**ใบงาน เรื่อง การสร้าง GUI Application ด้วยภาษา Python แบบ OOP**

**(ตอนที่ 2) คะแนนเต็ม 10 คะแนน**

**ชื่อ-สกุลนักศึกษา:** นายซุลกิฟลี อาแว เลขที่: 67319010002 **ห้องเรียน:** 1205 วันที่: 3/4/2568

**จุดประสงค์การเรียนรู้**

หลังจากทำใบงานนี้ นักศึกษาจะสามารถ:

• จัดการ Event ใน GUI Application ได้ถูกต้อง

• สร้าง Controller Class เพื่อจัดการข้อมูลและแยก Logic จากส่วน GUI ได้

• เชื่อมต่อและจัดการข้อมูล CRUD กับฐานข้อมูล MySQL ผ่าน GUI ได้

• เข้าใจและจัดการโครงสร้างโค้ดแบบ Modular (1 class ต่อ 1 ไฟล์) ได้อย่างมีประสิทธิภาพ

**เครื่องมือที่ใช้**

• Python 3.x

• Tkinter

• MySQL

• mysql-connector-python

**ติดตั้ง mysql-connector-python ด้วยคำสั่ง:**

pip install mysql-connector-python

**กิจกรรมที่ 1: สร้างฐานข้อมูลและตาราง (1 คะแนน)**

**ขั้นตอนการทำกิจกรรม**

สร้างฐานข้อมูลชื่อ gui\_db และตารางชื่อ users ด้วยโครงสร้างดังนี้:

• id INT AUTO\_INCREMENT PRIMARY KEY

• name VARCHAR(100)

• age INT

ตัวอย่าง SQL:

CREATE DATABASE gui\_db CHARACTER SET utf8 COLLATE utf8\_general\_ci; USE gui\_db;

CREATE TABLE users (

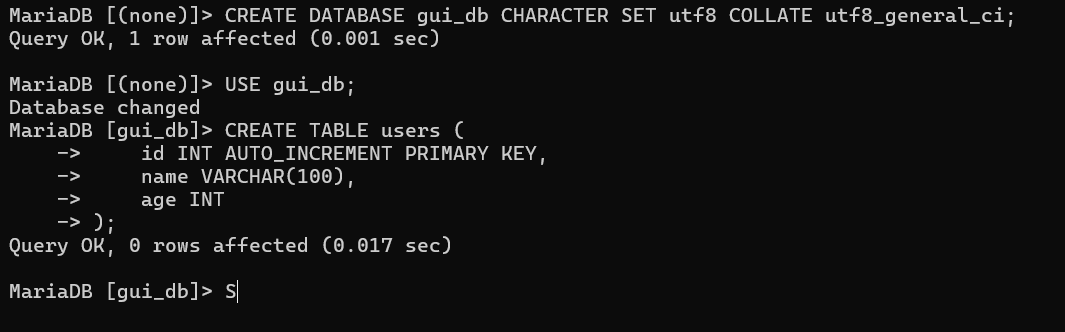
id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

age INT

);

**แนบภาพผลลัพธ์การสร้างฐานข้อมูลและตารางที่นี่**

****

**กิจกรรมที่ 2: สร้าง Database Class (1 คะแนน)**

สร้างไฟล์ Database.py และเขียนคลาส Database เพื่อจัดการฐานข้อมูล MySQL ตัวอย่างโครงสร้างไฟล์ Database.py:

import mysql.connector

class Database:

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

# เชื่อมต่อฐานข้อมูลที่นี่

def add\_user(self, name, age):

# เพิ่มข้อมูลผู้ใช้ที่นี่

def get\_all\_users(self):

# แสดงข้อมูลผู้ใช้ทั้งหมดที่นี่

def delete\_user(self, user\_id):

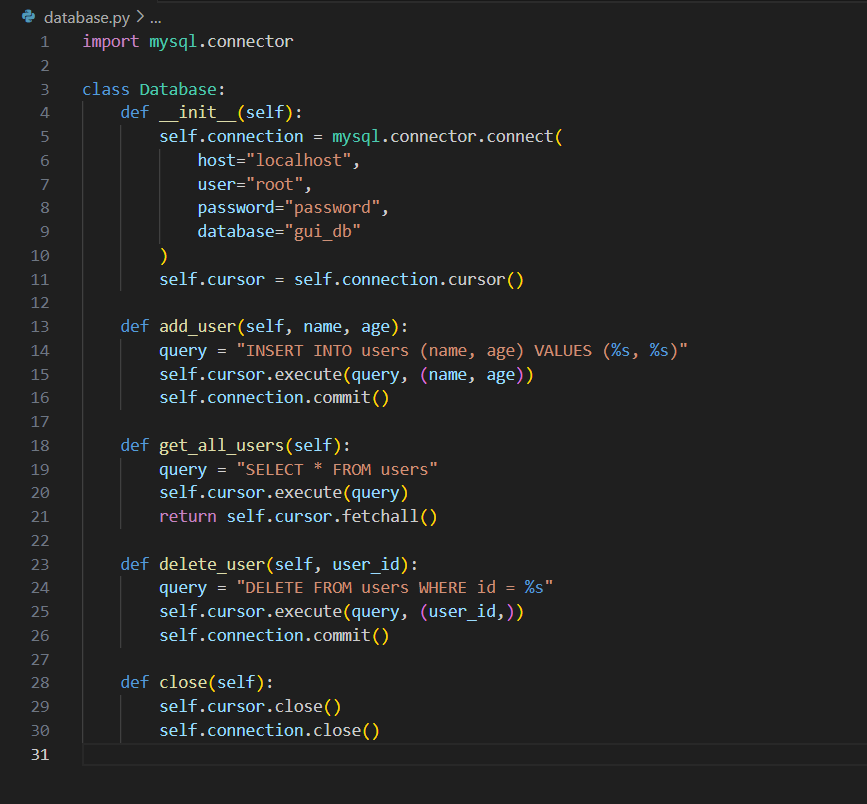
# ลบข้อมูลผู้ใช้ที่นี่

def close(self):

# ปิดการเชื่อมต่อที่นี่

**บันทึกโค้ดที่เขียน:**

# Database.py



**กิจกรรมที่ 3: สร้าง Controller Class (1 คะแนน)** สร้างไฟล์UserController.py เพื่อจัดการข้อมูลผ่าน Database Class ตัวอย่างโครงสร้างไฟล์ UserController.py:

from Database import Database

class UserController:

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

# เชื่อมต่อฐานข้อมูลที่นี่

def add\_user(self, name, age):

# เพิ่มข้อมูลผ่าน Database Class ที่นี่

def get\_all\_users(self):

# ดึงข้อมูลผ่าน Database Class ที่นี่

def delete\_user(self, user\_id):

# ลบข้อมูลผ่าน Database Class ที่นี่

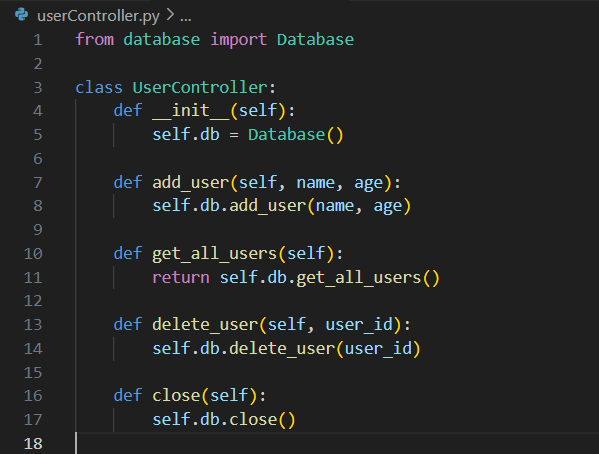
def close(self):

# ปิดการเชื่อมต่อที่นี่

**บันทึกโค้ดที่เขียน:**

# UserController.py

# เขียนโค้ดของคุณที่นี่



**กิจกรรมที่ 4: สร้าง GUI Form เชื่อมต่อกับ Controller (2 คะแนน)** สร้างไฟล์UserForm.py และเขียน GUI เพื่อจัดการข้อมูลผ่าน Controller Class ตัวอย่างโครงสร้างไฟล์ UserForm.py:

import tkinter as tk

from tkinter import messagebox

from UserController import UserController

class UserForm:

def \_\_init\_\_(self, parent):

# สร้าง GUI Form ที่นี่

def save\_user(self):

# จัดการ Event เพิ่มข้อมูลที่นี่

def show\_users(self):

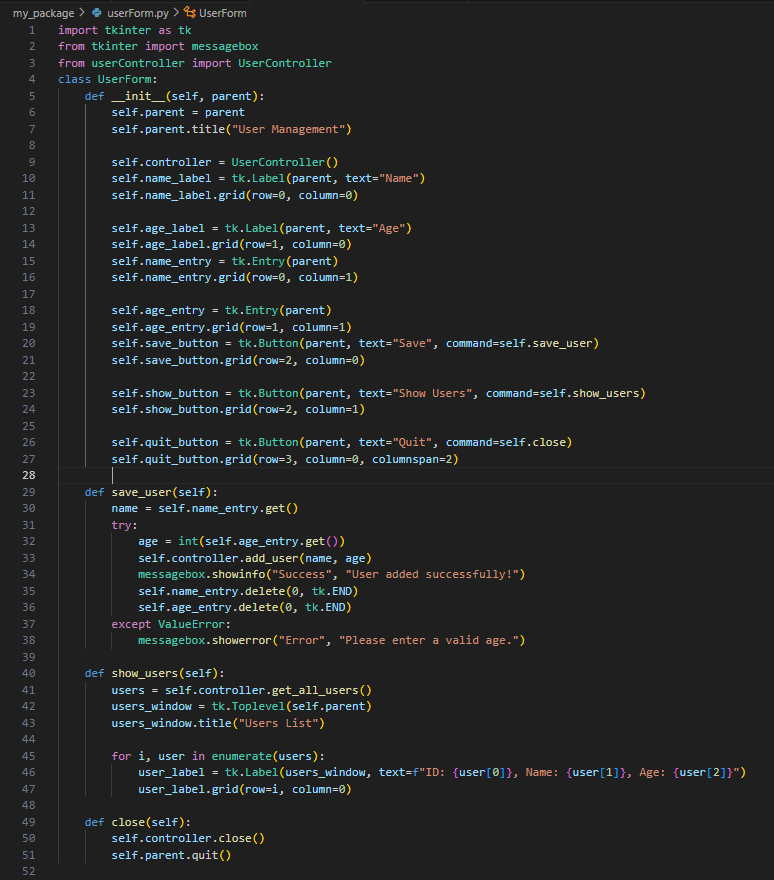
# จัดการ Event แสดงข้อมูลที่นี่

def close(self):

# ปิด Controller ที่นี่

**บันทึกโค้ดที่เขียน:**

# UserForm.py



**กิจกรรมที่ 5: สร้าง MainWindow Class (1 คะแนน)** สร้างไฟล์MainWindow.py สำหรับ GUI หลักของโปรแกรม ตัวอย่างโครงสร้างไฟล์ MainWindow.py:

import tkinter as tk

class MainWindow:

def \_\_init\_\_(self, title, geometry):

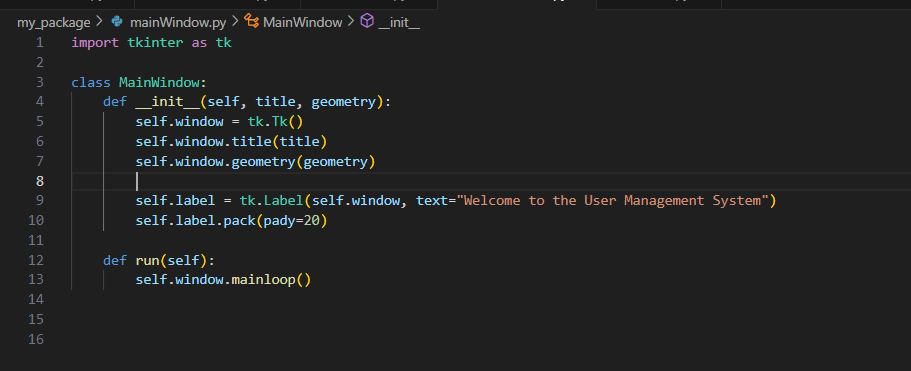
# สร้างหน้าต่างหลักที่นี่

def run(self):

# เริ่ม GUI ที่นี่

**บันทึกโค้ดที่เขียน:**

# MainWindow.py



**กิจกรรมที่ 6: เขียนไฟล์ main.py และทดสอบโปรแกรม (2 คะแนน)** สร้างไฟล์main.py เพื่อเรียกใช้งาน GUI Application แบบเต็มรูปแบบ ตัวอย่างไฟล์main.py:

from MainWindow import MainWindow

from UserForm import UserForm

def main():

# สร้างและแสดง GUI Application ที่นี่

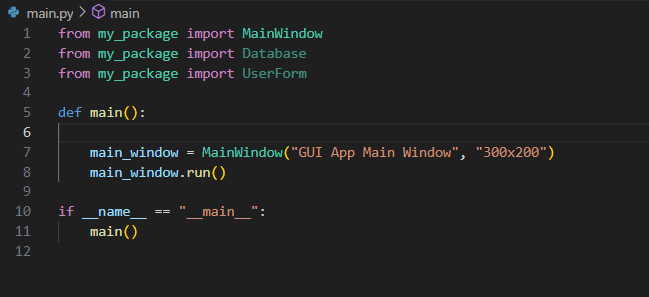
if \_\_name\_\_ == "\_\_main\_\_":

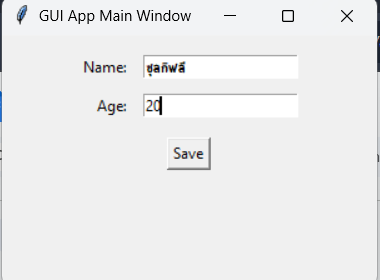
main()

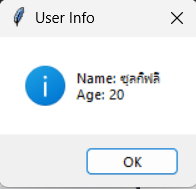
**บันทึกโค้ดที่เขียน:**

# main.py

# เขียนโค้ดของคุณที่นี่

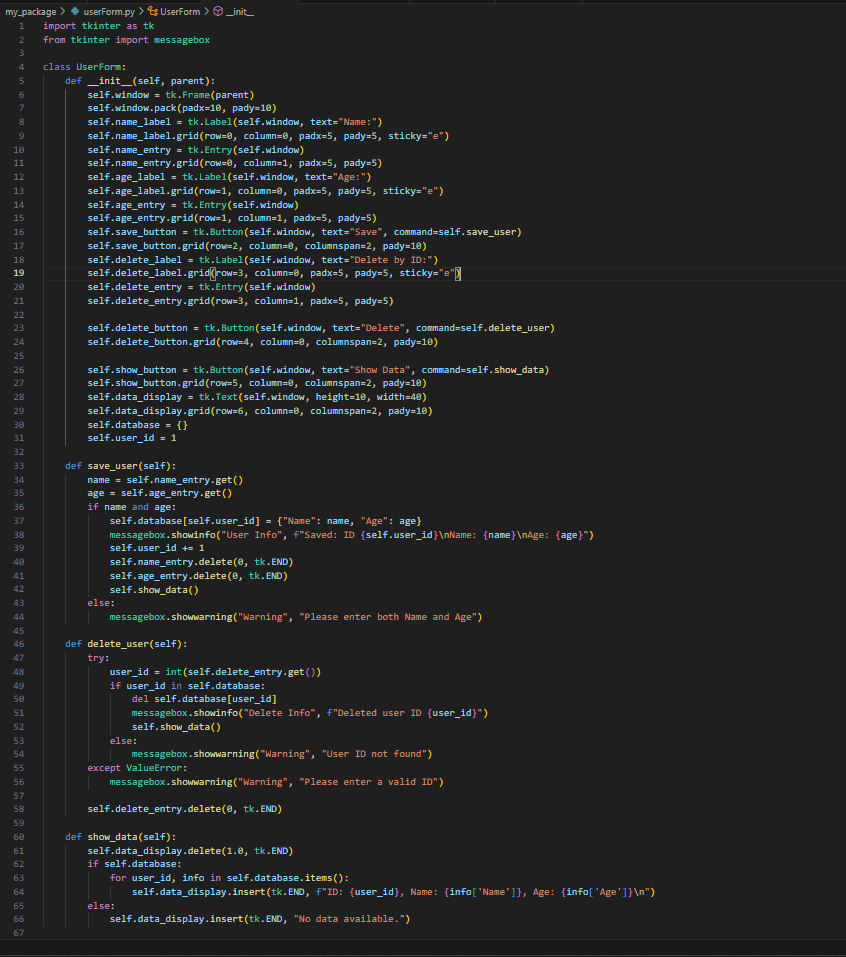


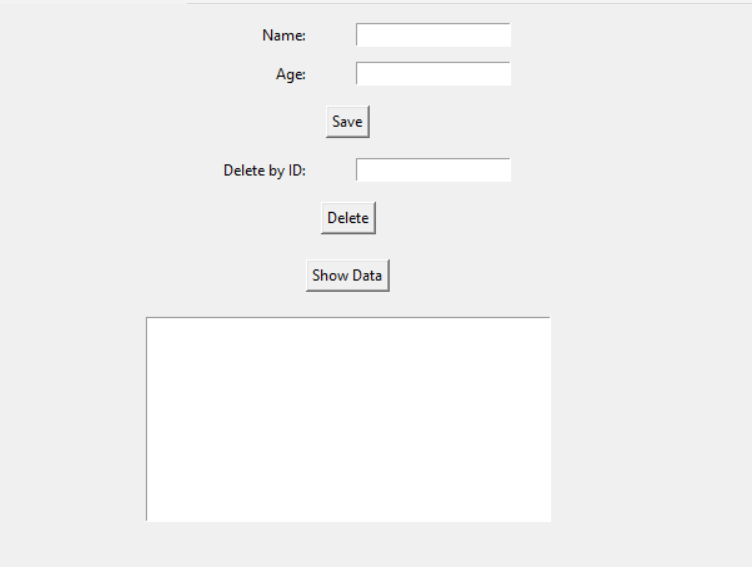
**แนบภาพผลลัพธ์ GUI Application ขณะทำงานที่นี่** 



**กิจกรรมที่ 7: เพิ่มฟีเจอร์ลบข้อมูลผ่าน GUI (1 คะแนน)**

• เพิ่ม Widget (Entry และ Button) ใน GUI เพื่อให้ผู้ใช้สามารถลบข้อมูลได้จาก ID **บันทึกโค้ดที่เพิ่มและผลลัพธ์การลบข้อมูลที่นี่**

# เขียนโค้ดเพิ่มเติมที่นี่   


**แนบภาพผลลัพธ์ที่ได้จากการลบข้อมูลที่นี่  
**

**กิจกรรมที่ 8: ตอบคำถามสรุปความรู้ (1 คะแนน)**

1. การแยก Controller ออกจาก GUI ส่งผลดีต่อโปรแกรมอย่างไร?

ตอบ: ช่วยให้โค้ดแยกส่วนชัดเจน แก้ไขและบำรุงรักษาง่าย ไม่กระทบกันระหว่างการแสดงผลกับการประมวลผล

2. การใช้ MySQL ใน GUI Application ช่วยเพิ่มประสิทธิภาพของโปรแกรมได้อย่างไร?

ตอบ: ช่วยจัดการข้อมูลได้รวดเร็ว รองรับข้อมูลเยอะและค้นหาข้อมูลได้เร็วขึ้น

**เกณฑ์การให้คะแนน (คะแนนเต็ม 10 คะแนน)**

| **เกณฑ์การประเมิน** | **คะแนนเต็ม** | **คะแนนที่ได้** |
| --- | --- | --- |

| กิจกรรมที่ 1: สร้างฐานข้อมูล | 1 |  |
| --- | --- | --- |
| กิจกรรมที่ 2: สร้าง Database Class | 1 |  |
| กิจกรรมที่ 3: สร้าง Controller Class | 1 |  |
| กิจกรรมที่ 4: สร้าง GUI Form เชื่อมต่อ Controller | 2 |  |
| กิจกรรมที่ 5: สร้าง MainWindow Class | 1 |  |
| กิจกรรมที่ 6: เขียนไฟล์ main.py และทดสอบ | 2 |  |
| กิจกรรมที่ 7: เพิ่มฟีเจอร์ลบข้อมูล | 1 |  |
| กิจกรรมที่ 8: ตอบคำถามสรุป | 1 |  |
| **รวมคะแนน** | **10** |  |

**ลงชื่อผู้ส่งงาน: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ วันที่: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ตรวจสอบและให้คะแนนโดย: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ วันที่: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ขอให้นักศึกษาสนุกกับการเรียนรู้ และประสบความสำเร็จในการทำใบงานนะครับ!**