



Chatbots with Personality



Notable Code!

The screenshot shows a code editor with a dark theme. On the left, a code editor window displays the following Python code:

```
#I found a thing  
  
import this  
print(this)
```

On the right, a terminal window shows the output of the code:

```
Python 3.6.1 (default, Dec 2015, 13:05:11)  
[GCC 4.8.2] on linux  
The Zen of Python, by Tim Peters  
  
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!  
<module 'this' from '/usr/local/lib/python3.6/this.py'>
```

I only post here what has been shared in the Slack channel. Please do share!



Question 1

What are 2 kinds of **errors** we should defend against?

Question 2

What does the **strip** function do?



Question 3

What could be wrong with this code?

```
if reply.lower().strip(" ") == "GOOD"  
    print("Good!")
```

Question 4

What does this code output?

```
movies = ["Superman", "Frozen", "X-Men"]  
print("x-men" in movies)
```

More on Chaining



A shortcut for code

```
1 # Chaining Example
2 # Author: Angelica Lim
3 # Date: Jan. 17, 2018
4
5 # Get the user reply and make it lowercase
6 # without extra characters
7 reply = input().lower().strip("!,.?")
8
9 # Print reply
10 print(reply)
```

==

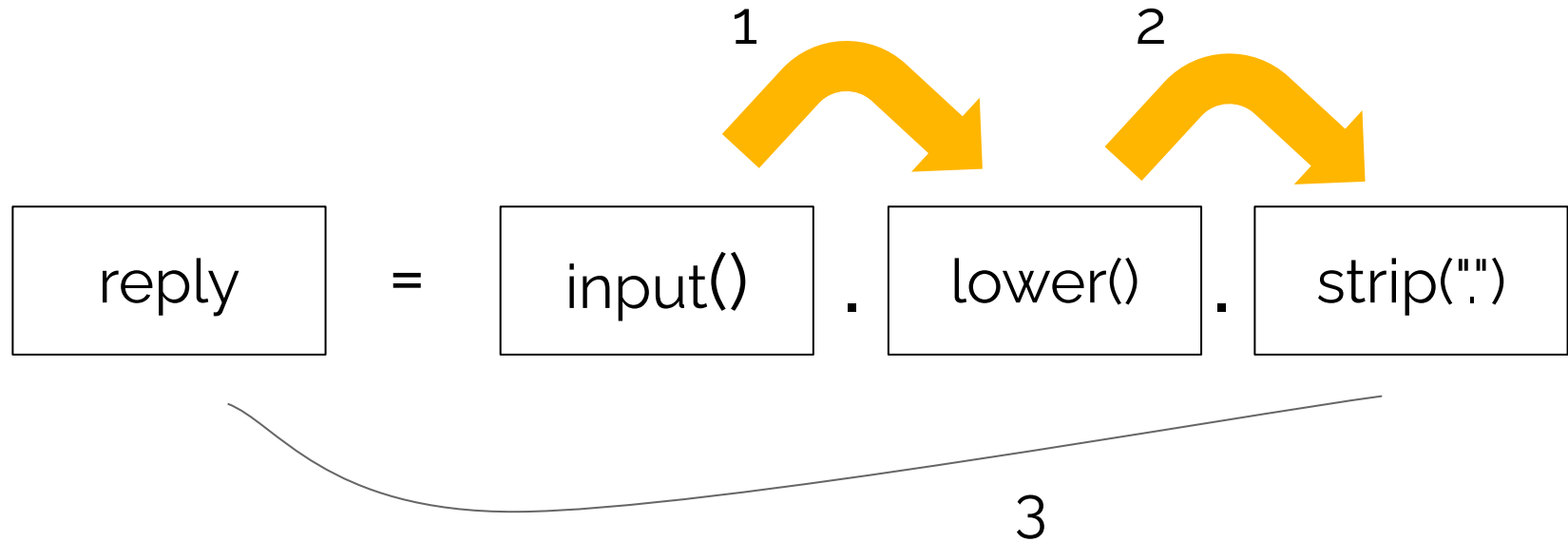
```
1 # Chaining Example
2 # Author: Angelica Lim
3 # Date: Jan. 17, 2018
4
5 # Get the user reply
6 reply = input()
7
8 # Make the reply lowercase
9 lowercase = reply.lower()
10
11 # Remove !,.? characters from the lowercased reply
12 stripped_lowercase = lowercase.strip("!,.?")
13
14 print(stripped_lowercase)
```

Both produce

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



Here's how it works (left to right)



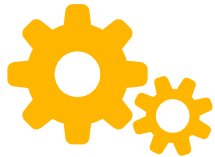


Chaining Examples

```
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 Party Game Bot



This lesson

- Creating lists on-the-fly
- For loop (basic)



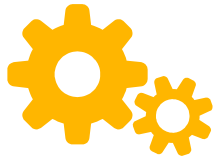
Mind Reader Game Host

This is a 2-player party game, hosted by your chatbot.

Rules

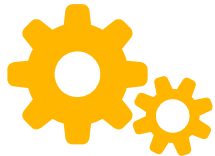
- The 1st player reads a **word**, and secretly enters 3 words they associate with it.
- The 2nd player must then try to guess at least one of the words. If it's a match, they win!





Mind Reader Game

```
1 # How Well Do You Know Me?
2 # Author: Angelica Lim
3 # Date: Jan. 16, 2018
4
5 # This is a 2-player game where you The 1st player reads a word,
6 # and secretly enters 3 words they associate with it.
7 # The 2nd player must then try to guess at least one of the words.
8 # If it's a match, they win!
9
10
11 # Introduce the game
12
13 # Ask the first player to enter 3 words associated with a given word
14
15 # Clear the screen
16
17 # Ask the 2nd player to guess an associated word
18
19 # Check if they match and tell them if they won!
20
```



Mind Reader Game

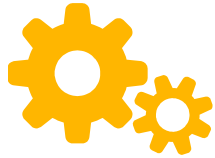
```
1 # Mind Reader
2 # Author: Angelica Lim
3 # Date: Jan. 16, 2018
4 # This is a 2-player game where you The 1st player reads a word,
5 # and secretly enters 3 words they associate with it.
6 # The 2nd player must then try to guess at least one of the words.
7 # If it's a match, they win!
8
9 import replit
10
11 # Introduce the game
12 print("Welcome to Mind Reader!")
13
14 # Ask the first player to enter 3 words associated with a given word
15 print("Player 1, enter 3 words you think when I say cat:")
16
17 # Get the 3 words
18 first = input("First word:")
19 second = input("Second word:")
20 third = input("Third word:")
21
22 # Clear the screen
23 replit.clear()
24
25 # Ask the 2nd player to guess an associated word
26 print("Player 2, what is one word you think Player 1 associates with cat?")
27 guess = input()
28
29 # Check if they match and tell them if they won!
30 if guess in [first, second, third]:
31     print("You've got it!")
32
```

Try playing it with 2 people! :)

Now, can you think of a way to choose the word randomly?

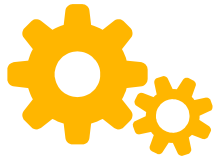
Yes, you can construct lists on-the-fly based on input!

Mind Reader Game



What if they got it wrong?

```
1 # Mind Reader Game
2 # Author: Angelica Lim
3 # Date: Jan. 16, 2018
4 # This is a 2-player game where you The 1st player reads a word,
5 # and secretly enters 3 words they associate with it.
6 # The 2nd player must then try to guess at least one of the words.
7 # If it's a match, they win!
8
9 import replit
10 import random
11
12 # Introduce the game
13 print("Welcome to Mind Reader!")
14
15 words = ["cat", "dog", "house", "apple"]
16 word = random.choice(words)
17
18 # Ask the first player to enter 3 words associated with a given word
19 print("Player 1, enter 3 words you think when I say " + word + "?")
20
21 # Get the 3 words
22 first = input("First word:")
23 second = input("Second word:")
24 third = input("Third word:")
25
26 # Clear the screen
27 replit.clear()
28
29 # Ask the 2nd player to guess an associated word
30 print("Player 2, what is one word you think Player 1 associates with " + word + "?")
31 guess = input()
32
33 # Check if they match and tell them if they won!
34 if guess in [first, second, third]:
35     print("You've got it!")
```



Mind Reader Game

How do we play the game multiple times?

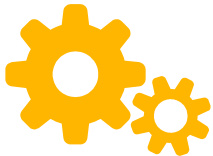
```
9 import replit
10 import random
11
12 # Introduce the game
13 print("Welcome to Mind Reader!")
14
15 words = ["cat", "dog", "house", "apple"]
16 word = random.choice(words)
17
18 # Ask the first player to enter 3 words associated with a given word
19 print("Player 1, enter 3 words you think when I say " + word)
20
21 # Get the 3 words
22 first = input("First word:")
23 second = input("Second word:")
24 third = input("Third word:")
25
26 # Clear the screen
27 replit.clear()
28
29 # Ask the 2nd player to guess an associated word
30 print("Player 2, what is one word you think Player 1 associates with " + word)
31 guess = input()
32
33 # Check if they match and tell them if they won!
34 if guess in [first, second, third]:
35     print("You've got it!")
36 else:
37     print("No match! They said " + first + ", " + second + " and " + third + "!")
38
```

Loops

For doing things over, and over, and over...!

<http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/TheforLoop.html>





Note: We removed the random statement here. Instead, we'll loop through all the words, one by one.

for

in

This means: **for** every word **in** the *words* list, do the following (indented)

```
4 # This is a 2-player game where you the 1st player reads a word,
5 # and secretly enters 3 words they associate with it.
6 # The 2nd player must then try to guess at least one of the words.
7 # If it's a match, they win!
8
9 import replit
10
11 # Introduce the game
12 print("Welcome to Mind Reader!")
13
14 # Create a list of words
15 words = ["cat", "dog", "apple"]
16
17 # Loop through all the items in words
18 for word in words:
19
20     # Ask the first player to enter 3 words associated with a given word
21     print("Player 1, enter 3 words you think when I say " + word)
22
23     # Get the 3 words
24     first = input("First word:")
25     second = input("Second word:")
26     third = input("Third word:")
27
28     # Clear the screen
29     replit.clear()
30
31     # Ask the 2nd player to guess an associated word
32     print("Player 2, what is one word you think Player 1 associates with " + word)
33     guess = input()
34
35     # Check if they match and tell them if they won!
36     if guess in [first, second, third]:
37         print("You've got it!")
38     else:
39         print("No match! They said " + first + ", " + second + " and " + third + "!")
40
```



A Nosy Question Bot

Nosy Question Bot

```
1 # A Nosy Question Bot
2 # Author: Angelica Lim
3 # Date: Jan. 17, 2018
4 # Description: Asks you from a list of questions.
5
6 import random
7
8 # Introduction
9 print("Hello! This is Nosy Question Bot. I'd like to ask you some questions! Here we go.")
10
11 # Create questions list
12 questions = ["Given the choice of anyone in the world, whom would you want as a dinner guest?", "Would you like to be famous? In what way?", "Before making a telephone call, do you ever rehearse what you are going to say? Why?", "What would constitute a "perfect" day for you?"]
13
14 # Make responses list
15 responses = ["Interesting.", "I see!", "Fascinating."]
16
17 # Make a loop that will ask the questions from the list
18 for question in questions:
19     # Print the question
20     print(question)
21
22     # Get the response
23     input()
24
25     # Make a reply
26     print(random.choice(responses))
27
28
29 # End it off
30 print("You are a super interesting person! Thanks for replying to my questions.")
```



Let's **review** some concepts

What are the **keywords** needed to make a **loop**?

In a loop, what do you need to do to the code that you want to repeat?

True or false? Methods can be **chained** from **left** to **right**.

Week 2 Exercise

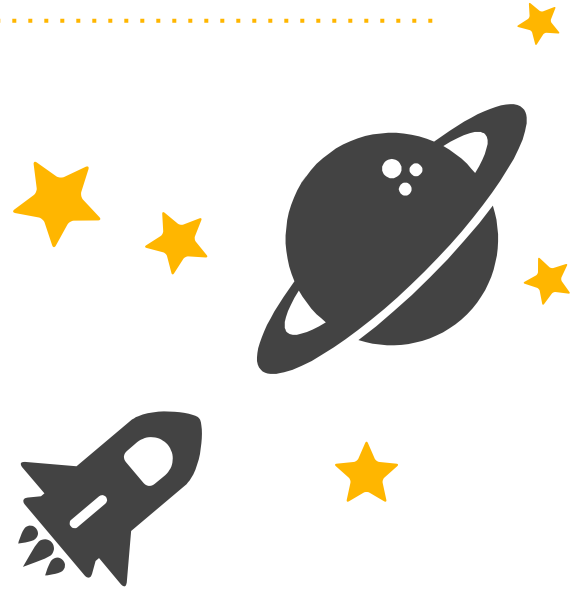


Create a ChatBot with a Loop!

Create a chatbot with a loop based on the examples in class in <http://repl.it>. The chatbot should start when you click on the Run button. It must use at least **2 string methods** for robustness, and a **for loop**. Design your algorithm in English first, then translate it to Python code. Tips: Code bit by bit, and test small pieces as you go. Add your robustness at the end.



Extras

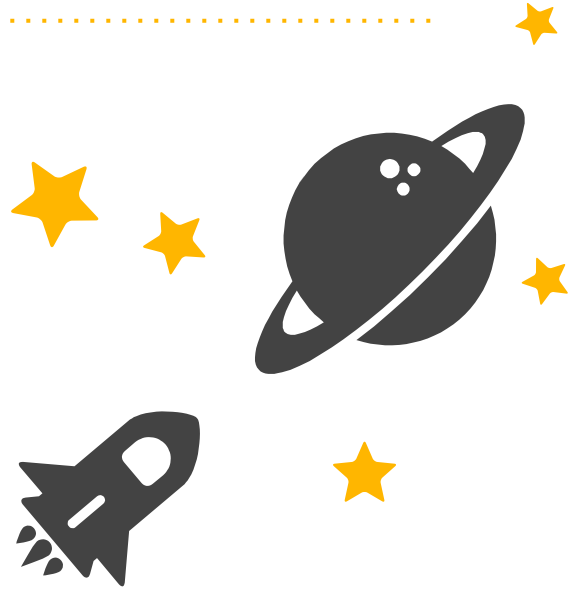


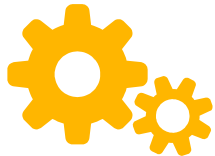
Integer

String, list, Boolean... and now Integer!

<http://interactivepython.org/runestone/static/thinkcspy/SimplePythonData/ValuesandDataTypes.html>

(Strings and Integers)





Keeping Score

The variable **score** is of type **integer**

You can only concatenate strings with strings, so you need to convert your integer to a **string** type using **str()**

```
9 import replit
10
11 # Introduce the game
12 print("Welcome to Mind Reader!")
13
14 words = ["cat", "dog", "apple"]
15 score = 0
16
17 for word in words:
18
19     # Ask the first player to enter 3 words associated with a given word
20     print("Player 1, enter 3 words you think when I say " + word)
21
22     # Get the 3 words
23     first = input("First word:")
24     second = input("Second word:")
25     third = input("Third word:")
26
27     # Clear the screen
28     replit.clear()
29
30     # Ask the 2nd player to guess an associated word
31     print("Player 2, what is one word you think Player 1 associates with " + word)
32     guess = input()
33
34     # Check if they match and tell them if they won!
35     if guess in [first, second, third]:
36         print("You've got it!")
37
38         # Add to their score
39         score = score + 1
40     else:
41         print("No match! They said " + first + ", " + second + " and " + third + "!")
42
43 # At the end, print out their score
44 print("You got " + str(score) + " right!")
45
```

This is how you can add to yourself, or "accumulate"

Nosy Question Bot

You can give the option to break out of the loop using an **if** statement and **break** keyword

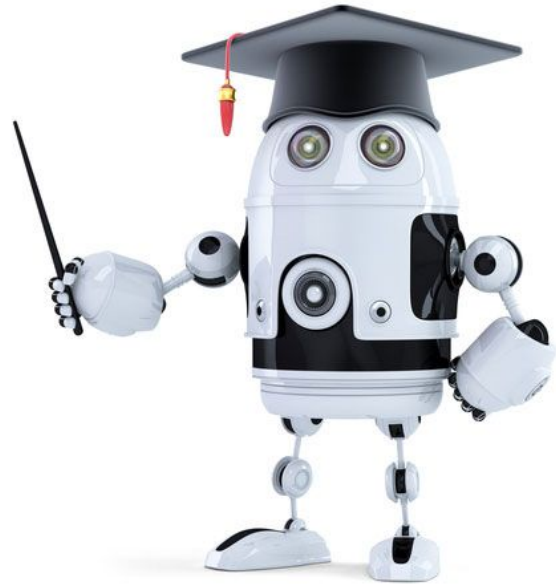
```
1 # A Nosy Question Bot
2 # Author: Angelica Lim
3 # Date: Jan. 17, 2018
4 # Description: Asks you from a list of questions.
5
6 import random
7
8 # Introduction
9 print("Hello! This is Nosy Question Bot. I'd like to ask you some questions! Here we go.")
10
11 # Create questions list
12 questions = ["Given the choice of anyone in the world, whom would you want as a dinner guest?", "Would you like to be famous? In what way?", "Before making a telephone call, do you ever rehearse what you are going to say? Why?", "What would constitute a \"perfect\" day for you?"]
13
14 # Make responses list
15 responses = ["Interesting.", "I see!", "Fascinating."]
16
17 # Make a loop that will ask the questions from the list
18 for question in questions:
19     # Print the question
20     print(question)
21
22     # Get the response
23     reply = input()
24
25     # If they say stop, break out of the loop
26     if reply == "stop":
27         break
28
29     # Make a reply
30     print(random.choice(responses))
31
32 # End it off
33 print("You are a super interesting person! Thanks for replying to my questions.")
34
```

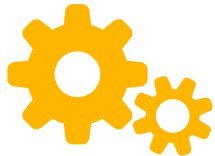


Flashcard Bot

So, you want to help yourself study for a test?

Make a flashcard bot that will test your knowledge and provide you encouragement!





A Training Bot

```
1 # A FlashCard Bot
2 # Author: Angelica Lim
3 # Date: Jan. 17, 2018
4
5 # Description: Asks you from a list of questions. If you type "Done", it
6 # will exit
7
8 # Introduction
9
10 # Create question list
11
12 # Make a loop that will ask the questions from the list
13
14 # If the user types Done, exit
```

Will this produce an error?

```
points = 5
```

```
print("You got " + points + "
right!")
```

How do we break
out of a loop?