Chatbots with Personality

A Robust Bot

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

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

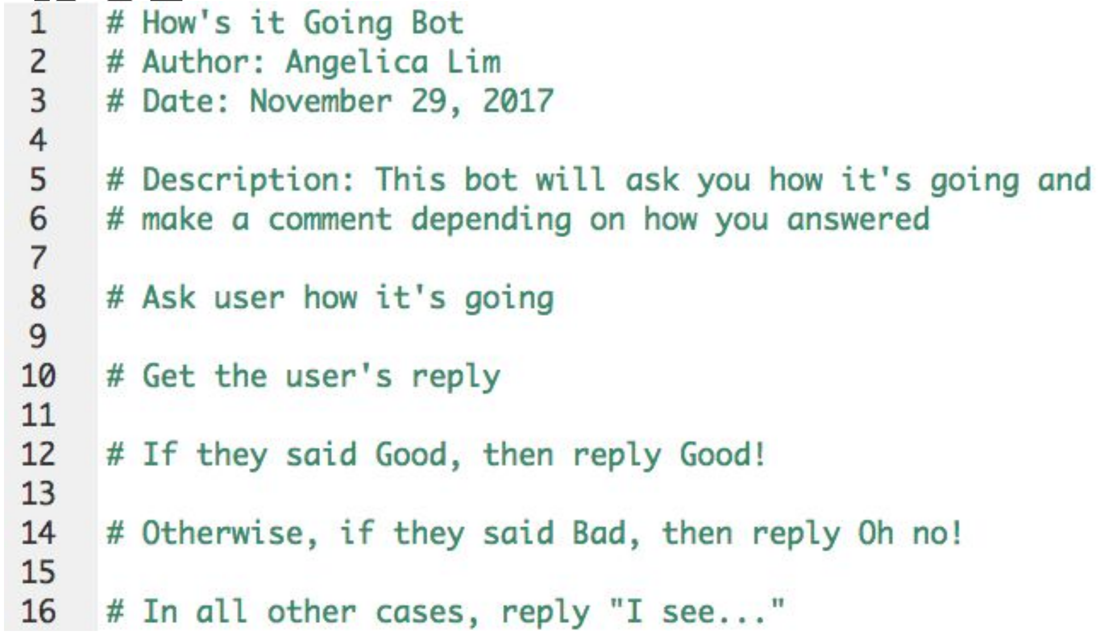
This lesson

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

Today, you'll learn:

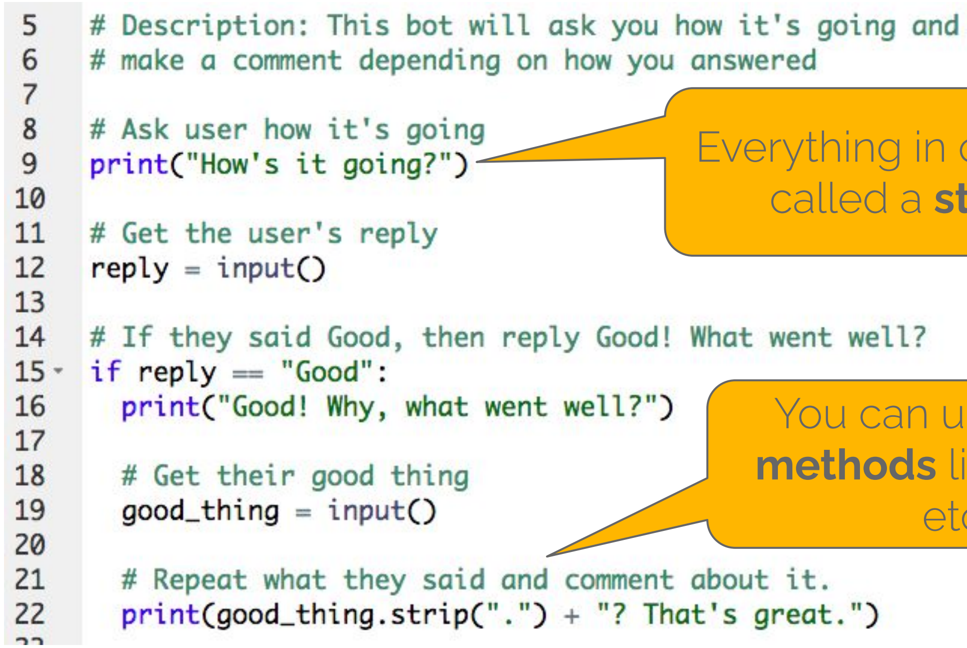
- String methods (**strip**, **lower**, **upper**)
- The **in** keyword

Previously... the How's it Going bot



```
# How's it Going Bot
# Author: Angelica Lim
# Date: November 29, 2017

# 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

# Get the user's reply

# If they said Good, then reply Good!

# Otherwise, if they said Bad, then reply Oh no!

# In all other cases, reply "I see..."
```

String methods

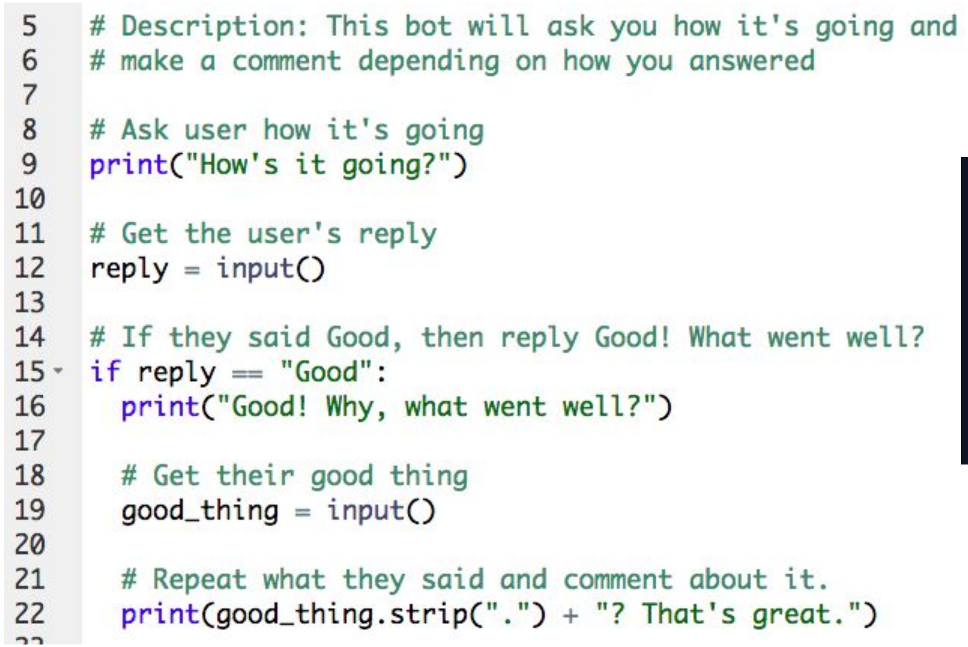


```
# Description: This bot will ask you how it's going and
# make a comment depending on how you answered

# 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.")
```

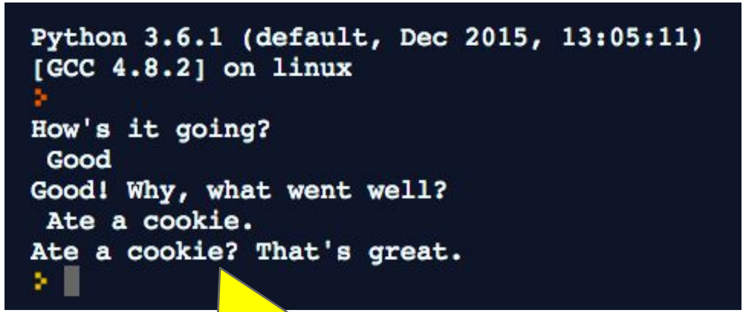
Everything in quotes is called a **string**.

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!

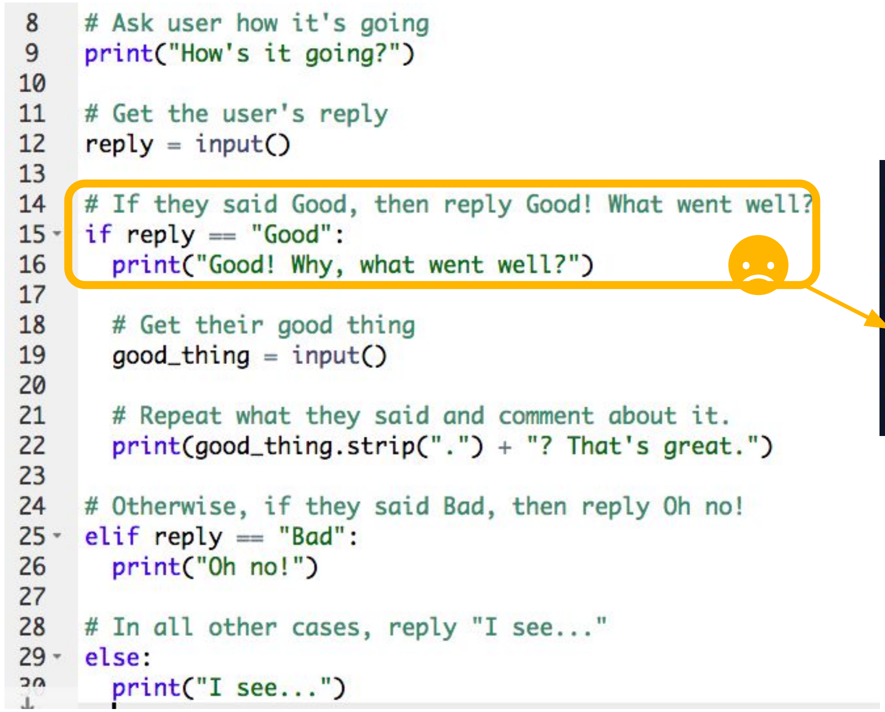
String methods

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

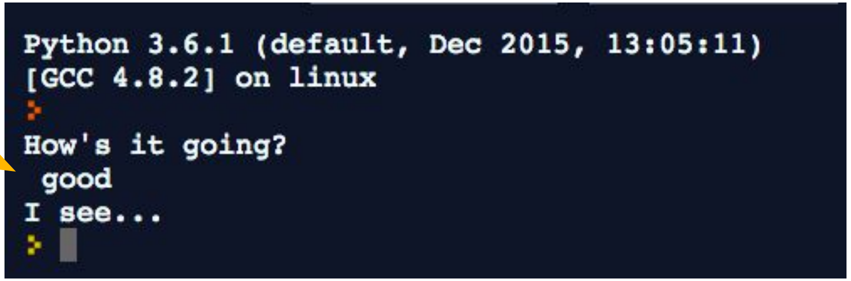
	Example
Remove the characters . ! ? and space at the end of myString	myString.strip(".!? ")
Convert all the letters into lowercase	myString.lower()
Convert all the letters into uppercase	myString.upper()

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

How might we use lower()?

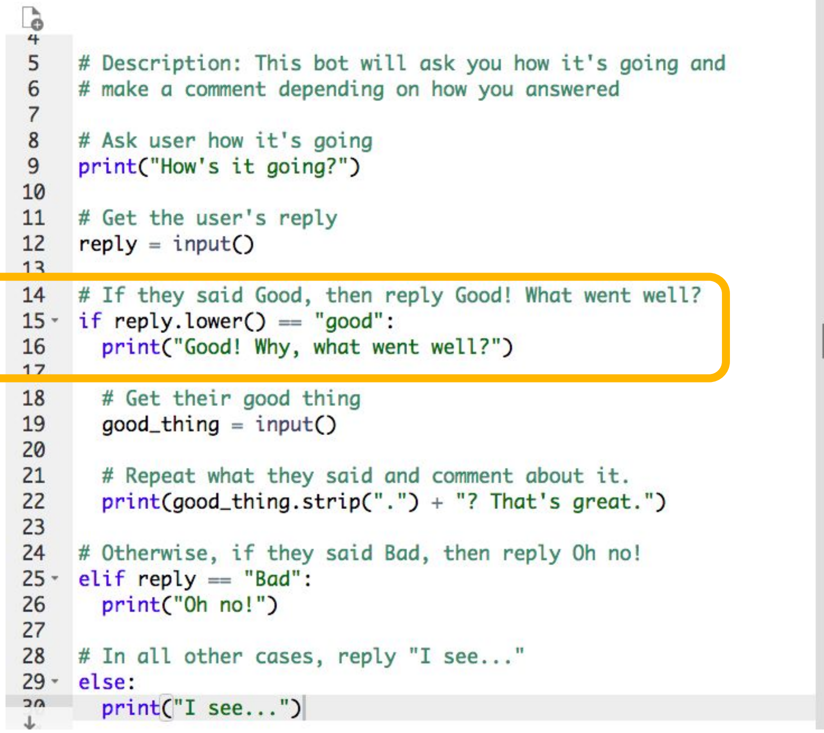


```python
# Ask user how it's going
print("How's it going?")

# 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...")
```

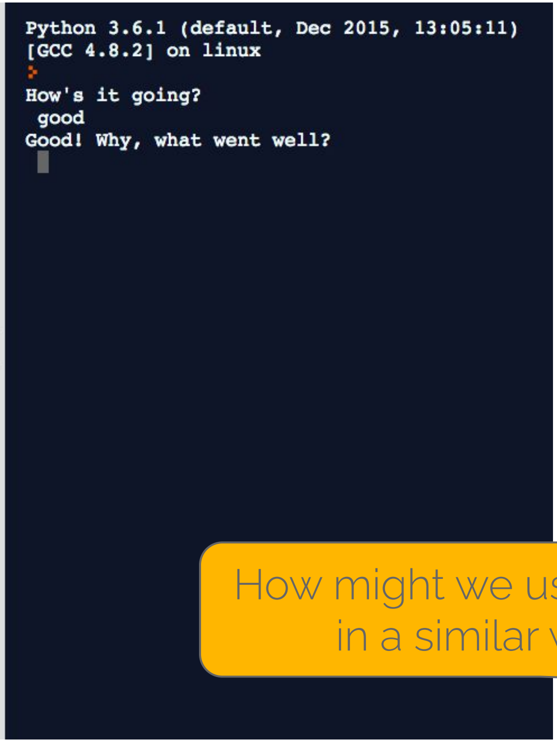


```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
How's it going?
 good
I see...
>
```

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...")
```

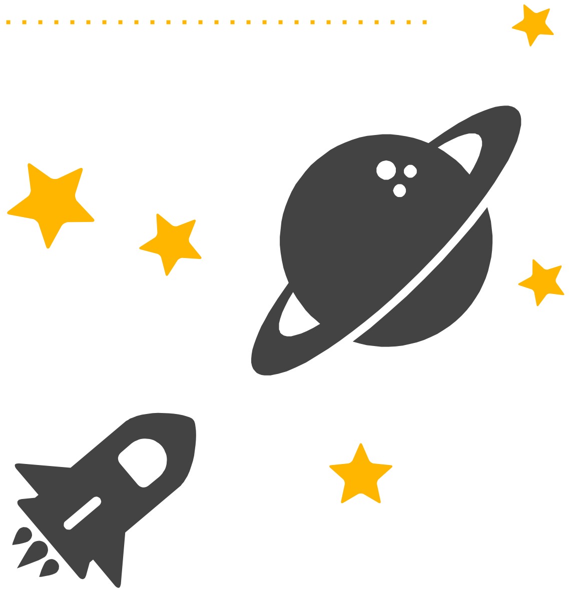


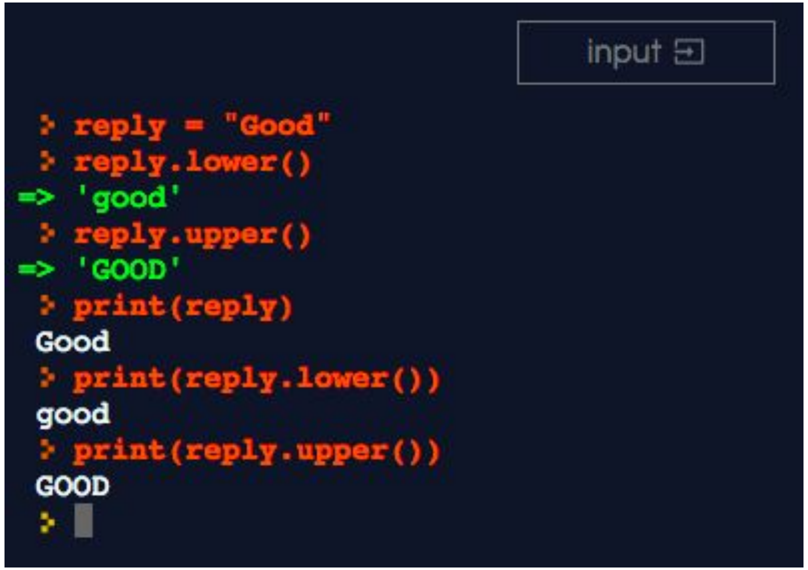
```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux

How's it going?
 good
Good! Why, what went well?
```

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

Your best friend has secrets

REPL: Read Evaluate Print Loop

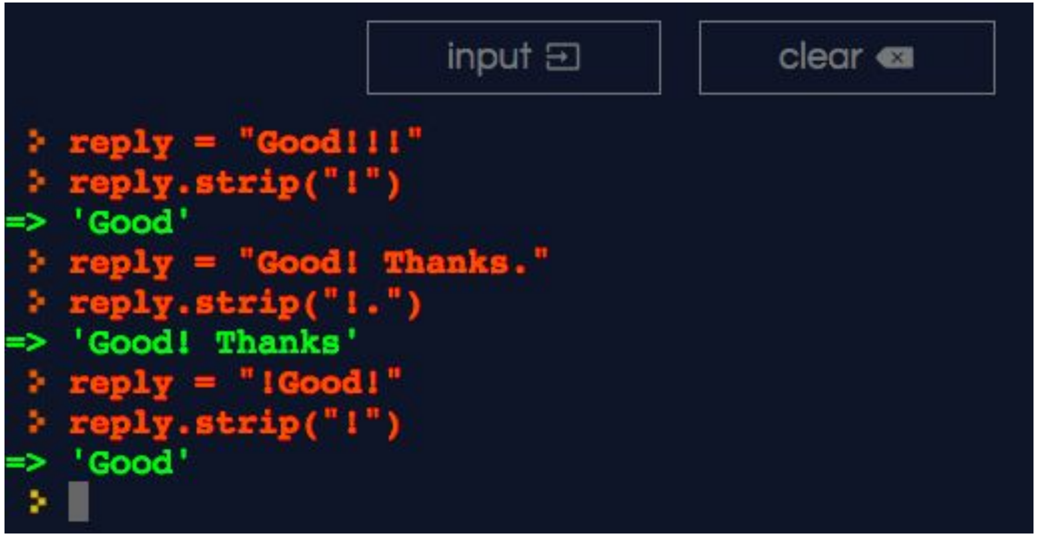




```
> reply = "Good"
> reply.lower()
=> 'good'
> reply.upper()
=> 'GOOD'
> print(reply)
Good
> print(reply.lower())
good
> print(reply.upper())
GOOD
>
```

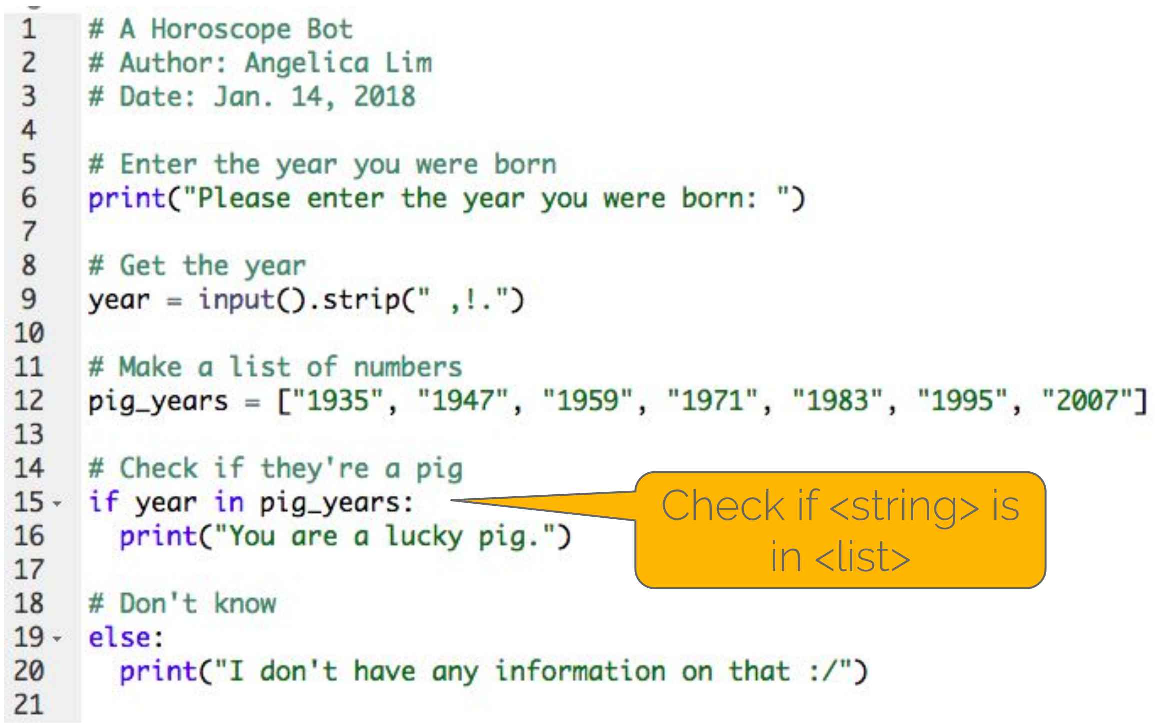
You have an **interactive** Python console to try things out!

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



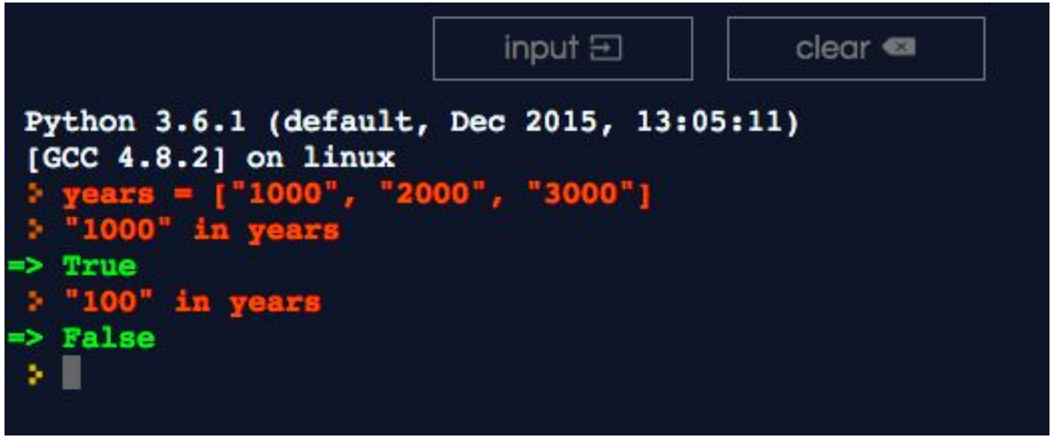
```
> reply = "Good!!!"
> reply.strip("!")
=> 'Good'
> reply = "Good! Thanks."
> reply.strip("!.")
=> 'Good! Thanks'
> reply = "!Good!"
> reply.strip("!")
=> 'Good'
>
```

The in keyword



```
# 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 :/")
```

Check if <string> is in <list>

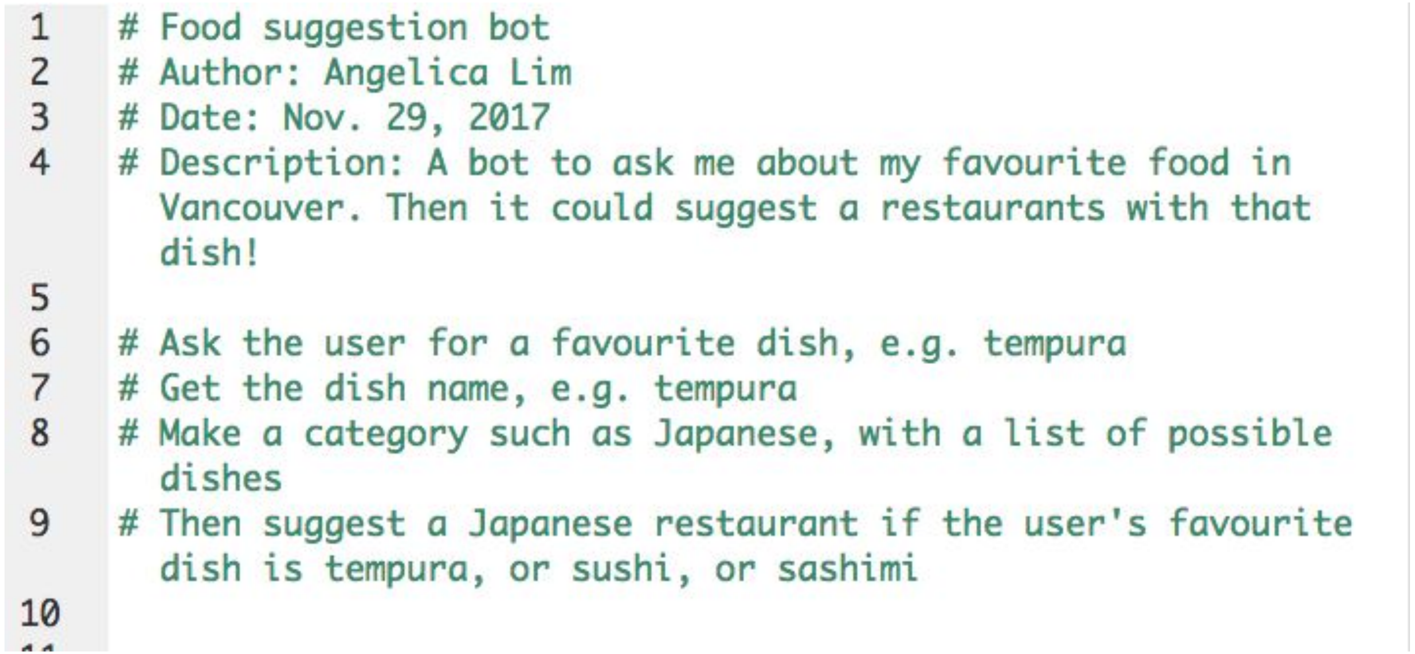


```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
 > years = ["1000", "2000", "3000"]
 > "1000" in years
=> True
 > "100" in years
=> False
 >
```

Let's **make** a FoodBot

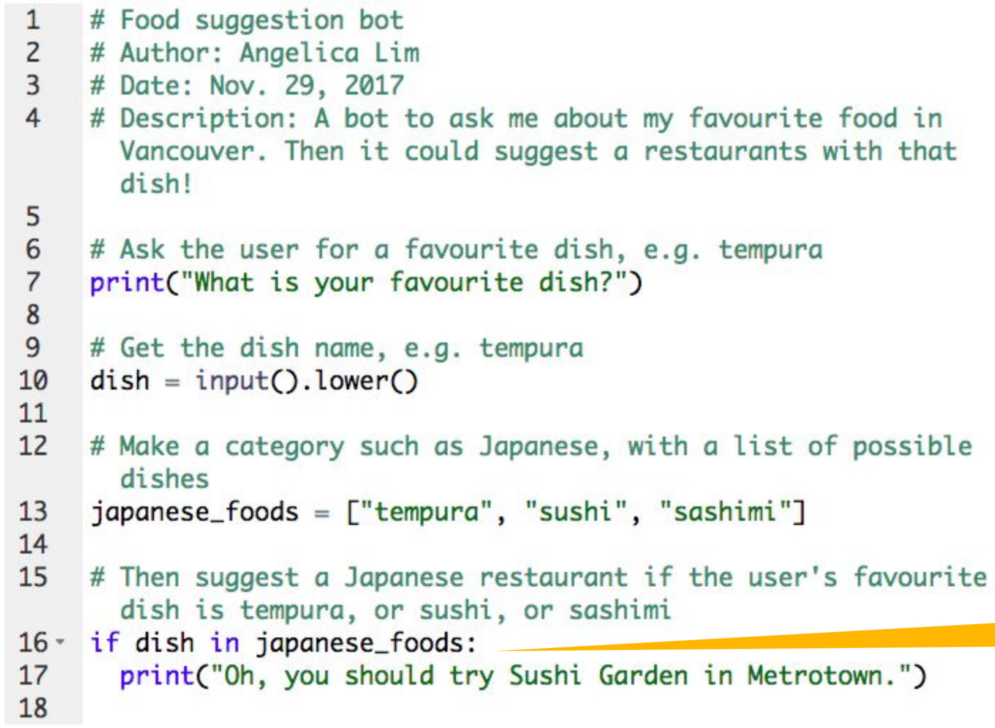
Make a bot that asks you about your favourite food dish. Then it suggests some restaurants in Vancouver with that dish!

Food Bot algorithm



```
# Food suggestion bot
# Author: Angelica Lim
# Date: Nov. 29, 2017
# Description: A bot to ask me about my favourite food in
  Vancouver. Then it could suggest a restaurants with that
  dish!

# Ask the user for a favourite dish, e.g. tempura
# Get the dish name, e.g. tempura
# Make a category such as Japanese, with a list of possible
  dishes
# Then suggest a Japanese restaurant if the user's favourite
  dish is tempura, or sushi, or sashimi

```

Food Bot in Python 3



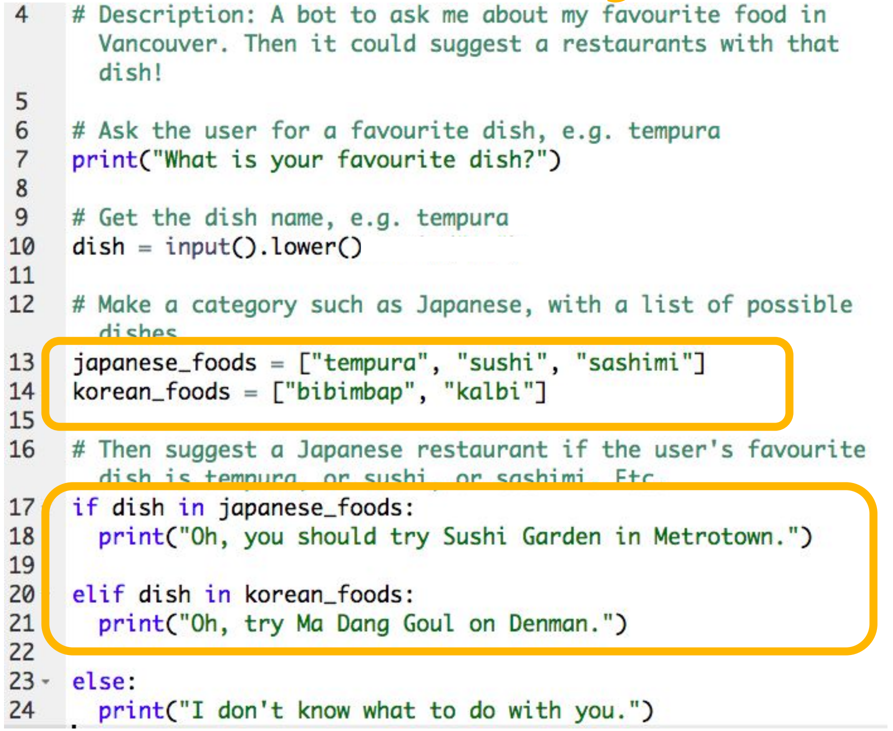
```python
# Food suggestion bot
# Author: Angelica Lim
# Date: Nov. 29, 2017
# Description: A bot to ask me about my favourite food in Vancouver. Then it could suggest a restaurants with that dish!

# Ask the user for a favourite dish, e.g. tempura
print("What is your favourite dish?")

# Get the dish name, e.g. tempura
dish = input().lower()

# Make a category such as Japanese, with a list of possible dishes
japanese_foods = ["tempura", "sushi", "sashimi"]

# Then suggest a Japanese restaurant if the user's favourite dish is tempura, or sushi, or sashimi
if dish in japanese_foods:
  print("Oh, you should try Sushi Garden in Metrotown.")

```

in

in also works with lists

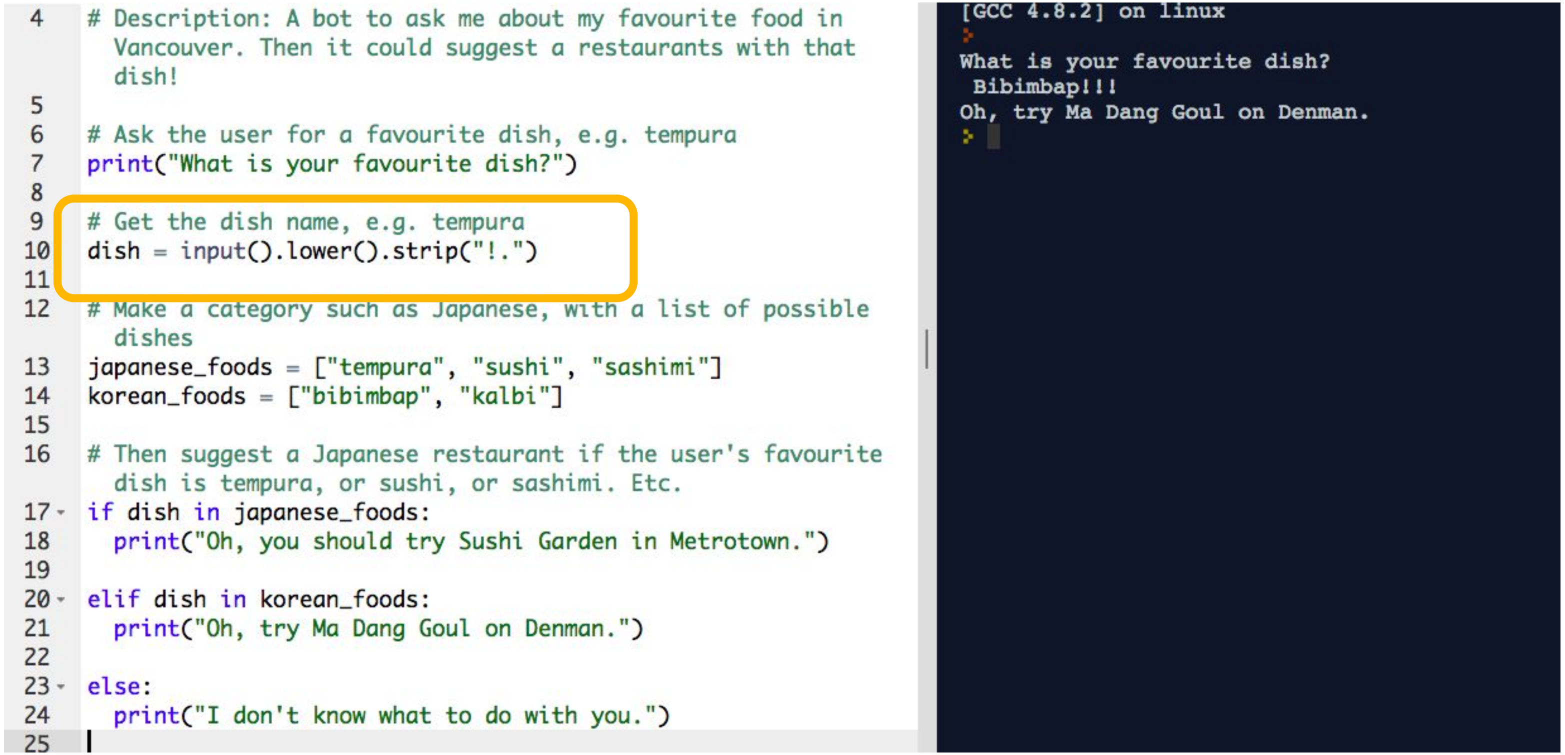
http://interactivepython.org/runestone/static/thinkcspy/Lists/ListMembership.html

Food Bot in Python 3



```
# Description: A bot to ask me about my favourite food in
  Vancouver. Then it could suggest a restaurants with that
  dish!

# Ask the user for a favourite dish, e.g. tempura
print("What is your favourite dish?")

# Get the dish name, e.g. tempura
dish = input().lower()

# Make a category such as Japanese, with a list of possible
  dishes
japanese_foods = ["tempura", "sushi", "sashimi"]
korean_foods = ["bibimbap", "kalbi"]

# Then suggest a Japanese restaurant if the user's favourite
  dish is tempura, or sushi, or sashimi. Etc.
if dish in japanese_foods:
  print("Oh, you should try Sushi Garden in Metrotown.")

elif dish in korean_foods:
  print("Oh, try Ma Dang Goul on Denman.")

else:
  print("I don't know what to do with you.")
```

Test your Food Bot



```python
# Description: A bot to ask me about my favourite food in
  Vancouver. Then it could suggest a restaurants with that
  dish!

# Ask the user for a favourite dish, e.g. tempura
print("What is your favourite dish?")

# Get the dish name, e.g. tempura
dish = input().lower().strip("!.")

# Make a category such as Japanese, with a list of possible
  dishes
japanese_foods = ["tempura", "sushi", "sashimi"]
korean_foods = ["bibimbap", "kalbi"]

# Then suggest a Japanese restaurant if the user's favourite
  dish is tempura, or sushi, or sashimi. Etc.
if dish in japanese_foods:
  print("Oh, you should try Sushi Garden in Metrotown.")

elif dish in korean_foods:
  print("Oh, try Ma Dang Goul on Denman.")

else:
  print("I don't know what to do with you.")

```

```
[GCC 4.8.2] on linux
›
What is your favourite dish?
 Bibimbap!!!
Oh, try Ma Dang Goul on Denman.
›
```

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 create a list of items?
4. How do we check that something is in that list?

For next time

Can you make your bot more robust using the **in** keyword and string methods?