Documentation of Cap Stone Project
(OTP Verification System)

NAME: CH.RAMA SAI PRAMODH

STUDENT_ID:S_8599

Modules used in project: In project random, AppJar these modules are used.

Coding language used: Python.

IDE Used: PyCharm, Google Colab.

Code breakdown and Explanation:

Step1: OTP Generation:

For generating the OTP(one time password) random module is used and the randint is used to generate 6-digit OTP.

```
# # for generating the OTP
import random
def generated_otp():
  return random.randint(100000,999999)
```

Step2: Sending OTP to User:

For this step User defined function is used.

The sending otp to email address and otp sent. Message is displayed.

```
# For sending otp to user:
def sending_otp(otp,email):
  print(f'sending_otp {otp} to Email address')
  print('otp sent')
```

Step3: prompting the OTP:

In this step we allow the user to enter the OTP received and allow the user to request for resending OTP. And please enter the otp sent to email address or type resend.

```python
# For prompting the OTP and allow the user to enter the user:
def prompt_the_user_to_enter_otp():
  return input(f'please enter otp sent to Email address or type resend:')
```

Step4: For Verify the OTP

In this step we verify the OTP sent and OTP entered by the user.

In this we have to observe one thing about try and except block.

```python
# for verify the OTP:
def verify_otp(generated_otp,entered_otp):
  try:
     entered_otp = int(entered_otp)
  except ValueError:
        return False
  return generated_otp == entered_otp
```

Try and Except block:

Try and Except block uses to: The try block lets you test a block of code for errors. The except block lets you handle the error. The final block lets you execute code, regardless of the result of the try- and except blocks.

For our case checks for the user enters the correct OTP and checks for the non-numeric values.

String conversion fails than returns fail and the otp verification fails.

Step5: Creating the main function and integrating the all above functions:

Interconnecting and joining all functions and creating the main function.

And the no of attempts to enter the correct otp is 3.

I used to loop and condition statements.

```python
# For prompting the OTP and allow the user to enter the user:
# OTP verification system:
def prompt_the_user_to_enter_otp():
  return input('please enter otp sent to Email address or type resend ')
# for verify the OTP:
def verify_otp(generated_otp,entered_otp):
  try:
     entered_otp = int(entered_otp)
  except ValueError:
        return False
  return generated_otp == entered_otp
# Complete code with main function:
def main ():
  email='nonuser@gmail.com'    # exmaple email
  otp = generated_otp()
  sending_otp (otp,email)
  attempts = 3
  while attempts > 0:
    entered_otp = prompt_the_user_to_enter_otp()
    if entered_otp.lower() == 'resend':
        otp = generated_otp()
        sending_otp(otp, email)
        print('A new OTP has been sent to your email address.')
        continue
    if verify_otp(otp, entered_otp):
      print('Access Granted')
      break
    else:
      attempts-= 1
      print('Access denied')
      print(f'Entered wrong otp please try again,You have {attempts}
attempt(s) left.')
  if attempts == 0:
    print(f'Too many Incorrect attempts.You have {attempts} attempt(s)
left.')
```

<u>Explanation of loop and condition statements:</u>

<u>Loop:</u>

In this I have used only while loop.

<u>While Loop:</u>

It is a control flow statement that allows to execute block of code repeatedly as a specified condition is True.

<u>Conditional Statement:</u>

In this IF conditional statement is used.

<u>IF:</u>

It will allow us to control the flow of program based on condition.

<u>Break and continue:</u>

Break: Break statement stops the entire process of the loop.

<u>Continue</u>: Continue statement only stops the current iteration of the loop.

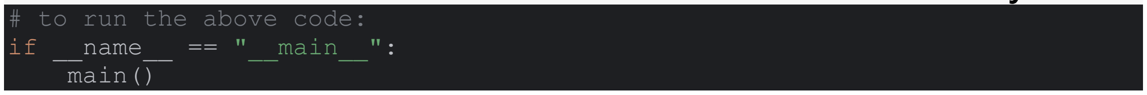
<u>One pre-defined function is used:</u>

Pre-defined function: this function are given and defined by Python.

The pre-defined function used is print.

Step6: Run the code:

if __name__ == "__main__" is used to run the code directly.

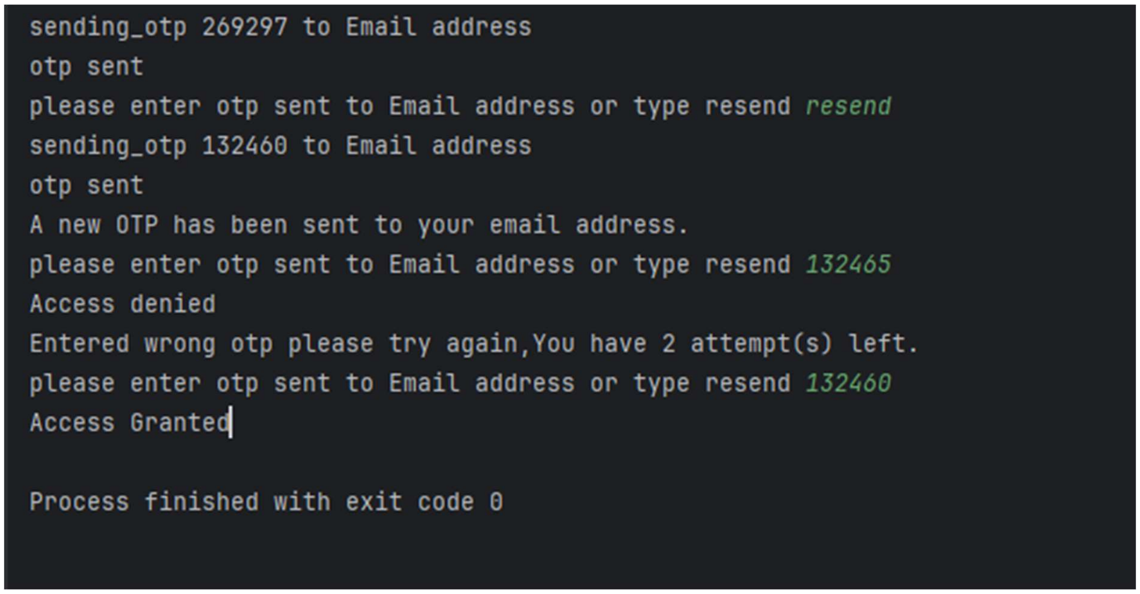


```
# to run the above code:
if __name__ == "__main__":
    main()
```

Appjar is used to design the application interface.

Output:

The code is working perfectly.



```
sending_otp 269297 to Email address
otp sent
please enter otp sent to Email address or type resend resend
sending_otp 132460 to Email address
otp sent
A new OTP has been sent to your email address.
please enter otp sent to Email address or type resend 132465
Access denied
Entered wrong otp please try again,You have 2 attempt(s) left.
please enter otp sent to Email address or type resend 132460
Access Granted

Process finished with exit code 0
```

The GUI (Graphical User Interface|) Code is provided below.

GUI Code:

```
from appJar import gui
import random


# OTP functions
def generated_otp():
    return random.randint(100000, 999999)


def sending_otp(otp):
    print(f'Sending OTP {otp} to the registered email address')
    print('OTP sent')


def verify_otp(generated_otp, entered_otp):
    try:
        entered_otp = int(entered_otp)
    except ValueError:
        return False
    return generated_otp == entered_otp


# handle button events for the OTP window
def press_otp(button):
    if button == "Cancel":
        app.stop()
    elif button == "Resend OTP":
        resend_otp()
    else:
        entered_otp = app.getEntry("OTP")
        if verify_otp(app.otp, entered_otp):
            print('Access Granted')
            app.infoBox("Success", "Access Granted")
            app.stop()
        else:
            app.attempts -= 1
            if app.attempts > 0:
                app.errorBox("Error", f'Access Denied. You have
{app.attempts} attempt(s) left.')
            else:
                app.errorBox("Error", 'Too many incorrect attempts. Access
Denied.')
                app.stop()


# OTP prompt window
def prompt_otp():
    app.otp = generated_otp()
    sending_otp(app.otp)

    app.setBg("White")
    app.setFont(18)
    app.attempts = 3
    app.addLabel("otp_title", "Enter the OTP sent to your email")
    app.setLabelBg("otp_title", "blue")
    app.setLabelFg("otp_title", "gray")

    app.addLabelSecretEntry("OTP")

    app.addButtons(["Submit", "Resend OTP", "Cancel"], press_otp)

    app.go()


# Resend OTP function
def resend_otp():
    app.otp = generated_otp()
    sending_otp(app.otp)
    app.infoBox("Info", "A new OTP has been sent to your email address.")


# create a GUI variable called app
app = gui("OTP Verification", "400x200")
prompt_otp()
```

Future scope:

By using SMTP Client Protocol, we can implement in online applications.