A Robust Bot

Review Questions

Question 1

What would this output if a user inputs **S**?

```
print("What is your clothing size? (S/M/L)")
size = input()
if size == "S":
    print("We have shirts in small.")
if size == "M":
    print("We have pants in medium.")
elif size == "L":
    print("We only carry small and medium sizes.")
else:
    print("I don't know that size.")
```

A. We have shirts in small.
B. We have pants in medium.
C. We only carry small and medium sizes.
D. I don't know that size.

Question 2

What is wrong with this code?

```
burger = "basic"
if burger == "classic" or "deluxe‟
    print("That’s a great choice!")
```

A. The strings should use single quotes.
B. The first line should use ==.
C. The `if` line should use =.
D. The `or` is not going to work as expected.

Question 3

What would this output if a user input **M**?

```
print("Tell us what size drink you want.")
size = input()
print(size == "s" or size == "m" or size == "l")
```

A. True
B. False
C. M
D. Nothing, the program crashes.

Robust

strongly formed or constructed

- Merriam Webster

able to **withstand** or **overcome adverse conditions**.

- Oxford Dictionary

Defend against bad input

Syntax errors

You'll catch these as soon as you try and run it.
E.g. missing colon after an ***if*** *condition*

Semantic errors
or **logic** errors

Usually you'll find these after testing it a few times or asking a friend to try it.
E.g. Hello, what year were you born? 1993!
Sorry, you didn't enter a year I know.

http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/Syntaxerrors.html
http://interactivepython.org/runestone/static/thinkcspy/GeneralIntro/SemanticErrors.html

This lesson

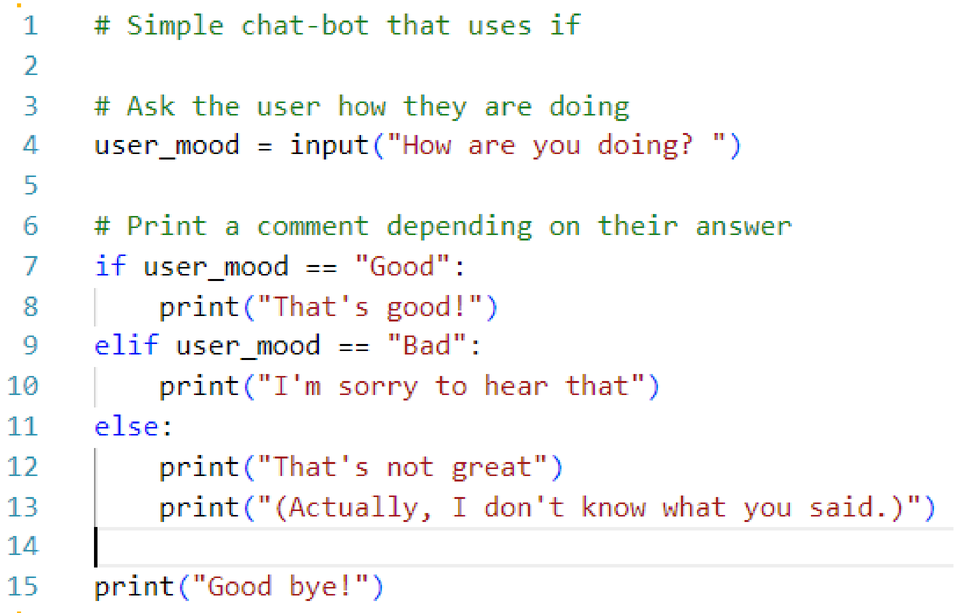
Our bot was good, but sometimes users input **unexpected** things. Let's make our bot more *robust* to various situations.

Today, you'll learn:

- String methods (**strip**, **lower**, **upper**)
- Method chaining
- The **in** keyword

Note: we will use the words **method/function/command** interchangeably in this course for now.

Previously... the How's it Going bot



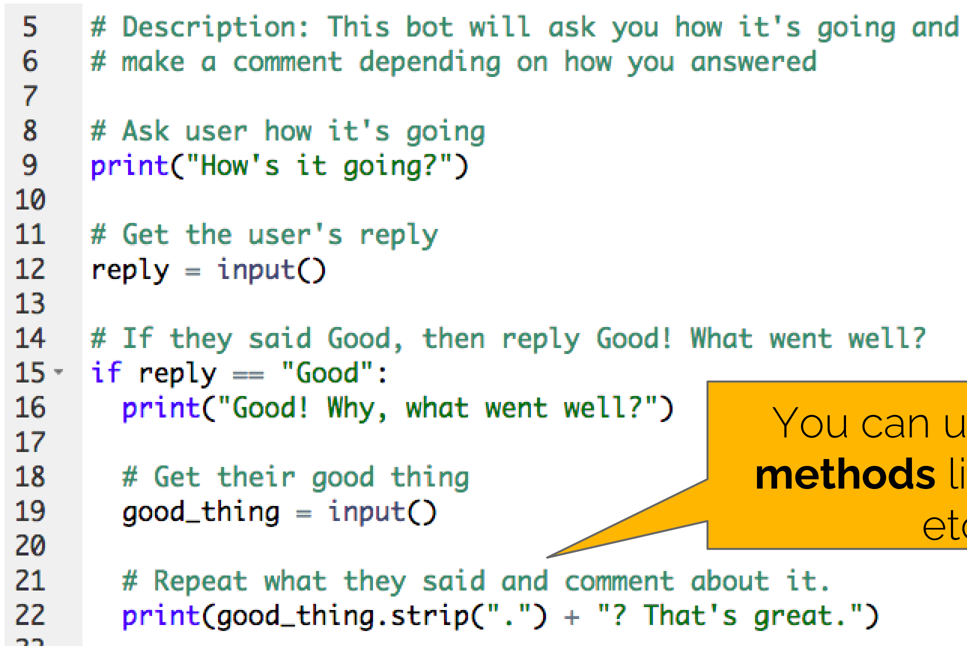
```python
# Simple chat-bot that uses if

# Ask the user how they are doing
user_mood = input("How are you doing? ")

# Print a comment depending on their answer
if user_mood == "Good":
    print("That's good!")
elif user_mood == "Bad":
    print("I'm sorry to hear that")
else:
    print("That's not great")
    print("(Actually, I don't know what you said.)")

print("Good bye!")
```

If "Good", **ask** the user what went well and **state it back** to them.

For example, if they say "*Food.*", We want to say "*Food? That's great!*"

Use **my_string.strip()**

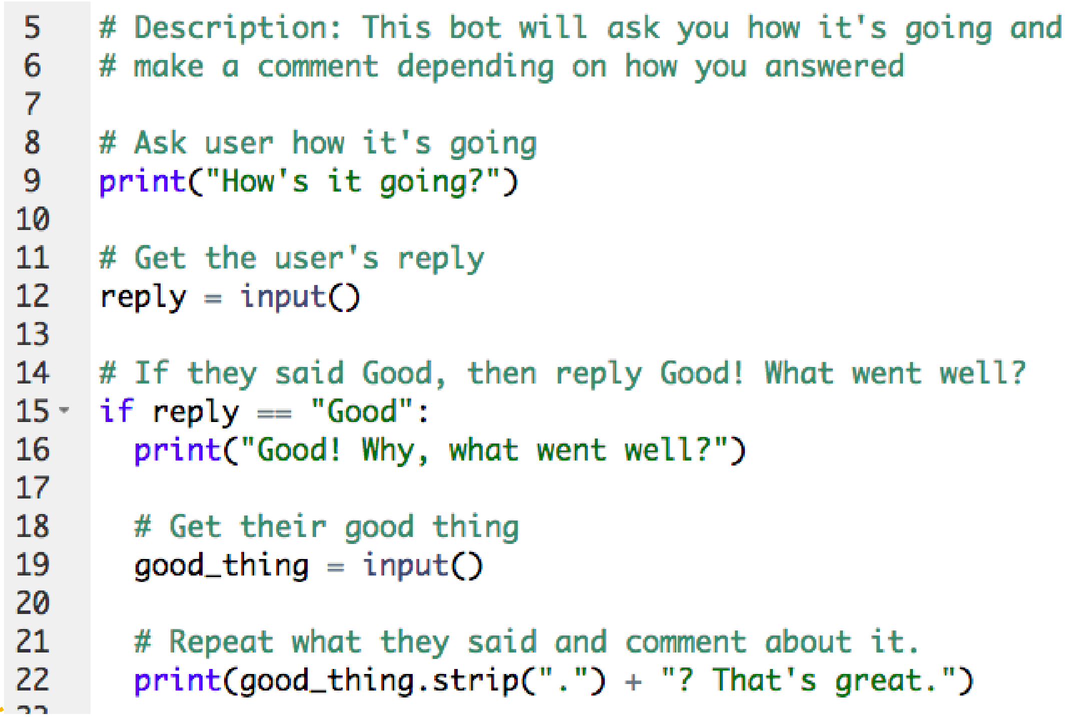
String methods



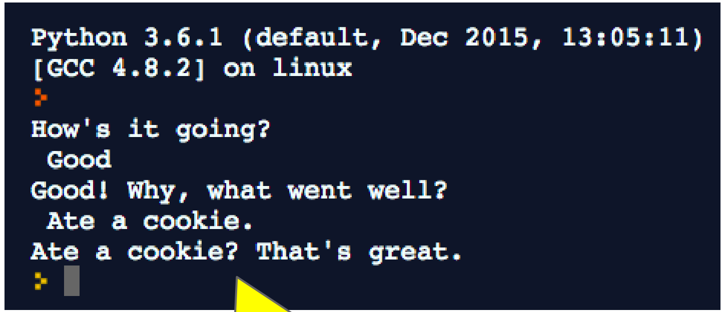
```
# Description: This bot will ask you how it's going and
# make a comment depending on how you answered

# Ask user how it's going
print("How's it going?")

# Get the user's reply
reply = input()

# If they said Good, then reply Good! What went well?
if reply == "Good":
  print("Good! Why, what went well?")

  # Get their good thing
  good_thing = input()

  # Repeat what they said and comment about it.
  print(good_thing.strip(".") + "? That's great.")
```

You can use **string methods** like **.strip**(), etc.

Test how it works



```python
# Description: This bot will ask you how it's going and
# make a comment depending on how you answered

# Ask user how it's going
print("How's it going?")

# Get the user's reply
reply = input()

# If they said Good, then reply Good! What went well?
if reply == "Good":
  print("Good! Why, what went well?")

  # Get their good thing
  good_thing = input()

  # Repeat what they said and comment about it.
  print(good_thing.strip(".") + "? That's great.")
```



```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
How's it going?
 Good
Good! Why, what went well?
 Ate a cookie.
Ate a cookie? That's great.
>
```

No more period!

String methods

You can remove more than just the period. Try it!

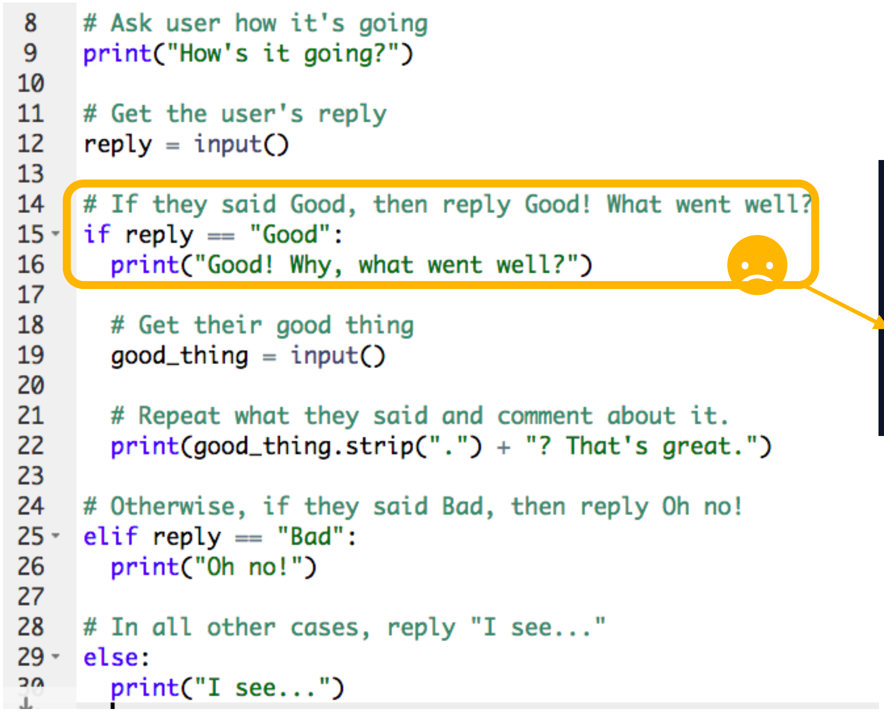
	Example
Remove the characters . ! ? and space at the start and end of myString	myString.strip(".!? ")
Convert all the letters into lowercase	myString.lower()
Convert all the letters into uppercase	myString.upper()

http://interactivepython.org/runestone/static/thinkcspy/Strings/StringMethods.html (not 9.5.1)

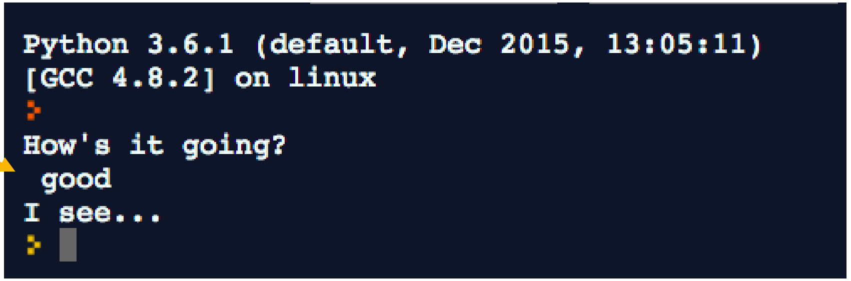
How might we use lower()?

- Use lower() on input strings to make it easier to handle answers with different case.

How might we use lower()?

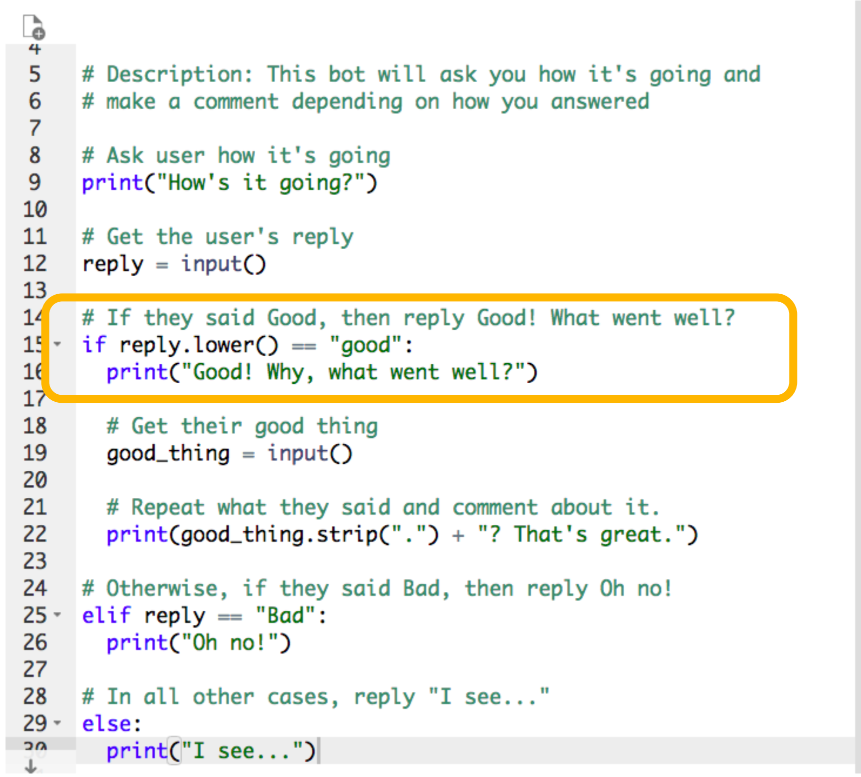


```python
# Ask user how it's going
print("How's it going?")

# Get the user's reply
reply = input()

# If they said Good, then reply Good! What went well?
if reply == "Good":
  print("Good! Why, what went well?")

  # Get their good thing
  good_thing = input()

  # Repeat what they said and comment about it.
  print(good_thing.strip(".") + "? That's great.")

# Otherwise, if they said Bad, then reply Oh no!
elif reply == "Bad":
  print("Oh no!")

# In all other cases, reply "I see..."
else:
  print("I see...")
```

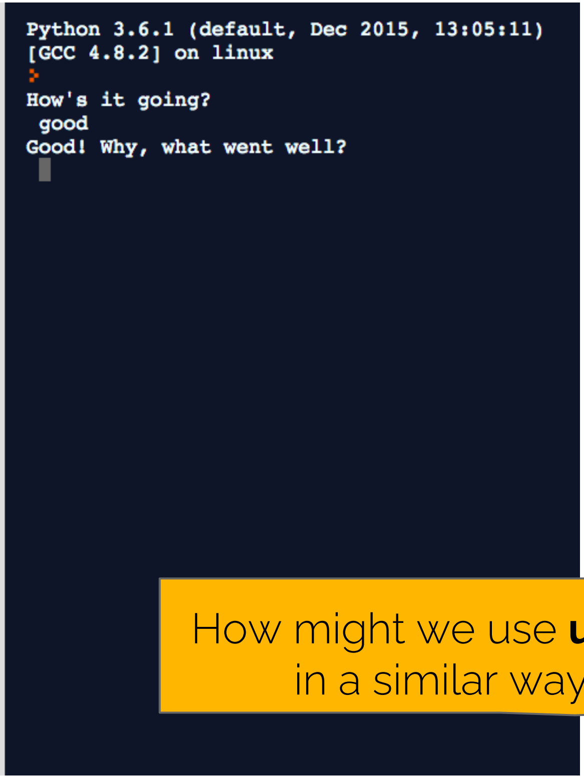


```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
›
How's it going?
 good
I see...
›
```

How might we use lower()?



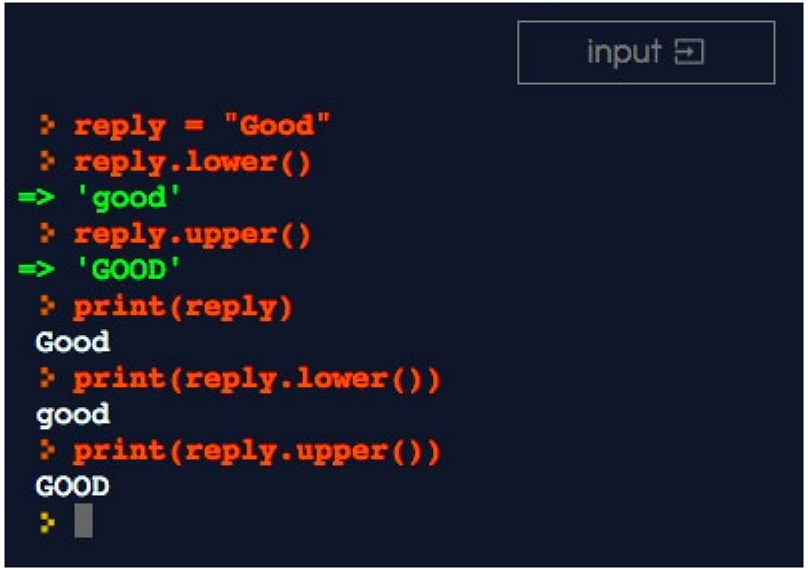
```
# Description: This bot will ask you how it's going and
# make a comment depending on how you answered

# Ask user how it's going
print("How's it going?")

# Get the user's reply
reply = input()

# If they said Good, then reply Good! What went well?
if reply.lower() == "good":
  print("Good! Why, what went well?")

  # Get their good thing
  good_thing = input()

  # Repeat what they said and comment about it.
  print(good_thing.strip(".") + "? That's great.")

# Otherwise, if they said Bad, then reply Oh no!
elif reply == "Bad":
  print("Oh no!")

# In all other cases, reply "I see..."
else:
  print("I see...")
```



```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
›
How's it going?
 good
Good! Why, what went well?
```

How might we use **upper**() in a similar way?

More examples

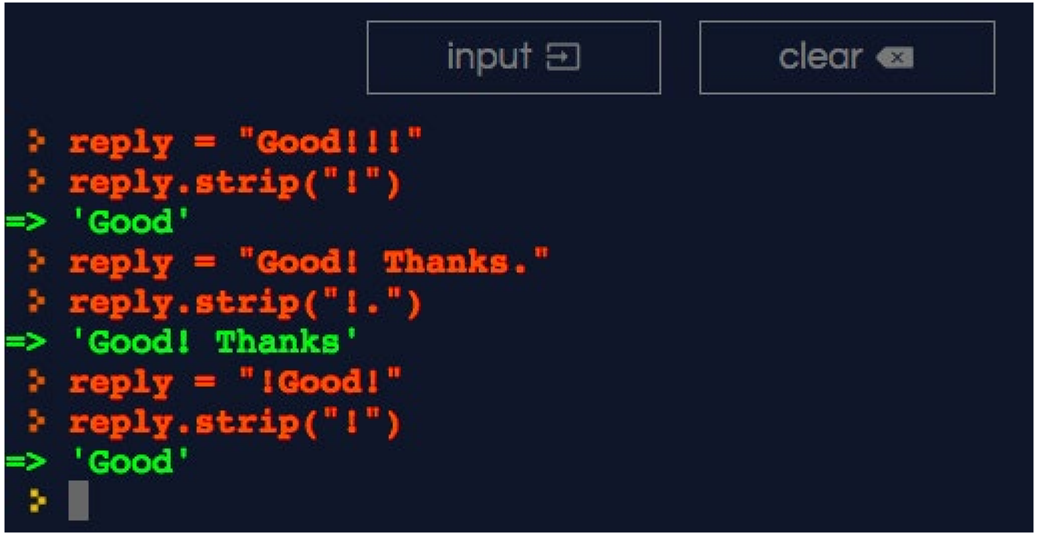


```
> reply = "Good"
> reply.lower()
=> 'good'
> reply.upper()
=> 'GOOD'
> print(reply)
Good
> print(reply.lower())
good
> print(reply.upper())
GOOD
>
```

REPL: Read Evaluate Print Loop

You have an **interactive** Python console to try things out!

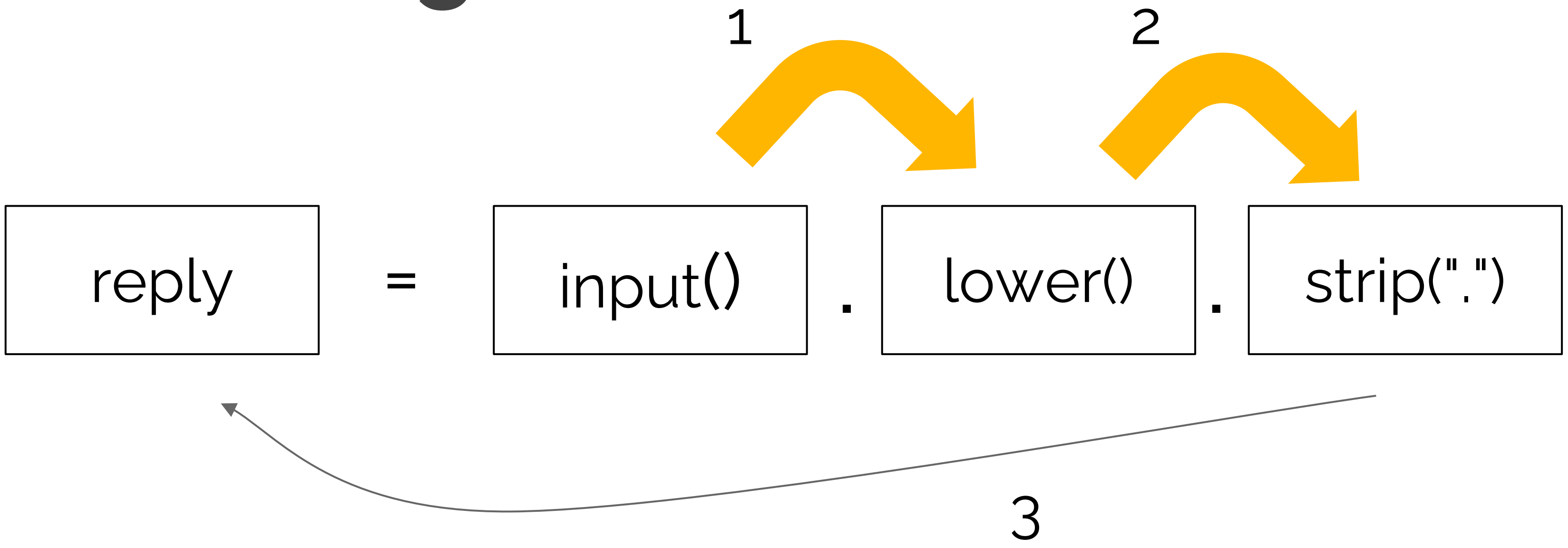
Note: You can either run your code or use the interactive console, but not both at the same time.



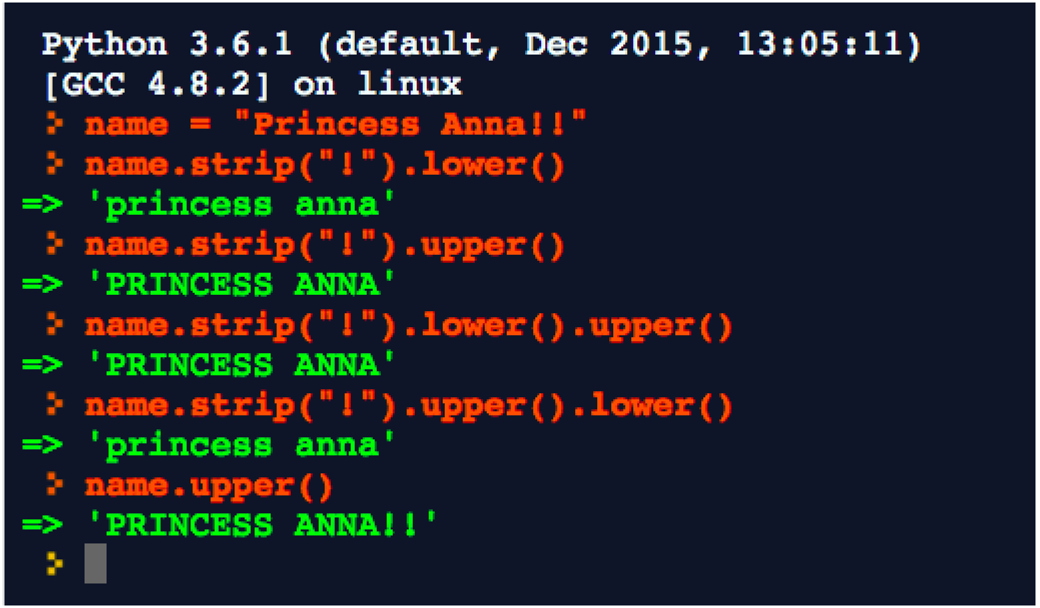
```
 > reply = "Good!!!"
 > reply.strip("!")
=> 'Good'
 > reply = "Good! Thanks."
 > reply.strip("!.")
=> 'Good! Thanks'
 > reply = "!Good!"
 > reply.strip("!")
=> 'Good'
 >
```

Method Chaining

Here's how chaining works (left to right)

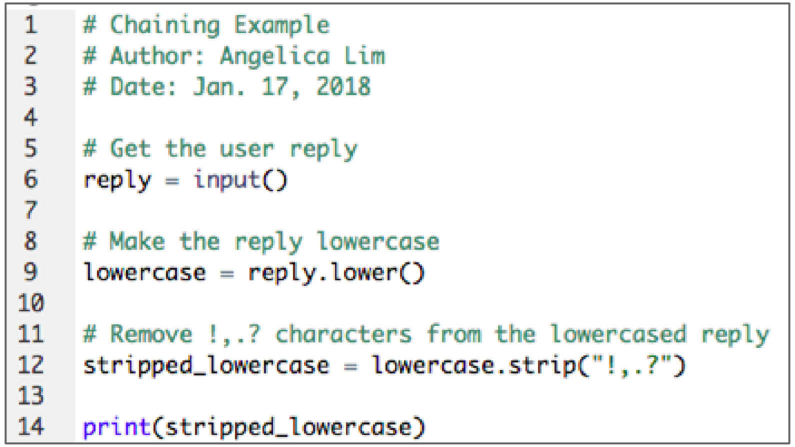


Chaining Examples



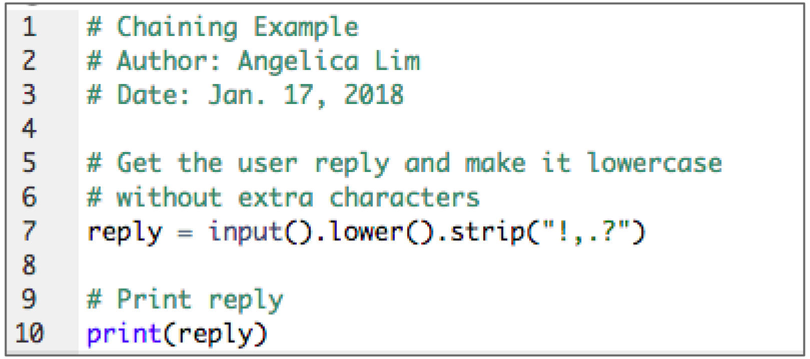
```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
 > name = "Princess Anna!!"
 > name.strip("!").lower()
=> 'princess anna'
 > name.strip("!").upper()
=> 'PRINCESS ANNA'
 > name.strip("!").lower().upper()
=> 'PRINCESS ANNA'
 > name.strip("!").upper().lower()
=> 'princess anna'
 > name.upper()
=> 'PRINCESS ANNA!!'
 >
```

A shortcut for code



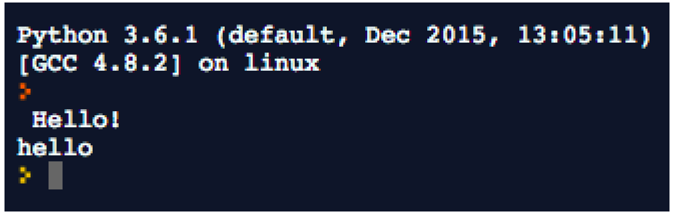
```python
# Chaining Example
# Author: Angelica Lim
# Date: Jan. 17, 2018

# Get the user reply
reply = input()

# Make the reply lowercase
lowercase = reply.lower()

# Remove !,.? characters from the lowercased reply
stripped_lowercase = lowercase.strip("!,.?")

print(stripped_lowercase)
```

==



```python
# Chaining Example
# Author: Angelica Lim
# Date: Jan. 17, 2018

# Get the user reply and make it lowercase
# without extra characters
reply = input().lower().strip("!,.?")

# Print reply
print(reply)
```

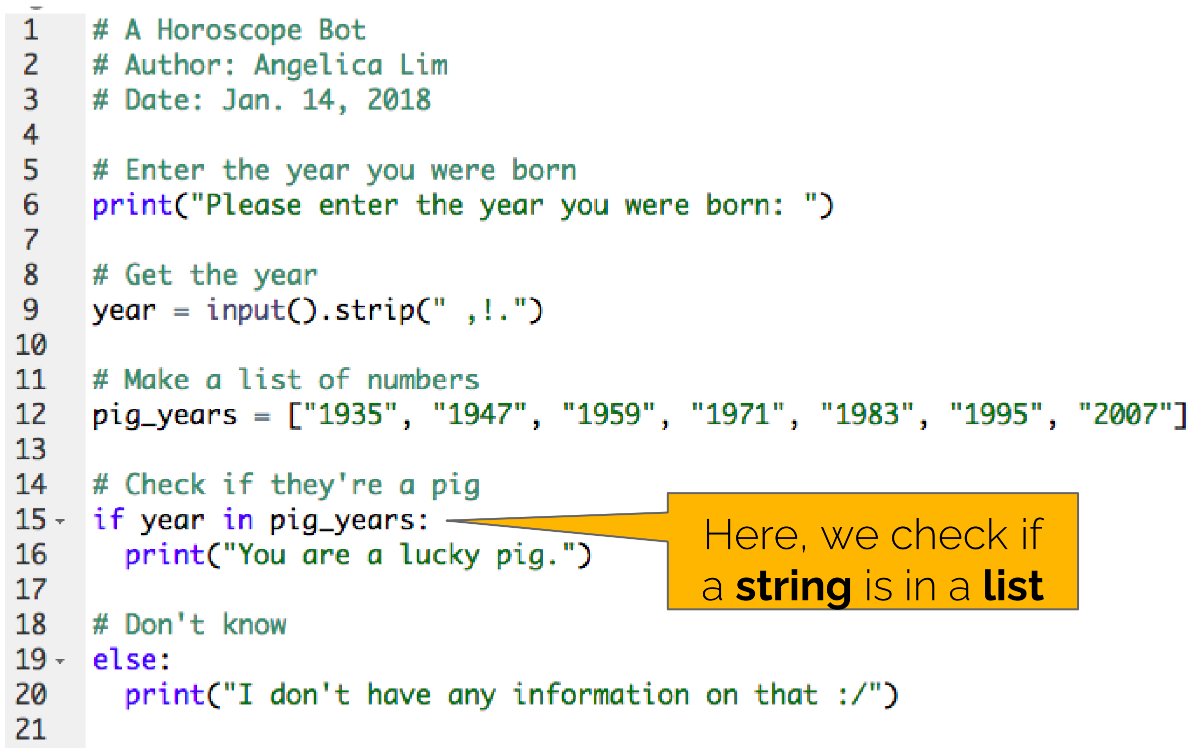
Both produce



```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
 Hello!
hello
>
```

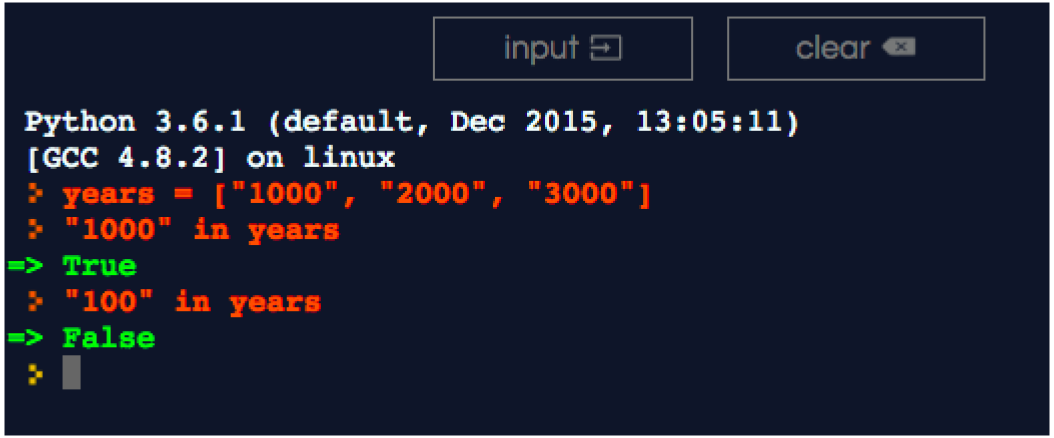
in keyword

The in keyword (list)



```python
# A Horoscope Bot
# Author: Angelica Lim
# Date: Jan. 14, 2018

# Enter the year you were born
print("Please enter the year you were born: ")

# Get the year
year = input().strip(" ,!.")

# Make a list of numbers
pig_years = ["1935", "1947", "1959", "1971", "1983", "1995", "2007"]

# Check if they're a pig
if year in pig_years:
  print("You are a lucky pig.")

# Don't know
else:
  print("I don't have any information on that :/")
```

Here, we check if a **string** is in a **list**



```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
> years = ["1000", "2000", "3000"]
> "1000" in years
=> True
> "100" in years
=> False
>
```

The **in** keyword (string)

```
# Checking if a string is in another string
reply = "I'm good, thanks!"
print("good" in reply)
```

```
True
```

The **in** keyword can also check if a **string** is contained in another **string**.

http://interactivepython.org/courselib/static/thinkcspy/Strings/Theinandnotinoperators.html

Code it!

```python
# Example: Check user's mood
# using `in` keyword

reply = input("How are you feeling?")
if "good" in reply:
    print("I'm glad you feel good.")

if reply == "Bad":
    print("I'm sorry you feel bad.")
```

- Modify your chatbot to make use of:
  - **in** for “Bad”
  - **.lower()**
  - **.strip(“!. “)**

http://interactivepython.org/courselib/static/thinkcspy/Strings/Theinandnotinoperators.html

Let’s **make** a FoodBot

Make a bot that asks you about your favourite food dish. Then it suggests some restaurants in Vancouver with that dish!

Food Bot algorithm

Try to write this in Python at home for practice, without looking!

- Description
  - Make a bot to ask user's favourite food in Vancouver. Then suggest a restaurant.
- Steps
  - Ask the user for a favourite dish (e.g. tempura)
  - Make a category, such as Japanese, with a list of possible dishes.

```
japanese = ["tempura", "sushi", "miso"]
```

  - Then, suggest a Japanese restaurant if the user's favourite dish is Japanese

How would you add another cuisine given other favourite foods?

Today's Review

1. Can we chain together **food = input().lower().strip("!.")**
2. If so, if the user input is **!!Ice cream!!!**, what does **print(food)** output?
3. How do we check that a string is in a list?
4. How do we check that a string is in another string?