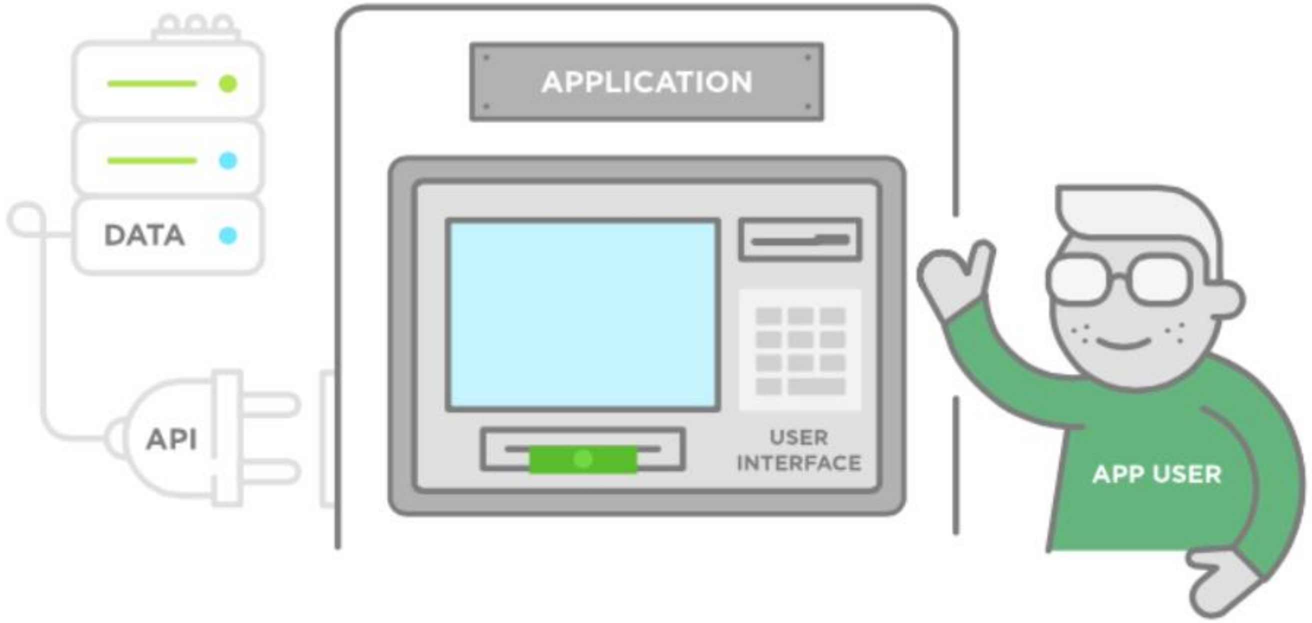
PRIME INTUIT
Finishing School

Building a Chatbot

PRIME INTUIT
Finishing School

What is an API



An API is a system of tools and resources in an operating system which lets developers create software applications that interact with other services. APIs are what allow two separate applications to communicate with each other. APIs let users interact with other services or websites without leaving your website.
APIs are important because without them, it would be very
difficult to understand the syntax of all the apps and services your tool interacts with. APIs explain what each object is called and which actions can be executed on each one.

PRIME INTUIT
Finishing School

What is an API

Application: *Think of an application like an ATM. When you walk up to an ATM, you expect it will allow you to access your account and complete a transaction like withdrawing cash. Like an ATM, an app provides a function, but it's not doing this all by itself—it needs to communicate both with the user, and with the "bank" it's accessing.*

Programming: *APIs allow the ATM to communicate with your bank. The programming is the engineering part of the app's software that translates input into output. In other words, it translates your request for cash to the bank's database, verifies there's enough cash in your account to withdraw the requested amount, the bank grants permission, then the ATM communicates back to the bank how much you withdrew so that the bank can update your balance.*

Interface: *A user interface (UI) is how we interact with an application. In the case of the ATM, it's the screen, keypad, and cash slot—where the input and output occurs. We enter our pin number, punch in how much cash we'd like to withdraw, then take the cash that's spit out. Interfaces are how we communicate with a machine. With APIs, it's much the same, only we're replacing users with software.*

PRIME INTUIT
Finishing School

What is an API

Application: *Think of an application like an ATM. When you walk up to an ATM, you expect it will allow you to access your account and complete a transaction like withdrawing cash. Like an ATM, an app provides a function, but it's not doing this all by itself—it needs to communicate both with the user, and with the "bank" it's accessing.*

Programming: *APIs allow the ATM to communicate with your bank. The programming is the engineering part of the app's software that translates input into output. In other words, it translates your request for cash to the bank's database, verifies there's enough cash in your account to withdraw the requested amount, the bank grants permission, then the ATM communicates back to the bank how much you withdrew so that the bank can update your balance.*

Interface: *A user interface (UI) is how we interact with an application. In the case of the ATM, it's the screen, keypad, and cash slot—where the input and output occurs. We enter our pin number, punch in how much cash we'd like to withdraw, then take the cash that's spit out. Interfaces are how we communicate with a machine. With APIs, it's much the same, only we're replacing users with software.*

PRIME INTUIT
Finishing School

Types

Remote APIs

Remote APIs are designed to interact through a communications network. By "remote," we mean that the resources being manipulated by the API are somewhere outside the computer making the request. Because the most widely used communications network is the internet, most APIs are designed based on web standards. Not all remote APIs are web APIs, but it's fair to assume that web APIs are remote.

Web APIs

Web APIs typically use HTTP for request messages and provide a definition of the structure of response messages. These response messages usually take the form of an XML or JSON file. Both XML and JSON are preferred formats because they present data in a way that's easy for other apps to manipulate. Web APIs are broken into two general categories: ***Remote Procedure Call (RPC)*** *and* ***REpresentational State Transfer (REST)****.*

PRIME INTUIT
Finishing School

API Keys

API keys are used for authentication for user agents that interact with or make requests to an API. These keys can be sent to the API using a query string, a request header or as a cookie. The company that controls the API can use the keys to only allow registered users to make requests, track who makes requests, track usage of the API and block or throttle users who exceed certain limits. When the API sees the key you're sending, it will then authenticate that you are who you say you are and authorize you to perform a certain action.

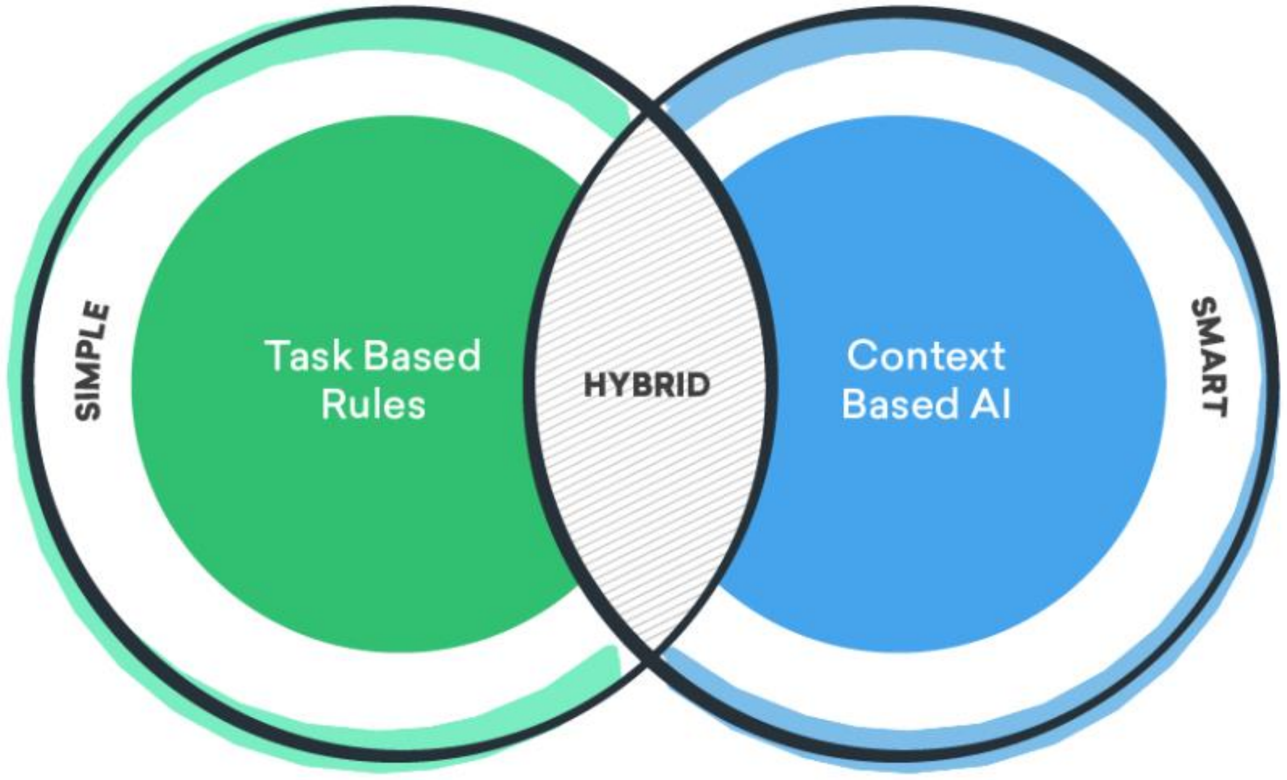
PRIME INTUIT
Finishing School

Chatbots

What is a chatbot?
Chatbots are software agents which communicate and collaborate with human users through text messaging using a natural language, say English, to accomplish specific tasks. Examples of common tasks in a business context are product inquiries, ordering, and troubleshooting.

PRIME INTUIT
Finishing School

Types of Chatbots



PRIME INTUIT
Finishing School

What is a framework

“A framework is a basic conceptual structure used to solve or address complex issues, usually a set of tools, materials or components. Especially in a software context the word is used as a name for different kinds of toolsets, component bases; it has since become a kind of buzzword or fashionable keyword.”

PRIME INTUIT
Finishing School

Flask Vs Django

KEY DIFFERENCES:

•Flask provides support for API while Django doesn’t have any support for API.
•Flask does not support dynamic HTML pages and Django offers dynamic HTML pages.
•Flask is a Python web framework built for rapid development whereas Django is built for easy and simple projects.
•Flask offers a diversified working style while Django offers a Monolithic working style.
•URL dispatcher of the Flask web framework is a RESTful request on the other hand, URL dispatcher of Django framework is based on controller-regex.
•Flask is WSGI framework while Django is a Full Stack Web Framework.

PRIME INTUIT
Finishing School

Building the components

Creating the python file and framework

```
C:\Users\LENOVO>pip install flask
C:\Users\LENOVO>cd OneDrive
C:\Users\LENOVO\OneDrive>cd Desktop
C:\Users\LENOVO\OneDrive\Desktop>mkdir PI_B2_BOT
C:\Users\LENOVO\OneDrive\Desktop>cd PI_B2_BOT
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>notepad C_BOT.py
```

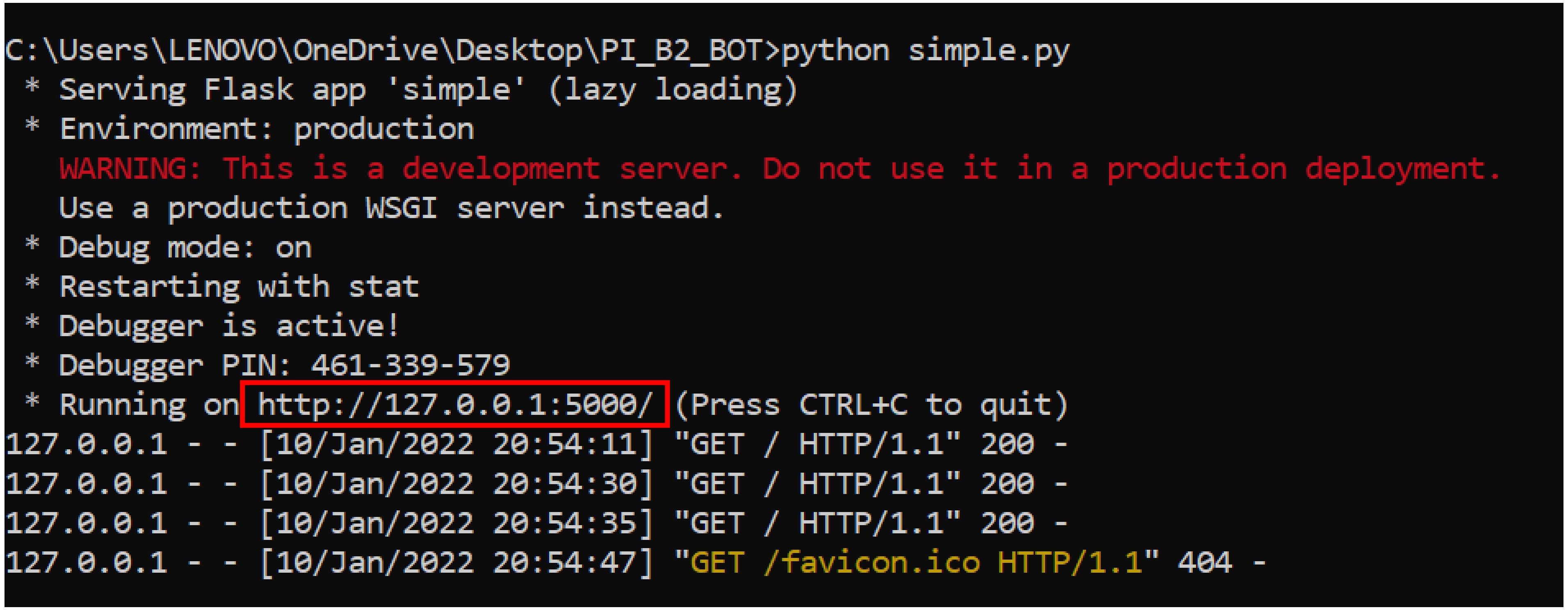
```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def index():
        return "Hello everyone"

if __name__ == "__main__":
   app.run(debug = True)
```

PRIME INTUIT
Finishing School

Building the components

Creating the python file and framework

C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>python C_BOT.py



```
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>python simple.py
 * Serving Flask app 'simple' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 461-339-579
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [10/Jan/2022 20:54:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jan/2022 20:54:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jan/2022 20:54:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jan/2022 20:54:47] "GET /favicon.ico HTTP/1.1" 404 -
```

PRIME INTUIT
Finishing School

Building the components

Creating the python file and framework

```
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>mkdir Templates
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>cd Templates
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT\Templates>notepad index.html
```

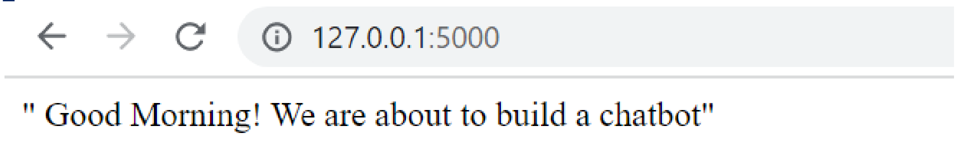
" Good Morning! We are about to build a chatbot"

```
from flask import Flask,render_template,request
app = Flask(__name__)

@app.route("/")
def index():
        return render_template(“index.html”)

if __name__ == "__main__":
   app.run(debug = True)
```

```
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT\Templates>cd..
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>python C_BOT.py
```

PRIME INTUIT
Finishing School

Building the components


127.0.0.1:5000

" Good Morning! We are about to build a chatbot"

Installing ChatterBot

C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>pip install chatterbot

C:\Users\LENOVO\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\chatterbot_corpus\data

```
from flask import Flask,render_template,request
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer
app = Flask(__name__)
english_bot = ChatBot("Chatterbot",storage_adapter="chatterbot.storage.SQLStorageAdap
trainer = ChatterBotCorpusTrainer(english_bot)
trainer.train("chatterbot.corpus.english")
trainer.train("data/data.yml")
@app.route("/")
def index():
        return render_template("index.html")
if __name__ == "__main__":
  app.run(debug = True)
```

PRIME INTUIT
Finishing School

Building the components

Creating the python file and framework

Creating the yml file

```
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>mkdir data
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT>cd data
C:\Users\LENOVO\OneDrive\Desktop\PI_B2_BOT\data>notepad data.yml
```

```
trainer.train("data/data.yml")
```

```
categories:
- conversations
conversations:
- - Good Morning
  - Good Morning, How are you
- - I am fine but boared
  - ohh what can I do to help you?
- - I want to see a movie?
  - OK, shall I give your favourite movie list?
- - Yes Avengers playlist.
  - Okay done enjoy the movie...
```

PRIME INTUIT
Finishing School

Building the components

Update the python file to receive response from Web page

```
from flask import Flask,render_template,request
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

app = Flask(__name__)
english_bot = ChatBot("Chatterbot",storage_adapter="chatterbot.storage.SQLStorageAdapter")
trainer = ChatterBotCorpusTrainer(english_bot)
trainer.train("chatterbot.corpus.english")
trainer.train("data/data.yml")

@app.route("/")
def index():
        return render_template("index.html")  # To send context or call to HTML

@app.route("/get")
def get_bot_response():
   userText = request.args.get("msg") # get dat from input, we write JS to index.html
   return str(english_bot.get_response(userText))
if __name__ == "__main__":
   app.run(debug = True)
```

PRIME INTUIT
Finishing School

Building the components

Update the index.html file

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="/static/style.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  </head>
  <body>
    <h1>Flask Chatterbot </h1>
    <h3>A web implementation of chatbot using flask</h3>
    <div>
      <div id="chatbox">
        <p class="botText"><span>Hi! I'm Chatterbot from NK.</span></p>
      </div>
      <div id="userInput">
        <input id="textInput" type="text" name="msg" placeholder="Message">
        <input id="buttonInput" type="submit" value="Send">
      </div>
```

PRIME INTUIT
Finishing School

Building the components

Update the index.html file

```
<script>
    function getBotResponse() {
     var rawText = $("#textInput").val();
     var userHtml = '<p class="userText"><span>' + rawText + '</span></p>';
     $("#textInput").val("");
     $("#chatbox").append(userHtml);
     document.getElementById('userInput').scrollIntoView({block: 'start', behavior: 'smooth'});
     $.get("/get", { msg: rawText }).done(function(data) {
      var botHtml = '<p class="botText"><span>' + data + '</span></p>';
      $("#chatbox").append(botHtml);
      document.getElementById('userInput').scrollIntoView({block: 'start', behavior: 'smooth'});
     });
    }
```

PRIME INTUIT
Finishing School

Building the components

Update the index.html file

```
        $("#textInput").keypress(function(e) {
          if ((e.which == 13) && document.getElementById("textInput").value != "" ){
            getBotResponse();
          }
        });
        $("#buttonInput").click(function() {
          if (document.getElementById("textInput").value != "") {
            getBotResponse();
          }
        })
      </script>
    </div>
  </body>
</html>
```