**CHAT BOT**

**Echo Bot I**

Hello, World!

* You'll begin learning how to build chat bots in Python by writing two functions to build the simplest bot possible: Echo Bot.
* **Echo Bot just responds by replying with the same message it receives.**
* In this exercise, you'll define a function that responds to a user's message

INSTRUCTIONS

* Write a function called respond () with a single parameter message which returns the bot's response. To do this, concatenate the strings "I can hear you! You said: “ and message.
* Store the concatenated strings in bot \_message, and return this result.

**Python code**

# Define a function that responds to a user's message: respond

* def respond(message):

# Concatenate the user's message to the end of a standard bot response

* bot\_message = "I can hear you! You said: " + message

# Return the result

* return bot\_message

# Test function

* print(respond("hello!"))

# Echo Bot II

* Having written your respond() function, you'll now define a function called send \_message() with a single parameter message which logs the message and the bot's response

**INSTRUCTIONS**

* Use the user \_template string's .format() method to include the user's message into the user template, and print the result.
* Call the respond () function with the message passed in and save the result as response.
* Log the bot's response using the bot \_template string's .format() method.
* Send the message "hello" to the bot.

**Python Code**

# create templates

* bot\_template = "BOT : {0}"
* user\_template = "USER : {0}"

# Define a function that sends a message to the bot: send\_message

* def send\_message(message):

# Print user\_template including the user\_message

* print(user\_template.format(message))

# Get the bot's response to the message

* response = respond(message)

# Print the bot template including the bot's response.

* print(bot\_template.format(response))

# Send a message to the bot

* send\_message("hello")

# Chitchat

* Now you're going to leave the simple EchoBot behind and create a bot which can answer simple questions such as "What's your name?" and "What's today's weather?"
* You'll use a dictionary with these questions as keys and the correct responses as values.
* This means the bot will only respond correctly if the message matches exactly, which is a big limitation. In later exercises you will create much more robust solutions.
* The send\_message() function has already been defined for you, as well as the bot\_template and user \_template variables.

**INSTRUCTIONS:[1/2]**

* Define a respond() function which takes in a message argument, checks if the message has a pre-defined response, and returns the response in the responses dictionary if there is a match, or the "default" message otherwise.

**PYTHON CODE**

# Define variables

* name = "Greg"
* weather = "cloudy"

# Define a dictionary with the predefined responses

* responses = {
* "what's your name?": "my name is {0}".format(name),
* "what's today's weather?": "the weather is {0}".format(weather),
* "default": "default message"
* }

# Return the matching response if there is one, default otherwise

* def respond(message):

# Check if the message is in the responses

* if message in responses:

# Return the matching message

* bot\_message = responses[message]
* else:

# Return the "default" message

* bot\_message = responses["default"]
* return bot\_message

**INSTRUCTION: [2/2]**

* Well Done! Your bot is now able to answer some simple questions. Hit 'Run Code' and call send\_message()(which utilizes the new respond() function) in the IPython Shell with the following questions:
  + "what's today's weather?"
  + "what's your name?"
  + "what's your favorite color?"
* Hit 'Submit Answer' when you are done.

**PYTHON CODE**

* send\_message("what's today's weather?")
* send\_message("what's your name?")
* send\_message("what's your favourite color?")

# Adding variety

* It can get a little boring hearing the same old answers over and over. In this exercise, you'll add some variation. If you ask your bot how it's feeling, the likelihood that it responds with "oh I'm great!" or "I'm very sad today" should be equal.
* Here, you'll use the random module - specifically random.choice(ls) - which randomly selects an element from a list ls.
* A dictionary called responses, which maps each message to a list of possible responses, has been defined for you.

**INSTRUCTIONS ½**

* Import the random module.
* If the message is in responses, use random.choice() in the respond() function to choose a random matching response.
* If the message is not in responses, choose a random default response.

**PYTHON CODE**

# Import the random module

* import random
* name = "Greg"
* weather = "cloudy"

# Define a dictionary containing a list of responses for each message

* responses = {
* "what's your name?": [
* "my name is {0}".format(name),
* "they call me {0}".format(name),
* "I go by {0}".format(name)
* ],
* "what's today's weather?": [
* "the weather is {0}".format(weather),
* "it's {0} today".format(weather)
* ],
* "default": ["default message"]
* }

# Use random.choice() to choose a matching response

* def respond(message):

# Check if the message is in the responses

* if message in responses:

# Return a random matching response

* bot\_message = random.choice(responses[message])
* else:

# Return a random "default" response

* bot\_message = random.choice(responses["default"])
* return bot\_message

**INSTRUCTIONS 2/2**

* Great job! Adding some variety makes your bot much more fun to talk to. Now, hit 'Run Code' and use send\_message() (which utilizes the new respond() function) to ask the bot "what's your name?" 3 times.
* Hit 'Submit Answer' when you're done.

responses = pd.DataFrame({'statement': ['tell me more!', 'why do you think that?', 'how long have you felt this way?', 'I find that extremely interesting', 'can you back that up?', 'oh wow!', ':)'], 'question': ["I don't know :(", 'you tell me!']})

# ELIZA I: asking questions

* Asking questions is a great way to create an engaging conversation. Here, you'll create the very first hint of ELIZA's famous personality, by responding to statements with a question and responding to questions with answers.
* A dictionary of responses with "question" and "statement" as keys and lists of appropriate responses as values has already been defined for you. Explore this in the Shell with responses.keys() and responses["question"]

**Instructions**

* Define a respond() function which takes in message as an argument, and uses the string's .endswith() method to check if a message ends with a question mark.
* If the message does end with a question mark, choose a random "question" from the responses dictionary. Else, choose a random "statement" from the responses.
* Send the bot multiple messages with and without a question mark - these have been provided for you. If you want to experiment further in the Shell, be sure to first hit 'Run Code'.

**PYTHON CODE**

* import random
* def respond(message):

# Check for a question mark

* if message.endswith("?"):

# Return a random question

* return random.choice(responses["question"])

# Return a random statement

* return random.choice(responses["statement"])

# Send messages ending in a question mark

* send\_message("what's today's weather?")
* send\_message("what's today's weather?")

# Send messages which don't end with a question mark

* send\_message("I love building chatbots")
* send\_message("I love building chatbots")

**rules = pd.DataFrame({**

**'I want (.\*)': ['What would it mean if you got {0}', 'Why do you want {0}',"What's stopping you from getting {0}"],**

**'do you remember (.\*)': ['Did you think I would forget {0}',"Why haven't you been able to forget {0}",'What about {0}', 'Yes .. and?'],**

**'do you think (.\*)': ['if {0}? Absolutely.', 'No chance'],**

**'if (.\*)': ["Do you really think it's likely that {0}", 'Do you wish that {0}','What do you think about {0}', 'Really--if {0}']})**

**ELIZA II: Extracting key phrases**

* The really clever thing about ELIZA is the way the program *appears* to understand what you told it by occasionally including phrases uttered by the user in its responses.
* In this exercise, you will match messages against some common patterns and extract phrases using re.search(). A dictionary called rules has already been defined, which matches the following patterns:
  + "do you think (.\*)"
  + "do you remember (.\*)"
  + "I want (.\*)"
  + "if (.\*)"
* Inspect this dictionary in the Shell before starting the exercise.

**INSTRUCTIONS**

* Iterate over the rules dictionary using its .items()method, with pattern and responses as your iterator variables.
* Use re.search() with the pattern and message to create a match object.
* If there is a match, use random.choice() to pick a response.
* If '{0}' is in that response, use the match object's .group() method with index 1 to retrieve a phrase.

**PYTHON CODE**

# Define match\_rule()

* def match\_rule(rules, message):
* response, phrase = "default", None

# Iterate over the rules dictionary

* for pattern, responses in rules.items():

# Create a match object

* match = re.search(pattern, message)
* if match is not None:

# Choose a random response

* response = random.choice(responses)
* if '{0}' in response:
  + phrase = match.group(1)

# Return the response and phrase

* return response.format(phrase)

# Test match\_rule

* print(match\_rule(rules, "do you remember your last birthday"))

# ELIZA III: Pronouns

* To make responses grammatically coherent, you'll want to transform the extracted phrases from first to second person and vice versa. In English, conjugating verbs is easy, and simply swapping "me" and 'you', "my" and "your" works in most cases.
* In this exercise, you'll define a function called replace\_pronouns() which uses re.sub() to map "me" and "my" to "you" and "your" (and vice versa) in a string.

**INSTRUCTIONS**

* If 'me' is in message, use re.sub() to replace it with 'you'.
* If 'my' is in message, replace it with 'your'.
* If 'your' is in message, replace it with 'my'.
* If 'you' is in message, replace it with 'me'

**PYTHON CODE**

# Define replace\_pronouns()

* def replace\_pronouns(message):
* message = message.lower()
* if 'me' in message:

# Replace 'me' with 'you'

* return re.sub('me','you',message)
* if 'my' in message:

# Replace 'my' with 'your'

* return re.sub('my','your',message)
* if 'your' in message:

# Replace 'your' with 'my'

* return re.sub('your','my',message)
* if 'you' in message:

# Replace 'you' with 'me'

* return re.sub('you','me',message)
* return message
* print(replace\_pronouns("my last birthday"))
* print(replace\_pronouns("when you went to Florida"))
* print(replace\_pronouns("I had my own castle"))

# ELIZA IV: Putting it all together

* Now you're going to put everything from the previous exercises together and experience the magic! The match\_rule(), send\_message(), and replace\_pronouns() functions have already been defined, and the rules dictionary is available in your workspace.
* Your job here is to write a function called respond() with a single argument message which creates an appropriate response to be handled by send\_message()

**INSTRUCTIONS**

* Get a response and phrase by calling match\_rule() with the rules dictionary and message.
* Check if the response is a template by seeing if it includes the string '{0}'. If it does:
  + Use the replace\_pronouns() function on phrase.
  + Include the phrase by using .format() on response and overriding the value of response

**PYTHON CODE**

# Define respond()

* def respond(message):

# Call match\_rule

* response, phrase = match\_rule(rules, message)
* if '{0}' in response:

# Replace the pronouns in the phrase

* phrase = replace\_pronouns(phrase)

# Include the phrase in the response

* response = response.format(phrase)
* return response

# Send the messages

* send\_message("do you remember your last birthday")
* send\_message("do you think humans should be worried about AI")
* send\_message("I want a robot friend")
* send\_message("what if you could be anything you wanted")