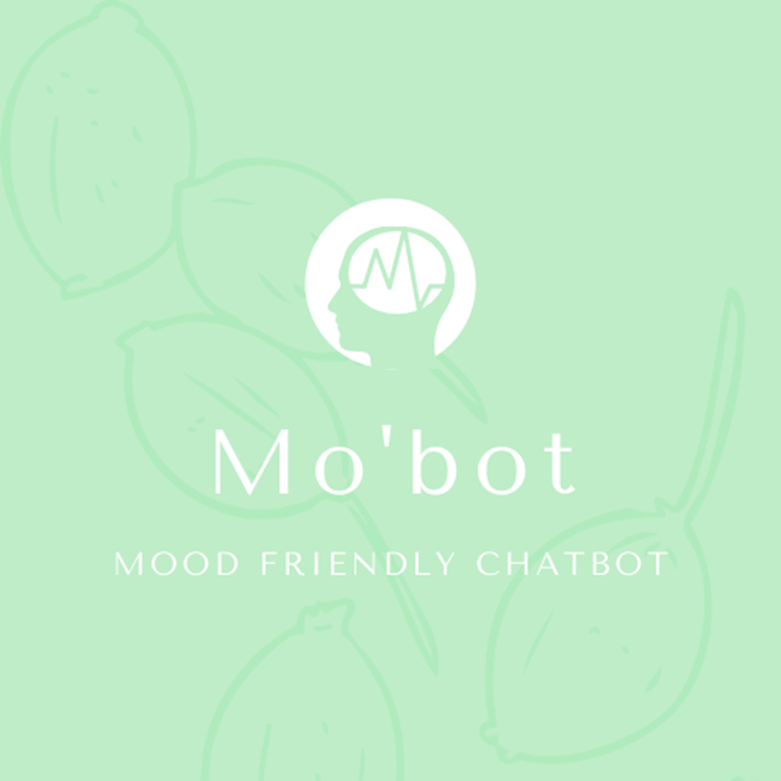
|  |
| --- |
| **Professional Portfolio** |

< Replace this text with the URL of your professional portfolio >

< ****

|  |  |
| --- | --- |
| **Project Name** | **Chatbot Project** |
| **Subject** | **Programming & Algorithms** |
| **Programming Language** | **Python 3** |
| **Number of Group Members** | **6** |
| **Position in Group** | **Group Leader** |

**Introduction to project:**

- Chatbot project is a **Coventry University** group project with a task to achieve. However, the methods adopted to reach that task is up to the students. There are no limits and no pre-defined end goals for the chatbot.

- **The task is** to come up with a computer program that is human-like in conducting conversations.

- There is uncountable amount of ideas to come up with for a responsive robot. Some of our first suggestions were: Personal Assistant, Recommendation for Cooking and Decision Making bot.

- The chatbot we ended up choosing is **Dr. Bot**. Which was supposed to be **a prototype of a doctor diagnosing and giving advice and treatment**. Due to the sensitivity of the subject and huge medical field this was not easy to conclude. Therefore, the chatbot we ended up making is mood-bot.

- **Mood-bot** has many functions, Greeting(), Mental(), deep breathing(), countmood(), feedmood(), drawmood()..etc

****

**Personal Contribution:**

From the first day in the project, everyone in my group nominated me as a **group leader**. I didn’t hesitate to take the position because our last project didn’t end with a good outcome although I did everything I could. So I thought maybe this time if I take the wheel, we can end up with something better.

First of all, I went through the project brief and picked the most suitable apps to use. Because everyone needs to be on the same page. I created a whatsapp group for all team members.

* I created a project board on Trello to organize the work and keep track of it.
* Github repository.
* I told everyone to go through Moodle to get the main idea of the project, get familiar with the tools and terms used.
* Influenced members to come up with ideas for (must, should and nice to have features). And try to develop own plans. The aim was to test members capabilities, understanding and develop a collaborative plan.
* Scheduled next meeting.

Sadly, not everyone was on the same level of excitement, eager and understanding. Some members never show up and other members are off point. Hence, at the beginning, the project outcome was escalating at a good pace. As the tasks got complicated, members contributions drooped. Anyhow, initiating the fundamentals of the chatbot in terms of plan, ideas and features worked out well.

Next, we started executing functions, I personally wrote a function that calculates BMI and Greets user. At the middle phase the vision was getting blurry and outcome decreased. A big boost and something to put us back in track was needed. I found out that other groups present their progress and methods adopted in other class. I went there and got exposed to new points of views. Every week a member of our group was assigned to this task, I considered it as a follow up to work.

Letter I tried to create a web hook for the source code so I can link it with messenger. that happened after I created an API key from google’s platform providing API’s. The procedure took me too long and I got stock for a whole week doing that so I dropped it. In addition, I approached nltk in chase of a more intelligent interactions. Chatterbot library and how it trained chatbot responses fascinated me for a while. But then I focused more on making the chatbot work on basics.

A person using a computer sitting on top of a desk

Description automatically generated

As an international fresher in University, I faced many obstacles as I approached this project. Some of these were:

* **Operating within project scope**: Throughout the weekly presentations for this project -which took place to follow up with the outcome and get advice and recommendations from module teacher- I noticed that sometimes I spend most of weekly study time on the presentation and still get a low grade. Other times I spend nearly nothing and get good grades. Many reasons for this is: working hard not smart, going off track in chase of interesting features and mainly getting stuck on complicated tasks for too long.
* **Research and information filtering**:

This project has no curriculum or slides elected by lecturer to give a clear path to follow. Therefore, it’s a self learning method based on other modules in the semester and mainly my own research. Filtering the huge amount of information on the web to adapt to my context was a challenge.

* **Explaining deep uncertainties**: languagebarrier was an issue for me explaining deep meanings.
* **Engagement with diversity**.
* **Managing the outcome**.

**Skills developed:**

**Project related:**

**Time management**: Distributing the work throughout the week makes it way easier to reach the weekly goal.

**Public speaking**: Personally, I used to fear speaking in public. For the first two-three presentations I kept evading the actual speaking. At some point I realized that for my ideas to reach everyone else I had to explain it in my own words. Its eagerness to standout that gave me the courage to do it. After the first three presentation I presented every after.

**Presentation**: Leaving out unnecessary details to stay within presentation time limit. Focusing on more important parts to convey message. And leaving some time for module leader to give recommendations.

**Locating effective members** : Assigning and expecting different jobs and outcomes from group members.

**SOS**: Rescuing presentation from zero grades in last minutes!

**Programming related:**

* Basic HTML structure.
* Learnt to use Basic Python libraries and methods.
* Facebook for developers.
* Github website enviroment and the use of git commands.
* Creating Virtual environment for projects using cmd.
* Exposed to advanced programming concepts and libraries ( API’s, Flask, ngrok, ntlk, pymessenger, urllib,
* Debugging.
* Creating an API key for google maps using google.cloud.api
* Good knowledge of NLTK (Tokenization, stopwords, Lemmatizers, Stemming, Wup\_similarity, path\_distance , hypernym tree, corpus).
* Object Oriented Programming (I wrote Mental function using OOP).
* Exposed to different programming environments like Heroku, ATOM text editor.
* Using pymessenger to send and receive messages.
* Flask code creating a web application(Hello World).

**RESULT:**

The Mo’bot is working properly, with five main services built in it.

1. **Feed mood:** selecting mood and getting mood lifting tips related to user’s selection.
2. **Count mood:** mood tracking using database to store previous moods.
3. **Mental Health Questionnaire**: tests user mental health and gives feed back based on a questionnaire.
4. **Draw mood:** mood drawing task for fun.
5. **Deep breathing:** breathing exercise to help user relief.

Anyhow, if more time was available I would focus on making it more intelligent using nltk.

**LEARNING:**

* **No courtesy**: as a new member of the university and its distinct environment, I had my own mores. These habits I don’t consider to be professional. For example: our module leader asked every group to mention their contribution in the last slide of every presentation, its preferred to be in percentage. I couldn’t do that in the beginning. Although it wasn’t honest and affected my grades, but I thought of it as ditching group members. I learned to be **decisive in a professional way**, and that satisfied me in terms of **claiming rights.**
* The periodic nature of the project (weekly presentation) was annoying in the start. This had brought many benefits as I got used to it. **Commitment, scheduling** and operating within the time frame.
* **Consistency** with group members.

If I had the choice to redo the project I would focus more on working smart, working within the context and fulfilling the “ABC’s” of the project. More than working on impressive hard topics. If I had the chance to solve problems in functions I wouldn’t get stuck within one method instead I’ll move on flexibly.

|  |
| --- |
| **Source Code** |

< https://github.com/YGAtabani1/Chatbot/blob/master/moodbotnew.py>

< **from** **nltk.corpus** **import** stopwords

**from** **nltk.tokenize** **import** word\_tokenize

**from** **nltk.stem** **import** PorterStemmer

**from** **nltk.corpus** **import** wordnet

**import** **random**

BotGreetList= ["Hey, How are you feeling today?","Hi, How can I help?","Hello, Need something?","Hola, I'm Dr.Bot", "Welcome to Dr.Bot", "Bonjour!, Dr.Bot at your service", "Aloha"]

UserGreetList= ["hey","hi","hello","hola", "aloha", "Hey", "Hi", "Hello", "Hola", "Aloha"]

#WRITTEN BY ME

entry = input()

**def** **BotGreeting**():

#Function to greet user

**for** entry **in** UserGreetList:

**if** entry **in** UserGreetList:

**return** random.choice(BotGreetList)

#WRITTEN BY ME

#Function to calculate users BMI

**def** **bmiCal**(height,weight):

height = float(input("whats your height in meters?. **\n**"))

weight = float(input("whats your weight in kg?. **\n** "))

gender = str(input("Are you a male or a female?. **\n**"))

Bmi = weight/(height\*height)

**print**("your bmi is: " + str(Bmi))

**if** gender == "male" **or** gender == "Male":

**if** Bmi < **15** **and** Bmi > **0**:

**print**("You're very severly underweight!")

**elif** Bmi >= **15** **and** Bmi <= **18.5**:

**print**("You're underweight!")

**elif** Bmi > **18.5** **and** Bmi < **25**:

**print**("Your BMI is normal!")

**elif** Bmi >= **25** **and** Bmi < **35**:

**print**("You're Overweight!")

**elif** Bmi >= **35**:

**print**("You're severly obese!")

**elif** gender == "female" **or** gender == "Female":

**if** Bmi < **20** **and** Bmi > **0**:

**print**("You're severly underweight")

**elif** Bmi >= **20** **and** Bmi < **28**:

**print**("You're normal")

**elif** Bmi >= **28** **and** Bmi < **35**:

**print**("You're average ")

**elif** Bmi >= **35**:

**print**("You're severly obese")

**return**(Bmi)

**print**(bmiCal(**160**,**80**))

#less than 15 (very severly underweight)

#lLess than 15 Very severely underweight

#Between 15 and 16 Severely underweight

#Between 16 and 18.5 Underweight

#Between 18.5 and 25 Normal (healthy weight)

#Between 25 and 30 Overweight

#Between 30 and 35 Moderately obese

#Between 35 and 40 Severely obese

#Over 40 Very severely obese

#FOUND ON THE WEB

#code for hosting using flask.Source: google. This code when run on terminal gives you an IP address. when you go to the address you see a webpage with (Hello World)

#TOKEN(WEBSITE)-HOST CODE

#TOKEN RECEIVED AFTER RUNNING CODE: http://127.0.0.1:5000/

**from** **flask** **import** Flask, request

app = Flask(\_\_name\_\_)

**@app.route**('/', methods=['GET', 'POST'])

**def** **receive\_message**():

**return** "Hello World!"

**if** \_\_name\_\_ == '\_\_main\_\_\_':

app.run(debug=True)

#FOUND ON THE WEB

#using NLTK to find noun(NN) and adjective(JJ) in a sentence input by the user.Source: google

**from** **nltk.corpus** **import** twitter\_samples

**from** **nltk.tag** **import** pos\_tag\_sents

tweets = twitter\_samples.strings('positive\_tweets.json')

tweets\_tokens = twitter\_samples.tokenized('positive\_tweets.json')

JJ\_count = **0**

NN\_count = **0**

**for** tweet **in** tweets\_tagged:

**for** pair **in** tweet:

tag = pair[**1**]

**if** tag == 'JJ':

JJ\_count += **1**

**elif** tag == 'NN':

NN\_count += **1**

**print**('toal number of adjectives= ' , JJ\_count)

**print**('total number of nouns = ' NN\_count)

#FOUND ON THE WEB AND EDITED BY ME

#flask\_app is a code that uses my own messenger generated key in line 102. and verify token to create post and get requests. source: google. key: cloud.google.api

**import** **random**

**from** **flask** **import** Flask, request

**from** **pymessenger.bot** **import** Bot

app = Flask(\_\_name\_\_)

key="AIzaSyCUhTMchel5jVPwobjgFYIRtPrjH\_ngb7I"

ACCESS\_TOKEN = ""

VERIFY\_TOKEN = 'VERIFY\_TOKEN'

bot = Bot(ACCESS\_TOKEN)

#We will receive messages that Facebook sends our bot at this endpoint

**@app.route**("/", methods=['GET', 'POST'])

**def** **receive\_message**():

**if** request.method == 'GET':

"""Before allowing people to message your bot, Facebook has implemented a verify token

that confirms all requests that your bot receives came from Facebook."""

token\_sent = request.args.get("hub.verify\_token")

**return** verify\_fb\_token(token\_sent)

#if the request was not get, it must be POST and we can just proceed with sending a message back to user

**else**:

# get whatever message a user sent the bot

output = request.get\_json()

**for** event **in** output['entry']:

messaging = event['messaging']

**for** message **in** messaging:

**if** message.get('message'):

#Facebook Messenger ID for user so we know where to send response back to

recipient\_id = message['sender']['id']

**if** message['message'].get('text'):

response\_sent\_text = get\_message()

send\_message(recipient\_id, response\_sent\_text)

#if user sends us a GIF, photo,video, or any other non-text item

**if** message['message'].get('attachments'):

response\_sent\_nontext = get\_message()

send\_message(recipient\_id, response\_sent\_nontext)

**return** "Message Processed"

**def** **verify\_fb\_token**(token\_sent):

#take token sent by facebook and verify it matches the verify token you sent

#if they match, allow the request, else return an error

**if** token\_sent == VERIFY\_TOKEN:

**return** request.args.get("hub.challenge")

**return** 'Invalid verification token'

#chooses a random message to send to the user

**def** **get\_message**():

sample\_responses = ["You are stunning!", "We're proud of you.", "Keep on being you!", "We're greatful to know you :)"]

# return selected item to the user

**return** random.choice(sample\_responses)

#uses PyMessenger to send response to user

**def** **send\_message**(recipient\_id, response):

#sends user the text message provided via input response parameter

bot.send\_text\_message(recipient\_id, response)

**return** "success"

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

app.run()

#WRITTEN BY ME

#Function to test users chest pain

**def** **chestPain**():

Advice = ["One reason for Muscular pain is heavy exercise, I'd recommend taking analgesics.**\n** Nemonia is caused for many reasons but its cured with anti-biotis ", "For peptic alser or gastric visit you doctor for treatment","Visit you doctor for treatment ASAP."]

**print**("Do you experience chest pain?")

ans = input("Y or N")

**if** ans == "Y" **or** ans == 'y':

**print**("What side of your chest do you feel the pain?")

**print**("Please pick a number:")

side = input("1- Sides**\n**2- Middle.**\n**")

**return** side

**if** side == **1**:

**print**("Chest pain on the sides is likely to be muscular pain or nemonia")

**print**(Advice[**0**])

**elif** side == **2**:

**print**("How does the pain feel like?")

type = input("1-Dull**\n** 2- Stabbing**\n** 3-Burning**\n**")

**return** type

**if** type == **1** **or** type == **3**:

**print**("This is more likely to be a peptic alser or Gastric ")

**print**(Advice[**1**])

**elif** type == **2**:

**print**("This is more likely to be a anginal pain")

**print**(Advice[**2**])

**elif** ans == "N" **or** ans == "n":

**print**("Great! Chest pain is one annoying symptom of Mi, Gastric, muscular pain and nemonia.")

#WRITTEN BY ME

#NLTK Function for stripping input sentences

**def** **word\_tokenize**(listOfWords):

engsw = stopwords.words('english') #

user\_input = listOfWords

token\_words1 = word\_tokenize(user\_input)

token\_words = lemmatizer(token\_words1, Pos = '')

filterArray= [item **for** item **in** token\_words **if** item **not** **in** engsw]

#WRITTEN BY ME

#NLTK Function to check similarity between two words

**def** **wordSimilarity**(word):

sim1 = wordnet.Synsets(word) #wordnet method that outputs list of words related to 'word'.

ref =

sim2 = wordnet.Synsets(Function) #this is the word that you want to compare with the previous to get path in the end

sim3 = sim1[**0**]

sim4 = sim2[**0**]

sim3.wup\_similarity(sim4) #wup\_similarity is a method from wordnet created by two japanese programmers. it returns an int wihich is the similarity between two words out of 10

ref = word.hypernyms() #hypernyms is a tree of words.

word.shortest\_path\_distance(ref) #this shows you the distance between your word and the refrence word you input in the tree of words

i = **0**

**for** entry **in** suggestions:

**if** entry.wup\_similarity()>

#WRITTEN BY ME

#Function to remind user of whatever they ask for

**def** **Reminder**():

**print**("What shall i remind you about?")

text = str(input())

**print**("in how many minutes?")

local\_time= float(input())

local\_time=local\_time\***60**

time.sleep(local\_time)

**print**(text)

#WRITTEN BY ME

#Function to check if user is going though any mental issues

**def** **Mental**():

**print**("First, I'm going to record your complaint")

**print**("All the information you share is private so please don't hesitate.")

**print**("I'm now going to ask you a couple of questions")

age = int(input('How old are you?**\n**'))

sleep = int(input( '**\n**On average, How many hours do you sleep every day**\n**'))

food = input('Please choose a number:**\n** On average, do you eat**\n** 1- normal**\n** 2- less than normal**\n** 3- more than normal?**\n**')

interest = input('Please choose a number:**\n**Do you experience**\n** 1- Loss of interest**\n** 2- Over interest**\n** 3- Normal**\n**')

speech = input('Please choose a number:**\n**Are you **\n** 1- Talkative **\n** 2- Quiet**\n** 3- Normal**\n** ')

mood = input("Please choose a number:**\n**Do you experience**\n** 1- High**\n** 2- Low mood**\n** 3- Normal mood?**\n**")

duration = input('How long does this last?**\n**1- Less than 2 weeks**\n** 2- More than 2 weeks**\n**')

**if** mood == '3' **and** duration == **2**:

**if** interest == "3" **and** duration == **2**:

**print**("Having a normal mood is great then!, Lets go though some tests to give you advice.")

**if** age > **18** **and** sleep < **8**:

**print**('As an adult, youre supposed to sleep about 7-8 hours a day**\n** Lack of sleep might be a sign of Bipolar or Mania ')

**if** interest == **2** **and** speech == **1**:

**print**("People suffering from Bipolar tend to be sleep less, more interested and takative.")

interest2 = input('Do you experience desire towards dangerous experiments?**\n** Y/N')

**if** interest2 == Y **or** interest2 == y:

**print**("You're more likely to suffer from Bipolar")

suicide = input('Have you ever had thoughts of suicide?**\n** Y/N')

**if** suicide == Y **or** suicide == y:

**print**("Call:1-800-273-8255**\n** You should see your local psycologist**\n** If you want to know more about bipolar go to this link**\n** 'https://www.healthline.com/health/could-it-be-bipolar-signs-to-look-for'")

**elif** suicide == N **or** suicide == n:

**print**("You should see your local psycologist**\n** If you want to know more about bipolar go to this link**\n** 'https://www.healthline.com/health/could-it-be-bipolar-signs-to-look-for'")

**elif** interest2 == N **or** interest2 == n:

**print**('Good, wanting to experiment dangerous adventures or sexual experiments is a sign of Bipolar. Not all the time though!')

**else**:

**print**("Sleeping less, talking more and being interested in dangerous adventures are signs of bipolar. But since you don't suffer from all these symptoms, you're more likely to be nomal")

**elif** age < **18** **and** sleep < **8**:

**print**("As a child, you are suposed to sleep more or equals to 8 hours a day")

**elif** age > **18** **and** sleep >**8**:

**if** food == **3** **and** interest == **1**:

**print**("Over sleeping when found with eating more, loss of interest and sleeping more, is more likely to be depression.")

**else**:

**print**("You're a normal person with lack of schedulling. Prioritizing stuff is a must for a better life.")

**elif** interest == **1** **and** duration == **1**:

**print**("Your mood is normal, anyhow, loss of interest is a sign of depression, specially when linked to more sleep, more eating and lack of speech. ")

**elif** mood == '2' **and** duration == **2**:

**print**("Low mood is a sign of depression. You need to schedule your day, exercise more and socialize more.")

**if** interest == **1** **and** food == **3**:

**print**("Loss of interest and eating more are also signs of depression.")

**if** sleep > **8** **and** speech ==**2**:

**print**("Sleeping more than usual and speacking less are strong signs of depression.")

**elif** mood == **1** **and** duration == **2**:

**print**("High mood, specially when it flunctuate between high and low for more than two weeks are signs of Bipolar\Mania")

**if** interest == **2** **and** sleep < **8**:

**print**("Being interested into almost everything with the desire to experiment dangerous activities is linked to Bipolar\Mania")

**if** speeck == **1** :

**print**("Talking more is also a sign of Bipolar\Mania")

**elif** mood == '3':

**pass**

#WRITTEN BY ME

#Another version of Mental function using OOP

**class** **Mental**:

"""This class stores methods that test the users mental health using their mood,age,sleep ..etc"""

**def** **\_\_init\_\_**(self):

self.mood = **0**

self.duration = **0**

self.sleep = **0**

self.age = **0**

self.speech = **0**

self.interest = **0**

self.food = **0**

**def** **preDepression**(self):

**print**("Experiencing a low mood for more than 2 weeks can be linked to depression if found with other symptoms:**\n**Lets find out")

user.interest = int(input('Please choose a number:**\n**Do you experience**\n** 1- Loss of interest**\n** 2- Over interest**\n** 3- Normal**\n**'))

user.speech = int(input('Please choose a number:**\n**Are you **\n** 1- Talkative **\n** 2- Quiet**\n** 3-Normal**\n** '))

user.sleep = int(input( '**\n**On average, How many hours do you sleep every day**\n**'))

**if** user.interest >=**1** **and** user.interest<=**3** **and** user.speech>= **1** **and** user.speech<=**3** **and** user.sleep>=**1** **and** user.sleep<=**24** **and** user.food>=**1and** user.food<=**3**:

**if** user.interest == **1** **and** user.speech == **2** **and** user.sleep > **8** :

Mental.depression(self)

**elif** user.interest== **2** **and** user.speech == **1** **and** user.sleep < **8**:

Mental.bipolar(self)

**elif** user.interest== **3** **and** user.speech == **3** **and** user.sleep == **8**:

Mental.normal(self)

**else**:

**print**("You've entered a wrong selection. Please try again:")

Mental.preDepression(self)

**def** **depression**(self):

**print**("Sleeping more..**\n**Eating more..**\n**Interacting less..**\n**Loss of interest...**\n**When experienced for more than two weeks are linked to depression.")

**def** **mania**(self):

**print**("Sleeping less..**\n**Eating less..**\n**Being interested in dangerous activities and sexual experiments..**\n**When experienced for more than two weeks are linked to mania")

**print**("Go to this link for more information")

**print**("https://www.verywellmind.com/symptoms-of-mania-380311")

**def** **preBipolar**(self):

**print**("High mood and flunctuations in mood are signs of bipolar. Lets proceed:")

user.interest = int(input('Please choose a number:**\n**Do you experience**\n** 1- Loss of interest**\n** 2- Over interest**\n** 3- Normal**\n**'))

user.speech = int(input('Please choose a number:**\n**Are you **\n** 1- Talkative **\n** 2- Quiet**\n** 3- Normal**\n** '))

user.sleep = int(input( '**\n**On average, How many hours do you sleep every day**\n**'))

user.food = int(input('Please choose a number:**\n** On average, do you eat**\n** 1- normal**\n** 2- less than normal**\n** 3- more than normal?**\n**'))

**if** user.interest >=**1** **and** user.interest<=**3** **and** user.speech>= **1** **and** user.speech<=**3** **and** user.sleep>=**1** **and** user.sleep<=**24** **and** user.food>=**1and** user.food<=**3**:

**if** user.interest == **2** **and** user.speech == **1** **and** user.sleep < **8** **and** user.food == **2**:

Mental.bipolar(self)

**elif** user.interest == **1** **and** user.speech == **2** **and** user.sleep > **8** **and** user.food ==**3**:

Mental.depression(self)

**elif** user.interest == **3** **and** user.speech == **3** **and** user.sleep == **8** **and** user.food ==**1** :

Mental.normal(self)

**else**:

Mental.mood\_swings(self)

**else**:

**print**("You've entered a wrong selection. Please try again:")

Mental.preBipolar(self)

**def** **bipolar**(self):

**print**("Sleeping less..**\n**Eating less...**\n**Being interested in dangerous activities and sexual experiments..**\n**When experienced for mor than two weeks are linked to bipolar")

**print**("Go to this website for more information about bipolar.")

**print**("https://www.healthline.com/health/could-it-be-bipolar-signs-to-look-for")

**def** **mood\_swings**(self):

**print**("Fluctuation in daily activities for less that 2 weeks are nothing but mood swingng"

)

**def** **normal**(self):

**print**("Your attributes are normal therefore, your mental health is on point")

**def** **sleep\_lack**(self):

**print**("You sleep less than normal. Adults must sleep about 7-8 hours everyday.")

**def** **diagnose**(self):

user = Mental()

user.mood = int(input("Please choose a number:**\n**Do you experience**\n** 1- High**\n** 2- Low mood**\n** 3- Normal mood?**\n**"))

user.duration = int(input('How long does this last?**\n**1- Less than 2 weeks**\n** 2- More than 2 weeks**\n**'))

**if** user.mood >=**1** **and** user.mood <=**3** **and** user.duration>=**1** **and** user.duration<=**3**:

**if** user.mood == **1** **and** user.duration == **2**:

Mental.preBipolar(user)

**elif** user.mood == **2** **and** user.duration == **2**:

Mental.preDepression(user)

**elif** user.mood == **3** **and** user.duration == **2**:

Mental.normal(user)

**elif** user.mood == **3** **and** user.duration == **2**:

Mental.mood\_swings(user)

**elif** user.duration == **1**:

**print**("Sometimes things things go wrong and you feel low. But if its been going on for less than two weeks then its not considered a mental issue.")

**else**:

**print**("You've entered a wrong selection, Please try again:")

Mental.diagnose(self)

user = Mental()

#would have been better to come up with a method that takes the input within the class example:raw\_input

**if** \_\_name\_\_ == '\_\_main\_\_':

Mental.diagnose(user)