

Why Does My Computer Do That? Intro to Coding with Python—Strings and String Methods

Dr. Ab Mosca (they/them)

Plan for Today

- operations on strings
- accessing individual letters
- handy methods

Discussion

```
*s-u-converter.py - /Users/jcrouser/Google Drive/Teaching/Course Material/CSC111...  
score = eval(input("Please enter score: "))  
isTakingSU = input("S/U? ")  
  
if (isTakingSU):  
    # calculate S or U  
    if (score >= 70):  
        print("S")  
    else:  
        print("U")  
else:  
    print(score)
```

```
Python 3.7.0 Shell  
Please enter score: 30  
S/U? False  
U  
Ln: 14 Col: 4
```

what's
going on?



(RECAP) Core concept 3: strings

- In CS, a sequence of characters that isn't a number is called a **string**
- In Python, a string is declared using **quotation marks**
- Strings can contain letters, numbers, spaces, and special characters
- Example:

```
x = "Jordan"
```

```
x = "Bates Hall"
```

Operations on strings

- **Concatenation:** join two strings together with +, e.g.

`"Ab" + " " + "Mosca"`

- **Repetition** (i.e. self-concatenation): use *, e.g.

`3 * "hi"`

Multi-line strings

- **Problem:** a string that looks ugly when you try to type it all on one line, e.g.

```
desc = "This course is an introduction to computer science and computer programming. The programming language Python (Version 3) is used to introduce basic programming skills and techniques."
```

- We can use **triple quotes** to make a multi-line string, e.g.

```
desc = """This course is an introduction to computer science and computer programming. The programming language Python (Version 3) is used to introduce basic programming skills and techniques."""
```

Escaping quotes

- **Problem:** you have a statement that contains both an apostrophe and double quotes, e.g.

```
"I can't!" he said
```

- What's the **issue** here?
 - If we try to wrap it in single quotes, Python thinks the apostrophe in should end the string:
- If we try to wrap it in double quotes, Python thinks the double quote at the beginning of the sentence should end the string

```
s = "I can't!" he said"
```

Escaping quotes

- **Problem:** you have a statement that contains both an apostrophe and double quotes, e.g.

`"I can't!" he said`

- **Solution:** protect ("escape") special characters using a backslash, e.g.

`s = '"I can\'t!" he said'`

or

`s = "\"I can't!\" he said"`

Accessing individual letters

- One way to think about a **string** is as a **list** of letters

Lists

- List is an ordered collection of objects
 - Ex. ["a", "b", "c"]
 - ["cat", "dog", "moose", "cow"]
 - [TRUE, FALSE, FALSE]
 - [90, 110, 14, 67]

Lists

- List is an ordered collection of objects
 - Ex. ["a", "b", "c"]
 - ["cat", "dog", "moose", "cow"]
 - [TRUE, FALSE, FALSE]
 - [90, 110, 14, 67]
- Items in a list are indexed

[X,	Y,	Z,	W,	V]
0	1	2	3	4

Lists

- List is an ordered collection of objects
 - Ex. ["a", "b", "c"]
 - ["cat", "dog", "moose", "cow"]
 - [TRUE, FALSE, FALSE]
 - [90, 110, 14, 67]

- Items in a list are indexed

lst = [X, Y, Z, W, V]

0 1 2 3 4

- To get one item, we use the index
 - Ex. `lst[2]` will give us Z

Accessing individual letters

- One way to think about a **string** is as a **list** of letters:

```
name = "Jordan"
      ≈ [ 'J', 'o', 'r', 'd', 'a', 'n' ]
          0   1   2   3   4   5
```

- **Question:** how would I print out the 3rd letter (position 2)?

```
print(name[2])
```

“Slicing” (getting a substring)

- What about the 2nd - 5th letters (positions 1-4)?

```
print(name[1:5])
```

- What happens if we do this?

```
print(name[2:])
```

- What about this?

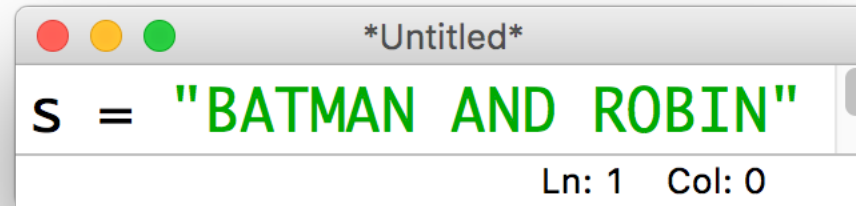
```
print(name[-2:])
```

up to, but
not including



15-minute exercise

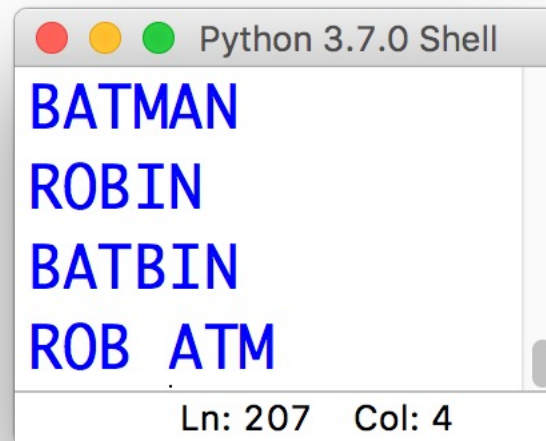
- Given this string:



A code editor window titled '*Untitled*' with a white background and a grey title bar containing three colored window control buttons (red, yellow, green). The text 's = "BATMAN AND ROBIN"' is displayed in green font. The status bar at the bottom right shows 'Ln: 1 Col: 0'.

```
s = "BATMAN AND ROBIN"
```

- Write a short program that uses **slicing** to produce:



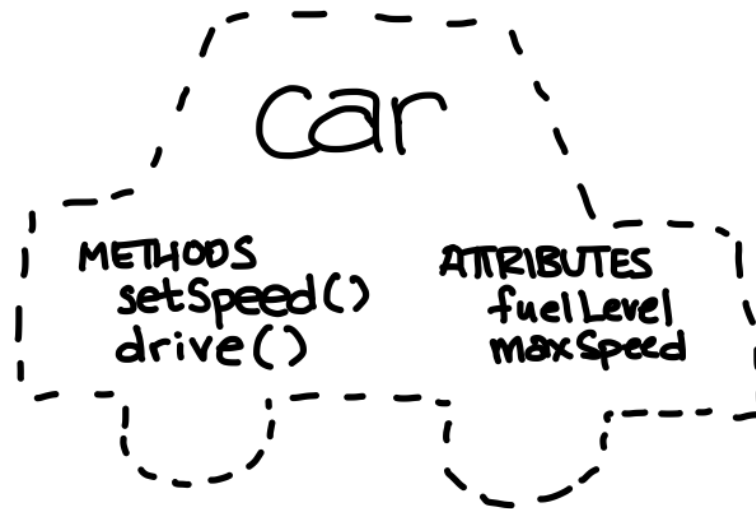
A Python 3.7.0 Shell window with a white background and a grey title bar containing three colored window control buttons (red, yellow, green) and the text 'Python 3.7.0 Shell'. The output of a program is shown in blue font: 'BATMAN', 'ROBIN', 'BATBIN', and 'ROB ATM' on four separate lines. The status bar at the bottom right shows 'Ln: 207 Col: 4'.

```
BATMAN  
ROBIN  
BATBIN  
ROB ATM
```

Discussion

What did you come up with?

Strings as objects



"object-oriented"

Useful methods for working with strings

- `s.lower()`: convert the string `s` to lowercase
- `s.upper()`: convert the string `s` to UPPERCASE
- `s.strip()`: remove whitespace from the start / end of `s`
- `s.replace('old', 'new')`: replace all occurrences of 'old' in `s` by 'new'
- `s.split(c)`: slice `s` into pieces using `c` as a delimiter
- `s.join(list)`: opposite of `split()`, join the elements in the list together using `s` as the delimiter, e.g.

```
'-'.join(['a', 'b', 'c']) # a-b-c
```

Fun fact

- **strings** in python are **immutable** (along with **ints**, **floats**, **bools**, and a few other built-in types)
- This means that when we call a method on them, the original isn't modified

15-minute exercise

- Work with 1 – 2 other people to write a short program that:
 - Takes as input from the user a string
 - Takes as input from the user a character (char1)
 - Takes as input from the user another character (char2)
 - Returns that the input string with all occurrences of char1 replaced with char2 and in all caps

```
Input a string: Ab Mosca  
Input a character: a  
Input another character: o  
OB MOSCO
```