: the Basics

String basics

- Review
- Formatting
- Conversion types
- String methods
- find
- join & split
- lower & upper
- replace**
- strip

10/11

strip

- Strip off newline characters from end of the phrase


```
phrase=phrase.strip('\r\n')
```
- Strip off any leading or trailing whitespace from the phrase, and convert to upper case


```
phrase=phrase.strip().upper()
```
- Strip off any leading or trailing whitespace from the haystack phrase, replace *smilin* with *frownin* and convert to upper case


```
phrase=phrase.strip().replace('smilin', 'frownin').upper()
```

Strings: the Basics

String basics

- Review
- Formatting
- Conversion types
- String methods
- find
- join & split
- lower & upper
- replace
- strip**

11/11