What is Chatbot?

“A **chatbot** (also known as a **talkbot**, **chatterbot**, **Bot**, **IM bot**, **interactive agent**, or **Artificial Conversational Entity**) is a [computer program](https://en.wikipedia.org/wiki/Computer_program) or an [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) which conducts a [conversation](https://en.wikipedia.org/wiki/Conversation) via auditory or textual methods.

Such programs are often designed to convincingly simulate how a human would behave as a conversational partner, thereby passing the [Turing test](https://en.wikipedia.org/wiki/Turing_test). Chatbots are typically used in [dialog systems](https://en.wikipedia.org/wiki/Dialog_system) for various practical purposes including customer service or information acquisition. Some chatterbots use sophisticated [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) systems, but many simpler systems scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a [database](https://en.wikipedia.org/wiki/Database).”

Chatbots are not very new, one of the foremost of this kind is ELIZA, which was created in the early 1960s and is worth exploring. In order to successfully build a conversational engine, it should take care of the following things:

1. Understand who is the target audience  
2. Understand the Natural Language of the communication.   
3. Understand the intent or desire of the user  
4. provide responses that can answer the user

First chatbot

import time

import random

name = input("Hello, what is your name? ")

time.sleep(2)

print("Hello " + name)

feeling = input("How are you today? ")

time.sleep(2)

if "good" in feeling:

print("I'm feeling good too!")

else:

print("I'm sorry to hear that!")

time.sleep(2)

favcolour = input("What is your favourite colour? ")

colours = ["Red","Green","Blue"]

time.sleep(2)

print("My favourite colour is " + random.choice(colours))

Second Chat

NLTK Library

NLTK has a module, nltk.chat, which simplifies building these engines by providing a generic framework.

Import from nltk.chat.util:

**Chat**: This is a class that has all the logic that is used by the chatbot.

**Reflections**: This is a dictionary that contains a set of input values and its corresponding output values. It is an optional dictionary that you can use. You can also create your own dictionary in the same format as below and use it in your code. If you check nltk.chat.util, you will see its values as below:

reflections = {  
 "i am" : "you are",  
 "i was" : "you were",  
 "i" : "you",  
 "i'm" : "you are",  
 "i'd" : "you would",  
 "i've" : "you have",  
 "i'll" : "you will",  
 "my" : "your",  
 "you are" : "I am",  
 "you were" : "I was",  
 "you've" : "I have",  
 "you'll" : "I will",  
 "your" : "my",  
 "yours" : "mine",  
 "you" : "me",  
 "me" : "you"  
}

You can also create your own reflections dictionary in the same format as above and use it in your code. Here is an example for this:

my\_dummy\_reflections= {  
 "go" : "gone",  
 "hello" : "hey there"  
}

and use it as :

chat = Chat(pairs, my\_dummy\_reflections)

Using above concept from python’s NLTK library, lets build a simple chatbot without using any of the Machine Learning or Deep Learning Algorithms. So obviously our chatbot will be a decent one but not an intelligent one.

Source Code :

from nltk.chat.util import Chat, reflections

pairs = [  
 [  
 r"my name is (.\*)",  
 ["Hello %1, How are you today ?",]  
 ],  
 [  
 r"what is your name ?",  
 ["My name is Chatty and I'm a chatbot ?",]  
 ],  
 [  
 r"how are you ?",  
 ["I'm doing good\nHow about You ?",]  
 ],  
 [  
 r"sorry (.\*)",  
 ["Its alright","Its OK, never mind",]  
 ],  
 [  
 r"i'm (.\*) doing good",  
 ["Nice to hear that","Alright :)",]  
 ],  
 [  
 r"hi|hey|hello",  
 ["Hello", "Hey there",]  
 ],  
 [  
 r"(.\*) age?",  
 ["I'm a computer program dude\nSeriously you are asking me this?",]  
   
 ],  
 [  
 r"what (.\*) want ?",  
 ["Make me an offer I can't refuse",]  
   
 ],  
 [  
 r"(.\*) created ?",  
 ["Nagesh created me using Python's NLTK library ","top secret ;)",]  
 ],  
 [  
 r"(.\*) (location|city) ?",  
 ['Chennai, Tamil Nadu',]  
 ],  
 [  
 r"how is weather in (.\*)?",  
 ["Weather in %1 is awesome like always","Too hot man here in %1","Too cold man here in %1","Never even heard about %1"]  
 ],  
 [  
 r"i work in (.\*)?",  
 ["%1 is an Amazing company, I have heard about it. But they are in huge loss these days.",]  
 ],

[  
 r"(.\*)raining in (.\*)",  
 ["No rain since last week here in %2","Damn its raining too much here in %2"]  
 ],  
 [  
 r"how (.\*) health(.\*)",  
 ["I'm a computer program, so I'm always healthy ",]  
 ],  
 [  
 r"(.\*) (sports|game) ?",  
 ["I'm a very big fan of Football",]  
 ],  
 [  
 r"who (.\*) sportsperson ?",  
 ["Messy","Ronaldo","Roony"]

],  
 [  
 r"who (.\*) (moviestar|actor)?",  
 ["Brad Pitt"]

],  
 [  
 r"quit",  
 ["BBye take care. See you soon :) ","It was nice talking to you. See you soon :)"]

],  
]  
def chatty():  
 print("Hi, I'm Chatty and I chat alot ;)\nPlease type lowercase English language to start a conversation. Type quit to leave ") #default message at the start

chat = Chat(pairs, reflections)  
 chat.converse()  
if \_\_name\_\_ == "\_\_main\_\_":

chatty()

You have created a simple rule-based chatbot, and the last step is to initiate the conversation. This is done using the code below where the converse() function triggers the conversation.

1

2

3

chat.converse()

if \_\_name\_\_ == "\_\_main\_\_":

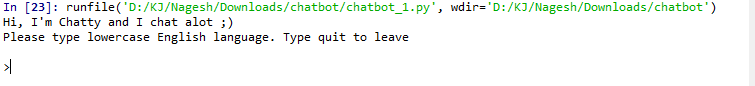
chatbot()

python

The code above will generate the following chatbox in your notebook, as shown in the image below.

The code is quite simple, still lets understand it.

Once the function chatty() is invoked , a default message will be displayed:



Next I’ve created an instance of Chat class containing **pairs**(list of tuples containing set of question and answers) and **reflections**(discussed above).

Next step is to trigger the conversation:

chat.converse()

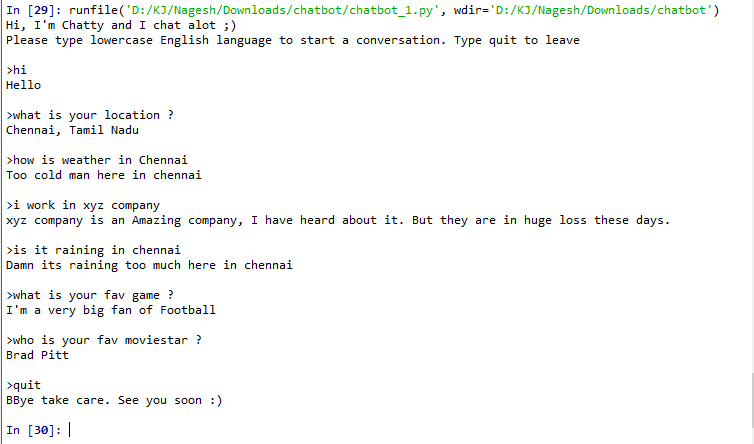
A simple conversation :



a simple conversation with Chatty

As you can see we have just hardcoded the probable question and answers in the list pairs.

Lets interact more with Chatty :



a simple conversation with Chatty

# **What does the if \_\_name\_\_ == “\_\_main\_\_”: do?**

Last Updated: 31-03-2020

Before executing code, Python interpreter reads source file and define few special variables/global variables.  
If the python interpreter is running that module (the source file) as the main program, it sets the special \_\_name\_\_ variable to have a value **“\_\_main\_\_”**. If this file is being imported from another module, \_\_name\_\_ will be set to the **module’s name.** Module’s name is available as value to \_\_name\_\_ global variable.  
A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended.

|  |
| --- |
| # Python program to execute  # main directly  print "Always executed"    if \_\_name\_\_ == "\_\_main\_\_":      print "Executed when invoked directly"  else:      print "Executed when imported" |

* All of the code that is at indentation level 0 [Block 1] gets executed. Functions and classes that are defined are, well, defined, but none of their code runs.
* Here, as we executed script.py directly \_\_name\_\_ variable will be**\_\_main\_\_**. So, code in this if block[Block 2] will only run if that module is the entry point to your program.
* Thus, you can test whether your script is being run directly or being imported by something else by testing \_\_name\_\_ variable.
* If script is getting imported by some other module at that time **\_\_name\_\_**will be module name.

**Why Do we need it?**

For example we are developing script which is designed to be used as module:

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # Python program to execute  # function directly  def my\_function():        print "I am inside function"    # We can test function by calling it.  my\_function() |

Now if we want to use that module by importing we have to comment out our call. Rather than that approach best approach is to use following code:

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| # Python program to use  # main for function call.  if \_\_name\_\_ == "\_\_main\_\_":      my\_function()    import myscripy    myscript.my\_function() |

**Advantages :**

1. Every Python module has it’s \_\_name\_\_ defined and if this is ‘\_\_main\_\_’, it implies that the module is being run standalone by the user and we can do corresponding appropriate actions.
2. If you import this script as a module in another script, the \_\_name\_\_ is set to the name of the script/module.
3. Python files can act as either reusable modules, or as standalone programs.
4. if \_\_name\_\_ == “main”: is used to execute some code **only** if the file was run directly, and not imported.