

Python Project on Chat Bot



Submitted by-:

Srinivasa V - 1BI17IS048

Prajit Singh - 1BI17IS065

Syed Affanulla - 1BI17IS051

Sri Hari N - 1BI17IS047





General Overview

- Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991.
- Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.
- Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks.

What is Chat Bot??



A chatbot is a software application used to conduct an on-line chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent.



The term "ChatterBot" was originally coined by Michael Mauldin (creator of the first Verbot) in 1994 to describe these conversational programs.



Chatbots are typically used in dialog systems for various purposes including customer service, request routing, or for information gathering.



While some chatbot applications use extensive word-classification processes, Natural Language processors, and sophisticated AI, others simply scan for general keywords and generate responses using common phrases

Objectives



All the information is maintained by the administrators based on the information available. Any action needed can be implemented by him only.



The project needs proper details of questions and the corresponding answers in a well structured form.



Various customer can register and take part in the chatbot

Purposes



Providing answers to questions



Viewing the "frequently asked questions".



Cheaper than "Call centers".



Takes less time to answer to the corresponding customers.

Software and Hardware Requirements

- Operating System - Windows 7 and above
- Language - Python
- Browser - Any of Microsoft edge, Google Chrome, etc.
- Scripting Language enable - HTML, CSS

- Processor - Intel i5 or above processor with 1.60 GHz
- RAM - Minimum 2GB
- Hard Disk - Minimum 250GB
- Monitor - color monitor
- Keyboard - 104 keys

Front End: CSS

- Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.
- CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.
- This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting
- S has a simple syntax and uses a number of English keywords to specify the names of various style properties.

Front End: HTML

- Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications.
- HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
- HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page.
- Front-end developers use HTML elements to specify what kind of information each item on a web page contains for instance.



Packages

ChatterBot

- ChatterBot is a Python library that makes it easy to generate automated responses to a user's input.
- ChatterBot uses a selection of machine learning algorithms to produce different types of responses
- This makes it easy for developers to create chat bots and automate conversations with users.

Numpy

- NumPy is a Python package which stands for 'Numerical Python'.
- It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc.
- NumPy array can also be used as an efficient multi-dimensional container for generic data.

SqlAlchemy

- SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.
- SQLAlchemy provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

Wheel

- Wheels are the new standard of Python distribution and are intended to replace eggs. Support is offered in pip ≥ 1.4 and setuptools ≥ 0.8 .
- This used to show the all-time most-downloaded packages.
- The all-time list is no longer available, and the packages in the last-365-days list will change to reflect more closely what the Python community is using.

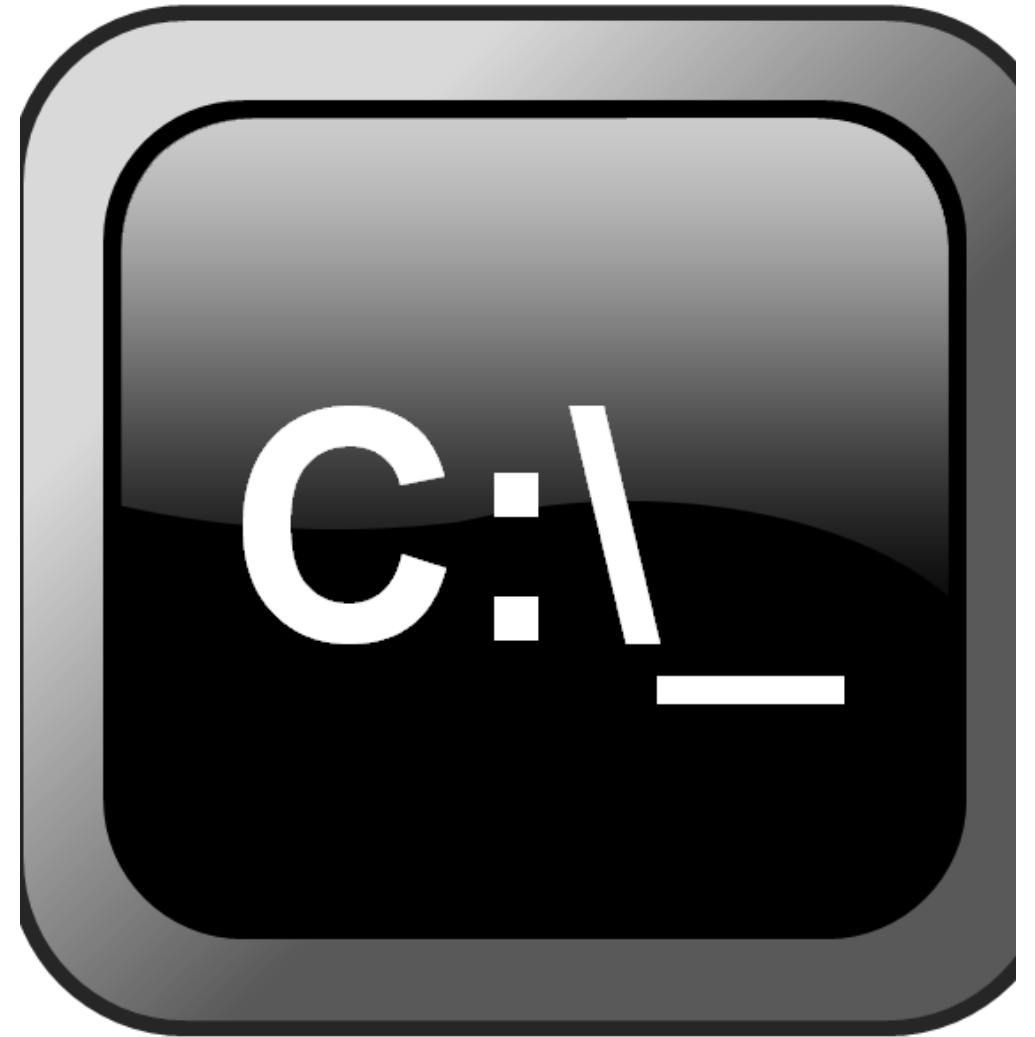
Commands

=> **Python---version-** If you have Python installed then the easiest way you can check the version number is by typing "python" in your command prompt. It will show you the version number and if it is running on 32 bit or 64 bit and some other information.

=> **Pip---version-** PIP is a package manager for Python packages, or modules if you like. If you have Python version 3.4 or later, PIP is included by default. This command is used to check the version of pip.

=> **Pip install chatterbot-** This command is used to install chatterbot package into our program.

=> **Pip install numpy-** This command is used to install numpy package into our program.



Commands

=>**Pip install SimpleWebSocketServer**- This command is used to install SimpleWebSocketServer package into our program.

=>**Pip install sqlalchemy**- This command is used to install SQLAlchemy package into our program.

=>**Pip install wheel**- This command is used to install Wheel package into our program.

=>**Pip install regex**- This command is used to install regex(regular expressions) package into our program.

=>**Python chattrainer.py**- This command is used to execute the chattrainer program(with necessary package)to see meaningful result.

=>**Python server.py**- This program is used to execute the server.py program.




Implementation:Index.html

```
<head>
<title>Chat Bot</title>
<meta charset="utf-8"/>
<script
  src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.
  1/jquery.min.js"></script>
<script>
  var ws = new WebSocket("ws://localhost:8000");
  // Close socket when window closes
  $(window).on('beforeunload', function(){
    ws.close();
  });
  ws.onerror = function(event) {
    location.reload();
  }
  ws.onmessage = function(event) {
    var message_received = event.data;
    chat_add_message(message_received, false);
  };
</script>
```


```
function chat_add_message(message, isUser) {
    var class_suffix = isUser ? '_user' : '';
    var html = '\
<div class="chat_line">\
    <div class="chat_bubble'+class_suffix+'">\
        <div class="chat_triangle'+class_suffix+'"></div>\
        '+message+'\
    </div>\
</div>\
';
    chat_add_html(html);
}


// Add HTML to the chat history
function chat_add_html(html) {
    $("#chat_log").append(html);
    chat_scrolldown();
}
```

```
// Scrolls the chat history to the bottom
function chat_scrolldown() {
    $("#chat_log").animate({ scrollTop: $("#chat_log")[0].scrollHeight }, 500);
}
// If press ENTER, talk to chat and send message to server
$(function() {
    $('#chat_input').on('keypress', function(event) {
        if (event.which === 13 && $(this).val() != ""){
            var message = $(this).val();
            $(this).val("");
            chat_add_message(message, true);
            ws.send(message);
        }
    });
});
</script>
<style>
* {
    box-sizing: border-box;
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
}
```

```
body {  
    font-family: Helvetica;  
}  
  
#chat_container {  
    overflow: hidden;  
    border-radius: 15px;  
    border: 1px solid black;  
    margin: 40px 80px 0px 80px;  
}  
  
#chat_log {  
    background-color: #F3F76F;  
    padding: 10px;  
    border-bottom: 1px solid black;  
    overflow-y: scroll;  
    height: 300px;  
    font-size: 26px;  
}  
  
#chat_input_container {  
    padding: 10px;  
}
```






```
#chat_input {
  padding: 2px;
  font-size: 18px;
  width: 100%;
}

.chat_line {
  overflow: hidden;
  width: 100%;
  margin: 2px 0 12px 0;
}

.chat_triangle, .chat_triangle_user {
  position: absolute;
  top: 0;
  width: 0;
  height: 0;
  border-style: solid;
  left: -18px;
  border-width: 0 18px 13px 0;
  border-color: transparent #ffffff transparent transparent;
```





```
.chat_triangle_user {  
    left: auto;  
    right: -18px;  
    border-width: 13px 18px 0 0;  
    border-color: #234b9b transparent transparent transparent;  
}  
  
.chat_bubble, .chat_bubble_user {  
    position: relative;  
    float: left;  
    background-color: #FFF;  
    margin-top: 10px;  
    line-height: 35px;  
    padding: 10px 25px 10px 25px;  
    margin-left: 20px;  
    font-size: 27px;  
}  
  
.chat_bubble_user {  
    float: right;  
    margin-left: 0px;  
    margin-right: 20px;  
    background-color: #234b9b;  
    color: #FFF;  
}
```



```
</style>
```

```
</head>
```

```
<body>
```

```
<div id="chat_container">
```

```
  <div id="chat_log">
```

```
  </div>
```

```
  <div id="chat_input_container">
```

```
    <div><input id="chat_input" /></div>
```

```
  </div>
```

```
</div>
```

```
</body>
```

```
</html>
```

Implementation:Chattrainer.py

```
from chatterbot import ChatBot
import os
def setup():
    chatbot = ChatBot('Bot',
        storage_adapter='chatterbot.storage.SQLStorageAdapter',
        trainer='chatterbot.trainers.ListTrainer')
    for file in os.listdir('C:/Users/Windows 8.1/chatbot/data/'):
        convData = open(r'C:/Users/Windows 8.1/chatbot/data/' +
            file,encoding='latin-1').readlines()
        chatbot.set_trainer(ListTrainer)
        chatbot.train(convData)
        #print("Training completed")
    setup()
```

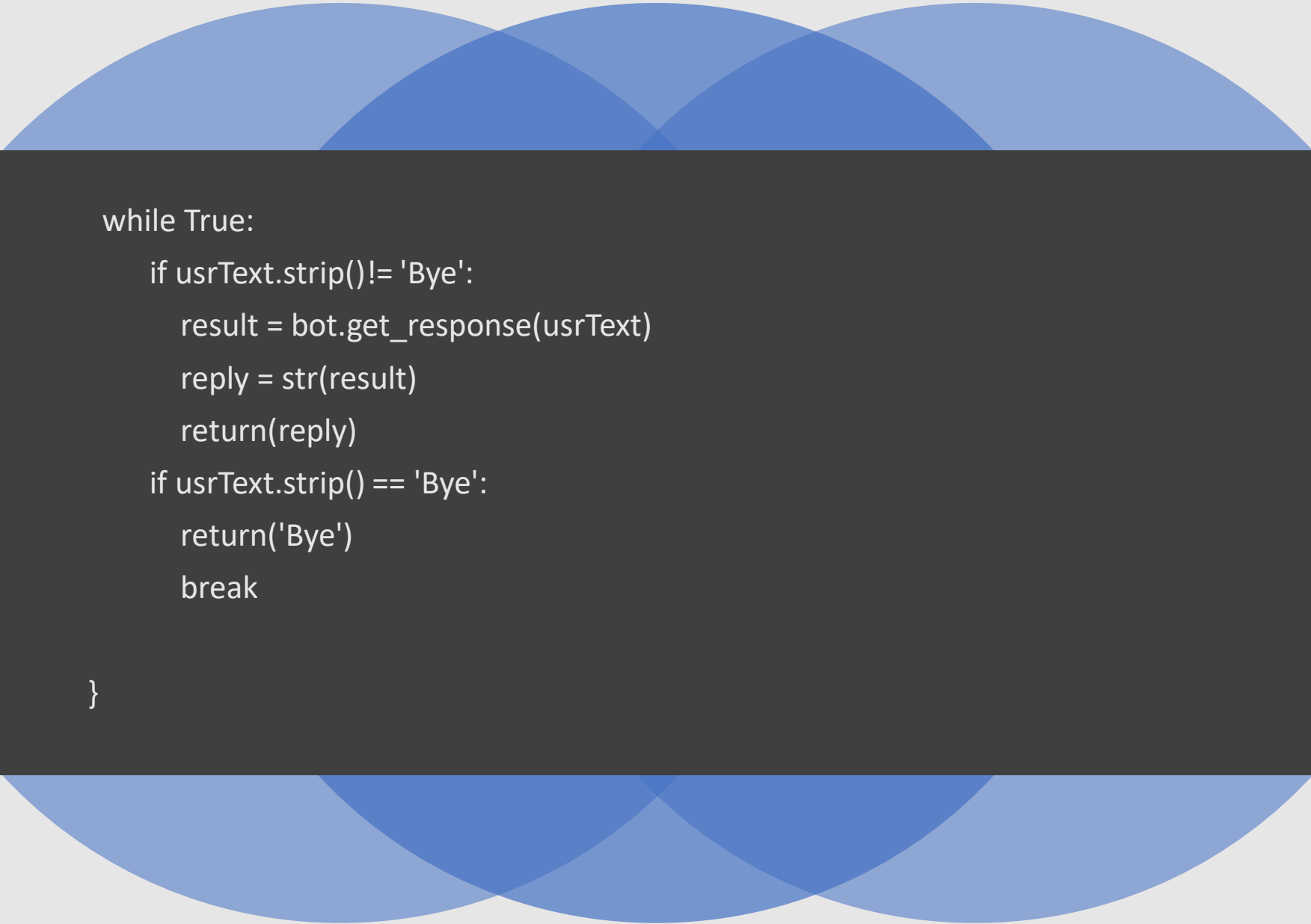
ChatterBot comes with built in adapter classes that allow it to connect to different types of databases. In this tutorial, we will be using the SQLStorageAdapter which allows the chat bot to connect to SQL databases.

You can run the training process multiple times to reinforce preferred responses to particular input statements. You can also run the train command on a number of different example dialogs to increase the breadth of inputs that your chat bot can respond to.

Implementation: ChatDirect.py

```
from chatterbot import ChatBot
def get_response(usrText):
    bot = ChatBot('Bot',
                  storage_adapter='chatterbot.storage.SQLStorageAdapter',
                  logic_adapters=[
                      {
                          'import_path': 'chatterbot.logic.BestMatch'
                      },
                      {
                          'import_path': 'chatterbot.logic.LowConfidenceAdapter',
                          'threshold': 0.70,
                          'default_response': 'I am sorry, but I do not understand.'
                      }
                  ],

    trainer='chatterbot.trainers.ListTrainer')
    bot.set_trainer(ListTrainer)
```



```
while True:
    if usrText.strip() != 'Bye':
        result = bot.get_response(usrText)
        reply = str(result)
        return(reply)
    if usrText.strip() == 'Bye':
        return('Bye')
        break
}
```

Implementation:Server.py

```
from SimpleWebSocketServer import SimpleWebSocketServer, WebSocket

class ChatServer(WebSocket):
    def handleMessage(self):
        # echo message back to client
        message = self.data
        response = get_response(message)
        self.sendMessage(response)

    def handleConnected(self):
        print(self.address, 'connected')

    def handleClose(self):
        print(self.address, 'closed')

server = SimpleWebSocketServer('', 8000, ChatServer)
server.serveforever()
```

One class, HTTPServer, is a socketserver.TCPServer subclass. It creates and listens at the HTTP socket, dispatching the requests to a handler.

- **class SimpleWebSocketServer(server_address, RequestHandlerClass)**

This class builds on the TCPServer class by storing the server address as instance variables named server_name and server_port. The server is accessible by the handler, typically through the handler's server instance variable.

The image features a large, dark blue, irregular shape that resembles a splash or a blot of ink. This shape is centered on a white background and has several smaller, lighter blue splatters and droplets scattered around its edges. The word "Snapshots" is written in a white, sans-serif font, centered within the dark blue shape.

Snapshots

```
Command Prompt - python chatbot_train.py
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Windows 8.1>python chatbot_train.py
'python' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Windows 8.1>cd appdata\local\programs\python\py

C:\Users\Windows 8.1\AppData\Local\Programs\Python\py>python chatbot_train.py
List Trainer: [##
                  1 11%
```

```
C:\> Command Prompt  
Microsoft Windows [Version 6.3.9600]  
(c) 2013 Microsoft Corporation. All rights reserved.  
  
C:\Users\Windows 8.1>python chatbot_train.py  
'python' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\Users\Windows 8.1>cd appdata\local\programs\python\py  
  
C:\Users\Windows 8.1\AppData\Local\Programs\Python\py>python chatbot_train.py  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
  
C:\Users\Windows 8.1\AppData\Local\Programs\Python\py>
```

```
Command Prompt - python server.py
```

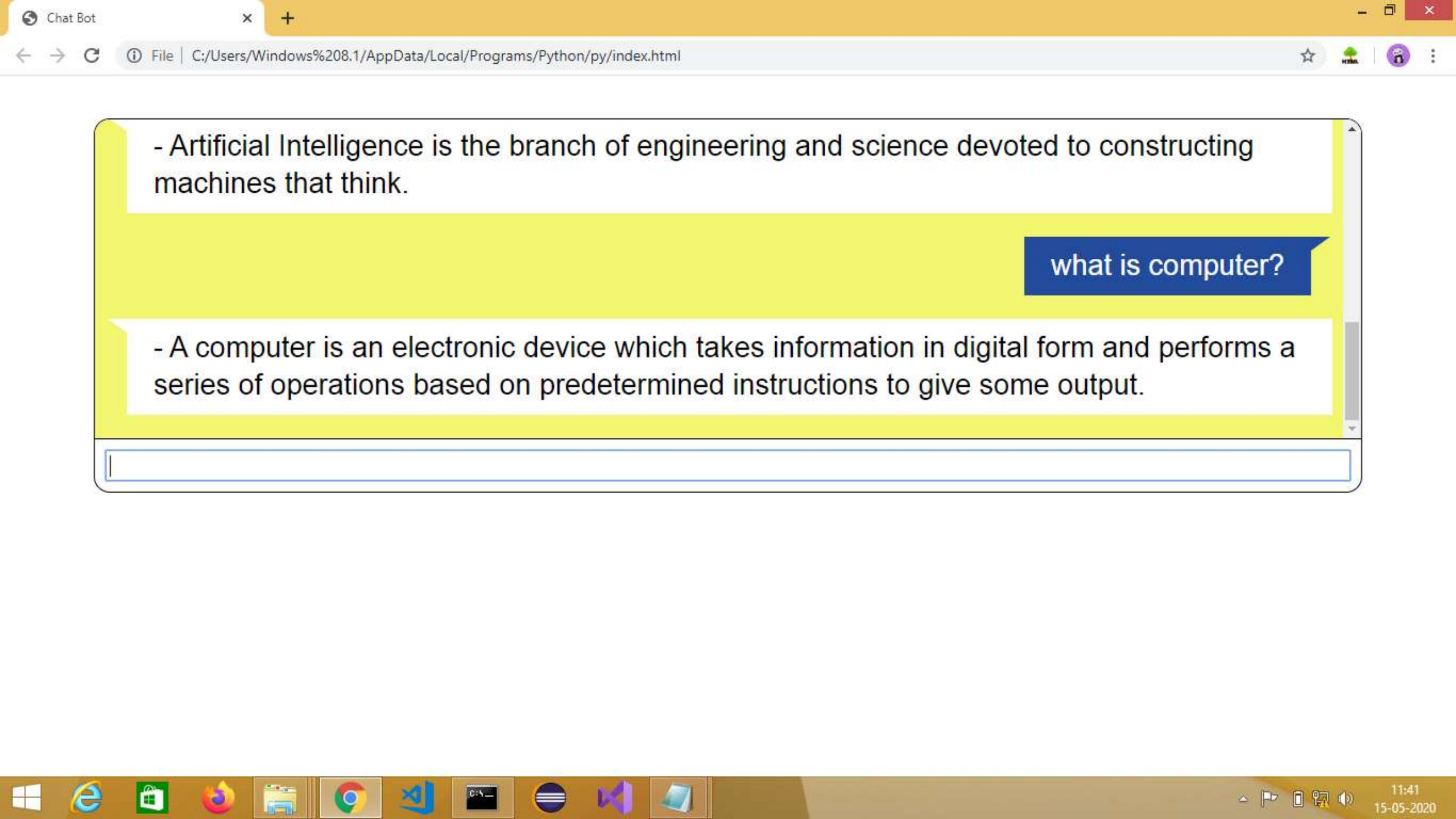
```
C:\Users\Windows 8.1>python chatbot_train.py  
'python' is not recognized as an internal or external command,  
operable program or batch file.
```

```
C:\Users\Windows 8.1>cd appdata\local\programs\python\py
```

```
C:\Users\Windows 8.1\AppData\Local\Programs\Python\py>python chatbot_train.py  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%  
List Trainer: [#####] 100%
```

```
C:\Users\Windows 8.1\AppData\Local\Programs\Python\py>python chatdirect.py
```

```
C:\Users\Windows 8.1\AppData\Local\Programs\Python\py>python server.py  
<'::1', 51206, 0, 0> connected  
<'::1', 51206, 0, 0> closed  
<'::1', 51207, 0, 0> connected  
<'::1', 51207, 0, 0> closed  
<'::1', 51208, 0, 0> connected
```

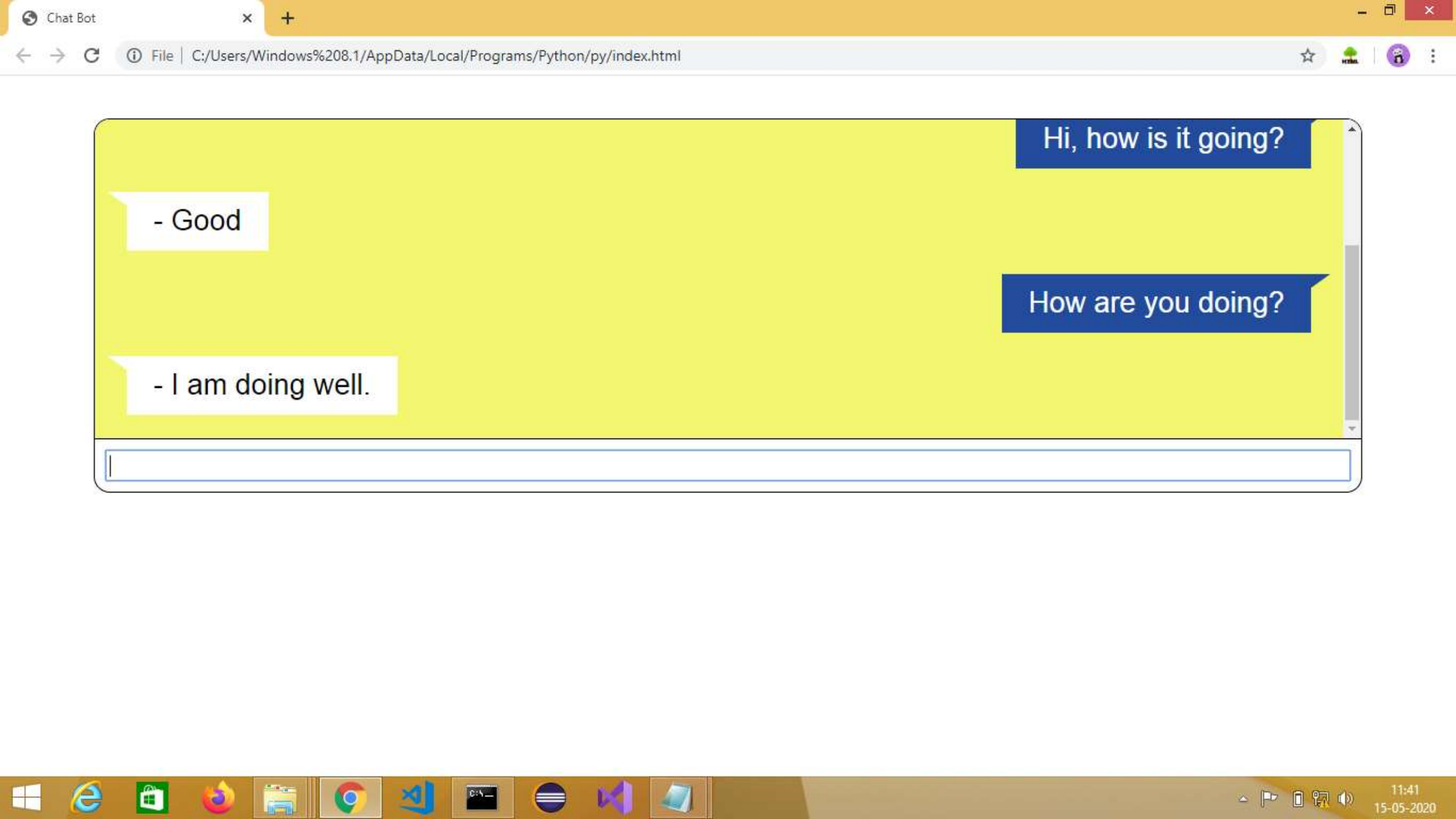


- Artificial Intelligence is the branch of engineering and science devoted to constructing machines that think.

what is computer?

- A computer is an electronic device which takes information in digital form and performs a series of operations based on predetermined instructions to give some output.



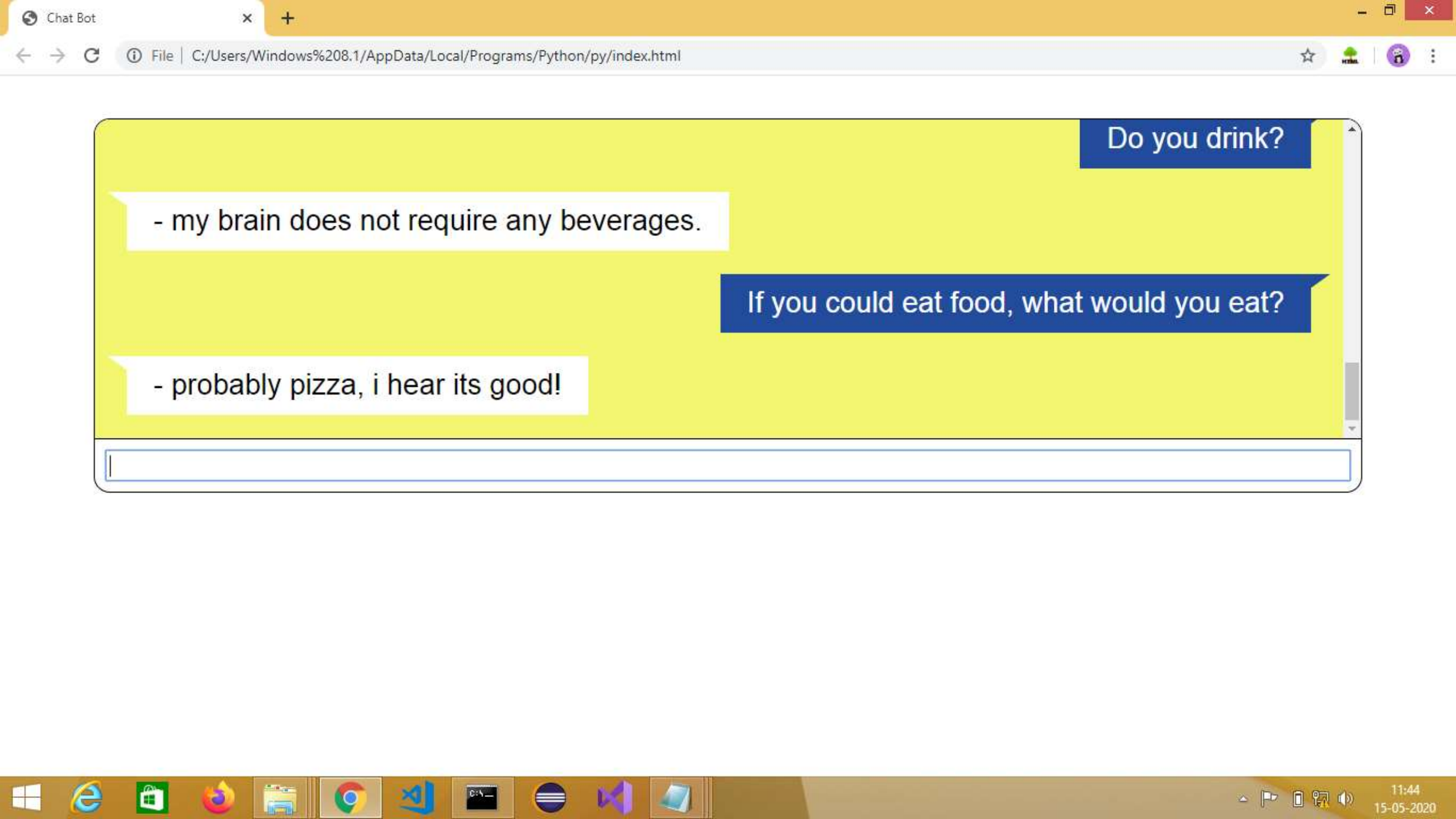


Hi, how is it going?

- Good

How are you doing?

- I am doing well.



Do you drink?

- my brain does not require any beverages.

If you could eat food, what would you eat?

- probably pizza, i hear its good!

THE END.

THANK YOU.