# การประยุกต์ใช้โปรแกรมไพธอนที่มีวิวัฒนาการในการพัฒนาแอปพลิเคชั่นเว็บ

# Applying Evolutionary Python Programs to Development of Web Applications


นาย ไกร ภาญจนวตี

นาย วรุตม์ สุดภู่ทอง



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

# การประยุกต์ใช้โปรแกรมไพธอนที่มีวิวัฒนาการในการพัฒนาแอปพลิเคชั่นเว็บ

# Applying Evolutionary Python Programs to Development of Web Applications


โดย

นาย ไกร กาญจนวตี

นาย วรุตม์ สุดภู่ทอง



อาจารย์ที่ปรึกษา

ดร. วิศิษฏ์ หิรัญกิตติ


ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้โปรแกรมไพธอนที่มีวิวัฒนาการในการพัฒนาแอปพลิเคชั่นชันเว็บ

Applying Evolutionary Python Programs to Web Applications

ผู้จัดทำ

1. นายไกร กาญจนวตี รหัสประจำตัว 40010086
2. นายวรุฒม์ สุดภู่ทอง รหัสประจำตัว 40010698

.....อาจารย์ที่ปรึกษา

( ดร. วิศิษฏ์ หิรัญกิตติ )

# การประยุกต์ใช้โปรแกรมไพธอนที่มีวิวัฒนาการในการพัฒนาแอปพลิเคชั่นเว็บ

นายไกร กาญจนวตี 40010086
นายวรุตม์ สุดภู่ทอง 40010698
ดร.วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

**บทคัดย่อ**

ในการสร้างซอร์ฟแวร์เอเย่นต์ (Software Agent) เพื่อให้เป็นเว็บแอปพลิเคชั่น (Web Application) ที่มีความยืดหยุ่นสูง สามารถเรียนรู้และปรับเปลี่ยนตัวเองได้นั้น จำเป็นต้องพัฒนาโดยใช้ภาษาคอมพิวเตอร์ที่เป็นอินเตอพรีเตอร์ (Interpreted Computer Language) โครงงานนี้ได้ศึกษาการใช้งานภาษาไพธอนซึ่งเป็นภาษาเชิงวัตถุในการพัฒนาซอร์ฟแวร์เอเย่นต์ที่ทำงานบนเว็บ และได้คิดค้นวิธีการสร้างโปรแกรมภาษาไพธอนที่มีวิวัฒนาการโดยสามารถเพิ่ม, เปลี่ยนแปลง หรือลบกระบวนการในคลาส (Class) ขึ้น ในการทดสอบวิธีการดังกล่าวได้มีการพัฒนาตัวอย่างโปรแกรมที่มีวิวัฒนาการสำหรับใช้งานบนเว็บ ขึ้นมาสองโปรแกรมด้วยกัน คือ โปรแกรมเอเย่นต์วาดภาพ (Drawing Agent) ที่สามารถเรียนรู้วิธีการวาดภาพง่าย ๆ ได้ และโปรแกรมเพอซันแนลแอซซิสแทนต์ (Personal Assistant) ที่สามารถปรับเปลี่ยนการทำงานได้

# Applying Evolutionary Python Programs to Development of Web Applications

Krai Kanchanawatee
Warut Sudpoothong
Dr. Visit Hirankitti Advisor
2000

**Abstract**

In order to develop a software agent as a web application, which is very flexible to learn and adapt itself, an interpreted computer language is needed. Hence, This project investigates the usage of Python, an interpreted object-oriented language, to develop a software agent running on the web. We have invented an approach to make a Python program evolve in a way that class's members can be added, replaced, and removed. To test this idea, we have applied this approach to develop two Python-based web applications, i.e. Drawing Agent, which can learn how to draw simple pictures and Personal Assistant whose functionalities are able to be changed.

# กิตติกรรมประกาศ

ปริญญานิพนธ์นี้คงจะสำเร็จลุล่วงไปมิได้ ถ้าไม่ได้รับความช่วยเหลือจากบุคคลหลายๆ ท่าน ด้วยกัน บุคคลแรกคือ ดร.วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษาซึ่งให้คำปรึกษา แนะนำ และเอาใจใส่ด้วยดีเสมอมา รวมถึงให้โอกาสในการแก้ไขข้อบกพร่องต่างๆ ในการทำโครงงานนี้

ขอขอบคุณเพื่อนๆ ทุกคนที่ช่วยเหลือในด้านต่างๆ ตลอดจนการจัดทำรายงานฉบับนี้ให้สำเร็จลงได้ด้วยดี ขอขอบพระคุณบิดา มารดาผู้ให้กำเนิด ที่คอยสนับสนุน ให้คำปรึกษาในด้านต่างๆ ตลอดมา สุดท้ายนี้ขอกราบขอบพระคุณบิดามารดาและอาจารย์ที่ปรึกษามา ณ ที่นี้

ไกร กาญจนวตี
วรุตม์ สุดภู่ทอง

# สารบัญ

หน้าที่

หน้าที่

# สารบัญตาราง

หน้าที่

# สารบัญรูปภาพ

หน้าที่

หน้าที่

# บทที่ 1
# บทนำ

## 1.1 ความสำคัญและที่มา

เป็นที่ยอมรับว่าอินเตอร์เน็ตเข้ามามีบทบาทในชีวิตประจำวันของเรา การทำกิจกรรมต่าง ๆ มีการนำสื่อทางอินเตอร์เน็ตมาประยุกต์ใช้งานมากขึ้น จึงทำให้ต้องมีการศึกษาและพัฒนาศักยภาพของโปรแกรมที่นำมาใช้งานบนอินเตอร์เน็ตให้สามารถทำงานได้อย่างมีประสิทธิภาพและเป็นอัตโนมัติมากยิ่งขึ้น

จากเหตุผลดังกล่าวจึงได้มีการศึกษาในเรื่องการพัฒนาโปรแกรมที่ใช้เทคโนโลยีทางด้านเอเย่นต์ (Agents) เพื่ออำนวยความสะดวกในการใช้งาน เอเย่นต์จะทำงานแทนคนที่เป็นผู้ใช้โดยผู้ใช้ไม่จำเป็นจะต้องสั่งงานทั้งหมด มีการสนับสนุนการเรียนรู้เพื่อเพิ่มขีดความสามารถให้กับตัวเอเย่นต์เองในการตอบสนองความต้องการของผู้ใช้ ทำให้ผู้ใช้สามารถระบุสิ่งที่ต้องการให้แก่เอเย่นต์ โดยไม่ต้องระบุวิธีการ เอเย่นต์สามารถหาสิ่งที่ผู้ใช้ต้องการได้จากความรู้ที่มี หรือติดต่อสื่อสารข้อมูลกับเอเย่นต์อื่น ๆ เพื่อให้ได้มาซึ่งผลคำตอบที่ผู้ใช้ต้องการ

เอเย่นต์แต่ละตัวจะมีกระบวนการทำงานของตัวเองเพื่อช่วยให้สามารถแก้ปัญหาที่ผู้ใช้งานต้องการ ซึ่งเอเย่นต์ตัวหนึ่งสามารถทำงานเฉพาะให้สำหรับแต่ละผู้ใช้งานทำให้ขาดความยืดหยุ่นในการใช้งาน ถ้าหากเอเย่นต์แต่ละตัวมีความสามารถในการแลกเปลี่ยนกระบวนการทำงานกับเอเย่นต์อื่นที่มีความสามารถแตกต่างกันหรือรับกระบวนการทำงานจากผู้ใช้งาน และนำมาพัฒนาตัวเองได้ก็จะทำให้การใช้งานเอเย่นต์มีความยืดหยุ่นและเกิดประโยชน์มากยิ่งขึ้น

โครงงานนี้ได้มีการคิดหาแนวทางในการพัฒนาเอเย่นต์ให้มีความสามารถสูงขึ้น โดยใช้หลักการของการเปลี่ยนแปลงกระบวนการทำงาน (Method) ที่มีวิวัฒนาการเข้ามาช่วยทำให้เอเย่นต์แต่ละตัวสามารถที่จะนำกระบวนการทำงานที่ได้รับมาจากเอเย่นต์อื่นหรือผู้ใช้งาน มาพัฒนากระบวนการของตัวเองให้มีความสามารถมากขึ้น

หลักการในการทำให้เอเย่นต์สามารถพัฒนากระบวนการทำงานให้มากขึ้น หรือสามารถเปลี่ยนวิธีการทำงานได้ตามที่ผู้ใช้ต้องการ เราได้ใช้แนวคิดของการวิวัฒนาการของกระบวนการของเอเย่นต์ โดยอาศัยคุณสมบัติการอินเฮริแทนซ์ (Inheritance) ซึ่งเป็นคุณสมบัติที่มีในภาษาที่เป็นโปรแกรมเชิงวัตถุ (Object Oriented)เข้ามาช่วยในการเปลี่ยนแปลงกระบวนการทำงาน

จากการที่ใช้หลักการของภาษาที่เป็นโปรแกรมเชิงวัตถุ เราจึงได้เลือกภาษาไพธอน (Python) ซึ่งเป็นภาษาเชิงวัตถุที่เป็นอินเตอพรีเตอร์ (Interpreter) ทำให้มีความยืดหยุ่นในการจัดการกับกระบวนการทำงานของโปรแกรม นอกจากนั้นไพธอนยังเป็นภาษาที่มีความสามารถสูงในการจัดการกับสายอักขระ (String) ซึ่งสามารถนำไปเขียนสคริปต์ซีจีไอ (Common Gateway Interface : CGI) ได้อย่างมีประสิทธิภาพ ทำให้ง่ายในการนำไปประยุกต์ใช้ในการพัฒนาโปรแกรมบนเว็บ

หลังจากการพัฒนาโปรแกรมเอเย่นต์ให้มีความสามารถในการพัฒนาตัวเองแบบมีวิวัฒนาการ ทำให้โปรแกรมเอเย่นต์มีความยืดหยุ่นในการทำงานมากขึ้น เราจะนำหลักการดังกล่าวไปประยุกต์ใช้กับแอปพลิเคชั่นที่ทำงานบนเว็บซึ่งคาดว่าจะทำให้เอเย่นต์ที่ทำงานบนเว็บมีความสามารถมากยิ่งขึ้นและสามารถใช้งานได้ง่ายดายมากยิ่งขึ้น

**1.2 วัตถุประสงค์ของงานวิจัย**

1. ศึกษาถึงวิธีการทำงานของโปรแกรมที่ทำงานเป็นเอเย่นต์
2. ศึกษาและพัฒนาหาวิธีการเปลี่ยนแปลงกระบวนการทำงานโดยใช้หลักการวิวัฒนาการ
3. ศึกษาและทดลองวิธีการในการรับกระบวนการทำงานและนำมาพัฒนาโปรแกรมเอเย่นต์ให้มีความสามารถในการทำงานมากขึ้น โดยการใช้ภาษาไพธอนในการพัฒนา
4. ทำการประยุกต์สร้างโปรแกรมเอเย่นต์ที่ทำงานบนเว็บซึ่งมีการใช้หลักการในการรับกระบวนการทำงานและนำมาพัฒนาตนเองโดยหลักการวิวัฒนาการให้มีความสามารถมากขึ้น

**1.3 ขอบเขตของงานวิจัย**

1. ศึกษาคุณสมบัติและการทำงานของโปรแกรมเอเย่นต์
2. ศึกษาคุณสมบัติและการใช้งานภาษาไพธอน
3. หารูปแบบวิธีการในการเปลี่ยนแปลงกระบวนการทำงานแบบมีวิวัฒนาการ
4. ศึกษาวิธีการในการรับกระบวนการทำงานและนำมาพัฒนากระบวนการทำงานของตัวโปรแกรมเอเย่นต์ให้มีความสามารถมากยิ่งขึ้น
5. พัฒนาตัวอย่างโครงงานเพื่อแสดงลักษณะการทำงานของเอเย่นต์ที่มีการประยุกต์ใช้แนวคิดในการพัฒนากระบวนการทำงานแบบมีวิวัฒนาการ โดยใช้ภาษาไพธอนในการพัฒนาโปรแกรม

**1.4 วิธีการดำเนินงาน**

1. ศึกษารายละเอียดต่างๆเกี่ยวกับเอเย่นต์ เช่น ความหมายของโปรแกรมที่เป็นเอเย่นต์ ประเภทของเอเย่นต์ คุณสมบัติของเอเย่นต์ และวิธีการทำงานของเอเย่นต์
2. ศึกษาหาวิธีในการทำให้เอเย่นต์สามารถที่จะนำกระบวนการทำงานจากเอเย่นต์อื่นหรือผู้ใช้งานมาพัฒนาความสามารถของตัวเอเย่นต์เอง โดยคิดหาวิธีและรูปแบบของการเปลี่ยนแปลงกระบวนการทำงานที่ใช้แนวความคิดของการวิวัฒนาการ
3. ศึกษาภาษาไพธอนในเรื่องคุณสมบัติของภาษา การเขียนโปรแกรมโดยใช้ภาษาไพธอน และแนวทางการประยุกต์ใช้งานเพื่อนำมาพัฒนาเป็นเอเย่นต์ที่มีความสามารถในการเปลี่ยนแปลงกระบวนการทำงาน

4. ศึกษาการทำงานของโปรแกรมที่ทำงานบนระบบเครือข่ายรวมทั้งรายละเอียดต่างๆของการทำงานของโปรแกรมที่ทำงานบนเว็บ และวิธีการพัฒนาโปรแกรมที่ทำงานโดยใช้งานบนเว็บ
5. นำภาษาไพธอนมาพัฒนาตัวอย่างโครงงานแสดงการทำงานของเอเย่นต์ที่มีการประยุกต์แนวความคิดในการพัฒนากระบวนการการทำงานของตัวเองโดยใช้แนวคิดของการมีวิวัฒนาการ

# บทที่ 2
# เอเย่นต์

## 2.1 เอเย่นต์

คำว่า "เอเย่นต์" ที่เราใช้กันในชีวิตประจำวันนั้นหมายถึงนายหน้าหรือบุคคลที่กระทำการใด ๆ แทนเรา แต่ความหมายของเอเย่นต์ที่ใช้ในรายงานฉบับนี้คือโปรแกรมที่สามารถรับข้อมูล และเป้าหมายของผู้ใช้ในการทำงานใด ๆ แล้ว ตัวเอเย่นต์จะทำการนำข้อมูลที่ได้ พร้อมทั้งรายละเอียดที่เก็บไว้ในตัวมารวมกัน และประมวลผลเพื่อทำงานให้ได้ผลลัพธ์เป็นเป้าหมายตามที่ผู้ใช้ได้ตั้งไว้

## 2.2 การแบ่งประเภทของเอเย่นต์

การแบ่งประเภทของเอเย่นต์นั้น มีอยู่หลายวิธี คือ อาจแบ่งโดยความสามารถของเอเย่นต์, วิธีในการ Process Agent หรืออาจแบ่งโดยหน้าที่ที่เอเย่นต์ทำงาน

- **ความสามารถของเอเย่นต์**

เราสามารถใช้สิ่งเหล่านี้ในการวัดความสามารถเอเย่นต์ กล่าวคือ Agency, Intelligence และ Mobility

- Agency เกี่ยวกับความเป็นอิสระของ Software Agent ในการเป็นตัวแทนของผู้ใช้ และทำ Action ที่มีประโยชน์ต่อผู้ใช้งาน
- Intelligent หมายถึงความสามารถของเอเย่นต์ ที่จะใช้ Knowledge and Processing ในการแก้ปัญหา ซึ่งเอเย่นต์สามารถทำงานที่ซับซ้อนได้โดยการใช้เทคนิคของ Artificial Intelligence
- Mobile การจัดว่าเอเย่นต์จะเป็น Mobile ได้นั้น เอเย่นต์จะต้องสามารถเคลื่อนที่ระหว่างระบบในเครือข่าย Mobility ทำให้ Intelligent Agent มีความซับซ้อน เนื่องจากสิ่งนี้จะเกี่ยวเนื่องไปถึงความปลอดภัย และค่าใช้จ่ายของงาน

- **วิธีการทำงาน (Processing Strategies)**

Agent Reactive or Reflex Agent เป็นเอเย่นต์ที่มีการตอบสนองต่อสิ่งเร้าภายนอก และข้อมูลของสิ่งแวดล้อมซึ่งได้จาก Sensor ของเอเย่นต์ โดยการตอบสนองนี้ Event–condition-action Mode การทำงานของ Reactive Agent นั้นคล้ายกับ Neural Network ที่ตอบสนองต่อพฤติกรรมที่ปรากฏ Application ของเอเย่นต์เหล่านี้จำกัดอยู่ที่ Robot ที่ใช้ Sensor ในการรับรู้

Deliberative or Goal-directed Agent เป็นเอเย่นต์ที่มี Domain Knowledge และความสามารถในการวางแผนการกระทำเพื่อให้ได้เป้าหมายที่กำหนด และสามารถติดต่อกับเอเย่นต์ตัวอื่น ๆ เพื่อรับหน้าที่ เอเย่นต์สามารถใช้เทคนิคของ Artificial Intelligent Reasoning

Collaborative Agent เป็นเอเย่นต์ที่ทำงานร่วมกันเพื่อแก้ปัญหาต่าง ๆ การติดต่อระหว่างเอเย่นต์เป็นสิ่งสำคัญ ในขณะที่เอเย่นต์แต่ละตัวทำงานอย่างอิสระ การร่วมมือกันทำให้เกิดประโยชน์ ในระบบ

Collaborative Agent เอเย่นต์ต่าง ๆ สามารถแก้ปัญหาใหญ่ๆได้ โดยใช้หน้าที่และ Domain Knowledge ของเอเย่นต์แต่ละตัว ซึ่งในระบบนี้เอเย่นต์ยังต้องสามารถที่จะแลกเปลี่ยนข้อมูลที่เกี่ยวกับ Benefits, Desires และจุดมุ่งหมายต่าง ๆ พร้อมทั้งสามารถใช้งาน Knowledge ร่วมกันได้

Mobile Agent เป็น Software process ที่สามารถเคลื่อนที่ผ่านระบบคอมพิวเตอร์ในเครือข่าย และทำงานของเอเย่นต์สำหรับเจ้าของเอเย่นต์นั้น ๆ ได้ ข้อดีของ Mobile Agent คือช่วยลดการติดต่อระหว่างระบบในบ้านกับ Remote System โดยการอนุญาติให้เอเย่นต์ไปยัง Remote System และเข้าถึงข้อมูลบนระบบนั้นโดยปัญหาที่สำคัญของ Mobile Agent คือ ความปลอดภัยในการใช้งาน

- **หน้าที่ในการทำงาน (Processing Function)**

เราสามารถแบ่งประเภทของเอเย่นต์ตามหน้าที่การทำงานเช่น Search Agent ซึ่งทำหน้าที่ในการเข้าไปในอินเตอร์เน็ต และหาข้อมูล (Documents) ที่เราต้องการ หรือ Filter Agent ซึ่งทำการค้นหาส่วนที่เราต้องการและกำจัดส่วนที่ไม่ต้องการออกไป หรือ Mail ที่รับมา หรือ Domain Specific Assistant ซึ่งอาจนำมาใช้ช่วยในการจัดตารางเวลา หรือการจัดเวลาเดินทางให้กับผู้ใช้

Interface Agent ซึ่งทำหน้าที่เป็นผู้ช่วยส่วนตัวเพื่อทำงานของผู้ใช้ให้สำเร็จ โดย Interface Agent จะเรียนและปรับตัวไปตามงานของผู้ใช้งานซึ่ง Patti Maes ได้อธิบายถึงวิธีในการเรียนรู้ที่สามารถเกิดขึ้นได้ คือ

1. เอเย่นต์ สามารถเรียนรู้ได้โดยการดูการทำงานของเจ้าของและเลียนแบบ
2. เอเย่นต์สามารถทำงานให้เจ้าของและเรียนรู้โดยการรับผลตอบสนองกลับมาจากผู้ใช้
3. เอเย่นต์สามารถรับคำสั่งโดยตรงจากผู้ใช้ เช่น เกิดเหตุการณ์นี้ขึ้นจะทำอย่างไร
4. ขอคำแนะนำจากเอเย่นต์ตัวอื่น และเรียนรู้จากประสบการณ์ของตัวเอเย่นต์เอง แต่ Interface Agent จะทำงานร่วมกับเจ้าของ ไม่ได้ทำงานร่วมกับเอเย่นต์ตัวอื่น โดยจะทำเฉพาะขอคำแนะนำเท่านั้น ซึ่งถ้า Interface Agent สามารถใช้ Knowledge ร่วมกันในการทำงานเมื่อคนหนึ่งในกลุ่มทำงานรู้วิธีในการทำงานบางอย่างผู้ใช้คนอื่นก็จะรู้ด้วยโดยผ่านทาง Interface Agent

Information Agent เอเย่นต์บางตัวทำงานโดยการเข้าไปค้นหาข้อมูลใน Internet ตามที่ผู้ใช้ต้องการ บางตัวจะทำหน้าที่ในการคัดเลือกข้อมูลที่มากับ E-mail ระบบ Information Agent จะแก้ปัญหาโดยรับข้อมูลที่ถูกต้องในเวลาที่เหมาะสม และไม่สนว่าข้อมูลจะมากหรือน้อย แต่จะสนใจในความถูกต้องของข้อมูล ตัวอย่างของ Information Agent เช่น Spider ซึ่งจะนำข้อมูลที่ผู้ใช้ต้องการกลับมา

## 2.3 คุณสมบัติของเอเย่นต์

1. Autonomous คือความเป็นอิสระ สามารถทำงานด้วยตัวของมันเองได้ตั้งแต่ต้นจนจบ
2. Mobility คือ เอเย่นต์สามารถเคลื่อนที่ผ่านไปในระบบเครือข่ายได้
3. Intelligent คือ ความฉลาด มีความสามารถในการตัดสินใจเพื่อทำการการแก้ปัญหาของเอเย่นต์

4. Persistent and Goal-Directed คือ เอเย่นต์จะทำงานตามที่ได้รับมอบหมายโดยไม่เปลี่ยนแปลงการทำงาน
5. Reaction คือ มีการตอบสนองต่อเหตุการณ์และมีการกระทำเพื่อตอบสนองต่อเหตุการณ์ที่เกิดขึ้น
6. Secure คือ ความปลอดภัยในการใช้งานเอเย่นต์ โดยเอเย่นต์ต้องไม่ก่อให้เกิดความเสียหายแก่ผู้ใช้

**2.4 เอเย่นต์ชาญฉลาด (Intelligent Agent)**

โปรแกรมที่ถูกสร้างขึ้นมาเพื่อทำงานตามเป้าหมายที่ผู้ใช้กำหนดขึ้นนั้น ถูกเรียกว่าเอเย่นต์ ซึ่งเอเย่นต์นี้ได้รับการคาดหวังจากผู้ใช้ให้ช่วยแบ่งเบาภาระต่าง ๆ หากแต่ความสามารถของเอเย่นต์ก็มีขีดจำกัดเนื่องจากเป็นเพียงโปรแกรม ไม่สามารถคิดอ่านได้อย่างมนุษย์ ได้มีความพยายามที่จะสร้างโปรแกรมที่มีความสามารถสูง เรียนรู้และพัฒนาตนเองให้ทำงานได้อย่างถูกต้องด้วยประสิทธิภาพสูงสุด จึงมีการคิดสร้างเอเย่นต์ชาญฉลาดขึ้น ถึงแม้เอเย่นต์นี้จะไม่สามารถทำงานแทนมนุษย์ได้ทุกอย่าง แต่ก็มีความสามารถสูงกว่าเอเย่นต์ทั่วไป ทำให้ผู้ใช้ไม่จำเป็นต้องตรวจสอบการทำงาน หรือหมายความว่าผู้ใช้ไม่จำเป็นต้องมีความรู้ทางงานนั้น ๆ เป็นพิเศษ เอเย่นต์ต้องสามารถทำงานแทนได้โดยมีการปรับปรุงแก้ไขข้อผิดพลาดเองได้ หลักการทำงานของเอเย่นต์ชาญฉลาดมีหลายวิธีการ แต่จะนำเสนอวิธีการ Events-Conditions-Actions ซึ่งเป็นวิธีที่ง่ายทั้งนี้เพื่อเป็นตัวอย่าง

เพื่อให้เกิดความเข้าใจในหลักการทำงานแบบ Events-Conditions-Actions จะขอยกตัวอย่างการทำงานของเอเย่นต์ชาญฉลาดที่มีหน้าที่แยกประเภทของจดหมายอิเล็กโทรนิกให้กับผู้ใช้ โดยตัวเอเย่นต์นี้มีการทำงานเกี่ยวข้องกับปัจจัย 3 อย่างด้วยกันคือ Events, Recognize conditions และ Actions

1. Events คือ เหตุการณ์ที่เกิดขึ้นที่เปลี่ยนแปลงสภาพแวดล้อมของเอเย่นต์ ซึ่งเอเย่นต์จำเป็นต้องรับรู้ เพื่อใช้เป็นตัวขับดัน (Trigger) ให้เกิดการทำงานต่อไป สำหรับเอเย่นต์ในตัวอย่างนั้น จะมี Events คือการเข้ามาถึงของจดหมายในตู้ไปรษณีย์อิเล็กทรอนิกส์ของผู้ใช้
2. Recognize Conditions คือ รูปแบบหรือเงื่อนไขในการจดจำเหตุการณ์ ซึ่งถูกกำหนดไว้ก่อนหน้านี้ ไม่ว่าผู้ใช้จะเป็นผู้กำหนด หรือเอเย่นต์กำหนดขึ้นเองจากการเรียนรู้ก็ตาม และจะใช้ช่วยเอเย่นต์ตัดสินใจเลือกปฏิบัติงานต่อไป จากตัวอย่างเมื่อมีการเข้ามาของจดหมาย ซึ่งก็คือ Event เอเย่นต์จะทำการตรวจสอบกับเงื่อนไขที่ได้กำหนดไว้เพื่อพิจารณาว่า Event นั้นมีความหมายอย่างไร เงื่อนไขนี้อาจมีหลายอย่างมากมาย ถ้ามีมากจะเรียกว่าข้อมูลความรู้ (Knowledge) ผู้ใช้อาจกำหนดให้เอเย่นต์เก็บเฉพาะจดมายจากผู้ที่ผู้ใช้รู้จักเท่านั้น เมื่อทราบเงื่อนไขนี้เอเย่นต์จะทำการตรวจสอบว่าเป็นจดหมายจากผู้ที่ผู้ใช้รู้จักหรือไม่ หากผู้ส่งไม่เป็นที่รู้จักเอเย่นต์ก็จะตอบสนองอย่างหนึ่ง เช่นลบจดหมายฉบับนั้นทิ้ง และหากเป็นผู้ที่รู้จักเอเย่นต์ก็จะตอบสนองอีกอย่างหนึ่ง
3. Actions คือ ผลตอบสนองของเอเย่นต์ที่กระทำเมื่อมีเหตุการณ์ต่าง ๆ ที่เกิดขึ้น จากตัวอย่างเมื่อมีการตรวจสอบเงื่อนไข และทราบว่าเป็นจดหมายจากผู้ที่ผู้ใช้รู้จักหรือไม่แล้ว การ

กระทำของเอเย่นต์หลังจากนั้นถือเป็น Actions ของ Event นั้น ๆ เช่นการลบจดหมายทิ้ง หรือการเปิดแสดงจดหมายให้กับผู้ใช้เป็นต้น

เอเย่นต์ที่อยู่ในตัวอย่างนี้มีความแตกต่างจากตัวกรองจดหมายทั่วไป เนื่องจากความสามารถของเอเย่นต์ที่ไม่ต้องพึ่งพาผู้ใช้ โปรแกรมธรรมดาทั่วไปนั้นผู้ใช้จำเป็นต้องป้อนคำสั่งให้กับโปรแกรมเพื่อทำงาน แต่สำหรับเอเย่นต์ชาญฉลาดนั้นมีการใช้ข้อมูลความรู้ที่เพิ่มเติมได้เองในการแก้ปัญหา ผู้ใช้อาจบอกเพียงว่า ผลการทำงานของเอเย่นต์นั้นมีความถูกต้องหรือไม่ก็ได้ และหากการทำงานของเอเย่นต์ไม่ถูกต้อง เอเย่นต์จะทราบจากผู้ใช้และทำการแก้ไขพัฒนาความถูกต้องให้กับตนเองต่อไป

จากการที่เอเย่นต์ชาญฉลาดต้องมีความสามารถในการพัฒนาตนเองได้นี้เอง จึงมีการพัฒนากระบวนการในการพัฒนาตนเองสำหรับเอเย่นต์ชาญฉลาดขึ้น กระบวนการดังกล่าวนี้มีหลากหลายแนวความคิดด้วยกัน แนวความคิดหนึ่งคือ การเปลี่ยนกระบวนการในการคิด (Method) ของเอเย่นต์ให้เอเย่นต์มีวิธีการคิดเปลี่ยนไปจากเดิม

# บทที่ 3
# ภาษาไพธอน (Python)

ภาษาไพธอนถูกกล่าวถึงไว้โดย Mark Lutz และ David Ascher ว่า "ภาษาไพธอนคือภาษาสคริปต์ที่มีความสามารถเชิงวัตถุ" (Learning Python, 1999) เนื่องจากภาษาไพธอนใช้การแปรความหมายในแบบอินเตอพรีเตอร์ (Interpreter) ประกอบกับการที่ภาษานี้มีการออกแบบให้ไม่ขึ้นกับระบบปฏิบัติการใด ๆ จึงเหมาะในการใช้งานเป็นภาษาสคริปต์ โดยเฉพาะการใช้งานคู่กับภาษาจาวา เนื่องจากได้มีการสร้างภาษาไพธอนให้ทำการประมวลผลโดยจาวา และสามารถเรียกใช้งานคำสั่งต่าง ๆ ในภาษาจาวาได้ ภาษาไพธอนมีการพัฒนาอย่างต่อเนื่องโดยกลุ่ม ไพธอนออแกนไนเซชัน(Python Organization) เวอร์ชันล่าสุดคือ 2.0 โดยมีผู้พัฒนาซอฟต์แวร์หลายแห่งให้การสนับสนุน จากการใช้งานพบว่าภาษานี้มีความสามารถในการประมวลผลสายอักขระสูง แต่การใช้งานในการพัฒนาอินเตอร์เฟส (Interface) กับผู้ใช้นั้นทำได้ยาก จึงต้องใช้งานควบคู่กับภาษา Tcl, Tk, COM หากใช้งานภาษาไจธอน (Jython) ซึ่งออกแบบมาให้ใช้งานกับจาวาก็สามารถใช้แอปเพรต (Applet) หรือสวิง (Swing) ของภาษาจาวาได้

ภาษาไพธอนมีความยืดหยุ่นสูง สามารถนำมาใช้งานได้หลากหลายไม่ว่าการพัฒนาโปรแกรมทั่วไปที่ทำได้ง่าย และนำกลับมาใช้ใหม่ได้ การรวมภาษาไพธอนเข้าไว้ในโปรแกรมทำให้สามารถเปลี่ยนแปลงโปรแกรมนั้นให้มีความเหมาะสมกับการใช้งานมากขึ้น (Customize) ใช้ภาษานี้ในการทำโปรแกรมต้นแบบ (Prototyping) ก็สามารถทำได้อย่างรวดเร็ว เนื่องจากภาษานี้สามารถใช้งานควบคู่กับภาษาอื่นได้เป็นอย่างดี ใช้ในการประมวลผลฐานข้อมูล และใช้สร้างระบบปัญญาประดิษฐ์ โครงงานนี้ได้ใช้ภาษาไพธอนเป็นภาษา CGI เนื่องด้วยความสามารถในการจัดการกับสายอักขระ และสามารถใช้งานคู่กับเซิร์ฟเวอร์ต่าง ๆ ได้ซึ่งโครงงานนี้ใช้ Apache เป็นเซิร์ฟเวอร์ ข้อดีที่สำคัญอีกข้อหนึ่งนอกเหนือจากความสามารถของตัวภาษาคือ ในการใช้งานภาษานี้ ผู้ใช้สามารถดาวน์โหลดตัวแปรภาษามาได้จาก www.python.org โดยไม่ต้องเสียค่าใช้จ่ายใด ๆ

## 3.1 คุณสมบัติของภาษาไพธอน

ภาษาไพธอน มีความสามารถที่เด่นหลายประการคือ

- มีความสามารถเชิงวัตถุ ภาษาไพธอนถูกสร้างมาเป็นภาษาเชิงวัตถุ (Object Oriented Language) เช่นเดียวกับภาษา C++ และจาวา และสามารถสร้างคลาสย่อยจากภาษาทั้งสองได้อีกด้วย จึงเหมาะแก่การเป็นภาษาสคริปต์ให้กับทั้งภาษา C++ และจาวา
- ไม่ขึ้นกับระบบปฏิบัติการใด ภาษาไพธอนพัฒนาโดยใช้ ANSI C โปรแกรมที่เขียนขึ้นจากภาษาไพธอนจะสามารถใช้ได้กับทุก ๆ ระบบปฏิบัติการที่มีตัวแปรภาษาไพธอนอยู่ โดยตัวแปรภาษาจะแปรคำสั่งให้เป็น bytecode ที่ระบบปฏิบัติการนั้นเข้าใจ
- มีการจัดการกับหน่วยความจำอัตโนมัติ และมีความสามารถในการเปลี่ยนชนิดของตัวแปร (Type) ได้เอง ทำให้การพัฒนาโปรแกรมทำได้ง่าย

- มีความสามารถในการใช้งานคู่กับภาษาต่าง ๆ ได้หลากหลาย มีการใช้งานภาษาไพธอนคู่กับภาษาต่าง ๆ เช่น จาวา, XML, C++ สามารถใช้กับ Common Object Model ได้

| | Execution speed | Coding speed | Object-Orient | GUI coding | Dev Environ | Suitability For large tasks | Libraries available |
|---|---|---|---|---|---|---|---|
| Python | Fair | Excellent | Excellent | Good * | Fair | Excellent | Good |
| Perl | Fair | Excellent | Fair | Good * | Fair | Fair | Excellent |
| Vis.Basic | Good | Excellent | Fair | Excellent | Excellent | Poor | Fair |
| C | Excellent | Poor | Poor | n/a | Excellent | Good | Good |
| C++ | Excellent | Fair | Excellent | n/a | Excellent | Excellent | Good |
| Java | Fair | Good | Excellent | Good | Excellent | Excellent | Good |

* Using Tk GUI package

***ตารางที่ 3.1 แสดงการเปรียบเทียบคุณสมบัติของภาษาไพธอนกับภาษาอื่น ๆ***

จากตารางสามารถแสดงคุณสมบัติของภาษาดังนี้

1. **Execution speed** PYTHON มีความเร็วในการทำงานพอๆกับ Perl ซึ่งเป็น Interpreter แต่จะทำงานได้ช้ากว่าภาษา C และ C++ ซึ่งเป็น Compiler
2. **Coding speed** แสดงถึงความสามารถในการเขียนโปรแกรมโดยใช้ภาษานั้นๆ ซึ่ง PYTHON มีความสามารถเท่ากับภาษา Perl ซึ่งแสดงให้เห็นว่าสามารถที่จะเขียนการทำงานของโปรแกรมโดยใช้ code ที่น้อยกว่าภาษาอื่นๆ โดย Visual Basic สามารถเขียนได้เร็วที่สุด
3. **Object Orientation** คือความสามารถของภาษาที่สามารถใช้ Object Orient ในข้อนี้ PYTHON , C++ และ Java สามารถทำได้ดี โดยสามารถออกแบบโปรแกรมเป็น Object ได้ดีกว่าภาษาอื่นๆ
4. **GUI coding** เป็นความสามารถในการนำ GUI มาใช้ในโปรแกรม โดย PYTHON และ Perl สามารถใช้ Tk GUI มาใช้ในการสร้าง GUI ซึ่งสามารถทำออกมาได้ดีในระดับหนึ่ง ซึ่งภาษา Visual Basic สามารถทำได้ดีโดยการใช้ Visual Development Environment ทำให้สามารถสร้าง GUI ได้อย่างรวดเร็ว
5. **Development Environment** คืออุปกรณ์ที่ช่วยในการสร้างโปรแกรม เช่น editor, interactive debugger, code management system รวมถึงส่วนของ Package ต่างๆ ซึ่ง Perl และ PYTHON ยังขาดในส่วนนี้อยู่ อย่างไรก็ตามเราสามารถใช้ IDLE (Python's new integrated development environment) มาช่วยในการเขียนได้
6. **Suitability for large tasks** คือความสามารถในการสนับสนุนใช้งานกับโปรแกรมที่มีขนาดใหญ่และซับซ้อน ซึ่ง PYTHON , C++ และ Java สามารถทำได้ดีเนื่องจากสามารถทำงานโดยการออก

แบบที่ใช้ Object Orientation โดย Perl สามารถทำได้ดีพอสมควร แต่ในการใช้ Visual Basic ซึ่งสามารถสร้าง Interface ได้อย่างรวดเร็ว จะมีปัญหาในการจัดการ Code ในงานขนาดใหญ่

Libraries available เป็นส่วนช่วยในการใช้งานภาษา ซึ่ง Perl มี package จำนวนมากในการนำมาใช้งาน สำหรับภาษาไพธอนนั้นมี Libraries ที่สนับสนุนอยู่พอสมควร

**3.2 ภาษาไพธอนและภาษาจาวา**

ในการเปรียบเทียบคุณสมบัติของภาษาไพธอน เราเลือกทำการเปรียบเทียบกับภาษาจาวา ทั้งสองภาษานี้มีคุณสมบัติที่เหมือนกันในหลาย ๆ ด้าน เช่นเป็นภาษาอินเตอพรีเตอร์ (Interpreter) ที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุทั้งคู่ แต่ภาษาไพธอนมีความแตกต่างจากภาษาจาวาอยู่หลายประการด้วยกันคือ

1. แม้ว่าทั้งภาษาไพธอน และภาษาจาวาจะเป็นภาษาอินเตอพรีเตอร์เหมือนกัน แต่ภาษาจาวาจะเป็นภาษาที่ใช้ในการสร้างโปรแกรมระบบ (System programming language) มากกว่า ความยืดหยุ่นของภาษาจาวาจึงมีน้อยกว่าภาษาไพธอน เนื่องจากการที่โปรแกรมภาษาจาวาต้องทำการจัดสร้างไฟล์คลาสขึ้นมาทุก ๆ ครั้งหลังจากเขียนโปรแกรมเสร็จ จึงทำให้การรีโปรแกรม (Reprogram) ยุ่งยาก หากต้องการทำรีโปรแกรมจะต้องทำการเขียนโปรแกรม และทำการคอมไพล์โดยคำสั่ง JAVAC เพื่อสร้างไฟล์คลาสใหม่ทุก ๆ ครั้ง สำหรับภาษาไพธอนถูกจัดให้เป็นภาษาสคริปต์ (Scripting language) เนื่องจากภาษาไพธอนสามารถรับคำสั่งต่าง ๆ ของผู้ใช้ได้โดยตรง และทำการประมวลผลคำสั่งนั้น ๆ ได้ทันทีโดยไม่ต้องคอมไพล์โปรแกรมใหม่แต่อย่างใด
2. จากที่ได้กล่าวมาแล้วในข้อแรก การรันโปรแกรมที่เป็นภาษาจาวานั้นจะต้องทำการคอมไพล์ก่อนหนึ่งครั้งจึงสามารถรันได้ แต่สำหรับภาษาไพธอนนั้น การรันในระบบปฏิบัติการวินโดวส์สามารถที่จะ double click บนไฟล์โค้ดเพื่อรันได้ทันที ทำให้การรันง่ายกว่าภาษาจาวามาก และเมื่อต้องการทำรีโปรแกรมก็สามารถรันใหม่ได้ทันทีหลังทำการแก้ไขเสร็จเช่นกัน
3. โดยปกติภาษาจาวาจะทำการสร้างไฟล์คลาสออกมาตามจำนวนคลาสที่มีอยู่ในโค้ดของโปรแกรม ซึ่งหากโปรแกรมมีจำนวนคลาสมากแล้ว จะทำให้มีจำนวนไฟล์คลาสมากขึ้นตามไปด้วย ซึ่งอาจส่งผลให้เกิดความยุ่งยากในการจัดการกับไฟล์เหล่านี้ได้ สำหรับภาษาไพธอนที่ไม่มีการคอมไพล์ไฟล์นั้น สามารถจัดเก็บคลาสหลาย ๆ คลาสไว้ในไฟล์ไพธอนเพียงไฟล์เดียวได้ หากผู้เขียนโปรแกรมต้องการแยกส่วนของคลาสต่าง ๆ ออกจากกันเพื่อความเป็นระเบียบ หรือต้องการนำโปรแกรมที่ผู้อื่นเขียนมาใช้ด้วยก็สามารถเรียกใช้ด้วยคำสั่ง IMPORT ไฟล์นั้น ๆ มาได้ ซึ่งโปรแกรมจะรู้จักคลาสทุก ๆ คลาสที่ทำการอิมพอร์ตมา รวมไปถึงแอททริบิวต์ (Attribute) ทุก ๆ ตัวในคลาสต่างๆ ด้วย หากไม่ต้องการใช้การอิมพอร์ต ผู้เขียนสามารถเรียกใช้คำสั่ง EXEC หรือ EXECFILE ได้ ซึ่งจะทำการประมวลผลคำสั่งนั้น ๆ และถ้าโค้ดที่ทำการ EXEC มีการประกาศคลาสต่าง ๆ ในโค้ดนั้น คลาสเหล่านั้นจะสามารถเรียกใช้ได้ทันทีหลังจากทำการ EXEC เช่นเดียวกับการอิมพอร์ต รายละเอียดของคำสั่งเหล่านี้จะกล่าวถึงในบทต่อไป

4. ภาษาไพธอนมีคุณสมบัติ Dynamic typing คือการกำหนดชนิดของข้อมูลให้กับตัวแปรต่าง ๆ โดยอัตโนมัติ ผู้ใช้สามารถกำหนดชนิดของข้อมูลชนิดใด ๆ ให้กับตัวแปรก็ได้ โดยไม่ต้องสนใจว่าตัวแปรนั้นจะเคยเก็บค่าข้อมูลชนิดใดมาก่อน ชนิดของข้อมูลในภาษาไพธอนจะถูกระบุไว้เป็นข้อมูลชนิดเดียวคือ ไพธอนอ๊อบเจ็ค (Python Object) ไม่ว่าจะเป็นตัวเลข, สายอักขระ หรือแม้แต่ซ๊อกเก็ตในการติดต่อสื่อสาร ก็สามารถเก็บลงในตัวแปรได้โดยผู้เขียนโปรแกรมไม่ต้องทำการแยกประเภทของตัวแปรแต่อย่างใด ทำให้การเขียนโปรแกรมด้วยภาษาไพธอนมีความง่าย และยืดหยุ่น แต่อย่างไรก็ตามผู้เขียนทราบว่าตัวแปรใดเก็บข้อมูลอะไรอยู่ เพื่อป้องกันการใช้งานตัวแปรผิดพลาด นอกจากนี้ภาษาไพธอนยังมีความสามารถในการจัดการกับหน่วยความจำโดยอัตโนมัติ เช่นเดียวกับภาษาจาวา
5. คำสั่งต่าง ๆ ในภาษาไพธอนง่ายต่อการใช้งาน และมีความสามารถสูง โดยเฉพาะคำสั่งที่ใช้ในการจัดการกับข้อมูลชนิดสายอักขระ ซึ่งจากการพัฒนาโปรแกรมตัวอย่างในโครงงานนี้ ได้ใช้ภาษาไพธอนร่วมกับภาษาจาวา ภาษาไพธอนสามารถเขียนคำสั่งเพื่อตัดอักษรบางตัวออกจากสายอักขระได้ภายใน 4 คำสั่ง โดยไม่ต้องใช้คำสั่งลูปเลย ในขณะที่การเขียนภาษาจาวาเพื่อใช้ในการทำงานเดียวกันต้องใช้ถึง 10 คำสั่งในการจัดการ และมีการใช้ for ลูปในคำสั่งที่เขียนด้วย ทั้งนี้ผู้เขียนโปรแกรมอาจไม่ได้ใช้ความสามารถของภาษาจาวาอย่างเต็มที่นัก อีกตัวอย่างที่แสดงถึงความสามารถของคำสั่งของภาษาไพธอนคือ for ลูปของภาษาไพธอนซึ่งมีความแตกต่างจากภาษาจาวา สำหรับภาษาจาวานั้นจะมีตัวแปรภายในลูปเป็นตัวแปรที่เก็บค่าตัวเลขที่จะเพิ่มขึ้นหรือลดลงเมื่อทำคำสั่งในลูปนั้นไปหนึ่งรอบ แต่สำหรับภาษาไพธอน ตัวแปรภายในลูปจะเก็บค่าของตัวแปรที่นำมาพิจารณา เช่น

```
list1 = ['aaa', 'bbb', 'ccc']
for i in list1:
        print i
```

จากโค้ดภาษาไพธอนข้างต้นผลลัพธ์ที่ได้จะเป็น

```
aaa
bbb
ccc
```

จะเห็นว่าค่าที่เก็บในตัวแปรของลูปจะเป็นค่าของตัวแปรไม่ใช่ค่าของตัวเลข สำหรับภาษาจาวาถ้าต้องการให้ได้ผลลัพธ์ออกมาเหมือนกับโค้ดข้างต้นต้องกำหนดตัวแปรในลูปให้เป็นค่าตัวเลขซึ่งเป็นการข้อกำหนดของภาษา จากนั้นจึงระบุหาค่าตัวแปรโดยใช้ค่าตัวเลขนั้นเป็นดัชนีอีกทีหนึ่ง วิธีการที่ได้มานั้นยุ่งยากกว่าภาษาไพธอนมาก
6. การเขียนโปรแกรมเพื่อสร้างส่วนติดต่อกับผู้ใช้แบบกราฟฟิก (Graphic User Interfaces) โดยใช้ภาษาไพธอนนั้น ยังทำได้ยาก และภาษาไพธอนไม่สามารถเขียนโปรแกรมที่มีลักษณะการทำงานคล้าย ๆ กับจาวาแอพเพล็ต (Java Applet) ซึ่งทำงานบนเวปเพจได้ การพัฒนาส่วนติดต่อกับผู้ใช้ในภาษาไพธอนต้องอาศัยตัวช่วยอื่น ๆ โดยโปรแกรมในโครงงานนี้ได้เลือกใช้ Tk ซึ่งมีไลบรารีสนับสนุนอยู่ในไพธอนเอง แต่เอกสารประกอบต่าง ๆ ในการใช้งานไลบรารี Tk นี้ยังมีอยู่น้อย ทาง

เลือกอื่น ๆ ก็สามารถทำได้เช่น การใช้งานร่วมกับภาษา Visual Basic ซึ่งเป็นภาษาที่มีความสามารถในการสร้างส่วนติดต่อกับผู้ใช้ในระบบปฏิบัติการวินโดวส์ การติดต่อกับโปรแกรม Visual Basic จะทำผ่าน COM ของระบบปฏิบัติการวินโดวส์ นอกจากนี้ยังสามารถใช้ภาษาไจธอน (Jython) ซึ่งเป็นภาษาไพธอนประเภทหนึ่งที่การแปรคำสั่งในภาษาไพธอนโดยใช้ JVM เป็นตัวประมวลผลโปรแกรมไพธอนที่พัฒนาขึ้นโดยใช้ภาษาไจธอนนี้สามารถที่จะเรียกใช้ไลบรารีได้ทั้งของภาษาไพธอน และของภาษาจาวา นั่นหมายความว่าผู้เขียนสามารถเรียกใช้งานไลบรารีแอปเพรต หรือสวิงค์ของภาษาจาวาในการพัฒนาส่วนติดต่อกับผู้ใช้ได้

7. ภาษาจาวาจะใช้เครื่องหมาย ';' ในการคั่นเพื่อแยกคำสั่งออกจากกัน แต่สำหรับภาษาไพธอนมีการกำหนดให้นำอักษรจำพวกไวท์สเปส (White space) มาใช้ในการแบ่งคำสั่งต่าง ๆ หากเป็นภาษาจาวาแล้วอักษรที่เป็นไวท์สเปสโดยมากจะถูกตัดทิ้ง อักษรไวท์สเปสได้แก่ เว้นวรรค (' '), การขึ้นบรรทัดใหม่ ('\n'), การย่อหน้า ('\t') ฯลฯ การแยกคำสั่งแต่ละคำสั่งของภาษาไพธอนนั้นจะใช้การขึ้นบรรทัดใหม่ หมายความว่าในภาษาไพธอนนี้หนึ่งบรรทัดจะมีคำสั่งอยู่หนึ่งคำสั่ง และการแบ่งว่าคำสั่งในอยู่ในระดับความลึกใดก็ทำได้โดยการใช้ย่อหน้า หรือการเว้นวรรคเข้าไปประมาณสี่ช่อง ตัวอย่างเช่น

```
class a:
        class b:
                y = 100
        x = 500
```

จากโค้ดตัวอย่างข้างต้นจะเห็นว่ามีคลาสอยู่ด้วยกันสองคลาสโดยที่คลาส b อยู่ในคลาส a เนื่องจากก่อนที่จะเขียนคลาส b นั้นได้มีการย่อหน้าเข้าไปก่อนจึงทำให้คลาส b อยู่ในระดับที่ลึกกว่าคลาส a และในคลาส b มีตัวแปร y อยู่ ส่วนคลาส a มีตัวแปร x อยู่ โดยสามารถสังเกตว่าตัวแปรใดเป็นของคลาสใดได้จากจำนวนย่อหน้าก่อนหน้าคำสั่งนั้น ๆ การใช้วิธีการแบ่งคำสั่งอย่างนี้ทำให้ผู้เขียนโปรแกรมสามารถอ่านโค้ดเข้าใจได้ง่าย เพราะเป็นวิธีการเดียวกันกับที่นักเขียนโปรแกรมใช้ในการเขียนโปรแกรมทั่วไป

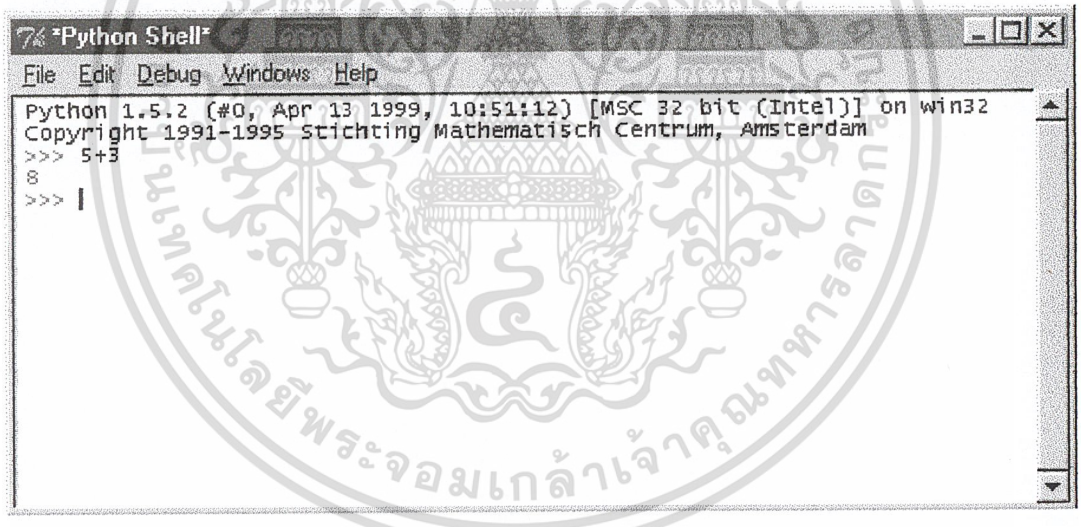
8. การพัฒนาโปรแกรมด้วยภาษาไพธอนนั้นสามารถทำได้อย่างรวดเร็วเนื่องจากมีอุปกรณ์ที่ช่วยในการทดสอบโปรแกรมมากับตัวโปรแกรมภาษาไพธอน อุปกรณ์ดังกล่าวคือ โปรแกรม IDLE ซึ่งเป็นส่วนติดต่อกับผู้ใช้ที่ให้ผู้ใช้สามารถพิมพ์คำสั่งเข้าไปทีละคำสั่ง จากนั้นโปรแกรมจะทำการประมวลผลคำสั่งนั้น และแสดงผลลัพธ์ออกมา ในการสร้างโปรแกรมต่าง ๆ ในโครงงานนี้ได้ใช้งานโปรแกรม IDLE เป็นตัวช่วยในการทดลองรันโปรแกรม โดยการคัดลอกโปรแกรมที่ต้องการทดสอบมาไว้ที่โปรแกรม IDLE จากนั้นให้โปรแกรม IDLE รันโค้ดนั้นและแสดงผลลัพธ์ เราสามารถทดสอบส่วนย่อย ๆ ของโปรแกรมว่ามีความถูกต้องหรือไม่ และยังสามารถสร้างข้อมูลเพื่อป้อนเข้าไปในส่วนของโปรแกรมที่ทำการทดลองได้ด้วย ทำให้ทราบว่าโปรแกรมที่สร้างขึ้นมานี้มีความถูกต้องมากน้อยเพียงใดได้ในเวลาที่รวดเร็ว ในขณะที่ภาษาจาวาเมื่อมีการแก้ไขโปรแกรมก็

ต้องทำการคอมไพล์ใหม่ และแม้จะใช้โปรแกรมเพื่อช่วยเขียนภาษาจาวา การกำหนดค่าอินพุตให้กับโปรแกรมภาษาจาวาก็ก่อให้เกิดความยุ่งยากกับผู้เขียนมากกว่าการเขียนภาษาไพธอน

จากการเปรียบเทียบคุณสมบัติของภาษาไพธอน และภาษาจาวาอาจกล่าวได้ว่า ภาษาไพธอนมีความโดดเด่นกว่าในด้านของความเร็วในการพัฒนาโปรแกรม เนื่องจากคุณสมบัติต่าง ๆ เช่น Dynamic typing และความสามารถของตัวคำสั่งแต่ละคำสั่ง ทั้งนี้ขึ้นกับความถนัดของผู้เขียนด้วย ในด้านการเขียนส่วนติดต่อกับผู้ใช้นั้น ภาษาไพธอนอาจทำได้ยากกว่ามากเนื่องจากตัวภาษาไม่มีการสนับสนุนมากนักต้องใช้ภาษาอื่นมาเป็นตัวช่วยในการพัฒนา อย่างไรก็ดีภาษาไพธอนถือเป็นทางเลือกหนึ่งในการพัฒนาโปรแกรมประเภทโปรโตไทป์ (Prototype) เพื่อทดลองใช้งาน เมื่อปรับเปลี่ยนโปรโตไทป์จนเป็นที่พอใจแล้วจึงทำการเขียนโปรแกรมใหม่เป็นภาษาที่ต้องการเช่น จาวา เป็นต้น

### 3.3 การใช้งานภาษาไพธอน

ภาษาไพธอนที่โครงงานนี้เลือกใช้คือภาษาไพธอนเวอร์ชัน 1.5.2 ในชุดของโปรแกรมจะประกอบไปด้วย คู่มือการใช้งานเป็นเอกสาร HTML, ไลบรารีต่าง ๆ และ IDLE ซึ่งเป็นหน้าจอโต้ตอบสำหรับผู้ใช้ ในการใช้งานภาษาไพธอนผู้ใช้สามารถใส่คำสั่งลงไปใน IDLE ได้เลย ซึ่งผลลัพธ์ของคำสั่งจะแสดงออกมาทันที ดังรูป




*รูปที่ 3.1 การใช้งานโปรแกรม IDLE*

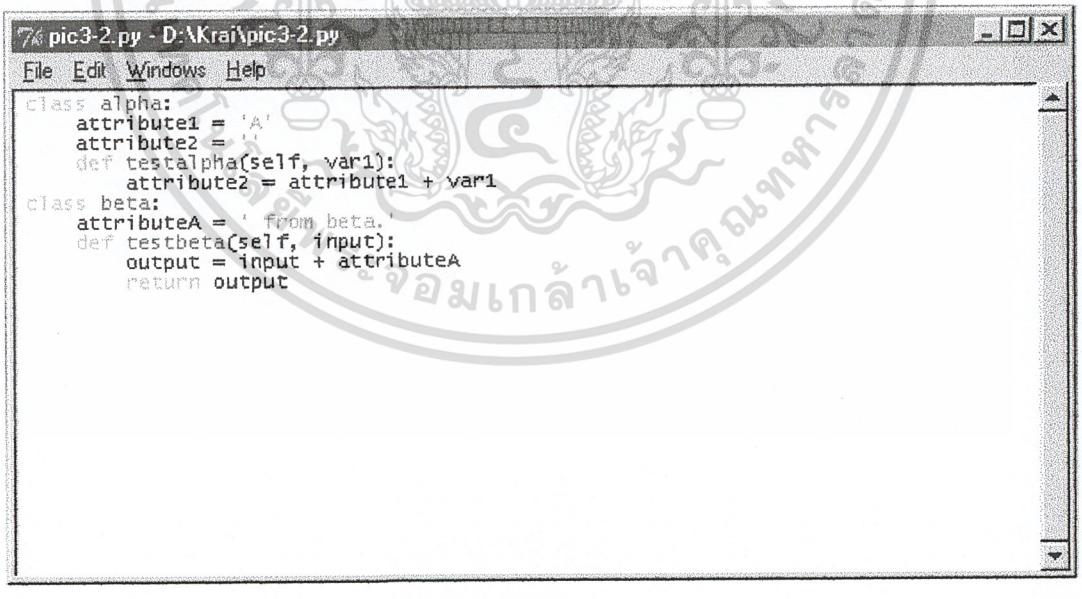
จากรูปเป็นการป้อนคำสั่งให้กับตัวแปรภาษาไพธอน คำสั่งดังกล่าวคือ 5 + 3 ตัวแปรภาษาจะทำการประมวลผลตามคำสั่งที่ได้มา และแจ้งผลลัพธ์กลับมาในบรรทัดถัดไป ในการป้อนคำสั่งให้กับโปรแกรม IDLE นั้นจะทำหลังเครื่องหมาย '>>>' เสมอ ยกเว้นเมื่อมีการป้อนโค้ดเป็นลูป, เป็นกระบวนการ, หรือเป็นคลาสเท่านั้น การเขียนคำสั่งหากต้องการเขียนเป็นโปรแกรมใหญ่ ๆ การใช้งาน IDLE ในลักษณะนี้จะไม่สะดวกนัก ผู้ใช้สามารถเลือกโปรแกรมจัดการเอกสารทั่วไปมาเขียนโปรแกรมลงไปและบันทึกเป็นไฟล์ .PY, .PYW หรือ .TXT ก็ได้ สำหรับโปรแกรม IDLE มีส่วนที่เป็นโปรแกรมจัดการเอกสารมาให้ด้วย สามารถเรียกใช้ได้โดยการกด CTRL + N เพื่อเรียกหน้าต่างขึ้นมาใหม่ และพิมพ์ชุดคำ

สั่งที่ต้องการลงไป จากนั้นทำการบันทึก และสามารถเรียกรันโปรแกรมที่สร้างขึ้นดังกล่าวได้ทันที หากบันทึกเป็น .PY หรือ .PYW แล้วสามารถเรียกรันได้โดยการกด Double click ที่ไฟล์ได้ทันที ข้อแตกต่างของไฟล์สองชนิดนี้คือ ไฟล์นามสกุล .PY จะมีหน้าต่างของ DOS ขึ้นมาให้ด้วย ส่วนไฟล์นามสกุล .PYW จะไม่มี ดังนั้นไฟล์ PY จึงเหมาะที่จะใช้เป็นนามสกุลสำหรับโปรแกรมที่กำลังพัฒนาอยู่ เนื่องจากสามารถดูผลได้อย่างใกล้ชิด เมื่อพัฒนาเสร็จจึงบันทึกเป็นไฟล์นามสกุล .PYW ข้อดีของการใช้ IDLE ในการพัฒนาโปรแกรมภาษาไพธอนคือ มีการตรวจสอบรูปแบบการเขียนโปรแกรมให้ด้วยเหมือนการใช้ทูลส์ (tools) ในการช่วยเขียนโปรแกรมทั่วไป คุณสมบัติดังกล่าวทำให้สามารถตรวจสอบ และแก้ไขโปรแกรมที่เขียนขึ้น และป้องกันการเขียนโปรแกรมผิดพลาดได้

ในการพัฒนาโปรแกรมต่าง ๆ ด้วยภาษาไพธอนนั้น IDLE จะมีประโยชน์อย่างมากในการตรวจสอบความถูกต้องของโปรแกรม โดยเราสามารถเรียกใช้คำสั่ง EXECFILE บนโปรแกรม IDLE เพื่อทำการประมวลผลชุดคำสั่งที่เราได้ทำการเขียนไว้ในไฟล์ โปรแกรมดังกล่าวจะถูกรันบนโปรแกรม IDLE เพื่อให้ผู้เขียนโปรแกรมตรวจสอบความถูกต้องของโปรแกรมนั้นได้ และยังสามารถแก้ไขข้อผิดพลาดได้ทันที นอกจากนี้การที่โปรแกรม IDLE สามารถรับคำสั่งต่าง ๆ จากผู้ได้โดยตรงทำให้ผู้ใช้สามารถกำหนดค่าของข้อมูลอินพุตได้เพื่อใช้ในการตรวจสอบโปรแกรม

### 3.4 โครงสร้างของภาษาไพธอน

ภาษาไพธอนเป็นภาษาที่มีคุณสมบัติเชิงวัตถุ สามารถเขียนโค้ดเพื่อสร้างคลาสต่าง ๆ ได้ หากต้องการสร้างคลาส 2 คลาส จะมีโค้ดดังต่อไปนี้

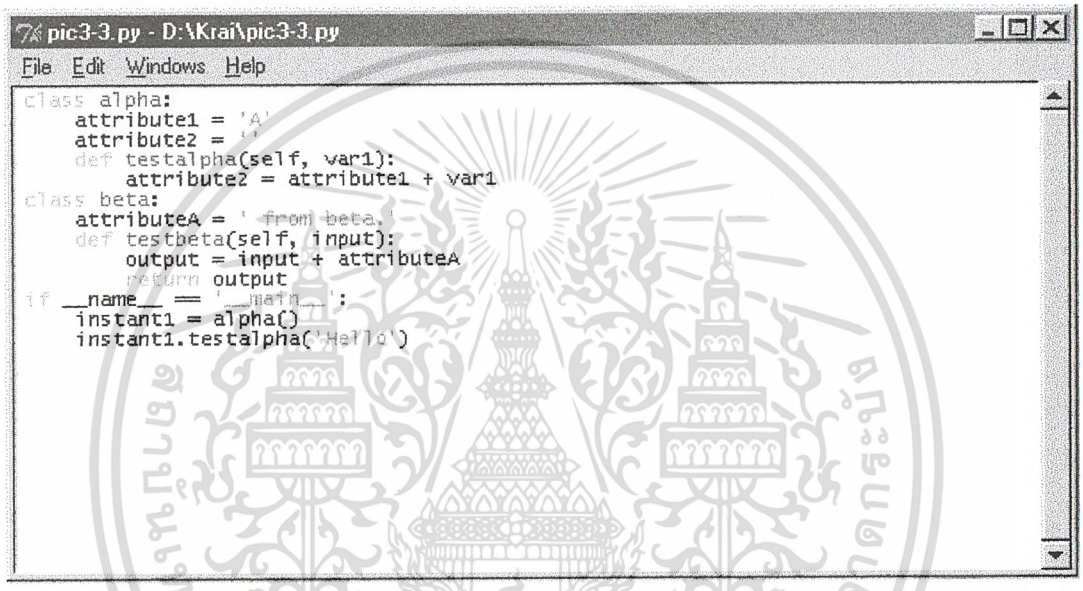



***รูปที่ 3.2 ตัวอย่างการเขียนคลาส 2 คลาสในภาษาไพธอน***

โค้ดข้างต้นเป็นการสร้างคลาส 2 คลาสคือ คลาส alpha และคลาส beta โดยในคลาส alpha นั้นมีแอททริบิวต์คือ attribute1 และ attribute2 และมีกระบวนการทำงานหนึ่งกระบวนการคือ testalpha กระบวนการดังกล่าวมีการรับค่าตัวแปร var1 เข้ามาด้วย ส่วนในคลาส beta นั้นมีแอททริบิวต์คือ

attributeA มีกระบวนการทำงานหนึ่งกระบวนการเช่นกันคือ testbeta ที่รับค่าตัวแปร input มา และส่งค่ากลับด้วยตัวแปร output ซึ่งเป็นการรวม attributeA และ input เข้าด้วยกัน การส่งค่ากลับนั้นใช้คำสั่ง return ตามด้วยค่าที่ต้องการให้ส่งกลับ โค้ดข้างต้นนี้ไม่มีการเริ่มต้นเรียกใช้งานคลาสใด ๆ ดังนั้นหากเขียนโค้ดเช่นนี้โปรแกรมจะทำการสร้างคลาสขึ้นมาเพียงอย่างเดียวไม่ทำการรันและแสดงผลใด ๆ จำเป็นต้องมีการเพิ่มโค้ดเพื่อให้โปรแกรมเริ่มต้นการทำงาน โค้ดที่เพิ่มอาจเป็นดังเช่นด้านล่าง

```
if __name__ == '__main__':
    instant1 = alpha()
    instant1.testalpha('Hello')
```



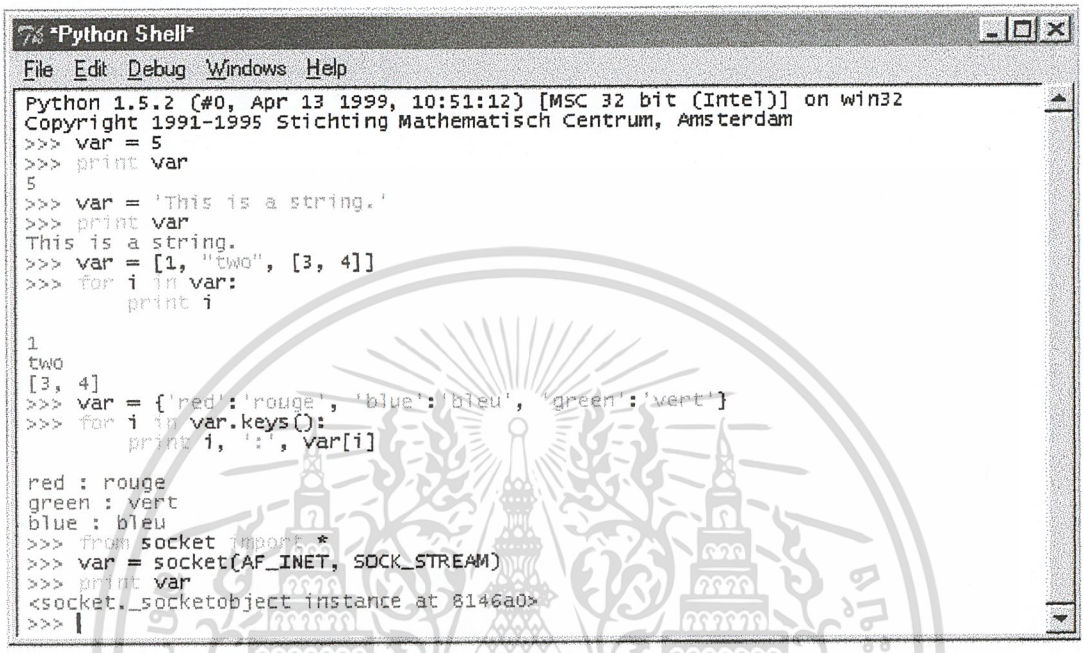

*รูปที่ 3.3 ตัวอย่างการเขียนคลาสที่มีการเรียกใช้งาน*

โค้ดข้างต้นนี้เป็นการเรียกใช้งานกระบวนการ testalpha ในคลาส alpha ให้ทำงาน วิธีการเรียกกระบวนการในคลาสให้ทำงานนั้นจำเป็นต้องสร้างอินสแตนส์ (Instant) ของคลาสนั้นขึ้นมาก่อน ในภาษาไพธอนตัวอินสแตนส์จะเป็นตัวแปรตัวหนึ่งซึ่งเก็บค่าที่อ้างอิงกับคลาสนั้น อินสแตนส์นี้สามารถอ้างอิงกระบวนการในคลาสทั้งหมด รวมไปถึงแอททริบิวต์ต่าง ๆ ของคลาสนั้นด้วย จากโค้ดข้างต้นมีการสร้างอินสแตนส์ที่มีชื่อว่า instant1 ซึ่งเป็นอิสแตนส์ของคลาส alpha เราสามารถอ้างอิงถึงกระบวนการ และแอททริบิวต์ได้โดยใช้จุดเชื่อมชื่ออินสแตนส์เข้ากับชื่อกระบวนการหรือชื่อแอททริบิวต์ตัวอย่างเช่น instant1.testalpha('Hello') หรือ instant1.attribute1 เป็นต้น

การเขียนโปรแกรมโดยภาษาไพธอนนั้น การขึ้นบรรทัดใหม่, การเว้นวรรค, การย่อหน้า หรือเว้นบรรทัดจะมีผลโดยตรงต่อโปรแกรมที่เขียน คำสั่งที่เขียนขึ้นอาจอยู่นอกคลาสที่มันควรจะอยู่ หากมีการย่อหน้าผิดไป ซึ่งผู้เขียนโปรแกรมจำเป็นจะต้องให้ความใส่ใจกับข้อกำหนดนี้ให้มาก

### 3.5 ชนิดของข้อมูล

ชนิดของข้อมูลในภาษาไพธอนจะอยู่ในอ๊อบเจ็คที่มีชื่อว่า ไพธอนอ๊อบเจ็ค การกำหนดชนิดของข้อมูลให้กับตัวแปรในภาษาไพธอนจะถูกกระทำโดยอัตโนมัติ ผู้เขียนสามารถเปลี่ยนแปลงตัวแปรหนึ่งให้เก็บค่าที่ต่างไปได้โดยไม่ต้องสนใจชนิดของตัวแปรที่เก็บอยู่เดิม ตัวอย่างเช่น




*รูปที่ 3.4 คุณสมบัติ Dynamic Typing ของภาษาไพธอน*

จากรูปตัวแปร var1 จะถูกเปลี่ยนให้เก็บค่าข้อมูลชนิดต่าง ๆ และมีข้อมูลชนิด socket อยู่ด้วยซึ่งก็ถือเป็นหนึ่งในชนิดของข้อมูลของไพธอนอ๊อบเจ็คเช่นกัน ชนิดของข้อมูลทั่วไปได้แก่

#### 3.5.1 ตัวเลข (Numeric)

ข้อมูลชนิดตัวเลข ได้แก่

- เลขฐาน 10 (Integer) คือเลขที่ประกอบไปด้วยเลข 0-9 ไม่ว่าจะเป็นจำนวนบวก หรือจำนวนลบ แต่ต้องไม่มีจุดทศนิยม ตัวอย่างเช่น 1, -3, 444586 ฯลฯ
- เลขฐาน 8 คือ ตัวเลขที่ขึ้นต้นด้วย 0 เช่น 0717 ฯลฯ
- เลขฐาน 16 คือ ตัวเลขที่ขึ้นต้นด้วย 0x เช่น 0xBF12, 0x1E ฯลฯ
- เลขทศนิยม (Floats) เช่น 3.2, -31e12 ฯลฯ
- จำนวนเต็มขนาดยาว (Long Integer) ได้แก่ เลขที่ลงท้ายด้วย 'L' หรือ 'l' เช่น 999999999L, 4l ฯลฯ
- จำนวนเชิงซ้อน (Complex Number) ได้แก่ 3j, 5 + 4j, 3.2 + 2.1j ฯลฯ

**3.5.2 จำนวนทางตรรกะ (Boolean)**

โดยปกติจำนวนทางตรรกะนั้นจะมีค่าคือจริง และเท็จ สำหรับภาษาไพธอนนั้นแทนค่าจริงด้วยเลข 1 และแทนค่าเท็จด้วย 0

**3.5.3 สายอักขระ (String)**

ข้อมูลชนิดสายอักขระ คือ ชุดของตัวอักษรเช่น 'This is a string.', "This is also a string." สายอักขระนี้จะครอบด้วยสัญลักษณ์ Single quoted หรือ Double quoted ก็ได้ แต่ต้องเหมือนกันทั้งหัว และท้าย หากใช้ Single quoted ในสายอักขระจะสามารถมี Double quoted ได้ และในทำนองเดียวกันหากใช้ Double quoted ก็สามารถมี Single quoted ในสายอักขระได้เช่นกัน

ในข้อมูลชนิดสายอักขระนั้นจะมีอักษรพิเศษอยู่ด้วยได้แก่

| | | |
|---|---|---|
| \n = Newline | \' = Single quoted | \b = Backspace |
| \t = Tap | \" = Double quoted | \f = Formfeed |
| \\ = Backslash | \a = Bell | \r = Carriage return |
| \v = Vertical tap | | |

สำหรับการใช้ "'" หรือ "" นั้นจะเป็นสายอักขระที่ หากมีการขึ้นบรรทัดใหม่ในสายอักขระหรือมีการย่อหน้าจะทำการเปลี่ยนให้เป็นตัวอักษรพิเศษ นอกจากนี้ยังใช้เป็น Comment อีกด้วย สำหรับ Comment นั้นจะใช้ '#' เป็นตัวบอก ข้อความที่อยู่หลัง '#' ไปจนสุดบรรทัดจะไม่นำมาทำการประมวลผลโดยภาษาไพธอน

**3.5.4 ลิสต์ (Lists)**

ข้อมูลชนิดลิสต์นี้จะเป็นการนำเอาข้อมูลชนิดอื่น ๆ มาเก็บไว้ในข้อมูลชนิดนี้ โดยจะมีลักษณะต่าง ๆ คล้ายกับข้อมูลชนิดอาเรย์ (Array) แต่จะต่างกันตรงที่ข้อมูลที่เก็บนี้ไม่จำเป็นต้องเก็บข้อมูลชนิดเดียวกันทุก ๆ สมาชิก ข้อมูลที่สามารถเก็บในลิสต์ได้แก่ ตัวเลข, ตัวอักษร, สายอักขระ หรือแม้แต่จะเป็นลิสต์เองก็ได้ การเขียนอ้างอิงถึงข้อมูลชนิดลิสต์ทำได้ดังนี้

| | |
|---|---|
| list1 = [] | เป็นการสร้างลิสต์เปล่า |
| list2 = [1, "two", [3, 4]] | เป็นการสร้างลิสต์ที่ประกอบไปด้วยสมาชิก 3 ตัวโดยตัวแรกเป็นตัวเลข, ตัวที่สองเป็นสายอักขระ และตัวที่สามเป็นลิสต์ที่มีสองสมาชิก |

การอ้างอิงถึงสมาชิกในลิสต์ทำได้ดังนี้

| | |
|---|---|
| list2[0] | จะได้ผลลัพธ์เป็น 1 |
| list2[1:2] | จะได้ผลลัพธ์เป็น "two" |
| list2[1:] | จะได้ผลลัพธ์เป็น ["two", [3, 4]] |
| list2[-1] | จะได้ผลลัพธ์เป็น [3,4] |
| list2[:-1] | จะได้ผลลัพธ์เป็น [1, "two"] |

### 3.5.5 ทัปเปิล (Tuples)

ข้อมูลชนิดทัปเปิลมีลักษณะเหมือนข้อมูลชนิดลิสต์ แต่จะต่างกันตรงที่ไม่สามารถแก้ไขข้อมูลที่เป็นสมาชิกได้ ทัปเปิลจึงเหมาะแก่การใช้เป็นข้อมูลอ้างอิงต่าง ๆ การกำหนดทัปเปิลมีวิธีการดังนี้

tuple1 = () เป็นการสร้างทัปเปิลเปล่า

tuple1 = ('one', 'two', 'three', 'four')

ส่วนการจัดการกับทัปเปิลนั้นมีวิธีการเช่นเดียวกันกับการจัดการลิสต์

### 3.5.6 ดิกชันนารี (Dictionary)

ข้อมูลชนิดดิกชันนารีจะประกอบไปด้วยคีย์ (Keys) และค่าที่เก็บ (Values) ในการอ้างอิงข้อมูลประเภทดิกชันนารีนั้นทำได้ดังนี้

| | |
|---|---|
| dic[key] | จะให้ผลลัพธ์เป็นอ๊อบเจ็คที่แสดงค่าที่เก็บโดยคีย์นั้น |
| dic.has_key(key) | จะส่งค่ากลับเป็น 1 เมื่อมีคีย์ที่ระบุใน dic และเป็น 0 เมื่อไม่มี |
| dic.keys() | ส่งค่ากลับเป็นลิสต์ของคีย์ต่าง ๆ ใน dic |
| dic.values() | ส่งค่ากลับเป็นลิสต์ของค่าที่เก็บต่าง ๆ ใน dic |
| dic.clear() | ทำการลบทุก ๆ ค่าที่เก็บใน dic |

โดยปกติแล้วข้อมูลชนิดลิสต์จะสามารถเรียงลำดับข้อมูลได้ แต่ข้อมูลชนิดดิกชันนารีนั้นจะไม่มีการเรียงลำดับข้อมูล การอ้างอิงถึงทำได้โดยผ่านทางคีย์เท่านั้น

### 3.5.7 ข้อมูลเปล่า (None)

ข้อมูลชนิดนี้เป็นข้อมูลที่ใช้ระบุค่าเริ่มต้นของตัวแปร หรือแสดงว่าไม่มีข้อมูล ซึ่งในการเปรียบเทียบค่านั้น None จะมีค่าเท่ากับ None เท่านั้น

## 3.6 ตัวกระทำ (Operators)

ตัวกระทำต่าง ๆ แบ่งออกเป็น 4 ประเภทคือ ตัวกระทำทางตรรกะ, ตัวกระทำทางการเปรียบเทียบ, ตัวกระทำทางบิตไวส์ และตัวกระทำทางคณิตศาสตร์

### 3.6.1 ตัวกระทำทางตรรกะ (Logical operators)

ตัวแปรทางตรรกะมีทั้งหมด 3 ตัวด้วยกันคือ and, or, not โดยทั้ง 3 ตัวกระทำนี้สามารถเรียกใช้งานได้ด้วยการเขียน and, or และ not โดยตรง

### 3.6.2 ตัวกระทำทางการเปรียบเทียบ (Comparisons)

ตัวกระทำนี้ใช้ในการเปรียบเทียบค่าต่าง ๆ มีดังต่อไปนี้

| | | | |
|---|---|---|---|
| < | น้อยกว่า | > | มากกว่า |
| <= | น้อยกว่าหรือเท่ากับ | >= | มากกว่าหรือเท่ากับ |
| == | เท่ากับ | <>, != | ไม่เท่ากับ (สามารถใช้ได้ทั้งสองแบบ) |

| | |
|---|---|
| in | x in y หมายความว่ามีค่า x เป็นสมาชิกใน y หรือไม่ |
| not in | แสดงผลตรงกันข้ามกับ in |
| is | x is y หมายความว่า x เป็นสิ่งเดียวกับ y ซึ่งไม่เหมือนกับ == |
| is not | แสดงผลตรงข้ามกับ is |

**3.6.3 ตัวกระทำทางบิตไวส์ (Bitwise operators)**

ตัวกระทำนี้ใช้ในการจัดการกับเลขฐานสอง โดยมีรายละเอียดดังต่อไปนี้

| | |
|---|---|
| << | integer << n ส่งค่ากลับเป็นตัวเลขที่เกิดจากการเลื่อน integer ไปทางซ้าย (Shift left) ไปเป็นจำนวน n ตัว |
| >> | ทำงานเช่นเดียวกับ << แต่เป็นการเลื่อนไปทางขวา (Shift right) |
| & | m & n ผลลัพธ์คือ การ and กันในรูปตัวเลขฐานสองของ m และ n |
| \| | m \| n ผลลัพธ์คือ การ or กันในรูปตัวเลขฐานสองของ m และ n |
| ^ | m ^ n ผลลัพธ์คือ การ exclusive-or กันในรูปเลขฐานสองของ m และ n |
| ~ | ~m ผลลัพธ์คือ การ not ของรูปเลขฐานสอง m |

**3.6.4 ตัวกระทำทางคณิตศาสตร์ (Arithmetic-Style operators)**

ตัวกระทำทางคณิตศาสตร์นี้สามารถใช้งานไม่เพียงกับตัวเลขเท่านั้น แต่ยังสามารถใช้งานกับข้อมูลชนิดอื่น ๆ เช่นสายอักขระได้ ตัวกระทำดังกล่าวได้แก่

| | |
|---|---|
| + | หากใช้กับตัวเลขจะเป็นการบวกกัน แต่หากใช้กับสายอักขระจะเป็นการนำมาเรียงต่อกัน และหากเป็นลิสต์ หรือทัปเปิลจะเป็นรวมสองลิสต์ หรือสองทัปเปิลเข้าด้วยกัน |
| - | ใช้กับตัวเลขเพื่อทำการลบกัน |
| * | หากใช้กับตัวเลขจะเป็นการคูณกัน แต่หากใช้กับสายอักขระ, ลิสต์ หรือทัปเปิลจะเป็นการเพิ่มจำนวนของข้อมูลนั้น เช่น 'A' * 5 จะได้ 'AAAAA' เป็นต้น |
| / | ใช้กับตัวเลขเพื่อทำการหาร โดยหากใช้กับจำนวนเต็มเท่านั้น ผลที่ได้ก็จะเป็นจำนวนเต็มด้วย เศษที่ได้จะถูกปัดลง ตัวกระทำนี้เหมือนกับ div ในภาษา Pascal |
| ** | ใช้กับตัวเลขเท่านั้น เป็นการยกกำลัง |
| % | หากใช้กับตัวเลขจะเป็นการหาเศษจากการหาร เหมือนการใช้คำสั่ง mod ในภาษา Pascal นั่นเอง หากใช้กับสายอักขระ จะเป็นการบอกรูปแบบในการพิมพ์ |

**3.6.5 ลำดับของตัวกระทำ (Precedence)**

ลำดับของตัวกระทำมีลำดับดังนี้

or

and

not

<, <=, ==, >=, >, !=, <>, is, in, not, not in

|

^

&

<<, >>

+, -

*, /, %

**

unary +, unary -, unary ~

**3.7 คำสงวน (Reserved words)**

คำสงวนในภาษาไพธอนแบ่งเป็น คีย์เวิร์ด(Keywords) และบิวท์อินฟังก์ชัน(Built-in function)

คีย์เวิร์ดได้แก่

'and', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while'

บิวท์อินฟังก์ชันได้แก่

'__import__', 'abs', 'apply', 'buffer', 'callable', 'chr', 'cmp', 'coerce', 'compile', 'complex', 'delattr', 'dir', 'divmod', 'eval', 'execfile', 'filter', 'float', 'getattr', 'globals', 'hasattr', 'hash', 'hex', 'id', 'input', 'int', 'intern', 'isins', 'issubclass', 'len', 'list', 'locals', 'long', 'map', 'max', 'min', 'oct', 'open', 'ord', 'pow', 'range', 'raw_input', 'reduce', 'reload', 'repr', 'round', 'setattr', 'slide', 'str', 'tuple', 'type', 'vars', 'xrange'

**3.8 คำสั่งลูป (Control flow)**

คำสั่งลูปเป็นคำสั่งที่ใช้ในการควบคุมลำดับการประมวลผลของคำสั่งต่าง ๆ โดยให้ประมวลผลวนรอบไปเรื่อย ๆ จนกว่าจะถึงเงื่อนไขที่กำหนด คำสั่งลูปต่าง ๆ ได้แก่ คำสั่ง IF, WHILE และ FOR

**3.8.1 คำสั่ง IF**

รูปแบบของคำสั่ง FOR มีดังนี้

if <EXPRESSION>:

    <STATEMENT>

elif <EXPRESSION>:

    <STATEMENT>

else:

    <STATEMENT>

โดย <EXPRESSION> นั้นหมายถึงเงื่อนไขในการหยุดลูป ส่วน <STATEMENT> คือคำสั่งที่อยู่ในลูป ตัวอย่างของการใช้คำสั่ง IF เช่น

```
if x >= 10:
        print 'More than 10'
elif x < 0:
        print 'Negative number'
else:
        print 'Answer accept'
```

**3.8.2 คำสั่ง WHILE**

คำสั่ง WHILE นั้นมีการตรวจสอบเงื่อนไขก่อนการทำคำสั่งในลูปทุกครั้ง โดยมีรูปแบบคือ

```
while <EXPRESSION>:
        <STATEMENT>
```

ตัวอย่างคำสั่ง WHILE เช่น

```
while 1:
        print 'This is an endless loop.'
```

**3.8.3 คำสั่ง FOR**

คำสั่ง FOR เป็นคำสั่งที่จะวนรอบการทำงานจนครบจำนวนที่กำหนด รูปแบบของคำสั่งคือ

```
for <VARIBLE> in <RANGE>:
    <STATEMENT>
```

โดย <RANGE> คือตัวแปรที่เป็นชนิดลิสต์ หรือคำสั่งเฉพาะบางคำสั่ง ส่วน <VARIBLE> คือตัวแปรที่เก็บค่าของสมาชิกทุกตัวของ <RANGE> การทำงานของคำสั่ง FOR จะทำงานวนรอบไปเป็นจำนวนเท่ากับจำนวนสมาชิกของ <RANGE> ตัวอย่างเช่น

```
list2 = [1, "two", [3, 4]]
for var1 in list2:
        print var1
```

จากโค้ดข้างต้นจะได้ผลลัพธ์เป็นดังนี้

```
1
two
[3, 4]
```

การทำงานของคำสั่ง FOR จากโค้ดข้างต้นจะทำงานเป็นจำนวน 3 ครั้ง เท่ากับจำนวนสมาชิกใน list2

```
for i in range(5):
        print i
```

คำสั่ง range(n) จะส่งค่ากลับเป็นตัวเลขตั้งแต่ 0 จนถึง n – 1 ผลลัพธ์ที่ได้จากโค้ดข้างต้นคือ

0

1

2

3

4

หากต้องการใช้คำสั่ง FOR ในภาษาไพธอนให้เหมือนกับคำสั่ง FOR ในภาษาอื่น ๆ เช่น Pascal วิธีหนึ่งที่จะเขียนได้คือ

```
from string import *
for i in range(len(list2)):
        list2[i] = upper(list2[i])
```

**3.9 mod_python**

เป็น Module ของภาษาไพธอนที่ใช้ในการติดตั้งในเว็บเซิร์ฟเวอร์ เช่น Apache เพื่อทำให้เว็บเซิร์ฟเวอร์สามารถทำการรัน CGI สคริปต์ที่เขียนด้วยภาษาไพธอนได้ โดยการทำสคริปต์ที่เขียนด้วยภาษาไพธอนสามารถใช้งาน Apache handler ได้ ซึ่งการใช้งาน mod_python จะมีประสิทธิภาพมากกว่าการใช้ module CGI ที่มีมากับภาษาไพธอน

เราสามารถทำการเขียนสคริปต์ที่มีการใช้งาน mod_python ได้โดยการเขียนโค้ดต่อไปนี้ไว้ในสคริปต์ที่จะทำการใช้งานเป็น CGI

```
from mod_python import apache
```

เมื่อเราทำการ import apache จาก module mod_python แล้วเราสามารถใช้งาน handler ของ Apache ได้ 2 วิธีคือ

1. Standard handler

   เป็น handler ปกติที่มีการใช้งาน โดยเมื่อเราทำการปรับแต่งให้ Apache สามารถใช้งาน Standard handler ได้(วิธีการปรับแต่งอยู่ในภาคผนวก)เราสามารถใช้งานได้โดยเขียนโปรแกรมดังนี้

   ```
   from mod_python import apache
   def handler(req):
       req.content_type = "text/plain"
       req.send_http_header()
       req.write("Hello World!")
       return apache.OK
   ```

   บรรทัดแรกเป็นการ import mod_python เข้ามาใช้งาน ใช้คำว่า req เพื่อใช้งาน handler เช่น req.content_type = "text/plain" คือกำหนด content_type ของข้อมูลที่ส่งเป็น Text ธรรมดา

req.send_http_header() เป็นการส่ง header ให้กับบราวเซอร์ req.write("Hello World!") เป็นการส่งข้อมูล Text "Hello World!" ไปให้กับบราวเซอร์ และ return apache.OK เป็นการบอกเว็บเซิร์ฟเวอร์ว่าการทำงานถูกต้อง แต่การใช้งานแบบนี้เราสามารถเรียกใช้งานได้ไฟล์เดียวในหนึ่งไดเรคทอรี ทำให้ไม่สะดวกในการใช้งาน

2. Public handler

เป็น handler ที่สะดวกต่อการใช้งานมากกว่าแบบแรกโดยสามารถเรียกใช้งานได้ทุกเมธอดในไฟล์ที่กำหนด (วิธีการปรับแต่ง Apache อยู่ในภาคผนวก) เราสามารถใช้งานได้โดยเขียนโปรแกรมดังนี้

```
from mod_python import apache
def say_yes(req, what="Yes"):
    return "I am saying %s" % what
def say_no(req,what="No"):
    return "I am saying %s" % what
```

ซึ่งถ้าเราทำการเก็บโค้ดนี้ไว้ในไฟล์ hello.py เราสามารถเรียกใช้งานได้ทั้ง 2 เมธอดคือ hello.py/say_yes และ hello.py/say_no โดยเราสามารถใช้งาน request handler ได้เช่นเดียวกับ Standard handler แต่เราต้องทำการส่งค่าที่เกิดจากการทำงานไปยังบราวเซอร์โดยคำสั่ง Return ในโครงงานนี้มีการส่งค่ากลับเป็นข้อมูลแฮสทีเอ็มแอล จึงต้องมีการกำหนดค่าชนิดข้อมูลและส่ง header ไปยัง บราวเซอร์ โดยการเขียนโปรแกรมดังนี้

```
req.content_type = "text/html"
req.send_http_header()
```

# บทที่ 4
# หลักการวิวัฒนาการของโปรแกรมไพธอน

การวิวัฒนาการของโปรแกรมคือการที่โปรแกรมมีความสามารถในการพัฒนาตนเองให้มีความสามารถตอบสนองต่อสิ่งแวดล้อม ซึ่งโดยมากคือการตอบสนองต่อความต้องการของผู้ใช้เป็นสำคัญ การที่โปรแกรมจะพัฒนาตนเองได้นั้น โปรแกรมจำเป็นจะต้องเรียนรู้สิ่งต่าง ๆ ได้ เพื่อจะใช้ในการพัฒนาตนเอง การเรียนรู้นี้อาจเป็นการเรียนรู้จากผู้ใช้งาน, โปรแกรมอื่น ๆ หรือข้อมูลที่เก็บอยู่ที่ใดที่หนึ่งก็ได้ หลักการวิวัฒนาการของโปรแกรมจึงเป็นหลักการสร้างโปรแกรม หรือกลุ่มของโปรแกรมที่พัฒนาตนเองได้ด้วยการเรียนรู้จากผู้ใช้ หรือจากโปรแกรมอื่น ๆ นั่นเอง โปรแกรมหรือกลุ่มของโปรแกรมที่สร้างภายใต้หลักการวิวัฒนาการของโปรแกรมนี้จะสามารถตอบสนองความต้องการของผู้ใช้งานได้มากขึ้นกว่าโปรแกรมธรรมดา

การใช้หลักการวิวัฒนาการของโปรแกรมนั้นจำเป็นต้องใช้ภาษาที่มีความยืดหยุ่นสูง สามารถแก้ไขโค้ดของโปรแกรมได้ด้วยตัวเอง นั่นคือสามารถทำรีโปรแกรมได้โดยไม่มีข้อจำกัดใด ๆ โปรแกรมภาษาไพธอนจึงถูกเลือกใช้ในการออกแบบโปรแกรมตามหลักการนี้ เนื่องจากมีความเหมาะสมที่สุด โดยอาจกล่าวได้ว่า นอกจากภาษาไพธอนแล้ว การใช้ภาษาอื่นเพื่อทำการสร้างโปรแกรมตามหลักการวิวัฒนาการของโปรแกรมเป็นไปได้ยาก หรืออาจเป็นไปไม่ได้เลย ภาษาไพธอนนั้นมีความสามารถในการจัดการกับสายอักขระสูง โครงงานนี้จึงเลือกที่จะออกแบบโปรแกรมที่มีวิวัฒนาการโดยใช้วิธีการแก้ไขโค้ดของโปรแกรมโดยตรงด้วยตัวโปรแกรมเอง วิธีการนี้ช่วยให้การออกแบบและการทำงานของโปรแกรมง่ายขึ้นมาก

## 4.1 โปรแกรมเชิงวัตถุ

ในการเขียนโปรแกรมตามหลักการวิวัฒนาการนั้นโปรแกรมจำเป็นต้องเขียนในเชิงวัตถุ เนื่องจากหลักการวิวัฒนาการได้ใช้คุณสมบัติของโปรแกรมเชิงวัตถุคือ การสืบทอดคุณสมบัติ (Inheritance) โปรแกรมเชิงวัตถุจะประกอบไปด้วยคลาส ซึ่งเป็นกลไกสำหรับกำหนดโครงสร้างและพฤติกรรมของวัตถุ การจำลองการทำงานของวัตถุทำได้โดยการสร้างอินสแตนส์ (Instance) ของคลาสนั้น อินสแตนส์นี้สามารถอ้างอิงข้อมูล และเรียกใช้กระบวนการของคลาสได้ สำหรับการสร้างคลาส และอินสแตนส์ในภาษาไพธอนนั้นได้กล่าวไปแล้วในบทที่ 3 สำหรับอินสแตนส์ของคลาสหนึ่งนั้นสามารถกำหนดให้มีความแตกต่างกันได้โดยการใช้ Constructor ภาษาไพธอนมีการเขียน Constructor ของคลาสด้วยการสร้างกระบวนการโดยใช้ '__init__' เป็นชื่อกระบวนการ และสามารถรับค่าตัวแปรต่าง ๆ ได้เหมือนการรับค่าตัวแปรของกระบวนการทั่วไป ตัวอย่างคือ

```
def __init__(self, x, y = 0, z = 0):
    result = x + y + z
```

การเขียนโปรแกรมโดยทั่วไปอาจมีความซ้ำซ้อนของโค้ดเกิดขึ้นได้ ทำให้โปรแกรมดูยุ่งและยังเสียเวลาในการเขียนด้วย จึงมีการนำคลาสที่มีอยู่แล้วมาปรับปรุงเปลี่ยนแปลงเพื่อให้เกิดคลาสใหม่ที่เหมาะสม โดยคลาสเดิมก็ยังใช้งานได้เช่นเดิม วิธีการสร้างคลาสดังกล่าวเรียกว่า การสืบทอดคุณสมบัติ การเขียนโค้ดให้คลาสในภาษาไพธอนมีการสืบทอดคุณสมบัติทำได้ดังตัวอย่างคือ

```
class parent_class:
        def method1(self):
                print 'Do method1'
        def method2(self):
                print 'Do method2'
class sub_class(parent_class):
        def method2(self)
                print 'This method is changed'
        def method3(self):
                print 'This is sub_class'
```

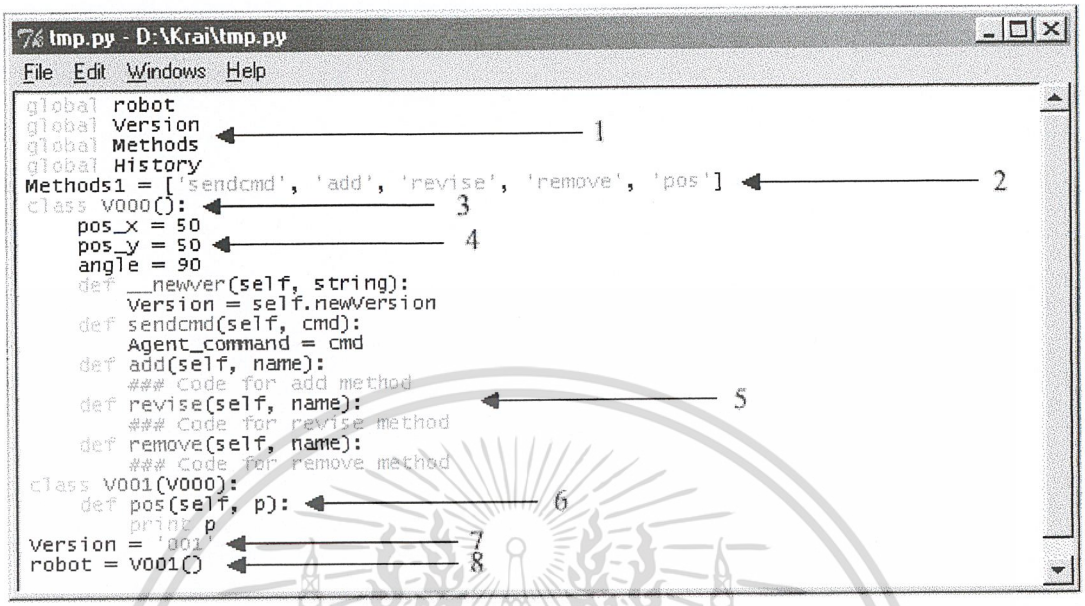
จากตัวอย่างเป็นการสร้างคลาสชื่อ sub_class ให้มีการสืบทอดคุณสมบัติจากคลาส parent_class ทำให้สามารถเรียกใช้กระบวนการต่าง ๆ ใน parent_class ซึ่งก็คือ method1 มีการเปลี่ยนแปลงกระบวนการชื่อ method2 และยังมีการเพิ่มกระบวนการที่ชื่อ method3 อีกด้วย ไม่มีการเปลี่ยนแปลงใด ๆ เกิดขึ้นกับคลาส parent_class สำหรับ Constructor ของคลาสจะประมวลผล Constructor ของทุก ๆ คลาส

**4.2 การทำงานของโปรแกรมที่มีวิวัฒนาการ**

โปรแกรมที่มีวิวัฒนาการนั้นจะต้องมีการเขียนโปรแกรมเป็นคลาส ในคลาสนั้นจะประกอบไปด้วย method หรือกระบวนการทำงานต่าง ๆ การวิวัฒนาการนั้นจะเป็นการเปลี่ยนแปลงกระบวนการทำงานที่มีอยู่ หรือเพิ่มกระบวนการทำงานใหม่ ๆ เข้าไปในโปรแกรม ซึ่งเท่ากับว่าโปรแกรมได้ทำการเรียนรู้ และพัฒนาตนเอง กระบวนการที่มีอยู่ในคลาสจะสามารถเรียกใช้ผ่านอินสแตนส์ของคลาสนั้น การเรียกใช้สามารถทำได้โดยตัวโปรแกรมเอง โดยโปรแกรมจะมีคำสั่งที่เรียกใช้กระบวนการต่าง ๆ ของตัวเองอยู่ นอกจากจะออกแบบให้โปรแกรมเรียกใช้งานกระบวนการของตนเองแล้ว ยังสามารถสร้างโปรแกรมอื่นเพื่อคอยควบคุม และเรียกใช้กระบวนการต่าง ๆ ของโปรแกรมอื่น วิธีการนี้เหมาะสำหรับระบบที่ประกอบไปด้วยโปรแกรมที่มีวิวัฒนาการหลาย ๆ ตัว โปรแกรมที่ใช้ในการควบคุมนี้จะเป็นตัวติดต่อกับผู้ใช้ เมื่อผู้ใช้ต้องการใช้งานโปรแกรม ผู้ใช้จะสั่งงานผ่านโปรแกรมที่คอยควบคุมนี้ ในโครงงานนี้เป็นการใช้งานโปรแกรมที่มีวิวัฒนาการร่วมกันหลายตัว จึงกำหนดให้มีโปรแกรมควบคุมเพื่อเรียกใช้งานกระบวนการทำงานของโปรแกรมที่มีวิวัฒนาการทุก ๆ ตัว และเป็นตัวติดต่อกับผู้ใช้งาน

เพื่อความสะดวกในการสื่อความหมาย ขอเรียกโปรแกรมที่มีวิวัฒนาการว่า เอเย่นต์ โดยเอเย่นต์นี้เป็นคลาสที่ประกอบไปด้วยกระบวนการทำงานต่าง ๆ และเอเย่นต์สามารถที่จะเปลี่ยนแปลง, เพิ่มหรือลดกระบวนการเหล่านี้ได้ตามหลักการวิวัฒนาการ และขอเรียกโปรแกรมที่คอยควบคุมเอเย่นต์ว่า

ซูเปอร์ไวเซอร์ (Supervisor) โปรแกรมนี้มีหน้าที่ควบคุมการทำงานของเอเย่นต์, เรียกใช้กระบวนการทำงานของเอเย่นต์ และติดต่อรับคำสั่งจากผู้ใช้




*รูปที่ 4.1 โครงสร้างตัวอย่างของโปรแกรมที่มีวิวัฒนาการ*

โครงสร้างหลัก ๆ ของเอเย่นต์ได้แก่

ส่วนที่ 1 คือ การประกาศตัวแปรหลักที่ใช้ ในโค้ดตัวอย่างนี้ได้มีการสร้างตัวแปรหลักไว้ในตัวซูเปอร์ไวเซอร์ซึ่งอยู่ในคนละไฟล์กัน เอเย่นต์จึงต้องทำการอ้างอิงถึงตัวแปรเหล่านี้ด้วยคำสั่งดังรูป ตัวแปรที่สำคัญได้แก่

- ตัวแปร robot เป็นตัวแปรที่เก็บค่าอินสแตนส์ของคลาสปัจจุบัน คลาสปัจจุบันในโปรแกรมตัวอย่างคือคลาส V001
- ตัวแปร Version เป็นตัวแปรที่เก็บค่าเวอร์ชันปัจจุบัน
- ตัวแปร Methods เก็บชื่อกระบวนการต่าง ๆ ของเอเย่นต์ที่สามารถเรียกใช้ได้
- ตัวแปร History เก็บประวัติการเปลี่ยนแปลงที่กระทำกับเอเย่นต์

ส่วนที่ 2 คือ ส่วนระบุถึงชื่อกระบวนการที่อยู่ในตัวแปร Methods ส่วนนี้มีไว้เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้จากการปิดโปรแกรม โดยเมื่อเปิดโปรแกรมขึ้นมาอีกครั้งค่าต่าง ๆ ของตัวแปร Methods จะยังคงอยู่เหมือนเดิม

ส่วนที่ 3 คือ คลาสที่สร้างขึ้นมา โค้ดตัวอย่างนี้มีการสร้างคลาสโดยใช้ชื่อตามเวอร์ชันเพื่อความสะดวกในการจัดการ ในคลาสนี้ประกอบด้วยแอททริบิวต์และกระบวนการทำงานต่าง ๆ ของคลาส

ส่วนที่ 4 คือ แอททริบิวต์ของคลาส

ส่วนที่ 5 คือ กระบวนการต่าง ๆ ของคลาสอันได้แก่

- กระบวนการ __newver เป็นกระบวนการภายใน ไม่สามารถเรียกจากภายนอกคลาสนี้ได้ กระบวนการภายในเหล่านี้ระบุโดยใช้อักษร '_' สองตัวนำหน้าชื่อกระบวนการ

- กระบวนการ sendcmd เป็นกระบวนการที่ให้ซูเปอร์ไวเซอร์ส่งคำสั่งมาให้กับเอเย่นต์ อาจไม่มีก็ได้ แต่กระบวนการนี้จะช่วยให้การทำงานร่วมกันของซูเปอร์ไวเซอร์และเอเย่นต์เป็นไปได้ง่ายขึ้น
- กระบวนการ add, revise, remove เป็นกระบวนการมาตรฐานที่เอเย่นต์ทุกตัวต้องมี เพื่อใช้เพิ่ม, เปลี่ยนแปลง และลบกระบวนการตามลำดับ อันที่จริงหลักการวิวัฒนาการยังมีกระบวนการในการ undo อีกด้วยแต่โปรแกรมโครงงานนี้ไม่ได้มีกระบวนการนี้อยู่ด้วย

ส่วนที่ 6 คือ คลาสใหม่ที่เกิดขึ้นจากการเพิ่มกระบวนการ กระบวนการที่เพิ่มขึ้นมาคือ pos คลาสนี้จะเป็นคลาสลูกของคลาสในส่วนที่ 3 และประกอบไปด้วยกระบวนการหนึ่งกระบวนการ คือ กระบวนการ pos

ส่วนที่ 7 คือ ส่วนที่ระบุถึงเวอร์ชันปัจจุบัน ใช้เพื่อป้องกันความผิดพลาดเช่นเดียวกับส่วนที่ 2

ส่วนที่ 8 คือ คำสั่งที่สร้างอินสแตนส์ (ในที่นี้คือตัวแปร robot) ให้อ้างอิงไปยังคลาสปัจจุบัน

ส่วนที่ 7 และ 8 นี้ไม่จำเป็นต้องอยู่ท้ายไฟล์ก็ได้

การทำงานของเอเย่นต์จะแบ่งเป็น 4 การทำงานด้วยกัน คือ

1. การเพิ่มกระบวนการทำงาน (Add Method)
2. การเปลี่ยนแปลงกระบวนการทำงาน (Revise Method)
3. การลบกระบวนการทำงาน (Remove Method)
4. การแก้ไขการเปลี่ยนแปลง (Undo)

**4.2.1 การเพิ่มกระบวนการทำงาน (Adding Method)**

การเพิ่มกระบวนการทำงานเป็นการทำให้เอเย่นต์มีความสามารถในการทำงานเพิ่มขึ้นจากเดิม โดยเมื่อมีความต้องการที่จะเพิ่มกระบวนการทำงานให้กับเอเย่นต์ ไม่ว่าจะเป็นจากผู้ใช้ หรือตัวเอเย่นต์เอง เอเย่นต์จะทำการเรียกใช้กระบวนการมาตรฐานที่มีชื่อว่า add กระบวนการนี้ใช้ในการเพิ่มกระบวนการให้กับเอเย่นต์ หากผู้ใช้เป็นผู้ต้องการเพิ่มกระบวนการผู้ใช้ต้องทำการแจ้งผ่านซูเปอร์ไวเซอร์อีกทีหนึ่ง และซูเปอร์ไวเซอร์จะเป็นผู้เรียกใช้กระบวนการ add ของเอเย่นต์

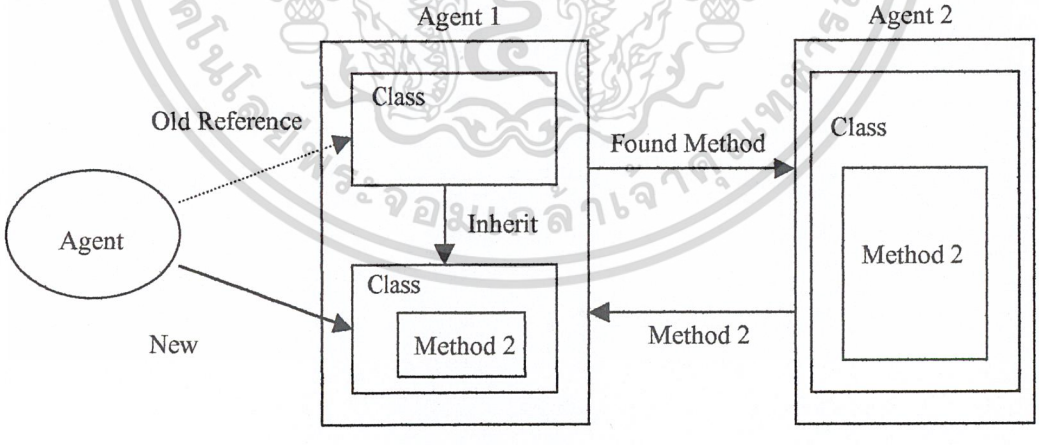
การทำงานของกระบวนการ add มีดังนี้คือ ผู้เรียกใช้กระบวนการจะเรียกใช้โดยแจ้งชื่อกระบวนการที่จะทำการเพิ่มมาด้วย กระบวนการ add จะทำการตรวจสอบชื่อกระบวนการว่ามีอยู่ในตัวแปร Methods หรือไม่ เมื่อไม่พบชื่อกระบวนการดังกล่าวในตัวแปร Methods จึงทำการตรวจสอบตัวแปร Methods ของเอเย่นต์อื่น ๆ ว่ามีกระบวนการที่ต้องการเพิ่มหรือไม่ หากมีก็จะทำการเปิดไฟล์โค้ดของเอเย่นต์ขึ้นมาก่อน เพื่อหาตัวโค้ดของกระบวนการดังกล่าว จากนั้นทำการตัดสายอักขระตั้งแต่คำว่า ‘ def <method_name>’ จนถึงคำว่า ‘ def’ หรือจนกว่าจะจบไฟล์ และทำการคัดลอกโค้ดของกระบวนการดังกล่าวมาใส่ไว้ในคลาสลูกของคลาสปัจจุบันที่มีชื่อเป็นเวอร์ชันถัดจากเวอร์ชันของคลาสพ่อแม่ เปลี่ยนตัวแปรเวอร์ชันให้เป็นเวอร์ชันปัจจุบัน เพิ่มชื่อกระบวนการลงในตัวแปร Methods เก็บข้อมูลการทำคำสั่ง add ไว้ในตัวแปร History ตัวแปรนี้จะเป็นชนิดดิกชันนารีที่มีคีย์คือเลขเวอร์ชัน และค่าที่เก็บคือลิสต์ที่มี 2 สมาชิก สมาชิกตัวแรกระบุถึงกระบวนการที่ทำ สมาชิกตัวที่สองระบุถึงชื่อกระบวน

การทำงานที่เกี่ยวข้อง ในกรณีของการเพิ่มกระบวนการนี้ ตัวแปร History จะเก็บข้อมูลดังเช่น ['add', <method_name>] โดยมีคีย์คือ ค่าเวอร์ชันนั่นเอง ทำการ EXEC คลาสที่สร้างขึ้นมา สุดท้ายเขียนคลาสนี้ลงในไฟล์โค้ดของเอเย่นต์ พร้อมทั้งแก้ค่าของตัวแปร Methods, Version และ History ที่อยู่ในโค้ดให้เป็นปัจจุบัน ทั้งนี้เพื่อให้เกิดความถูกต้องหากมีการหยุดทำงานของเอเย่นต์ เอเย่นต์ยังคงสามารถเก็บค่าในตัวแปรต่าง ๆ ได้อย่างถูกต้องเมื่อมีการรันเอเย่นต์อีกครั้ง

ข้อสังเกตในการเขียนโค้ด เพื่อตรวจสอบหาตำแหน่งของกระบวนการที่ต้องการ เราจะเขียนคำว่า ' def' แยกกันในลักษณะดังนี้ ' d' + 'ef ' เพื่อป้องกันการตรวจสอบผิดพลาด เช่นเมื่อเอเย่นต์ตัวอื่นมาตรวจโค้ดหากระบวนการ หากมีการเขียน def ไว้ติดกันจะทำให้เอเย่นต์ที่มาตรวจสอบเข้าใจว่าเป็นกระบวนการอีกกระบวนการหนึ่ง จึงทำให้เกิดความผิดพลาดขึ้น

ในกรณีที่เอเย่นต์ตรวจสอบไม่พบกระบวนการที่ต้องการในเอเย่นต์ตัวใด ๆ เลย เอเย่นต์จะทำการสอบถามการทำงานของกระบวนการที่จะเพิ่มจากผู้ใช้โดยตรง โค้ดที่ใส่นั้นต้องเป็นชุดคำสั่งที่เอเย่นต์รู้จัก เช่นคำสั่งต่าง ๆ ในภาษาไพธอน หรือเป็นกระบวนการทำงานอื่น ๆ ที่มีอยู่ในตัวเอเย่นต์ก่อนแล้ว

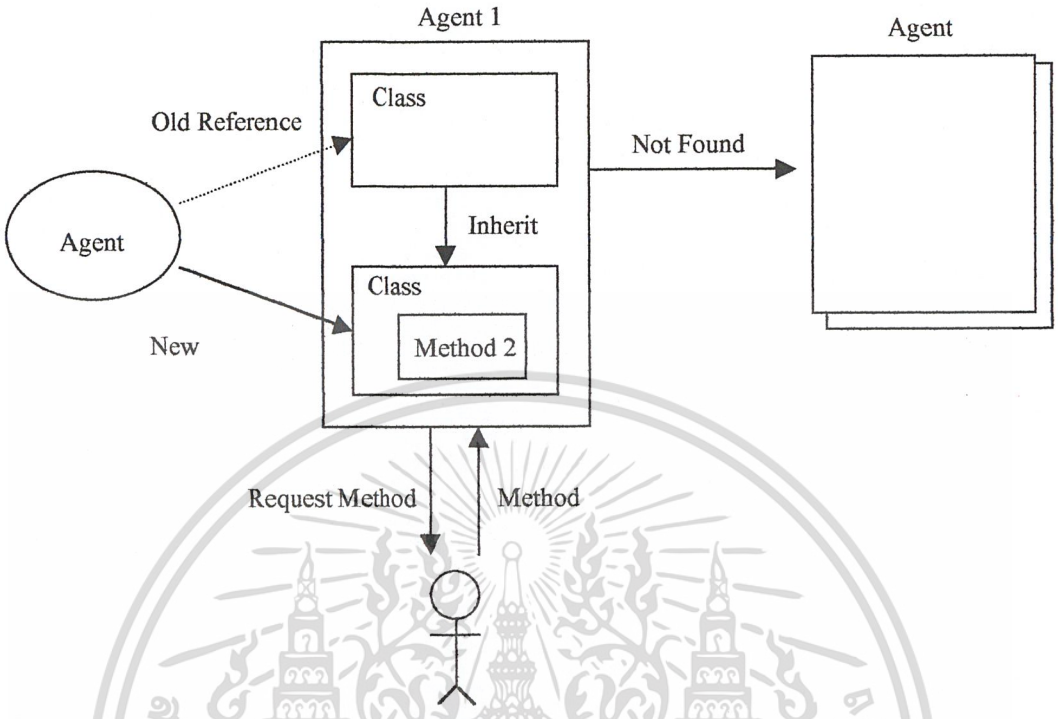
หลังจากเรียกใช้กระบวนการ add ของเอเย่นต์แล้ว ผลที่ได้คือ เอเย่นต์จะรู้จักคลาสใหม่ที่เพิ่มขึ้นมา และเนื่องจากในคลาสนั้นมีกระบวนการที่ต้องการเพิ่มอยู่ด้วย เอเย่นต์จึงรู้จักกระบวนการนั้นด้วย แต่ในขณะนี้เอเย่นต์เพียงแค่รู้จักกระบวนการเท่านั้น ไม่สามารถเรียกใช้กระบวนการดังกล่าวได้ หากต้องการเรียกใช้กระบวนการนั้น จำเป็นต้องสร้างอินสแตนส์ของคลาสใหม่ขึ้นมาก่อน อินสแตนส์ที่สร้างขึ้นมานี้จะเป็นตัวกลางเพื่อเรียกใช้กระบวนการต่าง ๆ ของคลาสที่อินสแตนส์นั้นอ้างอิงถึง ตลอดจนกระบวนการต่าง ๆ ที่มีในคลาสพ่อแม่ของคลาสนั้นด้วย ดังนั้นเราอาจกล่าวได้ว่า เอเย่นต์ก็คืออินสแตนส์ของคลาสนั่นเอง




*รูปที่ 4.2 การเพิ่มกระบวนการทำงานจากเอเย่นต์อื่น*

การทำงานของการเพิ่มกระบวนการมีดังรูปที่ 4.2 โดยเริ่มแรกนั้นอินสแตนส์ของเอเย่นต์จะอ้างอิงไปยังคลาสเก่า (ตามเส้นประ) เมื่อทำการเพิ่มกระบวนการเอเย่นต์จะไปหากระบวนการจากเอเย่นต์

อื่น เมื่อพบกระบวนการที่ต้องการแล้ว เอเย่นต์จะคัดลอกกระบวนการดังกล่าวมาไว้ในคลาสใหม่ที่เป็นคลาสลูกของคลาสเดิม สุดท้ายจึงสร้างให้อินสแตนส์ไปอ้างอิงกับคลาสใหม่




*รูปที่ 4.3 การเพิ่มกระบวนการทำงานจากผู้ใช้*

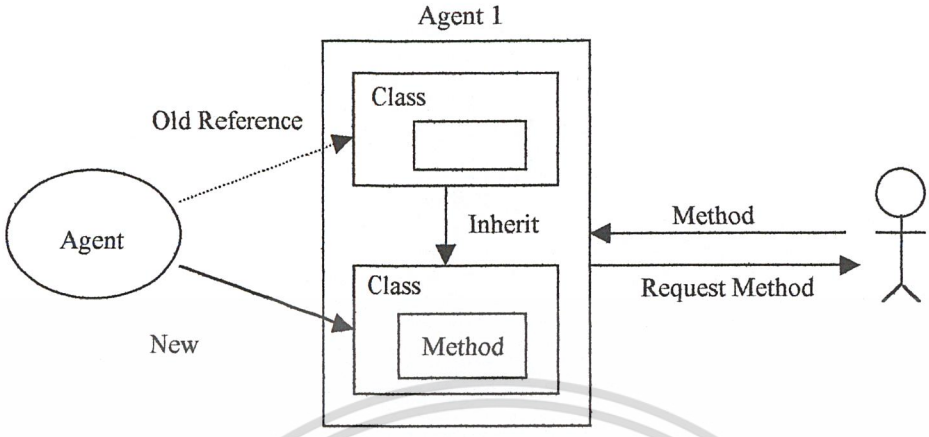
การทำงานของการเพิ่มกระบวนการในรูปที่ 4.3 นั้น คล้ายกับการทำงานในรูปที่ 4.2 ต่างกันที่ในรูป 4.3 นี้เอเย่นต์ไม่พบกระบวนการที่ต้องการในเอเย่นต์ตัวอื่น ๆ เลย หลังจากนั้นเอเย่นต์จึงทำการสอบถามการทำงานของกระบวนการที่จะเพิ่มจากผู้ใช้ เมื่อได้กระบวนการที่ต้องการแล้ว จะนำกระบวนการดังกล่าวไปใส่ในคลาสลูกของคลาสปัจจุบัน และทำการสร้างอินสแตนส์ให้อ้างอิงถึงคลาสดังกล่าว

**4.2.2 การเปลี่ยนแปลงกระบวนการทำงาน (Revising Method)**

การเปลี่ยนแปลงกระบวนการทำงาน คือ การเปลี่ยนแปลงการทำงานของกระบวนการใด ๆ ในเอเย่นต์ เช่นเดียวกับการเพิ่มกระบวนการทำงาน การเรียกใช้การเปลี่ยนแปลงกระบวนทำงานนี้ สามารถเรียกใช้โดยผู้ใช้หรือตัวเอเย่นต์เองก็ได้ และในการเรียกใช้จะต้องแจ้งชื่อของกระบวนการที่ต้องการเปลี่ยนแปลงด้วย กระบวนการที่เรียกใช้ คือ กระบวนการมาตรฐานที่ชื่อว่า revise

กระบวนการ revise นั้นเริ่มต้นเมื่อมีการเรียกใช้ เอเย่นต์จะทำการตรวจสอบตัวแปร Methods เพื่อดูว่ามีกระบวนการที่ต้องการเปลี่ยนแปลงอยู่ในตัวหรือไม่ หากมีอยู่จึงจะทำงานต่อไป โดยจะรับโค้ดการทำงานใหม่มาจากผู้ใช้ มาเก็บไว้ในคลาสลูกของคลาสปัจจุบันที่มีชื่อเป็นเวอร์ชั่นถัดจากเวอร์ชันของคลาสแม่ เพิ่มค่าที่เก็บในตัวแปร Version บันทึกการทำงานลงในตัวแปร History โดยค่าที่เก็บในตัวแปรนี้จะมีลักษณะคือ ['revise', <method_name>] จากนั้นทำการ EXEC คลาสที่สร้างขึ้น และทำการเขียนโค้ดของคลาสลงต่อไว้ในไฟล์โค้ดของเอเย่นต์ รวมถึงทำการแก้ไขค่าที่ถูกเขียนเพื่อใช้ในการเก็บ

Methods, Version และ History สุดท้ายทำการสร้างอินสแตนส์ของคลาสใหม่เพื่อให้เรียกใช้กระบวนการที่ทำการเปลี่ยนแปลงได้




*รูปที่ 4.4 การเปลี่ยนแปลงกระบวนการทำงาน*

การเปลี่ยนแปลงกระบวนการทำงานนั้นจะมีการทำงานคล้ายกับการเพิ่มกระบวนการ ต่างกันที่การเปลี่ยนแปลงกระบวนการจะทำงานเมื่อมีกระบวนการที่ต้องการเปลี่ยนแปลงในตัวแปร Methods แต่การเพิ่มกระบวนการทำงานจะตรวจสอบว่ายังไม่มีกระบวนการดังกล่าวอยู่ในตัวแปร Methods จึงจะเริ่มทำงาน นอกจากนี้การเปลี่ยนแปลงกระบวนการทำงานจะไม่มีการค้นหากระบวนการจากเอเย่นต์ตัวอื่น แต่จะทำการถามการทำงานจากผู้ใช้เลย

**4.2.3 การลบกระบวนการทำงาน (Removing Method)**

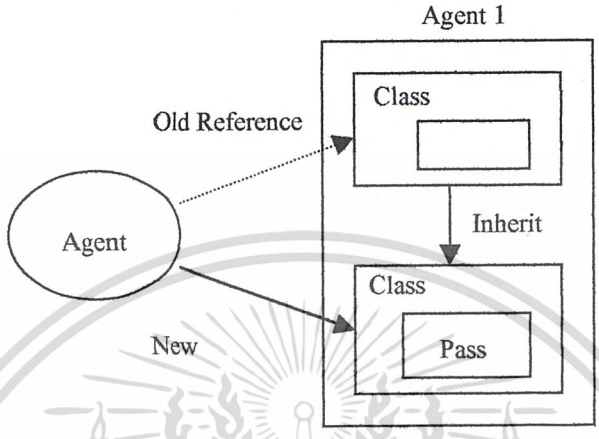
การลบกระบวนการทำงานนั้นคือการทำให้ผู้ใช้ไม่สามารถเรียกใช้กระบวนการนั้นได้อีก โดยมีหลักการคือ การทำให้กระบวนการดังกล่าวไม่มีวิธีการทำงาน โดยใช้คำสั่ง pass เช่น

```
class V005(V004):
        def mydef(self):
                pass
```

จากโค้ดข้างต้นจะเป็นการลบกระบวนการที่มีชื่อว่า mydef การลบด้วยวิธีดังกล่าวผู้ใช้ยังสามารถเรียกใช้กระบวนการนี้ได้อยู่ แต่กระบวนการดังกล่าวจะไม่ทำงานใด ๆ สำหรับการเขียนโปรแกรมจริงนั้นเอเย่นต์ต้องคอยตรวจสอบการเรียกใช้กระบวนการต่าง ๆ อยู่แล้ว โดยตรวจดูในตัวแปร Methods หากไม่มีกระบวนการนั้นในตัวแปรก็แปลว่าเอเย่นต์ไม่รู้จักกระบวนการดังกล่าว และจะไปค้นหาจากเอเย่นต์ตัวอื่น ๆ ต่อไป นั่นแปลว่า หากเราลบชื่อกระบวนการออกจากตัวแปร Methods แล้วผู้ใช้จะไม่สามารถเรียกใช้กระบวนการนี้ได้อีก ยกเว้นว่าผู้ใช้สามารถเรียกใช้กระบวนการทำงานของเอเย่นต์ได้โดยตรง

วิธีการในการลบกระบวนการ เริ่มจากมีการเรียกใช้กระบวนการมาตรฐานที่ชื่อว่า remove พร้อมกับแจ้งชื่อกระบวนการที่ต้องการลบออก เอเย่นต์ทำการตรวจสอบว่ามีกระบวนการนั้นอยู่หรือไม่ โดยผ่านตัวแปร Methods หากมีจะทำการสร้างคลาสที่เป็นคลาสลูกของคลาสปัจจุบันโดยบรรจุ

กระบวนการที่มีชื่อเดียวกับกระบวนการที่ต้องการลบออก แต่มีการทำงานเป็นคำสั่ง pass เพียงอย่างเดียว จากนั้นลบชื่อกระบวนการออกจากตัวแปร Methods เพิ่มค่าของตัวแปร Version และบันทึกการลบลงในตัวแปร History ซึ่งจะเก็บข้อมูล ['remove', <method_name>] จากนั้น EXEC คลาสที่สร้างขึ้นบันทึกโค้ดของคลาส พร้อมทั้งเปลี่ยนแปลงค่าของตัวแปร Methods, Version และ History ในโค้ดให้ตรงกับค่าปัจจุบัน สุดท้ายสร้างอินสแตนส์ให้อ้างอิงไปยังคลาสใหม่ที่สร้างขึ้น

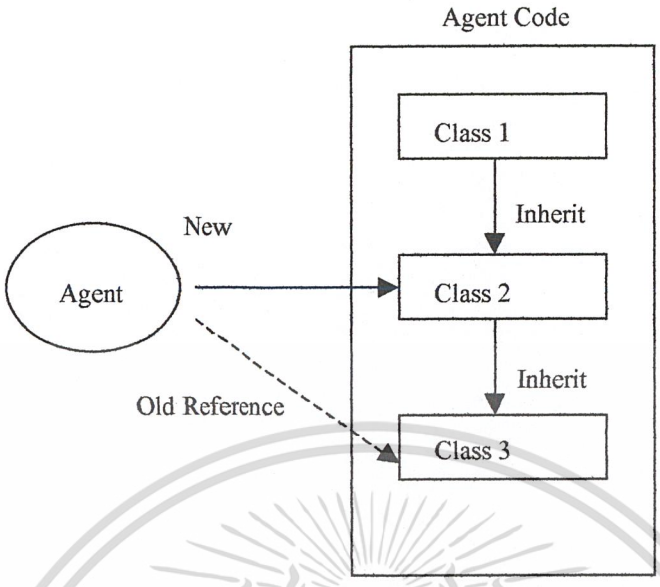



*รูปที่ 4.5 การลบกระบวนการทำงาน*

**4.2.4 การแก้ไขการเปลี่ยนแปลง (Undo)**

การแก้ไขการเปลี่ยนแปลงเป็นการทำให้การเปลี่ยนแปลงที่เคยทำไปแล้วในอดีตไม่เป็นผล ซึ่งต้องอาศัยตัวแปร History เป็นตัวอ้างอิง เช่น ต้องการแก้ไขการลบกระบวนการที่ชื่อ mydef เอเย่นต์จะทำการหาว่าได้ทำการลบกระบวนการ mydef ไปเมื่อเวอร์ชันใด ผ่านตัวแปร History จากจุดนี้จะเห็นว่าหากไม่ทำการบันทึกการเปลี่ยนแปลงทีละอย่างจะไม่สามารถใช้การแก้ไขการเปลี่ยนแปลงได้ ดังนั้นหากมีการเปลี่ยนแปลงใด ๆ หนึ่งอย่าง ต้องสร้างคลาสใหม่เพื่อรองรับการเปลี่ยนแปลงนั้นหนึ่งคลาสเสมอ เมื่อเอเย่นต์ทราบว่ามีการลบที่เวอร์ชันใดแล้ว จะสร้างอินสแตนส์ใหม่ให้อ้างอิงไปยังคลาสที่เป็นเวอร์ชันก่อนหน้าเวอร์ชันนั้น ผลก็คือ กระบวนการที่ถูกลบไปนั้นจะกลับมาใช้ได้ดังเดิม แต่กระบวนการใดก็ตามที่ผู้ใช้ทำการเพิ่มหรือเปลี่ยนแปลงไปหลังจากลบกระบวนการนั้นไปแล้วจะหายไป

อย่างไรก็ดีโปรแกรมที่พัฒนาขึ้นในโครงงานนี้ไม่ได้มีคุณสมบัติในการแก้ไขการเปลี่ยนแปลงอยู่ด้วย เนื่องจากการมีคุณสมบัตินี้ทำให้การจัดการกับโปรแกรมมีความยุ่งยากมาก และต้องใช้เวลานานในการพัฒนาเนื่องจากการเปลี่ยนคลาสอ้างอิงของอินสแตนส์มาก ๆ ทำให้สับสนได้ว่าอินสแตนส์ปัจจุบันเป็นอินสแตนส์ของคลาสใด ในส่วนนี้จึงเขียนไว้ในรายงานเพื่อให้ผู้ที่สนใจได้นำไปศึกษาและทำการพัฒนากระบวนการดังกล่าวต่อไป




*รูปที่ 4.6 การแก้ไขการเปลี่ยนแปลง*

**4.3 คำสั่งภาษาไพธอนที่ใช้ในการออกแบบ**

คำสั่งที่นำมาใช้ในการเขียนโปรแกรมแบบมีวิวัฒนาการ เป็นคำสั่งที่เกี่ยวกับการเรียกโค้ดที่เขียนไว้มาประมวลผลเป็นสำคัญ ซึ่งแบ่งออกเป็น 2 วิธีการ คือ

1. การใช้คำสั่ง IMPORT
2. การใช้คำสั่งจำพวก EXEC

**4.3.1 คำสั่ง IMPORT**

คำสั่งนี้เป็นคำสั่งที่ใช้ในการเรียกโมดูล (Module) ในภาษาไพธอนมาประมวลผล การใช้งานจะเหมือนกับคำสั่ง import ในภาษาจาวา ตัวอย่างของการใช้คำสั่งอิมพอร์ต ได้แก่

```
import string
from string import *
```

ทั้งสองคำสั่งจะมีผลแตกต่างกันคือ หากใช้ import string (เป็นการเรียกโมดูลที่ใช้ในการจัดการกับสายอักขระ) การใช้คำสั่งของโมดูล string จะต้องทำโดยอ้างชื่อโมดูลก่อน เช่น string.find(str, tmp) แต่ถ้าเป็นการอิมพอร์ตแบบที่สอง คือ from string import * การใช้คำสั่งในโมดูลจะใช้ find(str, tmp) เลย การเขียนโมดูลในภาษาไพธอนจะเขียนเป็นคลาสเช่นเดียวกับการเขียนโปรแกรมทั่วไปของภาษาไพธอน หลังการอิมพอร์ตโปรแกรมจะรู้จักคลาสทุกคลาสในโมดูลที่ทำการอิมพอร์ต ตลอดจนกระบวนการที่อยู่ในคลาสเหล่านั้นด้วย (เฉพาะกระบวนการที่สามารถเรียกใช้ได้จากภายนอกเท่านั้น)

**4.3.2 คำสั่งจำพวก EXEC**

คำสั่งจำพวก EXEC นี้ประกอบไปด้วย exec, execfile และ eval

- คำสั่ง exec คือ คำสั่งที่ใช้ในการประมวลผลโค้ดในรูปของสายอักขระ มีตัวอย่างการใช้งานคำสั่งดังเช่น

```
code = 'f = open('tmp.py','r')\ndata = f.read()\nf.close()\nprint data'
exec code
```

จากโค้ดตัวอย่างจะได้ผลลัพธ์เป็นการพิมพ์ข้อมูลในไฟล์ tmp.py

- คำสั่ง execfile คือ คำสั่งที่ใช้ในการประมวลผลโค้ดในรูปของไฟล์ ตัวอย่างเช่น

```
execfile('tmp.py')
```

จากโค้ดจะได้ผลลัพธ์เช่นเดียวกับการรันไฟล์ tmp.py

- คำสั่ง eval คือ คำสั่งที่ใช้ในการประมวลผลโค้ดประเภท Expression ตัวอย่างเช่น

```
n = eval('A'*5)
```

จากโค้ดจะให้ผลลัพธ์เช่นเดียวกับคำสั่ง

```
n = 'AAAAA'
```

ความแตกต่างของคำสั่ง IMPORT และคำสั่งจำพวก EXEC คือ คำสั่งอิมพอร์ตไม่ได้กระทำโค้ดนั้นจริง ๆ เพียงแค่เรียกมาเก็บไว้ในหน่วยความจำเพื่อให้สามารถอ้างอิงได้เท่านั้น แต่การใช้คำสั่งจำพวก EXEC นั้นจะเป็นการทำงานคำสั่งนั้น ๆ เช่น หากมีคำสั่ง print อยู่ในโค้ดด้วย คำสั่งอิมพอร์ตจะไม่ทำการพิมพ์ให้ แต่คำสั่งจำพวก EXEC จะทำการพิมพ์ออกมาด้วย

**4.4 ข้อดีและข้อจำกัดของหลักการวิวัฒนาการของโปรแกรม**

ข้อดีของหลักการวิวัฒนาการของโปรแกรมคือ

1. โปรแกรมที่ออกแบบตามหลักการวิวัฒนาการจะสามารถเรียนรู้ และปรับปรุงตนเองได้ ซึ่งมีส่วนช่วยอำนวยความสะดวกแก่ผู้ใช้งาน โดยผู้ใช้งานโปรแกรมประเภทนี้ไม่จำเป็นต้องมีความชำนาญในการใช้งานโปรแกรมก็สามารถที่จะใช้งานได้
2. โปรแกรมที่ออกแบบตามหลักการวิวัฒนาการจะสามารถตอบสนองความต้องการของผู้ใช้งานได้ดีกว่าโปรแกรมธรรมดา เนื่องจากความสามารถในการเรียนรู้ และพัฒนาตนเองจึงทำให้โปรแกรมประเภทนี้มีความสามารถเพิ่มขึ้นได้ไม่จำกัด และยังทำให้โปรแกรมมีความสามารถเฉพาะในส่วนที่ผู้ใช้ต้องการ ส่วนใดก็ตามที่ผู้ใช้ไม่ต้องการก็จะไม่มีอยู่ในโปรแกรม
3. ในการที่จะทำให้เอเย่นต์หลาย ๆ ตัวมีความสามารถเพิ่มขึ้นนั้น ผู้ใช้อาจต้องทำการสอนเอเย่นต์ทุก ๆ ตัวถึงวิธีการทำงาน หากมีเอเย่นต์หลาย ๆ ตัวแล้วจะสร้างปัญหาให้กับผู้ใช้ได้ แต่สำหรับโปรแกรมที่ออกแบบตามหลักการวิวัฒนาการ ผู้ใช้ทำการสอนเอเย่นต์เพียงตัวเดียวก็พอ เอเย่นต์ที่เหลือจะเรียนรู้จากเอเย่นต์อื่นจนมีความสามารถเท่าเทียมกัน

ข้อจำกัดของหลักการวิวัฒนาการของโปรแกรมคือ

1. ในการเปลี่ยนแปลงหนึ่งอย่างนั้น ต้องมีโค้ดของคลาสเพิ่มขึ้นหนึ่งคลาส ทำให้โปรแกรมมีความยาวไปเรื่อย ๆ ไม่จำกัด หากมีความยาวมากจะทำให้ลำบากต่อการจัดการ จึงต้องมีการรวมเอาคลาสเหล่านั้นมาเป็นคลาสเดียว ทุก ๆ ครั้งที่โปรแกรมยาวเกินไป
2. ชื่อกระบวนการนั้นจะซ้ำกันไม่ได้ ไม่เช่นนั้นอาจเกิดความผิดพลาดขึ้นได้ เช่น จะตั้งชื่อกระบวนการว่า add ไม่ได้ เนื่องจากกระบวนการ add เป็นหนึ่งในกระบวนการมาตรฐานของโปรแกรมประเภทนี้ หรือหากมีผู้เพิ่มกระบวนการชื่อ mymethod เข้าไปในเอเย่นต์ตัวใดตัวหนึ่ง แล้วผู้ใช้อีกคนทำการเพิ่มกระบวนการชื่อเดียวกันเข้าไปในเอเย่นต์อีกตัวหนึ่ง จะเกิดข้อผิดพลาดขึ้น คือ เอเย่นต์ 2 ตัวมีกระบวนการทำงานชื่อ mymethod เหมือนกัน แต่อาจมีการทำงานคนละอย่างกัน หากมีผู้ใช้อื่นต้องการเพิ่มกระบวนการดังกล่าวเข้าไปในเอเย่นต์ อาจได้ผลไม่เหมือนกันในทุก ๆ ครั้งที่เพิ่มก็ได้
3. ในกระบวนการแก้ไขการเปลี่ยนแปลง เมื่อทำการแก้ไขสิ่งที่ทำไป จะเป็นเสมือนการย้อนเวลากลับไป ฉะนั้นการแก้ไขเพียงบางกระบวนการโดยไม่ส่งผลถึงกระบวนการอื่นอาจทำไม่ได้ เช่น ต้องการแก้ไขการลบกระบวนการที่ชื่อ mymethod โดยสมมติให้เวอร์ชันปัจจุบันคือ 10 และได้ทำการลบกระบวนการ mymethod ไปในเวอร์ชันที่ 5 การแก้ไขการเปลี่ยนแปลงจะทำให้เอเย่นต์มีอินสแตนส์อ้างอิงกับคลาสเวอร์ชัน 4 ดังนั้นการเปลี่ยนแปลงที่เกิดขึ้นระหว่างเวอร์ชันที่ 6 ถึง 10 จะหายไป
4. ถึงแม้โปรแกรมที่ออกแบบตามหลักการวิวัฒนาการจะมีความสามารถสูง แต่การเพิ่มกระบวนการใหม่ ๆ นั้นยังต้องอาศัยการเขียนโค้ดจากผู้ใช้อยู่ดี
5. ในการพัฒนาโปรแกรมที่มีวิวัฒนาการ จำเป็นต้องใช้ภาษาไพธอน และเนื่องจากภาษาไพธอนเป็นภาษาอินเตอพรีเตอร์ จึงทำให้การทำงานไม่เร็วเท่าไร หากโปรแกรมมีขนาดใหญ่ และมีการใช้เทรดมากอาจก่อให้เกิดความผิดพลาดได้

**4.5 สรุปขั้นตอนในการสร้างโปรแกรมที่มีวิวัฒนาการ**

หลักการวิวัฒนาการของโปรแกรมไพธอน สามารถสรุปเป็นขั้นตอนเพื่อใช้ในการสร้างโปรแกรมที่มีวิวัฒนาการ (เอเย่นต์) ได้ดังนี้

ขั้นตอนที่ 1 สร้างตัวเอเย่นต์ให้มีลักษณะเป็นคลาส และกำหนดชื่อคลาสให้สื่อถึงเวอร์ชัน

ขั้นตอนที่ 2 สร้างตัวแปรที่จะใช้เป็นอินสแตนส์ของคลาส 1 ตัว

ขั้นตอนที่ 3 สร้างตัวแปร 3 ตัวคือ

- ตัวแปร Methods เป็นลิสต์ ใช้เก็บชื่อกระบวนการที่สามารถเรียกใช้งานได้
- ตัวแปร Version เป็นสายอักขระ หรือตัวเลข ใช้เก็บค่าเวอร์ชันปัจจุบัน
- ตัวแปร History เป็นดิกชันนารี ใช้เก็บการเปลี่ยนแปลงที่กระทำต่อเอเย่นต์

ขั้นตอนที่ 4 สร้างโค้ดเพื่อเก็บค่าสมาชิกในตัวแปร Methods, ตัวแปร Version

ขั้นตอนที่ 5 สร้างกระบวนการมาตรฐานทั้ง 3 กระบวนการ

การสร้างกระบวนการมาตรฐานเพื่อเพิ่มกระบวนการทำงาน

ขั้นตอนที่ 5.1.1 สร้างคลาสที่มีการรับค่าของชื่อกระบวนการที่ต้องการเพิ่ม

ขั้นตอนที่ 5.1.2 ตรวจสอบดูว่าค่าที่รับมามีอยู่ในตัวแปร Methods หรือไม่ หากมีทำการแจ้งผู้ใช้ และจบการทำงาน

ขั้นตอนที่ 5.1.3 ตรวจสอบชื่อกระบวนการที่ต้องการเพิ่มว่ามีอยู่ในเอเย่นต์ตัวอื่นหรือไม่ หากมีทำการคัดลอกกระบวนการดังกล่าวมา หากไม่มีทำการสอบถามวิธีการทำงานของกระบวนการดังกล่าวจากผู้ใช้

ขั้นตอนที่ 5.1.4 นำกระบวนการที่ได้จากขั้นตอนที่ 5.1.3 มาใส่ในคลาส โดยคลาสดังกล่าวเป็นคลาสลูกของคลาสปัจจุบัน และมีชื่อสื่อถึงเวอร์ชันถัดจากเวอร์ชันปัจจุบัน อาจเก็บคลาสนี้ไว้ในตัวแปรในรูปของสายอักขระ

ขั้นตอนที่ 5.1.5 เพิ่มชื่อกระบวนการนั้นลงในตัวแปร Methods

ขั้นตอนที่ 5.1.6 เพิ่มค่าในตัวแปร Version เพื่อแสดงถึงเวอร์ชันถัดไป

ขั้นตอนที่ 5.1.7 ทำการบันทึกการเปลี่ยนแปลงลงในตัวแปร History ให้คีย์เป็นค่าของตัวแปร Version และให้ค่าที่เก็บเป็นลิสต์ 2 สมาชิก สมาชิกแรกคือ กระบวนการมาตรฐานที่ทำ และสมาชิกตัวหลังคือชื่อกระบวนการที่เกี่ยวข้อง

ขั้นตอนที่ 5.1.8 ทำการ EXEC คลาสที่สร้างขึ้นในขั้นตอนที่ 5.1.4

ขั้นตอนที่ 5.1.9 บันทึกข้อมูลต่าง ๆ ที่ได้ทำการเปลี่ยนแปลงในขั้นตอนที่ 5.1.4 ถึง 5.1.7 ลงในไฟล์ที่เก็บโค้ดของเอเย่นต์

การสร้างกระบวนการมาตรฐานเพื่อเปลี่ยนแปลงกระบวนการทำงาน

ขั้นตอนที่ 5.2.1 สร้างคลาสที่มีการรับค่าของชื่อกระบวนการที่ต้องการเปลี่ยนแปลง

ขั้นตอนที่ 5.2.2 ตรวจสอบว่าค่าที่รับมามีอยู่ในตัวแปร Methods หรือไม่ หากไม่มีทำการแจ้งผู้ใช้ และจบการทำงาน

ขั้นตอนที่ 5.2.3 ทำการสอบถามการทำงานของกระบวนการจากผู้ใช้

ขั้นตอนที่ 5.2.4 นำกระบวนการที่ได้ในขั้นตอนที่ 5.2.3 มาใส่ในคลาส โดยคลาสดังกล่าวเป็นคลาสลูกของคลาสปัจจุบัน และมีชื่อสื่อถึงเวอร์ชันถัดจากเวอร์ชันปัจจุบัน อาจเก็บคลาสนี้ไว้ในตัวแปรในรูปของสายอักขระ

ขั้นตอนที่ 5.2.5 เพิ่มค่าในตัวแปร Version เพื่อแสดงถึงเวอร์ชันถัดไป

ขั้นตอนที่ 5.2.6 ทำการบันทึกการเปลี่ยนแปลงลงในตัวแปร History ให้คีย์เป็นค่าของตัวแปร Version และให้ค่าที่เก็บเป็นลิสต์ 2 สมาชิก สมาชิกแรกคือ กระบวนการมาตรฐานที่ทำ และสมาชิกตัวหลังคือชื่อกระบวนการที่เกี่ยวข้อง

ขั้นตอนที่ 5.2.7 ทำการ EXEC คลาสที่สร้างขึ้นในขั้นตอนที่ 5.2.4

ขั้นตอนที่ 5.2.8 บันทึกข้อมูลต่าง ๆ ที่ได้ทำการเปลี่ยนแปลงในขั้นตอนที่ 5.2.4 ถึง 5.2.6 ลงในไฟล์ที่เก็บโค้ดของเอเย่นต์

การสร้างกระบวนการมาตรฐานเพื่อลบกระบวนการทำงาน

ขั้นตอนที่ 5.3.1 สร้างคลาสที่มีการรับค่าของชื่อกระบวนการที่ต้องการลบ

ขั้นตอนที่ 5.3.2 ตรวจสอบว่าค่าที่รับมามีอยู่ในตัวแปร Methods หรือไม่ หากไม่มีทำการแจ้งผู้ใช้ และจบการทำงาน

ขั้นตอนที่ 5.3.3 สร้างกระบวนการที่มีชื่อเดียวกับค่าที่รับมา แต่มีการทำงานเป็นคำสั่ง pass

ขั้นตอนที่ 5.3.4 นำกระบวนการที่สร้างในขั้นตอนที่ 5.3.3 มาใส่ในคลาส โดยคลาสดังกล่าวเป็นคลาสลูกของคลาสปัจจุบัน และมีชื่อสื่อถึงเวอร์ชันถัดจากเวอร์ชันปัจจุบัน อาจเก็บคลาสนี้ไว้ในตัวแปรในรูปของสายอักขระ

ขั้นตอนที่ 5.3.5 ลบชื่อกระบวนการนั้นออกจากตัวแปร Methods

ขั้นตอนที่ 5.3.6 เพิ่มค่าในตัวแปร Version เพื่อแสดงถึงเวอร์ชันถัดไป

ขั้นตอนที่ 5.3.7 ทำการบันทึกการเปลี่ยนแปลงลงในตัวแปร History ให้คีย์เป็นค่าของตัวแปร Version และให้ค่าที่เก็บเป็นลิสต์ 2 สมาชิก สมาชิกแรกคือ กระบวนการมาตรฐานที่ทำ และสมาชิกตัวหลังคือชื่อกระบวนการที่เกี่ยวข้อง

ขั้นตอนที่ 5.3.8 ทำการ EXEC คลาสที่สร้างขึ้นในขั้นตอนที่ 5.3.4

ขั้นตอนที่ 5.3.9 บันทึกข้อมูลต่าง ๆ ที่ได้ทำการเปลี่ยนแปลงในขั้นตอนที่ 5.3.4 ถึง 5.3.7 ลงในไฟล์ที่เก็บโค้ดของเอเย่นต์

ขั้นตอนที่ 6 สร้างอินสแตนส์อ้างอิงคลาสที่สร้างขึ้นในขั้นตอนที่ 5.1.4, 5.2.4 และ 5.3.4 โดยใช้ตัวแปรในขั้นตอนที่ 2

ขั้นตอนที่ 7 สร้างซูเปอร์ไวเซอร์ เพื่อติดต่อกับผู้ใช้ และควบคุมการทำงานของเอเย่นต์ต่าง ๆ

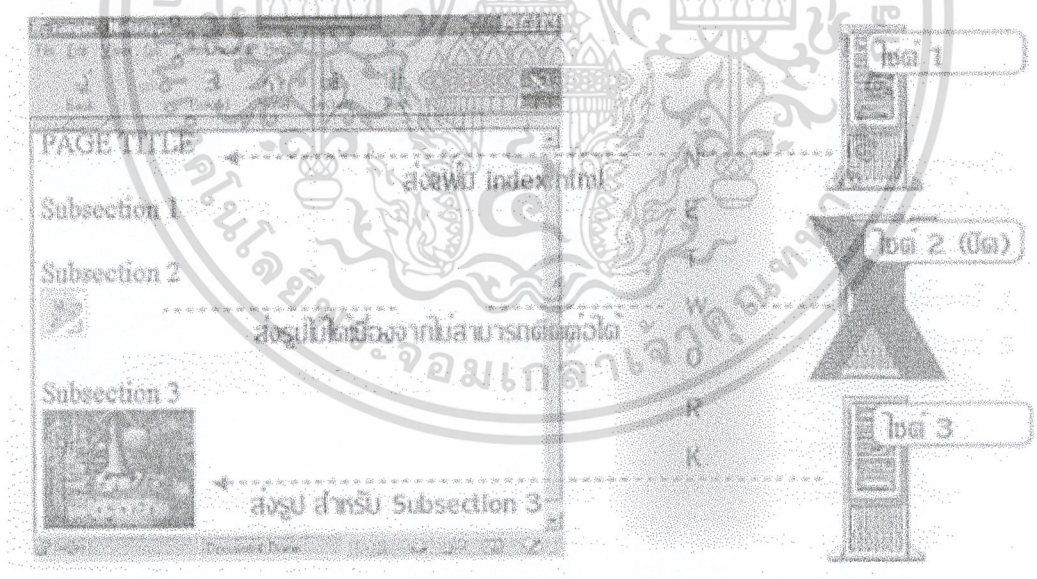
# บทที่ 5

# ทฤษฎีที่เกี่ยวข้องกับการสร้างแอปพลิเคชั่นเว็บ

ในโครงงานนี้ได้มีการคิดวิธีการในการเปลี่ยนกระบวนการทำงานของโปรแกรมเอเย่นต์ซึ่งมีการทำงานบนเว็บ ดังนั้นจึงต้องมีการศึกษาหลักการที่ใช้ในการพัฒนาโปรแกรมที่มีการใช้งานบนเว็บ โดยหลักการที่สำคัญได้แก่ HTTP Protocol และ CGI

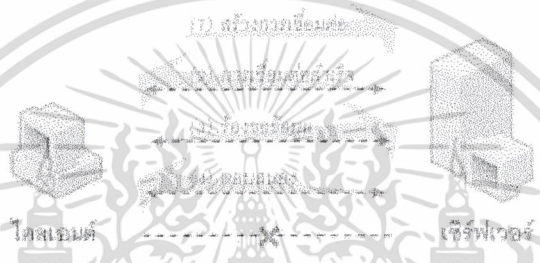
## 5.1 HTTP (Hyper Text Transfer Protocol)

HyperText Transfer Protocol คือระเบียบวิธีในการติดต่อสื่อสาร และการจัดการรับส่งข้อมูลในรูปของ HyperText หรือเว็บเพจ (Web Page) นั่นเอง โดยรูปแบบในการรับส่งข้อมูลของ HTTP นั้นจะเป็นแบบ Connection Oriented และการทำงานพื้นฐานจะเป็นไปในลักษณะ Transaction Oriented กล่าวคือจะใช้หลักการพื้นฐานของ ไคลเอนต์ – เซิร์ฟเวอร์ ในการร้องขอบริการ ซึ่งข้อมูลต่าง ๆ ที่เป็นส่วนประกอบของเว็บเพจที่ร้องขอบริการ เช่น อักษร, ภาพ, เสียง หรืออื่น ๆ จะมีการเปิดการติดต่อใหม่ และเป็นอิสระต่อกัน




*รูปที่ 5.1 แสดงการเปิดการติดต่อย่อยในการร้องขอโฮมเพจ*

จากรูป เป็นตัวอย่างเพื่อแสดงให้เห็นว่า การส่งข้อมูลต่าง ๆ ของ HTTP โพรโตคอลนั้น เป็นอิสระต่อกัน เว็บเพจบนเครื่องไคลเอนต์ ที่มีการติดต่อขอข้อมูลจากแหล่งอื่น 3 แห่ง โดยที่ไคลเอนต์ใช้บราวเซอร์ (Browser) ร้องขอข้อมูลจากแต่ละแหล่ง โดยแหล่งข้อมูลที่ร้องขอไปนั้นจะสร้างการติดต่อที่เป็นอิสระต่อกันขึ้นมาเพื่อส่งข้อมูล หากแหล่งข้อมูลใดเกิดความผิดพลาดไม่สามารถส่งข้อมูลที่ร้องขอมาให้ได้ ก็ไม่มีผลกระทบต่อส่วนอื่น เว็บเพจที่ฝั่งไคลเอนต์ยังคงแสดงผลได้ปกติ ข้อมูลที่ไม่สามารถส่งมาได้ก็จะไม่ถูกแสดงผล




*รูปที่ 5.2 แสดงการทำงานของ HTTP โพรโตคอล*

**วิธีการติดต่อของโพรโตคอล HTTP**

ด้วยเหตุที่การทำงานของโพรโตคอล HTTP เป็นแบบไคลเอนต์และเซิร์ฟเวอร์ (client/server) ดังนั้นการติดต่อสื่อสารใด ๆ ผ่านโพรโตคอลนี้จำเป็นต้องมีเครื่องตัวลูกกับตัวแม่ การสื่อสารจึงจะสมบูรณ์ได้ ไคลเอนต์จะสร้างการเชื่อมต่อ (connection) กับเซิร์ฟเวอร์ผ่านซ็อกเก็ต (socket) เมื่อซ็อกเก็ตทั้งสองฝั่งเชื่อมต่อกันได้สำเร็จ ไคลเอนต์จะส่งคำร้องขอข้อมูล (request) ไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์จะไปหาข้อมูลที่ไคลเอนต์ต้องการ ซึ่งไม่ว่าจะมีหรือไม่มีข้อมูลตามที่ไคลเอนต์ร้องขอ เซิร์ฟเวอร์ก็จะต้องส่งข้อมูลตอบสนอง (response) กลับมายังไคลเอนต์เสมอ สุดท้ายการเชื่อมต่อจะถูกตัดขาดหรือปลดการเชื่อมต่อของซ็อกเก็ตทั้งสองฝั่งออกนั่นเอง

ด้วยการทำงานของโพรโตคอล HTTP ที่มีการเชื่อมต่อในระยะเวลาเพียงสั้นๆ หรือที่เรียกว่าเป็นโพรโตคอลแบบ connectionless ในลักษณะดังกล่าว ทำให้ในช่วงเวลาหนึ่งๆ เซิร์ฟเวอร์ที่ให้บริการ www สามารถรองรับไคลเอนต์ได้จำนวนมากพร้อมๆ กัน เพราะไม่มีใครได้ทำการเชื่อมต่ออย่างถาวร


*รูปที่ 5.3 การติดต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์*

หลักการทำงานของโพรโตคอล HTTP คือ ฝั่งไคลเอนต์ใช้บราวเซอร์ติดต่อกับเว็บเซิร์ฟเวอร์ ซึ่งโปรแกรมบราวเซอร์นี้จะถูกเรียกว่า HTTPD (Hyper Text Transfer Protocol Daemon) ปกติแล้วจะทำงานที่พอร์ต (Port) 80 การติดต่อกับเว็บเซิร์ฟเวอร์นั้นปกติแล้วไคลเอนต์ต้องเป็นฝ่ายเริ่มเปิดการพูดคุยด้วยการส่งข้อมูลเพื่อทำการร้องขอ (Request) ไปยังเซิร์ฟเวอร์ และรอการตอบสนอง (Response) การร้องขอบริการจากเว็บเซิร์ฟเวอร์ของโพรโตคอล HTTP มีคำสั่งหลัก ๆ ที่ใช้กันทั่วไปได้แก่

- การร้องขอเพื่อให้เซิร์ฟเวอร์ส่งไฟล์มาให้ แบบนี้จะเรียกว่าร้องขอแบบ GET
- การร้องขอเพื่อถามเซิร์ฟเวอร์ว่า มีไฟล์ที่ต้องการอยู่ในเซิร์ฟเวอร์หรือไม่ (เป็นการถามเฉย ๆ ไม่ต้องการให้เซิร์ฟเวอร์ส่งไฟล์จริงมาให้) แบบนี้จะเรียกว่าร้องขอแบบ HEAD
- การร้องขอให้เซิร์ฟเวอร์รับข้อมูลจากไคลเอนต์ เรียกว่าร้องขอแบบ POST ซึ่งการร้องขอแบบนี้หมายความว่า ไคลเอนต์ต้องการส่งข้อมูลไปให้เซิร์ฟเวอร์

การร้องขอบริการจะมีความเกี่ยวข้องกับการระบุสถานที่ที่ต้องการใช้บริการ และที่อยู่ของข้อมูลต่าง ๆ ตามหลักการของ URL (Uniform Resource Locators) นอกจากนี้ยังต้องระบุโฮสต์ (Host) ซึ่งจะเป็นเครือข่ายปลายทาง และรายละเอียดนี้จะสัมพันธ์กับส่วนเพิ่มเติมของ URL

หลังจากเว็บเซิร์ฟเวอร์ได้รับการร้องขอบริการจากไคลเอนต์แล้วจะทำการตอบสนองการร้องขอนั้น ซึ่งจะมีรูปแบบการใช้งานเหมือนกับโพรโตคอลอื่น ๆ ที่ใช้หลักการของไคลเอนต์ – เซิร์ฟเวอร์ รูปแบบของการตอบสนองจะมีการส่งข้อมูลกลับ นำโดยหมายเลขตอบสนอง (Response Tags Number) ซึ่งจะเป็นค่ามาตรฐานมีความหมายตามตารางที่ 5.1 ตามด้วยข้อความอธิบายรูปแบบของข้อมูลที่ส่ง และส่วนสุดท้ายคือ ข้อมูลที่ถูกร้องขอ

| Response Tags Number | รายละเอียด |
|---|---|
| 100 | Continue |
| 101 | Switching |
| 200 | OK |
| 201 | Created |
| 202 | Accepted |
| 203 | Non-Authoritative information |
| 204 | No Content |
| 205 | Reset Content |
| 206 | Partial Content |
| 300 | Multiple Choices |
| 301 | Moved Permanently |
| 302 | Moved temporarily |
| 303 | See Other |
| 304 | Not Modified |
| 305 | Use Proxy |

*ตารางที่ 5.1 แสดงรายละเอียดของหมายเลขสถานภาพการทำงานของ HTTP โพรโตคอล*

โพรโตคอล HTTP จะระบุประเภทข้อมูล (data type) มากับข้อมูลเสมอ ดังนั้นเราจึงสามารถส่งไฟล์ทุกประเภทผ่านทางโพรโตคอลนี้ เมื่อไคลเอนต์ได้รับไฟล์ไปแล้วทราบวิธีการแสดงผลของไฟล์นั้น ก็จัดการตามกรรมวิธีของไฟล์นั้นได้เลย แต่ถ้าหากไม่ทราบ ก็เป็นหน้าที่ของผู้ใช้เองที่ต้องหาโปรแกรมหรือแอปพลิเคชั่นมาจัดการกับไฟล์นั้นเอง

**5.2 CGI (Common Gateway Interface)**

ความต้องการที่จะส่งข้อมูลผ่านระบบเครือข่าย เพื่อสร้างการสื่อสารระหว่างโปรแกรมซึ่งปฏิบัติการอยู่บนเครื่องคอมพิวเตอร์คนละเครื่อง เป็นสาเหตุให้มีการพัฒนาเทคโนโลยีต่าง ๆ ขึ้นเพื่อตอบสนอง เทคโนโลยีหนึ่งที่ใช้คือ เทคโนโลยี Common Gateway Interface โดยการใช้ CGI นี้เราสามารถพัฒนาโปรแกรมเพื่อตอบสนองความต้องการหลากหลายไม่ว่าจะเป็น การค้นหาข้อมูลในระบบเครือข่าย การให้คำปรึกษาผ่านระบบเครือข่ายโดยระบบผู้เชี่ยวชาญ (Expert System) ฯลฯ

Common Gateway Interface หรือ CGI อาจกล่าวได้ว่าคือ หลักการหรือวิธีการของการพัฒนาโปรแกรมประยุกต์ ที่ทำหน้าที่เสมือนประตู (Gateway) เชื่อมโยงการติดต่อกับการทำงานอื่น ๆ เพื่อให้เกิดการทำงานที่หลากหลายในการใช้งาน โดยอาศัยพื้นฐานของระบบเครือข่าย หรือกล่าวได้ว่าเป็นการทำงานร่วมกันกับเซิร์ฟเวอร์ของระบบเครือข่าย (Web Server) โปรแกรมที่พัฒนาขึ้นโดยใช้แนวความคิดของ CGI นั้นจะเป็นโปรแกรมที่ทำงานบนฝั่งของเซิร์ฟเวอร์ โดยมีส่วนที่ทำหน้าที่ติดต่อกับผู้ขอใช้

บริการ (Client) ข้อดีของวิธีการแบบ CGI คือ การปรับปรุงโปรแกรมสามารถทำได้ที่เครื่องเซิร์ฟเวอร์โดยตรง ไม่ต้องยุ่งเกี่ยวกับเครื่องของผู้ขอใช้บริการ จึงทำให้การปรับปรุงโปรแกรมเป็นไปได้ง่าย การเกิดขึ้นของเทคโนโลยี CGI นี้ทำให้โปรแกรมบนระบบเครือข่ายมีการพัฒนาไปจากเดิมเป็นอย่างมาก เนื่องจากสามารถพัฒนาโปรแกรมให้โต้ตอบ (Interact) กับผู้ใช้ได้

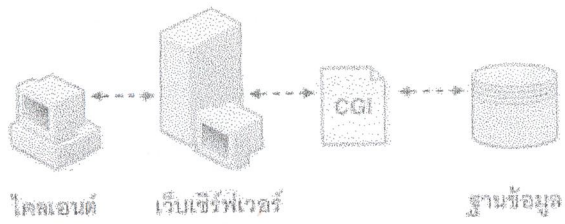
การทำงานของ CGI อาศัยหลักการของ ไคลเอนต์ – เซิร์ฟเวอร์ (Client - Server) โดยเว็บเซิร์ฟเวอร์จะเป็นผู้ติดต่อขอใช้บริการและรับรองผลลัพธ์ของ CGI กลับมา จากนั้นจึงส่งต่อให้กับผู้ใช้ที่ใช้งานผ่านบราวเซอร์ (Browser) ที่ฝั่งไคลเอนต์ ผลลัพธ์ของ CGI ที่ได้จะถูกส่งให้กับผู้ใช้โดยผ่านเซิร์ฟเวอร์ก่อน การส่งนั้นจะส่งโดยรูปแบบของ MIME ซึ่งระบุส่วนหัว (Header) ไว้ ส่วนหัวจะมีอยู่ 2 แบบคือ แบบเต็ม (Full Header) และแบบบางส่วน (Partial Header)

CGI เป็นแอปพลิเคชั่น ซึ่งทำงานเป็นตัวกลางระหว่างเซิร์ฟเวอร์กับไคลเอนต์ ข้อมูลที่จะให้บริการผ่านเว็บได้นั้นจะต้องถูกเก็บอยู่ในรูปแบบของเอกสาร HTML เมื่อเซิร์ฟเวอร์ได้รับการร้องขอไฟล์จากไคลเอนต์ เว็บเซิร์ฟเวอร์จะค้นหาและส่งไฟล์ที่ไคลเอนต์ต้องการกลับไปให้ ดังรูปที่ 5.4


***รูปที่ 5.4 การร้องขอเอกสาร HTML จากเซิร์ฟเวอร์***

ปัญหาของข้อมูลที่เก็บด้วยรูปแบบ HTML คือ เมื่อจะต้องมีการอัปเดตข้อมูลจะเป็นงานที่ยุ่งยากและเสียเวลาเป็นอย่างมาก เพราะว่าไฟล์เอกสาร HTML มีลักษณะการจัดเก็บแบบตายตัว (static) ยิ่งถ้ามีข้อมูลมากด้วยแล้ว การจัดเก็บข้อมูลแยกออกเป็นไฟล์ๆ ยิ่งจะทำให้ดูแลแก้ไขได้ยากมากขึ้น เป้าหมายของการใช้ CGI อย่างหนึ่งก็คือ ทำให้เอกสาร HTML ที่ผู้ร้องขอเข้ามามีความยืดหยุ่นหรือที่เรียกว่าเป็นแบบ dynamic

วิธีการทำเอกสารให้เป็นแบบ dynamic คือ แทนที่จะเก็บข้อมูลแยกเป็นไฟล์ HTML หลายๆ ไฟล์ เราก็อาจจะเก็บข้อมูลทั้งหมดไว้ในไฟล์เดียว เมื่อผู้ใช้ต้องการข้อมูลอะไร ก็กำหนดให้ผู้ใช้ป้อนเงื่อนไขที่ต้องการเข้ามาให้แก่ CGI หลังจากนั้น CGI จะไปค้นหรือดึงเอาเฉพาะข้อมูลที่ตรงตามที่ผู้ใช้ต้องการ จากนั้นจึงนำข้อมูลมาสร้างเป็นเอกสาร HTML แล้วส่งกลับไปแสดงให้ผู้ใช้ดู ดังนั้นเอกสาร HTML ที่ผู้ใช้แต่ละคนได้รับอาจไม่เหมือนกัน ขึ้นอยู่กับเงื่อนไขความต้องการของผู้ใช้ ในกรณีนี้ CGI จะทำหน้าที่เป็นประตูหรือ gateway ระหว่างฐานข้อมูลในเซิร์ฟเวอร์กับไคลเอนต์นั่นเอง ดังรูปที่ 5.5




*รูปที่ 5.5 การร้องขอข้อมูลจากเซิร์ฟเวอร์ผ่าน CGI*

ลักษณะการทำงานของ CGI ต้องอาศัยการประมวลผลที่เซิร์ฟเวอร์ แล้วสร้างคำตอบออกมาเป็นเนื้อหาแบบ HTML จากนั้นจึงส่งเนื้อหากลับไปให้ไคลเอนต์ เซิร์ฟเวอร์ใดที่ยอมให้มีการรัน CGI ได้ จึงต้องทำงานหนักกว่าเซิร์ฟเวอร์ที่ให้บริการเอกสาร HTML เพียงอย่างเดียว แนวความคิดการทำงานของ CGI จะเป็นแบบที่เรียกว่ารวมศูนย์ หรือ centralize งานทุกอย่างต้องวิ่งเข้ามารันที่เซิร์ฟเวอร์หมด ไคลเอนต์เพียงแต่ทำหน้าที่ส่งคำร้องขอและรอรับผลการทำงานเท่านั้น ซึ่งดูเหมือนว่า CGI จะขัดกับแนวคิดในการทำงานแบบกระจายศูนย์หรือ distribute ที่พยายามลดการเข้าไปรันงานในเซิร์ฟเวอร์ แต่ให้ไคลเอนต์ทำงานให้มากที่สุด

แม้ว่าการเขียนโปรแกรมด้วยเทคโนโลยี CGI นี้จะช่วยกำจัดข้อจำกัดต่าง ๆ ของการเขียนโปรแกรมบนระบบเครือข่ายได้ แต่ก็มิได้หมายความว่าข้อจำกัดต่าง ๆ จะหมดไป เทคโนโลยี CGI มีข้อจำกัดบางประการคือ

1. ไม่สามารถทำงานแบบวนรอบภายในตัวเองได้ (Loop Execution)
2. ไม่เหมาะในการสร้างโปรแกรมที่มีความจำเป็นต้องประมวลผลแบบทรานแซกชัน (Transaction Processing) ทั้งนี้เพราะในส่วนของฟอร์ม (Form) หรือ HTML ที่ฝั่งไคลเอนต์มิได้ถูกออกแบบมาให้ประมวลผลด้วยตัวเอง แต่เป็นการส่งข้อมูลให้กับ CGI ที่ฝั่งเซิร์ฟเวอร์ประมวลผล จากนั้นจึงทำการส่งผลลัพธ์กลับไปยังฝั่งไคลเอนต์ การทำงานในรูปแบบนี้อาจทำให้เกิดการผิดพลาดจากการติดต่อสื่อสารขึ้น และทำให้การประมวลผลแบบทรานแซกชันเกิดความผิดพลาด
3. เนื่องจากฟอร์มที่ฝั่งไคลเอนต์ไม่สามารถกำหนดชนิดของข้อมูลได้ ดังนั้นจึงเป็นหน้าที่ของผู้เขียนโปรแกรม CGI เองที่จะคอยจัดการกับชนิดของข้อมูลที่รับมา ซึ่งปัญหานี้เกิดขึ้นกับภาษาคอมพิวเตอร์หลายภาษา แต่การใช้ภาษา เช่น C, Perl หรือภาษาไพธอน สามารถที่จะกำจัดข้อจำกัดนี้ได้ เนื่องจากภาษาที่กล่าวมานี้มีความสามารถในการจัดการกับชนิดข้อมูลได้อัตโนมัติ ภาษาไพธอนเรียกความสามารถนี้ว่า Dynamic Typing

# บทที่ 6
# การทดลองสร้างแอปพลิเคชั่นเว็บ

ในการนำหลักการการเปลี่ยนกระบวนการทำงานของเอเย่นต์มาใช้งานเพื่อพัฒนาเป็นแอปพลิเคชั่นที่ทำงานบนเว็บ โครงงานนี้ได้มีการนำหลักการดังกล่าวไปพัฒนาโปรแกรมซึ่งแบ่งเป็น 2 ชิ้นงานได้แก่

- เอเย่นต์วาดรูป (Drawing Agent)
- เพอซันแนลแอซซิสแทนต์ (Personal Assistant)

โปรแกรมตัวแรกคือ เอเย่นต์วาดรูป สร้างขึ้นมาเพื่อแสดงการทำงานร่วมกันและการแลกเปลี่ยนกระบวนการทำงานแล้วนำมาพัฒนาตัวเองของเอเย่นต์ โปรแกรมนี้มีการทำงานคือมีเอเย่นต์ที่ทำงานร่วมกันในการวาดรูปโดยแต่ละตัวจะมีความสามารถที่แตกต่างกันและสามารถแลกเปลี่ยนกระบวนการทำงานระหว่างกันได้ ส่วนโปรแกรมเพอซันแนลแอซซิสแทนต์เป็นโปรแกรมที่ประยุกต์หลักการดังกล่าวไปใช้กับแอปพลิเคชั่นบนเว็บ โดยเป็นโปรแกรมที่สามารถจัดการกับตารางนัดหมายของผู้ใช้งานโปรแกรมผ่านทางอินเตอร์เน็ต ซึ่งผู้ใช้งานสามารถบันทึกตารางเวลาของตัวเองลงบนเว็บเพจและระบุวิธีการที่ต้องการให้เอเย่นต์ทำการแจ้งเตือนไปยังผู้ใช้งานเมื่อเวลาที่กำหนด รวมทั้งสามารถเพิ่มและลดกระบวนการทำงานของโปรแกรมได้

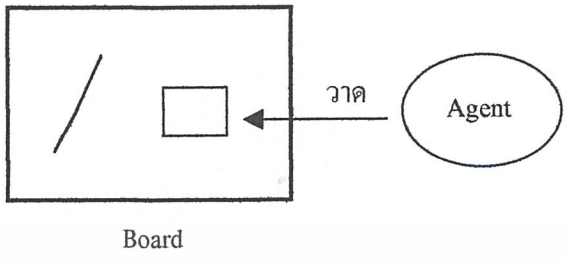
สำหรับบทนี้จะทำการอธิบายแนวคิดรวมทั้งการออกแบบและพัฒนาโปรแกรมทั้ง 2 ตัวโดยจะแบ่งออกเป็น 2 หัวข้อตามโปรแกรมที่ได้พัฒนาขึ้นมา

## 6.1 เอเย่นต์วาดรูป (Drawing Agent)

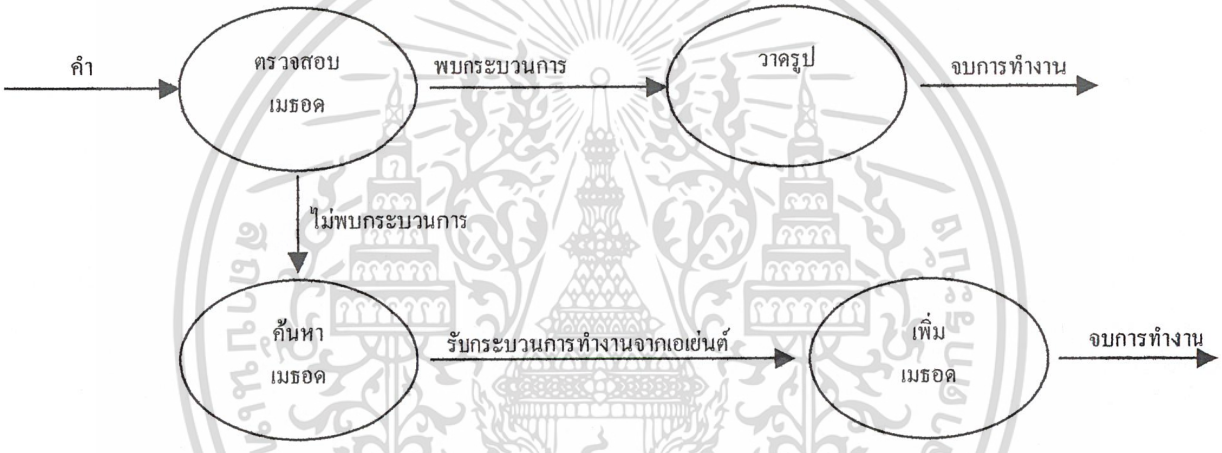
### แนวคิดและหลักการทำงาน

โปรแกรมเอเย่นต์วาดรูปเป็นโปรแกรมแสดงการทำงานร่วมกันของเอเย่นต์หลายๆตัว ซึ่งจะช่วยกันทำการวาดรูปทรงเรขาคณิตตามที่ผู้ใช้งานกำหนด โดยเอเย่นต์ทุกตัวจะสามารถวาดรูปลงบนบอร์ดสำหรับวาดรูปเดียวกันเพื่อแสดงผลการทำงานของเอเย่นต์ที่ได้ตามความต้องการของผู้ใช้งาน เอเย่นต์แต่ละตัวจะมีความสามารถในการวาดรูปที่แตกต่างกันอันได้แก่ รูปเส้นตรง รูปสี่เหลี่ยม และรูปสามเหลี่ยม เมื่อผู้ใช้งานกำหนดรูปที่ต้องการให้เอเย่นต์ทำการวาดลงบนบอร์ด คำสั่งจากผู้ใช้งานจะถูกแบ่งให้กับเอเย่นต์แต่ละตัวโดยซูเปอร์ไวเซอร์ (Supervisor) เพื่อทำการวาดรูปตามที่ได้รับคำสั่งมา เมื่อเอเย่นต์แต่ละตัวได้รับคำสั่งให้ทำการวาดรูปถ้าเอเย่นต์ตัวนั้นมีความสามารถในการวาดรูปที่ได้รับมอบหมายเอเย่นต์ก็จะทำการวาดรูปนั้นทันที แต่ถ้าเอเย่นต์นั้นไม่มีความสามารถในการวาดรูปตามที่ได้รับมอบหมายมาจากซูเปอร์ไวเซอร์ เอเย่นต์ตัวนั้นจะไม่ทำการวาดรูปตามที่ผู้ใช้งานต้องการแต่จะทำการตรวจสอบไปยังเอเย่นต์ตัวอื่นว่ามีความสามารถนั้นหรือไม่ ถ้าเอเย่นต์พบว่ามีเอเย่นต์ตัวอื่นที่มีความสามารถที่ต้องการก็จะทำการดึงกระบวนการทำงานนั้นๆมาจากเอเย่นต์ที่มีความสามารถนั้นแล้วทำการเปลี่ยนแปลงกระบวน

การทำงานของตัวเองโดยใช้หลักการของการวิวัฒนาการ ทำให้เอเย่นต์มีความสามารถนั้นด้วย เมื่อมีการสั่งให้ทำการวาดรูปชนิดนี้อีกครั้งเอเย่นต์ก็จะสามารถทำการวาดรูปนั้นได้โดยไม่ต้องนำมาจากเอเย่นต์ตัวอื่นอีก




*รูปที่ 6.1 แสดงการทำงานของเอเย่นต์บนบอร์ด*




*รูปที่ 6.2 แสดงขั้นตอนการทำงานของเอเย่นต์*

โปรแกรมเอเย่นต์วาดรูปประกอบด้วยส่วนประกอบสำคัญ 4 ส่วนคือ

- ซูเปอร์ไวเซอร์
- เอเย่นต์วาดรูป
- ส่วนรับอินพุตจากผู้ใช้งานโปรแกรม
- ส่วนแสดงผลการทำงาน

โดยแต่ละส่วนมีการทำงานดังนี้

ซูเปอร์ไวเซอร์ เป็นโปรแกรมที่ทำหน้าที่ในการแบ่งงานให้กับเอเย่นต์แต่ละตัว โดยจะรับคำสั่งที่ผู้ใช้งานป้อนเข้ามาแล้วทำการแบ่งงานเป็นส่วนๆแล้วกระจายงานให้เอเย่นต์แต่ละตัวเท่าๆกัน นอกจากนั้นจะนำผลที่ได้จากการทำงานของเอเย่นต์แต่ละตัวแสดงลงในส่วนแสดงผลเพื่อแสดงให้กับผู้ใช้งานโปรแกรม

เอเย่นต์วาดรูป จะเป็นเอเย่นต์ที่มีหน้าที่ในการรับคำสั่งจากซูเปอร์ไวเซอร์แล้วนำไปวาดรูปตามที่ได้รับมอบหมาย เอเย่นต์แต่ละตัวจะมีกระบวนการทำงานมาตรฐานที่เหมือนกัน เช่น กระบวนการสำหรับจัดการกับการเปลี่ยนแปลงกระบวนการทำงานของตัวเองคือ add, revise, remove นอกจากนั้นยังมีกระบวนการมาตรฐานสำหรับใช้วาดรูปอีกหลายกระบวนการทำงาน

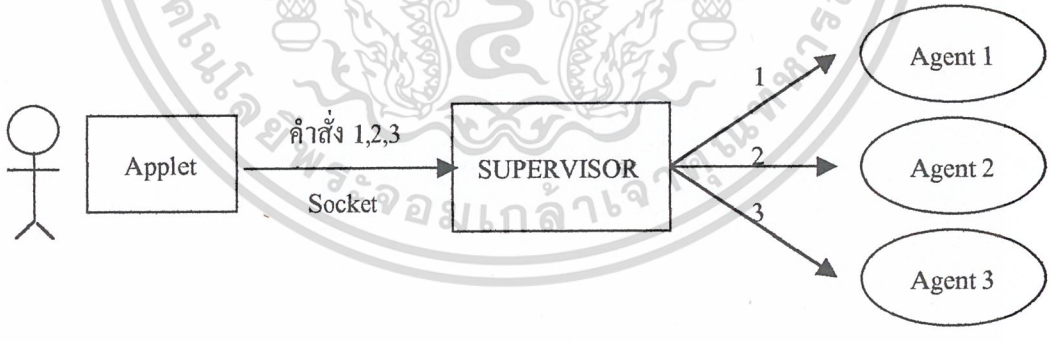
ส่วนรับอินพุตจากผู้ใช้งานโปรแกรม ส่วนนี้จะเป็นส่วนที่ทำหน้าที่ในการรับคำสั่งจากผู้ใช้ซึ่งได้แก่รูปต้นแบบที่ต้องการให้เอเย่นต์แต่ละตัวทำการวาด โดยผู้ใช้สามารถสั่งงานเอเย่นต์ได้โดยการวาดรูปต้นแบบให้กับตัวซูเปอร์ไวเซอร์เพื่อนำไปแบ่งให้เอเย่นต์ทำการวาดต่อไป

ส่วนแสดงผลการทำงาน เป็นส่วนที่นำผลการทำงานจากเอเย่นต์แต่ละตัวไปแสดงให้ผู้ใช้งานโปรแกรมได้ทราบผลจากการทำงานของเอเย่นต์

**ส่วนประกอบของโปรแกรมเอเย่นต์วาดรูป**

**ซูเปอร์ไวเซอร์**

เป็นส่วนที่ทำการควบคุมการทำงานของโปรแกรม ซึ่งจะทำการแบ่งงานที่ได้รับมอบหมายจากผู้ใช้งานโปรแกรมไปยังเอเย่นต์แต่ละตัว ซูเปอร์ไวเซอร์สามารถรับคำสั่งได้ทั้งจากผู้ใช้งานผ่านทางเว็บและทางการสั่งงานที่เซิร์ฟเวอร์ ในการรับคำสั่งผ่านทางเว็บซูเปอร์ไวเซอร์จะทำการสร้างซอกเก็ตเพื่อติดต่อกับส่วนรับอินพุตจากผู้ใช้งานโปรแกรมที่เป็นจาวาแอพเพล็ต ตัวซูเปอร์ไวเซอร์จะทำงานตลอดเวลาโดยโปรแกรมจะเป็นเธรดที่ทำการรับคำสั่งของผู้ใช้งานซึ่งจะเป็นคำสั่งที่เป็นสายอักขระ แล้วทำการตัดชุดคำสั่งให้แก่เอเย่นต์แต่ละตัวโดยการเรียกใช้เมธอดของเอเย่นต์ที่ใช้รับคำสั่ง สำหรับโปรแกรมที่ทำงานเป็นซูเปอร์ไวเซอร์จะอยู่ในโปรแกรมภาษาไพธอนที่ชื่อ paintbook.py




***รูปที่ 6.3 แสดงการทำงานของซูเปอร์ไวเซอร์***

**เอเย่นต์วาดรูป**

เป็นโปรแกรมเอเย่นต์ที่ทำหน้าที่ในการรับคำสั่งจากซูเปอร์ไวเซอร์แล้วนำมาทำงานตามที่ได้รับมอบหมายซึ่งได้แก่การวาดรูปตามกระบวนการทำงานที่มีในเอเย่นต์แต่ละตัว เอเย่นต์แต่ละตัวจะมีกระบวนการทำงานมาตรฐานสำหรับใช้ในการวาดรูปบนบอร์ดดังนี้

- goto เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์ไปยังตำแหน่งที่ต้องการบนบอร์ด

- turnto เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์หันไปในมุมองศาที่ต้องการบนบอร์ด
- forward เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์เลื่อนไปข้างหน้าตามระยะทางที่กำหนดโดยมีทิศทางตามมุมที่กำหนด
- rotate_l เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์หมุนทวนเข็มนาฬิกาไปเป็นมุมตามที่กำหนด
- rotate_r เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์หมุนตามเข็มนาฬิกาไปเป็นมุมตามที่กำหนด
- pen_up เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์ทำการวาดรูปลงบนบอร์ดในขณะที่เอเย่นต์มีการใช้คำสั่ง forward
- pen_down เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์ไม่ทำการวาดรูปลงบนบอร์ดในขณะที่เอเย่นต์มีการใช้คำสั่ง forward
- eraser เป็นกระบวนการทำงานที่กำหนดให้เอเย่นต์ทำการวาดรูปโดยใช้สีพื้นของบอร์ดในขณะที่เอเย่นต์มีการใช้คำสั่ง forward
- color เป็นกระบวนการทำงานสำหรับกำหนดสีที่ใช้ในการวาดรูปของเอเย่นต์

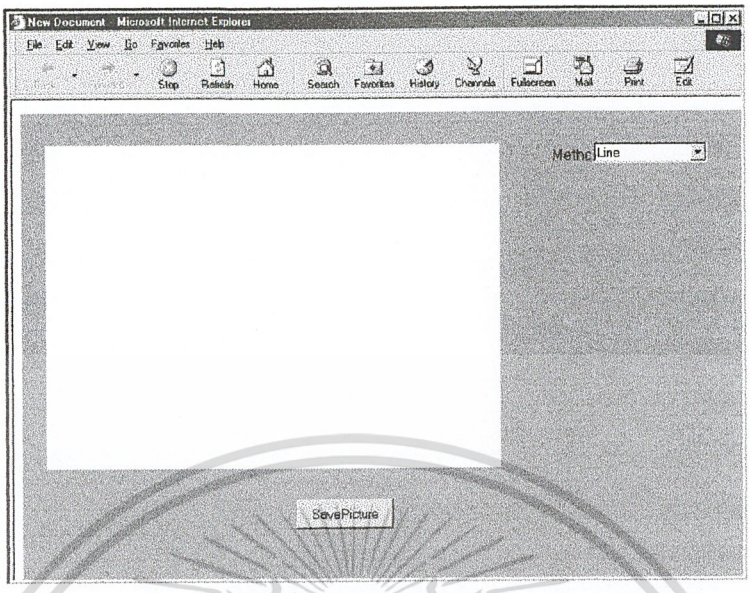
นอกจากนั้นเอเย่นต์แต่ละตัวจะมีกระบวนการทำงานเฉพาะสำหรับใช้ในการวาดรูปที่มีลักษณะแตกต่างกัน โดยกระบวนการที่เพิ่มเข้ามานั้นจะทำการเรียกใช้กระบวนการมาตรฐานที่มีอยู่ในเอเย่นต์ทุกตัว เมื่อมีการแลกเปลี่ยนกระบวนการทำงานระหว่างเอเย่นต์แต่ละตัวจึงสามารถนำไปใช้งานได้ทันที กระบวนการทำงานที่เพิ่มขึ้นมาเช่น Line ซึ่งประกอบไปด้วยกระบวนการมาตรฐานของเอเย่นต์คือ goto, turnto, pen_down และ forward เมื่อเอเย่นต์ได้รับคำสั่ง line เอเย่นต์จะมีการทำงานตามกระบวนการที่กำหนดเพื่อทำการวาดรูปเส้นตรงลงบนบอร์ด

นอกจากกระบวนการทำงานสำหรับใช้ในการวาดรูปลงบนบอร์ดแล้วเอเย่นต์จะมีกระบวนการทำงานมาตรฐานสำหรับใช้ในการจัดการกับการเปลี่ยนแปลงกระบวนการทำงานของตัวเอง กระบวนการมาตรฐานเหล่านี้ได้แก่

- Add เป็นกระบวนการทำงานสำหรับทำการนำกระบวนการทำงานที่ต้องการมาเพิ่มให้กับตัวเอง
- Revise เป็นกระบวนการทำงานสำหรับเปลี่ยนแปลงกระบวนการทำงานที่มีอยู่แล้วในตัวเอเย่นต์
- Remove เป็นกระบวนการทำงานสำหรับลบกระบวนการทำงานที่ต้องการออกจากตัวเอเย่นต์

**ส่วนรับอินพุตจากผู้ใช้งานโปรแกรม**

ในส่วนที่รับอินพุตจากผู้ใช้งาน จะแบ่งเป็น 2 ส่วนคือ ส่วนที่รับอินพุตจากผู้ใช้งานผ่านทางเว็บเพจ และส่วนที่รับอินพุตจากผู้ใช้งานที่เซิร์ฟเวอร์ ซึ่งจะมีการทำงานเหมือนกันคือ จะมีส่วนแสดงผลของการวาดจากผู้ใช้งาน โดยผู้ใช้งานสามารถกำหนดชนิดของรูปที่ต้องการวาดลงบนบอร์ดได้ ซึ่งได้กำหนดไว้ 3 ชนิดคือ รูปเส้นตรง รูปสี่เหลี่ยม และรูปสามเหลี่ยม เมื่อผู้ใช้งานกำหนดชนิดของรูปแล้วก็สามารถใช้เมาส์ทำการวาดรูปลงบนบอร์ดสำหรับรับคำสั่ง หลังจากนั้น ผู้ใช้งานสามารถสั่งคำสั่งให้เอเย่นต์ทำการวาดภาพที่กำหนดให้โดยการกดปุ่มซับมิท ซึ่งถ้าเป็นผู้ใช้งานผ่านทางเว็บเพจ ส่วนรับอินพุตจะทำการส่งคำสั่งผ่านทางซ็อคเก็ตไปยังตัวซูเปอร์ไวเซอร์ เพื่อแบ่งงานให้กับเอเย่นต์นำไปทำงานต่อไป




*รูปที่ 6.4 input interface*

**ส่วนแสดงผลการทำงาน**

ส่วนแสดงผลการทำงานจะแบ่งเป็น 2 ส่วนคือ ส่วนแสดงผลทางเว็บเพจซึ่งเป็นโปรแกรมจาวาแอพเพล็ต และส่วนแสดงผลที่เซิร์ฟเวอร์ซึ่งเป็น ทีเคอินเทอเฟส ซึ่งส่วนแสดงผลทางเว็บเพจจะทำการติดต่อกับตัวซูเปอร์ไวเซอร์โดยการติดต่อผ่านทางซ็อกเก็ต

ส่วนแสดงผลทางเว็บเพจ

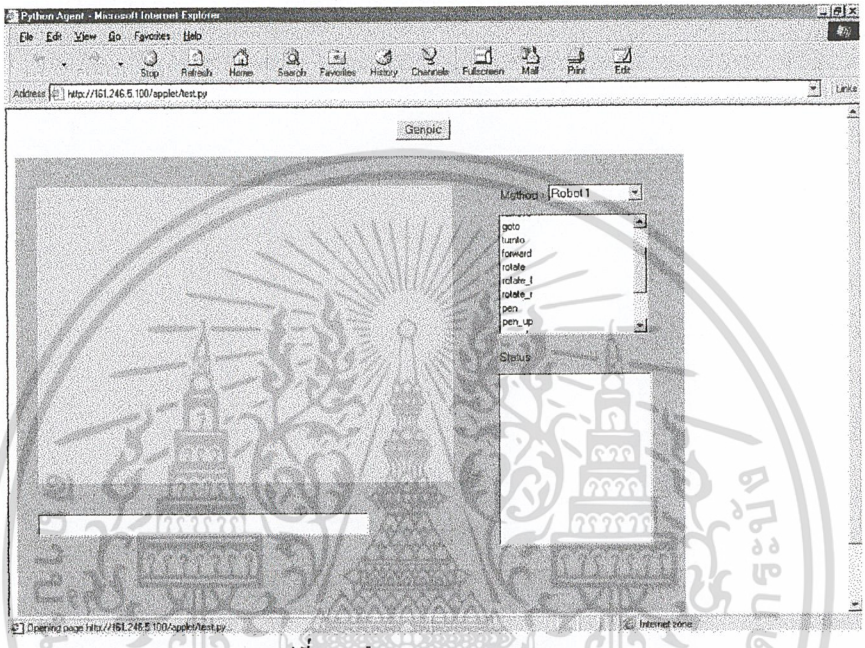
มีส่วนต่างๆของแอพเพล็ตดังนี้

- บอร์ดแสดงรูป ซึ่งจะเป็นแคนวาส (Canvas) สำหรับแสดงรูปกราฟฟิกของจาวาซึ่งจะแสดงรูปที่เอเย่นต์ทำการวาดโดยรับคำสั่งจากซูเปอร์ไวเซอร์ให้วาดภาพตามที่เอเย่นต์ได้ทำการวาดลงบนบอร์ดที่เป็นทีเคอินเทอเฟส
- ส่วนแสดงเมธอดของเอเย่นต์ ผู้ใช้งานสามารถเลือกให้แอพเพล็ตทำการแสดงเมธอดที่มีอยู่ในเอเย่นต์ที่ต้องการ โดยเลือกเอเย่นต์ภายในเมนูบาร์ซึ่งระบุตัวเอเย่นต์ และจะแสดงเมธอดของเอเย่นต์ลงในเท็กแอเรียสำหรับแสดงเมธอด
- ส่วนแสดงขั้นตอนการทำงานของเอเย่นต์ เนื่องจากการทำงานของเอเย่นต์จริงๆจะทำงานโดยแสดงผลไปยังทีเคอินเทอเฟส เมื่อนำไปแสดงบนเว็บเพจผู้ใช้งานจะไม่เห็นถึงขั้นตอนการทำงานของเอเย่นต์ที่ชัดเจน ดังนั้นจึงมีส่วนนี้เพื่อให้เอเย่นต์ส่งข้อความระบุขั้นตอนการทำงานที่ได้ทำเสร็จสิ้นไปยังซูเปอร์ไวเซอร์เพื่อส่งไปแสดงยังส่วนแสดงผลบนเว็บเพจต่อไป
- ปุ่มแสดงหน้าต่าง genpic ซึ่งจะเป็นจาวาสคริปต์ เมื่อผู้ใช้งานกดปุ่มจะทำการสร้างหน้าต่างบราวเซอร์แสดงส่วนรับอินพุตจากผู้ใช้งาน

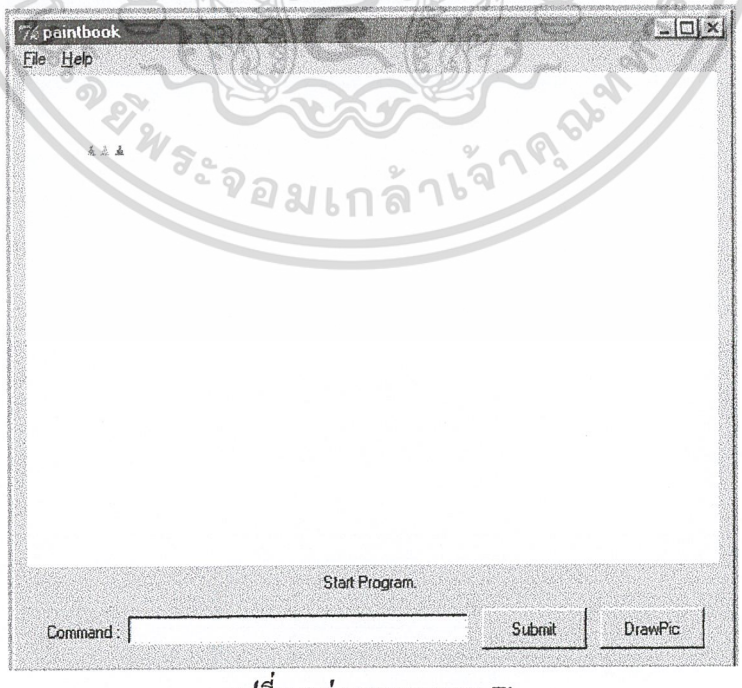
ส่วนแสดงผลที่เป็นทีเคอินเทอเฟส

มีส่วนต่างๆดังนี้

- บอร์ดแสดงผลการทำงาน เป็นแคนวาสวิดเจ็ท (Canvas Widget) ของทีเค ที่แสดงผลการทำงานของเอเย่นต์
- ส่วนแสดงขั้นตอนการทำงาน เป็นลาเบลวิดเจ็ท (Label Widget) ของทีเคแสดงผลการทำงานทีละขั้นตอนที่ตอบกลับมาจากเอเย่นต์
- ส่วนรับคำสั่ง เป็นเท็กฟิลด์วิดเจ็ท (Text Field Widget) ซึ่งสามารถรับคำสั่งจากผู้ใช้งานทีละคำสั่ง โดยสามารถอ้างถึงเอเย่นต์แต่ละตัวโดยใช้อินสแตนส์ชื่อ robot



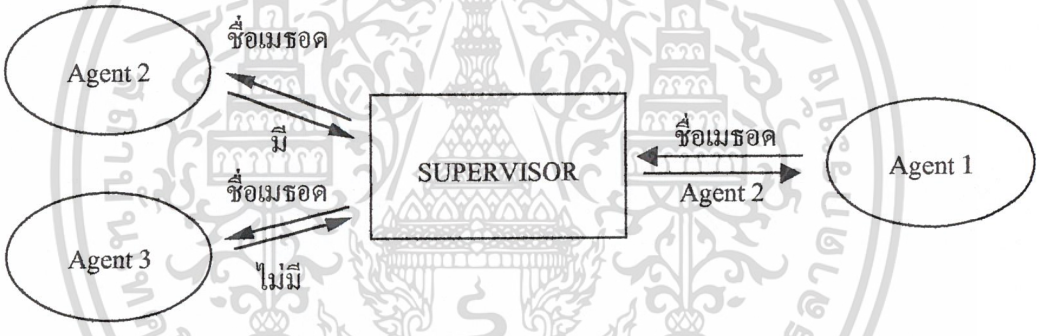

*รูปที่ 6.5 ส่วนแสดงผลบน applet*



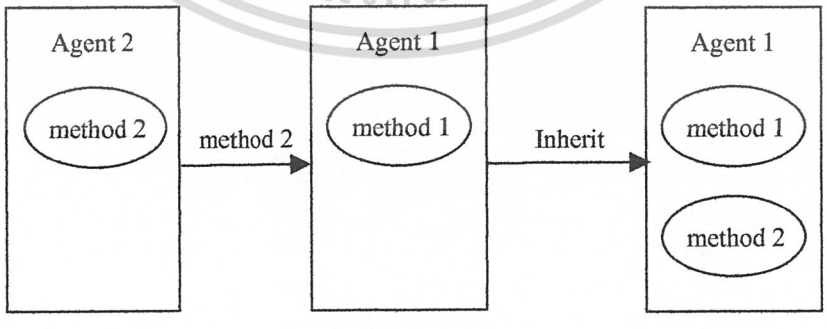

*รูปที่ 6.6 ส่วนแสดงผลบน Tk*

**การแลกเปลี่ยนกระบวนการทำงานและพัฒนากระบวนการทำงานของเอเย่นต์วาดรูป**

เมื่อเอเย่นต์ได้รับคำสั่งในการวาดรูปจากซูเปอร์ไวเซอร์ ถ้าเอเย่นต์มีความสามารถในการวาดรูปตามที่ได้รับมอบหมาย เอเย่นต์ก็จะวาดรูปที่ได้รับมอบหมายลงบนบอร์ดแสดงผลการทำงาน แต่ถ้าเอเย่นต์ไม่มีกระบวนการทำงานที่สามารถทำการวาดรูปที่กำหนดให้ได้ เอเย่นต์ตัวนั้นก็จะทำการค้นหากระบวนการทำงานจากเอเย่นต์ตัวอื่น โดยการแจ้งไปยังซูเปอร์ไวเซอร์เพื่อตรวจสอบว่ามีเอเย่นต์ตัวใดที่มีกระบวนการทำงานในการวาดรูปที่กำหนด เมื่อเอเย่นต์รู้ว่าเอเย่นต์ตัวใดมีกระบวนการทำงานที่ต้องการก็จะทำการดึงเอากระบวนการทำงานจากเอเย่นต์ที่มีการทำงานนั้น แล้วนำมาพัฒนากระบวนการทำงานของตัวเองโดยใช้หลักการวิวัฒนาการ โดยจะทำการดึงโค้ดจากเอเย่นต์ที่มีกระบวนการทำงานที่ต้องการมาบันทึกลงในโค้ดของตัวเองและทำการสร้างคลาสใหม่ที่มีการอินเฮริทมาจากคลาสเดิมของมันเอง แต่จะทำการเพิ่มกระบวนการทำงานที่ต้องการลงไป แล้วทำการสร้างอินสแตนส์ตัวใหม่ที่มีกระบวนการทำงานเพิ่มขึ้นมากกว่าเดิม โดยซูเปอร์ไวเซอร์จะทำการเปลี่ยนการอ้างถึงอินสแตนส์ของเอเย่นต์ไปเป็นตัวใหม่ที่มีวิวัฒนาการมาจากเอเย่นต์ตัวเดิม เมื่อเอเย่นต์ตัวใหม่ได้รับคำสั่งให้ทำการวาดรูปชนิดเดิมเป็นครั้งที่ 2 เอเย่นต์ก็จะสามารถทำการวาดภาพตามที่ได้รับมอบหมายได้ในทันที




*รูปที่ 6.7 แสดงการค้นหากระบวนการทำงาน*

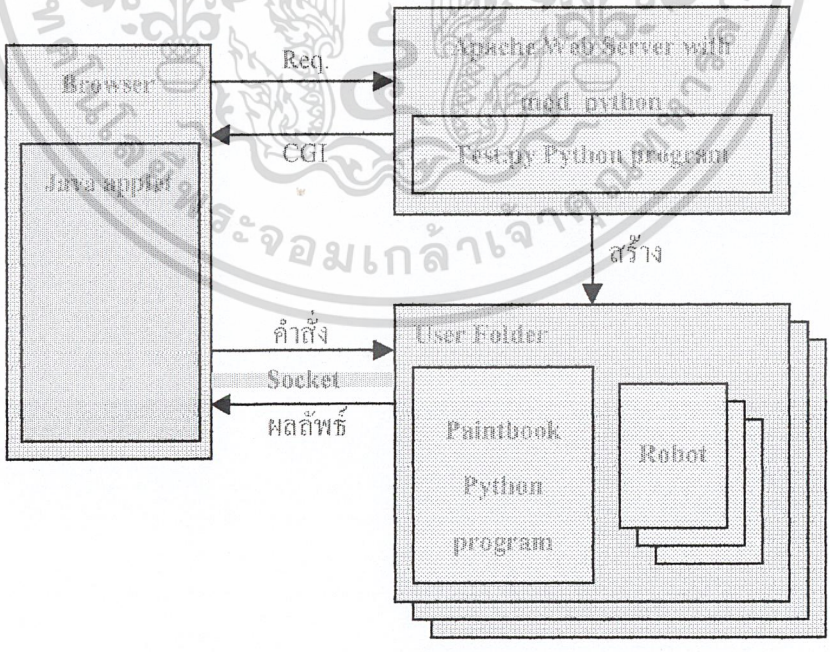



*รูปที่ 6.8 แสดงการวิวัฒนาการกระบวนการทำงาน*

**ขั้นตอนการทำงานของโปรแกรมเอเย่นต์วาดรูป**

เมื่อผู้ใช้งานโปรแกรมทำการเปิดเว็บเพจเพื่อใช้งานโปรแกรมเอเย่นต์วาดรูปบราวเซอร์จะทำการร้องขอไปยังเซิร์ฟเวอร์ซึ่งโครงงานนี้ได้ใช้ Apache เป็นเว็บเซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ได้รับการร้องขอจากบราวเซอร์ เซิร์ฟเวอร์จะทำการสั่งให้โปรแกรม test.py ซึ่งเป็น CGI ทำงาน ซึ่งจะมีกระบวนการทำงานดังนี้

1. สร้างไดเรคทอรี (Directory) ขึ้นใหม่เพื่อแยกการทำงานระหว่างแต่ละผู้ใช้งานให้มีเอเย่นต์ที่แตกต่างกัน
2. ทำการคัดลอกโค้ดที่ทำหน้าที่เป็นเอเย่นต์ ซูเปอร์ไวเซอร์ และส่วนที่ทำหน้าที่แสดงผลและรับคำสั่งจากผู้ใช้งานผ่านเว็บซึ่งเป็นโค้ดภาษาจาวาลงในไดเรคทรอรีที่ได้สร้างขึ้นมาใหม่
3. เปลี่ยนแปลงไฟล์ที่ทำหน้าที่เป็นซูเปอร์ไวเซอร์และส่วนแสดงผลการทำงานบนเว็บเพจให้มีพอร์ตที่แตกต่างจากผู้ใช้งานคนอื่นเพื่อใช้ในการติดต่อระหว่างซูเปอร์ไวเซอร์กับส่วนรับอินพุตและแสดงผลบนเว็บเพจ
4. ทำการรันโปรแกรม paintbook.py ซึ่งจะทำหน้าที่เป็นซูเปอร์ไวเซอร์
5. ทำการคอมไพล์ไฟล์ pyapplet.java ซึ่งเป็นโปรแกรมแอพเพล็ตเพื่อใช้งานเป็นส่วนแสดงผลการทำงานบนเว็บเพจ
6. ส่งเฮดเดอร์ไปยังเซิร์ฟเวอร์เพื่อส่งข้อมูลภาษาเอชทีเอ็มแอล (HTML) ไปยังบราวเซอร์ให้แสดงเว็บเพจในส่วนของการแสดงผลการทำงานของเอเย่นต์
7. เมื่อบราวเซอร์ทำการรันจาวาแอพเพล็ตจะมีการสร้างคอนเน็กชั่นจากจาวาแอพเพล็ตไปยังตัวซูเปอร์ไวเซอร์ซึ่งได้แก่โปรแกรม paintbook.py




***รูปที่ 6.9 แสดงการทำงานของโปรแกรมเอเย่นต์วาดรูป***

**6.2 เพอซันแนลแอซซิสแทนต์ (Personal Assistant)**

**แนวคิดและหลักการทำงาน**

โปรแกรมเพอซันแนลแอซซิสแทนต์เป็นโปรแกรมที่ได้นำวิธีการเปลี่ยนแปลงการทำงานแบบมีวิวัฒนาการมาประยุกต์เพื่อใช้ในการพัฒนาแอปพลิเคชั่นที่มีการทำงานบนเว็บ โดยโปรแกรมเพอซันแนลแอซซิสแทนต์จะมีลักษณะดังต่อไปนี้

- เป็นโปรแกรมเอเย่นต์ ซึ่งทำหน้าที่ช่วยในการจัดการตารางนัดหมายโดยผู้ใช้งานสามารถบันทึกและเปลี่ยนแปลงรวมทั้งการกำหนดฟิลด์ที่จะใช้ในการบันทึกรายละเอียดต่างๆได้
- โปรแกรมเพอซันแนลแอซซิสแทนต์สามารถที่จะทำการแจ้งเตือนผู้ใช้งานได้อัตโนมัติ ในกรณีที่มีการกำหนดให้แจ้งเตือนเมื่อถึงเวลาที่นัดหมาย
- เป็นโปรแกรมที่สามารถพัฒนากระบวนการทำงานได้ โดยรับการะบวนการทำงานจากผู้ใช้คนอื่นๆ
- เป็นโปรแกรมที่ใช้งานบนเว็บ

โปรแกรมเพอซันแนลแอซซิสแทนต์ประกอบด้วยส่วนประกอบสำคัญ 4 ส่วนคือ

- maincgi.py
- ทาสก์เมเนเจอร์ (Task Manager)
- พับบลิคไลบรารี (Public Library)
- ยูสเซอร์เมธอด (User Method)

โดยแต่ละส่วนมีการทำงานดังนี้

เพอซันแนลแอซซิสแทนต์เป็น CGI SCRIPT ที่ทำงานเป็นโปรแกรมหลักสำหรับทำการติดต่อกับผู้ใช้งานโปรแกรม ซึ่งจะทำการรับอินพุตจากผู้ใช้งานโปรแกรมแล้วทำการสร้างเว็บเพจเพื่อตอบสนองการสั่งงานของผู้ใช้งาน

ทาสก์เมเนเจอร์ เป็นโปรแกรมที่จัดการกับการสั่งงานในส่วนของการแจ้งเตือนที่ผู้ใช้งานต้องการให้เอเย่นต์ทำการแจ้งเตือนไปยังผู้ใช้งานในเวลาที่กำหนด

พับบลิคไลบรารี เป็นไฟล์ที่ทำการเก็บรวบรวมวิธีการทำงานของเอเย่นต์ของทุกยูสเซอร์ไว้เพื่อใช้ในการเปลี่ยนแปลงวิธีการทำงานของเอเย่นต์ที่ต้องการ

ยูสเซอร์เมธอด เป็นไฟล์ที่เก็บกระบวนการทำงานของเอเย่นต์ของยูสเซอร์ ซึ่งในแต่ละยูสเซอร์จะมีกระบวนการทำงานเฉพาะของตัวเอง โดยจะทำการเก็บไว้ภายในไดเรคทอรีของแต่ละยูสเซอร์

**ส่วนประกอบของโปรแกรมเพอซันแนลแอซซิสแทนต์**

**Maincgi.py**

เป็น CGI Script ที่ใช้สำหรับแสดงเว็บเพจตามที่ได้รับคำสั่งมาจากผู้ใช้งานโปรแกรม โดยเมื่อผู้ใช้งานโปรแกรมทำการกรอกข้อมูลหรือคลิ๊กที่จุดเชื่อมต่อจะมีการเรียกใช้งานเมธอดต่างๆภายในเมนโปรแกรมเพื่อทำการประมวลผลหากผู้ใช้งานส่งข้อมูลเข้ามาหรือแสดงเว็บเพจในส่วนที่ผู้ใช้งานต้องการ เมธอดที่มีอยู่ในสคริปต์ maincgi.py มีดังนี้

1. signup เป็นเมธอดที่ทำงานในขณะที่ผู้ใช้งานทำการสมัครสมาชิก โดยผู้ใช้งานจะต้องกรอกข้อมูลของตัวเองลงในเท็กฟิลด์บนเว็บเพจหลังจากนั้นเมธอด signup จะทำงานดังต่อไปนี้
   - ตรวจสอบชื่อและพาสเวิร์ดของผู้ใช้งานโปรแกรมว่าถูกต้องหรือไม่ โดยชื่อผู้ใช้งานจะต้องไม่ซ้ำกับชื่อที่มีอยู่แล้ว
   - ทำการเก็บชื่อผู้ใช้เพื่อตรวจสอบในการเข้าใช้งานโปรแกรม
   - สร้างไดเรคทอรีเฉพาะสำหรับผู้ใช้งานรายนั้นๆ
   - เก็บข้อมูลผู้ใช้งานลงในไฟล์ userdetail.txt ภายในไดเรคทอรีของผู้ใช้รายนั้น
   - สร้างไฟล์ task.txt และ job.txt สำหรับเก็บรายละเอียดการนัดหมายของผู้ใช้และการนัดหมายที่เอเย่นต์ต้องทำการตรวจสอบเพื่อทำการแจ้งเตือนไปยังผู้ใช้งาน
   - สร้างไฟล์ user_met.py ไว้ภายในไดเรคทอรีของผู้ใช้รายนั้นสำหรับเก็บกระบวนการทำงานของเอเย่นต์ในแต่ละยูสเซอร์
   - สร้างไฟล์ submittask.py สำหรับทำการรับรายการนัดหมายซึ่งจะมีฟิลด์ที่แตกต่างกันในผู้ใช้งานแต่ละราย โดยจะทำการสร้างเป็นไฟล์ที่แตกต่างกันตามชื่อผู้ใช้งาน
2. login เป็นเมธอดที่ใช้สำหรับตรวจสอบชื่อผู้ใช้งานและพาสเวิร์ดของผู้ใช้งานก่อนที่จะเข้าไปใช้งานโปรแกรม
3. showtask เป็นเมธอดที่ทำการแสดงรายการภายในตารางนัดหมายของผู้ใช้งานลงในเว็บเพจ โดยจะนำข้อมูลการนัดหมายมาจากไฟล์ task.txt ภายในไดเรคทอรีของผู้ใช้งานรายนั้น
4. addtask เป็นเมธอดสำหรับแสดงหน้าเว็บเพจสำหรับเพิ่มรายการนัดหมายของผู้ใช้งาน โดยจะทำการเรียกสคริปต์สำหรับเพิ่มรายการนัดหมายซึ่งได้สร้างไว้ในไดเรคทอรีของผู้ใช้งานในขั้นตอนการ login เข้าใช้งานโปรแกรม
5. removetask เป็นเมธอดสำหรับลบรายการนัดหมายของผู้ใช้งานออกจากตารางนัดหมาย โดยจะมีรายการนัดหมายของผู้ใช้งานให้เลือกเพื่อทำการลบออกจากตาราง
6. submitmodify เป็นเมธอดสำหรับให้ผู้ใช้งานสามารถเปลี่ยนแปลงข้อมูลของตัวเองได้ เมื่อผู้ใช้งานทำการแก้ไขข้อมูล เมธอดนี้จะทำการเก็บข้อมูลเดิมเช่น รายการนัดหมาย แล้วทำการลบออกทั้งหมดแล้วสร้างขึ้นมาใหม่เหมือนในขั้นตอนการ signup
7. addmethod เป็นเมธอดสำหรับทำการเปลี่ยนแปลงกระบวนการทำงานของเอเย่นต์ซึ่งก็คือกระบวนการในการแจ้งเตือนผู้ใช้งาน โดยจะทำการเพิ่มกระบวนการทำงานที่ผู้ใช้งานต้องการลงในไฟล์ user_met.py ซึ่งจะใช้หลักการเปลี่ยนกระบวนการทำงานแบบมีวิวัฒนาการ
8. removemethod เป็นเมธอดสำหรับทำการเปลี่ยนแปลงกระบวนการทำงานของเอเย่นต์ โดยจะทำการลบกระบวนการทำงานที่ผู้ใช้งานไม่ต้องการออกจากกระบวนการทำงานของเอเย่นต์
9. savetopublib เป็นเมธอดที่ทำการรับโค้ดของโปรแกรมที่ผู้ใช้งานส่งเข้ามาลงในไลบรารีรวมของทุกยูสเซอร์ เมธอดนี้จะทำการเรียกใช้โปรแกรม pubmanager โดยคำสั่ง execfile เพื่อให้ทำการจัดเก็บกระบวนการทำงานลงในไลบรารีรวม

10. delpublicmethod เป็นเมธอดสำหรับลบกระบวนการทำงานในไลบรารีรวม โดยผู้ที่ได้ทำการจัดเก็บเอาไว้เท่านั้นจึงจะสามารถทำการลบกระบวนการทำงานในไลบรารีรวมได้ โดยเมธอดนี้จะทำการเรียกใช้โปรแกรม pubmanager โดยคำสั่ง execfile เช่นเดียวกัน

เมธอดที่มีอยู่ในสคริปต์ maincgi.py เมื่อทำการประมวลผลข้อมูลจากผู้ใช้งานตามเมธอดที่มีอยู่แล้ว จะมีการส่งข้อมูลที่เป็นเอชทีเอ็มแอลไปยังบราวเซอร์เพื่อแสดงผลการทำงานและสร้างฟอร์มรับข้อมูลจากผู้ใช้งานเพื่อทำงานต่อไป อินเทอเฟสที่มีการสร้างขึ้นเพื่อติดต่อกับผู้ใช้งานโปรแกรมมีดังนี้

1. sign up page

เป็นส่วนที่ผู้ใช้งานใส่ข้อมูล เพื่อทำการสร้างโฟลเดอร์เก็บข้อมูลโดยชื่อของผู้ใช้งานแต่ละคนจะต้องไม่ซ้ำกัน

2. login page

เป็นส่วนที่ผู้ใช้งานใส่ข้อมูลชื่อผู้ใช้งานและ password เพื่อทำการ login เข้าสู่โปรแกรม

3. personal page

เป็นส่วนที่แสดงข้อมูลของผู้ใช้โดยแสดงจุดเชื่อมต่อไปยัง CGI Script ในส่วนของ task page และ method page

4. task page

เป็นส่วนที่แสดงข้อมูลเหตุการณ์และเวลาในการ นัดหมายโดยจะแสดงฟิลด์ต่างๆดังนี้

time แสดงเวลานัดหมาย

event แสดงรายละเอียดในการนัดหมาย

action แสดงกระบวนการที่เอเย่นต์ใช้ในการแจ้งเตือนผู้ใช้โปรแกรม

action time แสดงเวลาที่ต้องการ ให้แจ้งเตือนผู้ใช้งาน

นอกจากนั้นจะแสดงฟิลด์ต่างๆ ตามที่ผู้ใช้งาน เพิ่มเติม ในขั้นตอน การsign up

5. add task page

เป็นส่วนที่แสดงฟอร์มเพื่อรับข้อมูลจากผู้ใช้โปรแกรมเกี่ยวกับข้อมูลของแต่ละเหตุการณ์ ไปยังผู้ใช้งาน ซึ่งมีฟอร์มดังนี้

time เวลาของเหตุการณ์

event เหตุการณ์นัดหมาย

action กระบวนการในการแจ้งเตือน

action time เวลาในการแจ้งเตือนและฟิลด์ที่ผู้ใช้งานเพิ่มเติมเพื่อใช้เก็บข้อมูล

6. remove task page

แสดงข้อมูลในตารางนัดหมาย หรือผู้ใช้งานสามารถเลือกรายการที่ต้องการลบออกจากตารางนัดหมายได้จากคอมโบบอกซ์

7. Method Page

แสดงกระบวนการทำงานที่เอเย่นต์สามารถทำได้ในขณะนั้นหรือจะแสดงกระบวนการที่เอเย่นต์มีนอกจากนั้นจะแสดงจุดเชื่อมต่อไปยังสคริปต์ที่ทำหน้าที่ เพิ่มและลดกระบวนการภายในเอเย่นต์

8. Add Method page

แสดงกระบวนการทำงานที่ผู้ใช้สามารถนำไปใช้ได้ซึ่งประกอบไปด้วยชื่อของกระบวนการทำงาน คำอธิบายที่บอกถึงหน้าที่การทำงานของกระบวนการนั้น และวันเวลาที่กระบวนการทำงานนั้นถูกป้อนให้กับเซิร์ฟเวอร์ ผู้ใช้สามารถเลือกใช้กระบวนการทำงานต่างๆ ได้จากคอมโบบอกซ์ หลังจากเลือกแล้วกด " Add Method" เพื่อทำการเพิ่มกระบวนการทำงานนั้นลงในเอเย่นต์ของผู้ใช้ ผลลัพธ์ของการเพิ่มจะถูกแสดงในหน้าถัดไป ซึ่งจะบอกว่าสามารถเพิ่มกระบวนการนั้นได้หรือไม่ ในหน้าจอจะมีปุ่มเพื่อกลับไปยังหน้าจอของ Method page อีกครั้ง

9. Remove method page

จะถูกเลือกได้โดยการกด Remove ในหน้าจอของMethod page โดย Remove Method page นี้เป็นหน้าจอที่แสดงฟังก์ชันการถอดกระบวนการทำงานออกจากตัวเอเย่นต์ของผู้ใช้ ซึ่งประกอบไปด้วยส่วนที่แสดงกระบวนการทำงานทั้งหมดที่มีอยู่ในเอเย่นต์ของผู้ใช้ คอมโบบอกซ์เพื่อใช้ในการเลือกว่าจะถอดกระบวนการทำงานใดออกจากเอเย่นต์ของผู้ใช้ และปุ่ม Remove เพื่อใช้ในการถอดกระบวนการทำงาน เมื่อทำการถอดกระบวนการทำงานออกไปแล้ว โปรแกรมจะทำการแสดงผลว่าสามารถถอดได้หรือไม่ เช่นเดียวกับการเพิ่มกระบวนการทำงาน

10. Send Method page

เป็นหน้าที่ให้ผู้ใช้ทำการส่งกระบวนการทำงานของตนเข้ามาสู่เซิร์ฟเวอร์ได้ และยังสามารถให้ผู้อื่นนำกระบวนการนั้นไปใช้ได้ด้วย ผู้ใช้ต้องทำการป้อนข้อมูลของชื่อกระบวนการทำงาน และใส่กระบวนการทำงานโดยกระบวนการทำงานนี้มีการเขียนด้วยภาษาไพธอน และมีหลักเกณฑ์คือ

- ต้องเป็นคลาสที่มีชื่อเดียวกับชื่อกระบวนการ
- ในคลาสนั้นสามารถมีกระบวนการได้เพียง 1 กระบวนการเท่านั้นแต่จะมีคลาสย่อยอยู่กี่คลาสก็ได้ไม่จำกัด
- ชื่อกระบวนการนั้นต้องเหมือนกับชื่อที่กรอกลงในแบบฟอร์ม
- สามารถเขียนอ้างอิงถึงข้อมูลต่างๆได้ด้วย ตัวแปรระบบจะต้องเป็นตัวพิมพ์ใหญ่ทั้งหมดตัวแปรระบบทั้งหมดมีดังนี้
    1. TASK ID รหัสที่ระบุถึงข้อมูลของทาสก์ที่ผู้ใช้กำหนด
    2. USER NAME ที่ระบุถึงผู้ใช้งาน
    3. TIME ข้อมูลเวลาที่ผู้ใช้กรอก
    4. EVENT ข้อมูลที่ผู้ใช้ต้องการให้แสดง
    5. ACTION กระบวนการทำงานที่ผู้ใช้กำหนด
    6. ACTION TIME เวลาที่ผู้ใช้กำหนดให้กระบวนการนั้น
    7. EXTEND ข้อมูลที่อยู่ในฟิลด์เสริมที่ผู้ใช้กำหนด
    8. NAME ชื่อจริงของผู้ใช้
    9. E-MAIL อีเมลล์ของผู้ใช้ (อยู่ในรูปแบบของ list )

PAGER เบอร์เพจเจอร์ของผู้ใช้ (อยู่ในรูปแบบของ list)

11. MOBILE เบอร์มือถือของผู้ใช้ (อยู่ในรูปแบบของ list)

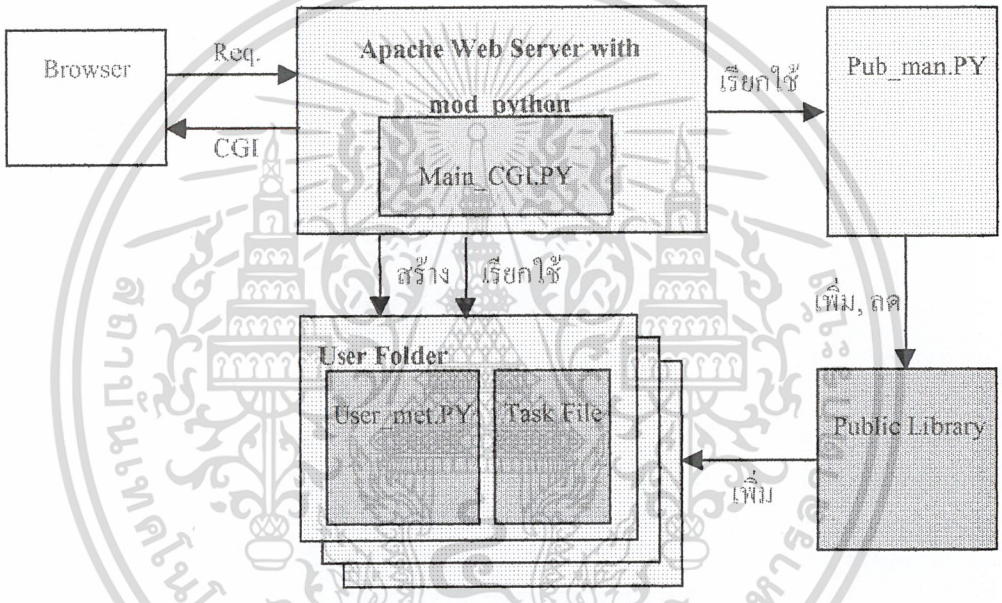
12. FIELD ชื่อฟิลด์เสริมที่ผู้ใช้เพิ่มเข้าไป (อยู่ในรูปแบบของ list)

11. Remove Public method

จะแสดงกระบวนการที่เป็นส่วนรวมซึ่งเอเย่นต์ของผู้ใช้งานสามารถรับไปใช้งานได้โดยหน้านี้จะรับเมทธอดที่ผู้ใช้ต้องการลบออกจากเมทธอดสาธารณะ โดยเลือกชื่อกระบวนการที่ต้องการลบจากคอมโบบอกซ์

12. Modify User Data page

จะแสดงแบบฟอร์มสำหรับกรอกรายละเอียดข้อมูลของผู้ใช้โปรแกรม เพื่อเปลี่ยนแปลงข้อมูลส่วนตัวของผู้ใช้งานโปรแกรม




*รูปที่ 6.10 การทำงานของสคริปต์ maincgi.py*

**ทาสก์เมเนเจอร์**

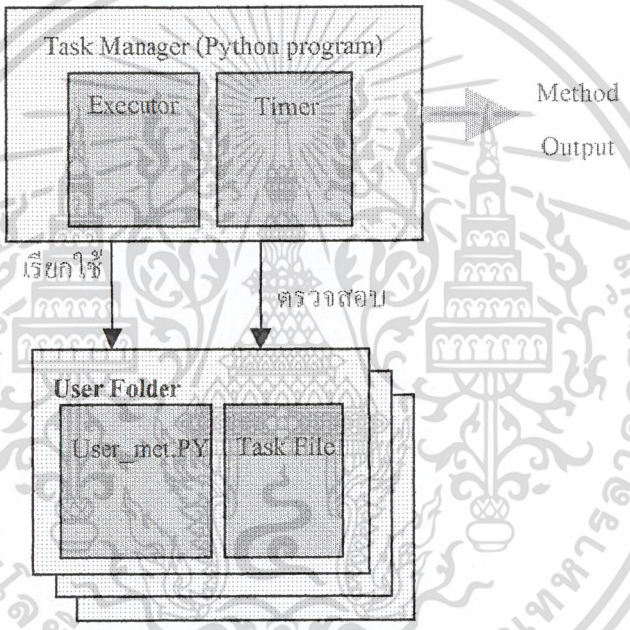
เป็นโปรแกรมสำหรับจัดการกับเหตุการณ์ในตารางนัดหมายของผู้ใช้งาน เมื่อผู้ใช้งานมีการเพิ่มรายการนัดหมายลงในตารางนัดหมายของตัวเองซึ่งถ้าผู้ใช้งานระบุให้เอเย่นต์ทำการแจ้งเตือนเมื่อถึงเวลา สคริปต์ maincgi.py จะทำการเก็บเหตุการณ์ที่ผู้ใช้ทำการเพิ่มเข้ามาลงในไฟล์ task.txt และ job.txt ภายในไดเรคทอรีของผู้ใช้งานรายนั้น หลังจากนั้นโปรแกรมในส่วนทาสก์เมเนเจอร์จะทำการตรวจสอบ และจัดการกับเหตุการณ์เมื่อถึงเวลาที่ผู้ใช้งานกำหนด โปรแกรมทาสก์เมเนเจอร์สามารถแบ่งเป็นส่วนย่อยๆ ได้ดังนี้

1. ทาสก์เมเนเจอร์ เป็นเธรดที่ทำงานตลอดเวลา โดยจะทำการตรวจสอบเวลาของปัจจุบันว่าตรงกับเหตุการณ์ที่รับมาจากผู้ใช้งานคนใดบ้างซึ่งจะทำการเก็บไว้ในตัวแปรแวดล้อม task_list ถ้ามีเหตุการณ์ใดที่มีเวลาตรงกับเวลาปัจจุบัน โปรแกรมทาสก์เมเนเจอร์จะทำการสร้างเธรด task_handler ขึ้น

มาเพื่อทำงานให้กับเหตุการณ์นั้นๆ หลังจากนั้นจะทำการลบเหตุการณ์ที่ได้ทำเสร็จเรียบร้อยแล้วออกจาก task_list

2. ทาสก์เอเย่นต์ เป็นเธรดที่ทำงานตลอดเวลาเพื่อทำการตรวจสอบเหตุการณ์ที่ผู้ใช้งานทำการเก็บลงในไฟล์ job.txt ภายในไดเรคทอรีของผู้ใช้งานโปรแกรมทุกคน ซึ่งเมื่อผู้ใช้งานมีการเพิ่มเหตุการณ์ลงในตารางนัดหมายและต้องการให้เอเย่นต์ทำการแจ้งเตือน ทาสก์เอเย่นต์จะทำการเพิ่มเหตุการณ์ที่มีการเพิ่มขึ้นมาใหม่ลงในตัวแปรแวดล้อม task_list

ในการทำงานของทาสก์เมเนเจอร์ เมื่อมีการสร้างเธรดสำหรับจัดการกับเหตุการณ์ที่ต้องมีการแจ้งเตือนผู้ใช้งานโปรแกรม เธรดที่สร้างขึ้นมาจะมีการนำกระบวนการทำงานที่เป็นของผู้ใช้ที่เป็นเจ้าของรายการนัดหมายที่มีการแจ้งเตือนมาใช้งาน โดยคำสั่ง execfile เพื่อสร้างเอเย่นต์ที่มีกระบวนการทำงานของผู้ใช้รายนั้น และทำการแจ้งเตือนผู้ใช้ด้วยกระบวนการที่ผู้ใช้งานต้องการ




*รูปที่ 6.11 การทำงานของทาสก์แมนเจอร์*

**พับบลิคไลบรารี**

เป็นไฟล์ที่เก็บกระบวนการทำงานของผู้ใช้งานทุกคนเอาไว้ ซึ่งถ้าผู้ใช้งานคนใดต้องการเพิ่มกระบวนการทำงานให้กับเอเย่นต์ของตัวเองสามารถค้นหากระบวนการทำงานที่ต้องการได้ในพับบลิคไลบรารี นอกจากนั้นผู้ใช้สามารถทำการเพิ่มกระบวนการทำงานลงในพับบลิคไลบรารีได้ โดยจะต้องมีรูปแบบของกระบวนการดังนี้

- ในคลาสนั้นสามารถมีกระบานการได้เพียง 1 กระบวนการเท่านั้นแต่จะมีคลาสย่อยอยู่กี่คลาสก็ได้ไม่จำกัด
- สามารถเขียนอ้างอิงถึงข้อมูลต่างๆได้ด้วย ตัวแปรแวดล้อมของระบบจะต้องเป็นตัวพิมพ์ใหญ่ทั้งหมด ตัวแปรระบบที่สามารถนำไปใช้ได้ทันทีมีดังนี้

TASK ID รหัสที่ระบุถึงข้อมูลของทาสก์ที่ผู้ใช้กำหนด

2. USER NAME ที่ระบุถึงผู้ใช้งาน
3. TIME ข้อมูลเวลาที่ผู้ใช้กรอก
4. EVENT ข้อมูลที่ผู้ใช้ต้องการให้แสดง
5. ACTION กระบวนการทำงานที่ผู้ใช้กำหนด
6. ACTION TIME เวลาที่ผู้ใช้กำหนดให้กระบวนการนั้น
7. EXTEND ข้อมูลที่อยู่ใน filed เสริมที่ผู้ใช้กำหนด
8. NAME ชื่อจริงของผู้ใช้
9. E-MAIL อีเมลล์ของผู้ใช้ (อยู่ในรูปแบบของ list )
10. PAGER เบอร์เพจเจอร์ของผู้ใช้ (อยู่ในรูปแบบของ list )
11. MOBILE เบอร์มือถือของผู้ใช้ (อยู่ในรูปแบบของ list )
12. FIELD ชื่อฟิลด์เสริมที่ผู้ใช้เพิ่มเข้าไป (อยู่ในรูปแบบของ list )

- ต้องมีคำอธิบายกระบวนการทำงาน (Comment) เพื่อให้ผู้อื่นสามารถเข้าใจการทำงานของกระบวนการนั้นได้

**ยูสเซอร์เมธอด**

เป็นไฟล์ที่ทำการเก็บกระบวนการทำงานของเอเย่นต์อยู่ในไฟล์ user_met.py ซึ่งแต่ละยูสเซอร์จะมีกระบวนการทำงานเฉพาะของตัวเอง โดยจะเป็นคลาสที่มีกระบวนการทำงานต่างๆอยู่ภายใน ชื่อของคลาสจะเป็นคำว่า user ต่อด้วยเวอร์ชั่นที่มีการเปลี่ยนแปลงกระบวนการทำงานแบบมีวิวัฒนาการมา ภายในคลาสจะมีตัวแปรแวดล้อมคือ ชื่อผู้ใช้งาน เวอร์ชั่น ไดเรคทอรีที่ทำการจัดเก็บกระบวนการทำงาน และกระบวนการทำงานที่มีอยู่ภายในคลาส เมื่อมีการ signup ผู้ใช้ใหม่ขึ้นจะมีการสร้างไฟล์ user_met.py ไว้ภายในไดเรคทอรีของผู้ใช้รายนั้น ซึ่งจะมีกระบวนการทำงานมาตรฐานของเอเย่นต์ดังนี้

- __find_methods เป็นกระบวนการทำงานสำหรับค้นหาชื่อกระบวนการภายในคลาสที่เก็บกระบวนการทำงานของเอเย่นต์ไว้
- add_method เป็นกระบวนการทำงานสำหรับทำการเพิ่มกระบวนการทำงานให้กับเอเย่นต์ ซึ่งใช้หลักการวิวัฒนาการ โดยจะทำการสร้างคลาสใหม่ที่มีการอินเฮริทมาจากคลาสกระบวนการทำงานตัวเก่าแต่จะทำการเพิ่มกระบวนการทำงานใหม่ลงไป
- revise_method เป็นกระบวนการทำงานสำหรับทำการเปลี่ยนแปลงการทำงานของกระบวนการทำงานที่มีอยู่แล้วในเอเย่นต์โดยจะทำการ override กระบวนการทำงานที่มีอยู่แล้วด้วยกระบวนการทำงานใหม่
- remove_method เป็นกระบวนการทำงานสำหรับทำการลบกระบวนการทำงานของเอเย่นต์ โดยจะทำการ override กระบวนการทำงานเดิมด้วยคำสั่ง pass ทำให้เมื่อเรียกใช้งานกระบวนการทำงานที่ถูกลบออกไปแล้วจะไม่มีการทำงานเกิดขึ้น

นอกจากส่วนประกอบเหล่านี้โครงงานนี้ได้ทำการสร้างกระบวนการทำงานตัวอย่างของเอเย่นต์ที่ใช้งานในการแจ้งเตือนผู้ใช้งานเมื่อถึงเวลาขึ้นมา 2 ตัวอย่างคือ notify_email และ notify_gsm โดยเก็บไว้ในพับบลิคไลบรารี

**notify_email**

เป็นกระบวนการทำงานตัวอย่างที่ใช้สำหรับส่งข้อความเตือนผู้ใช้งาน โดยเมื่อถึงเวลาที่ผู้ใช้งานต้องการให้ทำการแจ้งเตือนเอเย่นต์จะทำการเขียนอีเมล์ที่มีรายละเอียดของรายการในตารางนัดหมายที่ผู้ใช้งานต้องการให้เตือนส่งไปยังอีเมล์แอสเดรสของผู้ใช้งาน ซึ่งกระบวนการทำงานนี้จะใช้โปรโตคอลเอชเอ็มทีพี (SMTP) ในการส่งอีเมล์

**notify_gsm**

เป็นกระบวนการทำงานตัวอย่างที่ใช้สำหรับส่งข้อความเตือนผู้ใช้งาน โดยเมื่อถึงเวลาที่ผู้ใช้งานต้องการให้ทำการแจ้งเตือน เอเย่นต์จะทำการส่งข้อความรายการนัดหมายไปยังโทรศัพท์มือถือที่เป็น GSM โดยการส่งข้อความจะทำการส่งผ่านเซิร์ฟเวอร์ http://202.183.251.250/cgi-bin/web2sms.cgi โดยการส่งข้อความร้องขอไปยัง CGI ของเว็บให้ทำการส่งข้อความเข้าไปยังโทรศัพท์มือถือของผู้ใช้งานที่ต้องการให้แจ้งเตือน

**การพัฒนากระบวนการทำงานของโปรแกรมเพอซันแนลแอซซิสแทนต์**

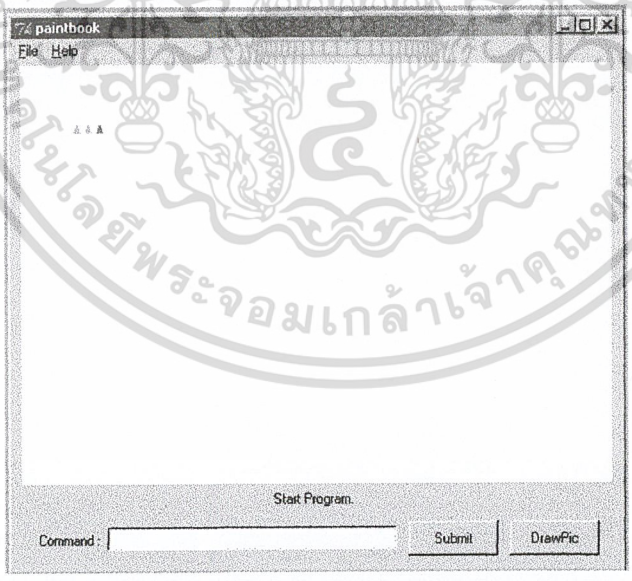
โปรแกรมเพอซันแนลแอซซิสแทนต์จะมีการพัฒนากระบวนการทำงานของเอเย่นต์ที่ใช้ในการแจ้งเตือนผู้ใช้งานโปรแกรม ผู้ใช้งานโปรแกรมสามารถเพิ่มกระบวนการทำงานใหม่ๆลงไปในพับบลิคไลบรารี ซึ่งผู้ใช้รายอื่นๆสามารถทำการเพิ่มกระบวนการทำงานของตนโดยการเลือกกระบวนการทำงานจากพับบลิคไลบรารี เมื่อผู้ใช้ทำการเลือกกระบวนการทำงานแล้วเอเย่นต์จะทำการเพิ่มกระบวนการทำงานลงในไฟล์ user_met.py โดยการเพิ่มกระบวนการทำงานตามหลักการวิวัฒนาการคือจะทำการสร้างคลาสใหม่ขึ้นมาโดยมีการอินเฮริทมาจากคลาสกระบวนการทำงานตัวเก่าและมีการเพิ่มกระบวนการทำงานใหม่ลงไป เมื่อมีการเรียกใช้งานเอเย่นต์ตัวเดิมจะมีการอ้างอิงถึงอินสแตนส์ตัวใหม่ที่มีการเพิ่มกระบวนการทำงานเรียบร้อยแล้ว

# บทที่ 7
# การสาธิตการทำงานของโปรแกรม

## 7.1 การทำงานของโปรแกรมเอเย่นต์วาดรูป

การทำงานของโปรแกรมเอเย่นต์วาดรูปจะเริ่มจากการที่ผู้ใช้งานโปรแกรมทำการเปิดเว็บเพจที่ทำการเรียกใช้งานโปรแกรม หลังจากนั้นบราวเซอร์จะทำการร้องขอไปยังเซิร์ฟเวอร์เพื่อขอใช้งานสคริปต์ test.py เมื่อเซิร์ฟเวอร์ทำการเรียกสคริปต์ test.py ให้ทำงานจะมีการทำงานดังต่อไปนี้

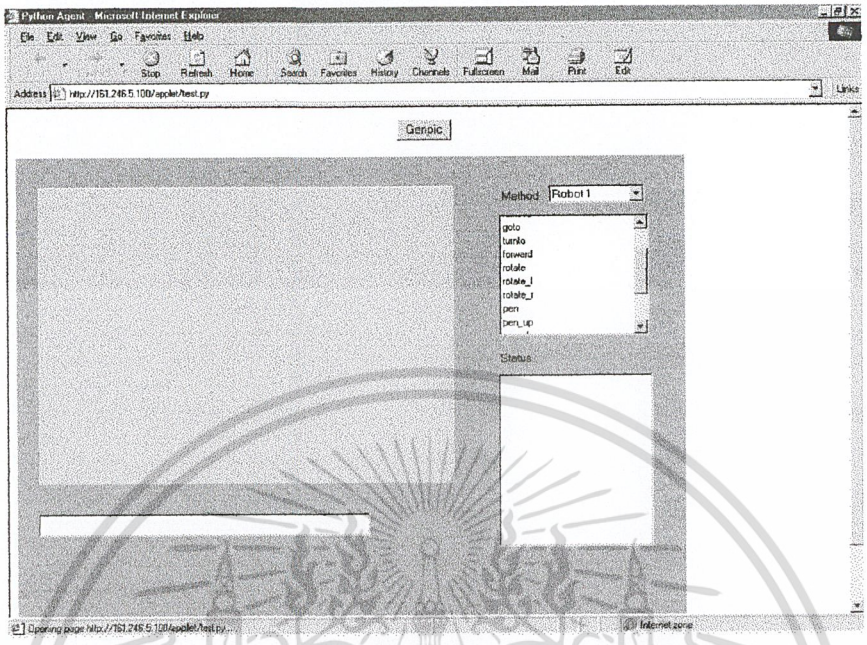
1. โปรแกรมจะทำการสร้างไดเรคทอรีสำหรับผู้ใช้งานโปรแกรมรายนั้น เสร็จแล้วทำการคัดลอกโปรแกรมที่ต้องใช้งานลงในไดเรคทอรีที่ได้สร้างขึ้นมา ซึ่งได้แก่ไฟล์
   - paintbook.py ใช้สำหรับทำงานในส่วนที่เป็นซูเปอร์ไวเซอร์
   - robot1.txt , robot2.txt , robot3.txt ใช้สำหรับทำงานในส่วนที่เป็นเอเย่นต์
   - pyapplet.java ใช้สำหรับทำงานในส่วนที่เป็นอินเทอเฟสติดต่อผู้ใช้งานบนเว็บซึ่งเป็นโปรแกรมจาวาแอพเพล็ตเพื่อแสดงผลการทำงานของเอเย่นต์
   - genpic.java ใช้สำหรับทำงานในส่วนที่ทำหน้าที่รับคำสั่งจากผู้ใช้งานโปรแกรม
   - genpic.html ใช้เพื่อแสดงหน้าต่างบราวเซอร์รับข้อมูลที่เป็นจาวาแอพเพล็ต
2. โปรแกรม test.py ทำการรัน โปรแกรม paintbook.py ซึ่งจะมีอินเทอเฟสที่เป็นทีเค ดังรูปที่ 7.1




*รูปที่ 7.1 ทีเคอินเทอเฟสของโปรแกรม paintbook.py*

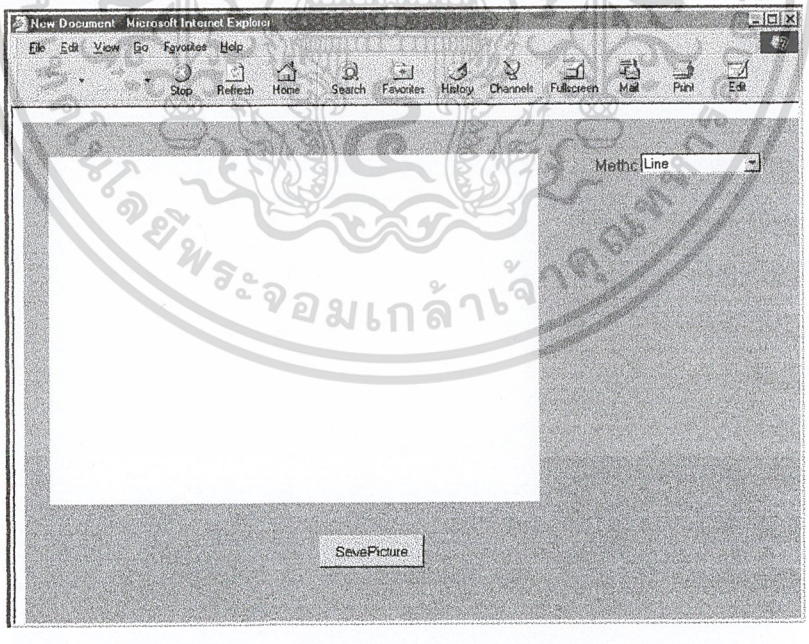
3. โปรแกรม test.py ทำการคอมไพล์โปรแกรม pyapplet.java โดยใช้คำสั่ง os.system เรียกใช้โปรแกรม javac.exe ซึ่งจะได้แอพเพล็ตที่เป็นส่วนแสดงผลการทำงานของเอเย่นต์
4. โปรแกรม test.py ทำการคอมไพล์โปรแกรม genpic.py ซึ่งจะได้แอพเพล็ตที่เป็นส่วนรับคำสั่งจากผู้ใช้งานโปรแกรม

5. ส่งโค้ดเอชทีเอ็มแอลไปยังบราวเซอร์เพื่อสั่งให้บราวเซอร์รันแอพเพล็ต pyapplet.class จะได้ส่วนแสดงผลการทำงานดังรูปที่ 7.2 ซึ่งประกอบด้วยบอร์ดแสดงรูปที่ได้จากการวาด ส่วนแสดงชื่อกระบวนการทำงานของเอเย่นต์ และส่วนแสดงสถานะการทำงาน



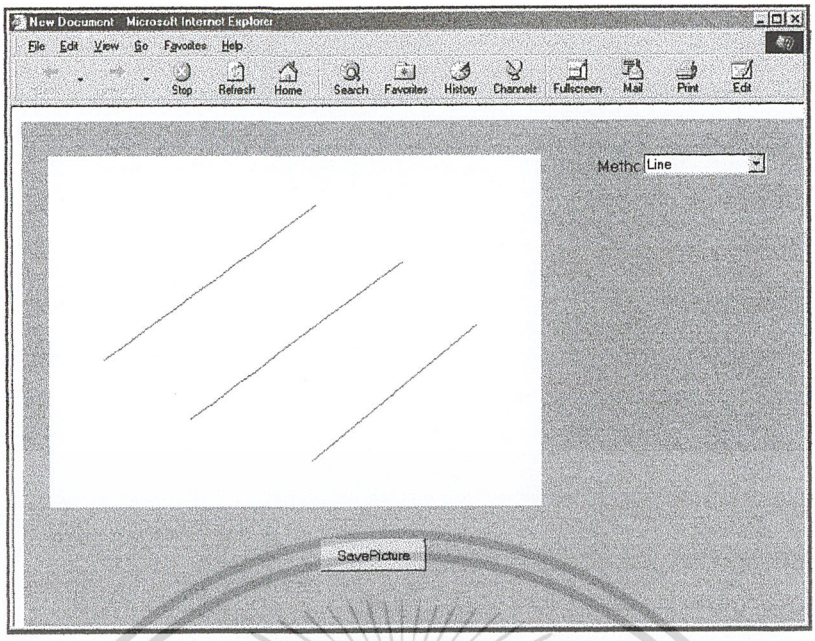

*รูปที่ 7.2 ส่วนแสดงผลการทำงานของเอเย่นต์บนเว็บ*

6. เมื่อผู้ใช้งานต้องการสั่งให้เอเย่นต์ทำการวาดรูป ผู้ใช้งานต้องเปิดหน้าต่างของส่วนรับคำสั่งโดยการคลิกที่ปุ่ม genpic ซึ่งจะได้หน้าต่างดังแสดงในรูปที่ 7.3



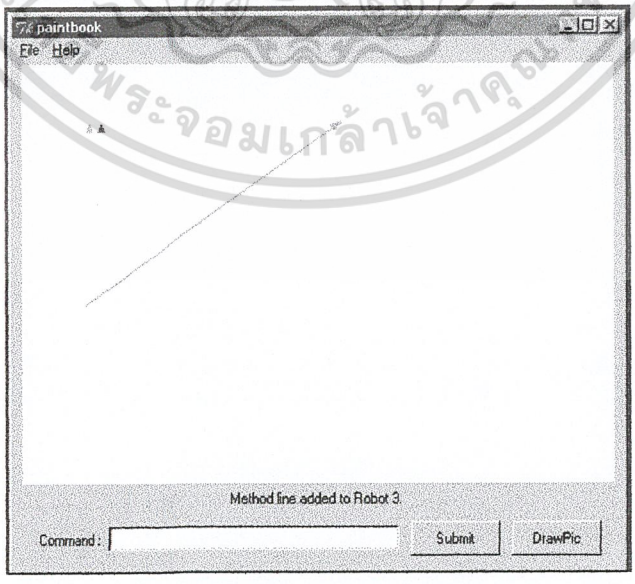

*รูปที่ 7.3 หน้าต่างรับคำสั่งจากผู้ใช้งานโปรแกรม*

7. หลังจากนั้นเราสามารถวาดรูปเพื่อให้เอเย่นต์นำไปวาดได้ โดยเราจะทำการวาดเส้นตรง 3 เส้นดังแสดงในรูป 7.4 แล้วคลิกที่ปุ่ม savepicture

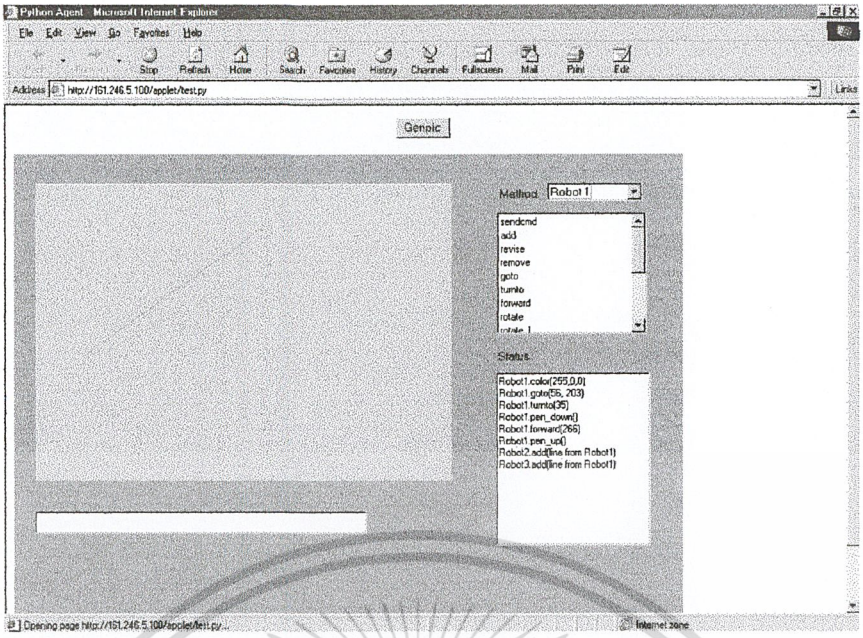



*รูปที่ 7.4 แสดงการใส่คำสั่งวาดเส้นตรง*

8. เอเย่นต์จะทำการวาดรูปเส้นตรงลงบนบอร์ดทั้งที่เป็นทีเคอินเทอเฟส และส่วนที่เป็นแอพเพล็ต ดังแสดงในรูปที่ 7.5 และ 7.6 โดยเอเย่นต์จะส่งสถานะการทำงานของเอเย่นต์ไปยังส่วนที่เป็นแอพเพล็ตซึ่งจะแสดงอยู่ในช่องสถานะ จะเห็นว่ารูปที่วาดมีเส้นตรงอยู่เพียงเส้นเดียว เนื่องจากตอนนี้มีเพียงเอเย่นต์ตัวแรกตัวเดียวที่มีกระบวนการทำงานในการวาดเส้นตรงดังนั้นเอเย่นต์ตัวอื่นจะไม่ทำการวาดเส้นตรงแต่จะทำการเปลี่ยนแปลงกระบวนการทำงานของตัวเองโดยนำกระบวนการทำงานจากเอเย่นต์ตัวที่ 1 มาทำการเพิ่มให้กับตัวเอง ในช่องสถานะจะมีข้อความบอกว่าอเย่นต์ทำการเพิ่มกระบวนการทำงานจากเอเย่นต์ตัวแรกเรียบร้อยแล้ว

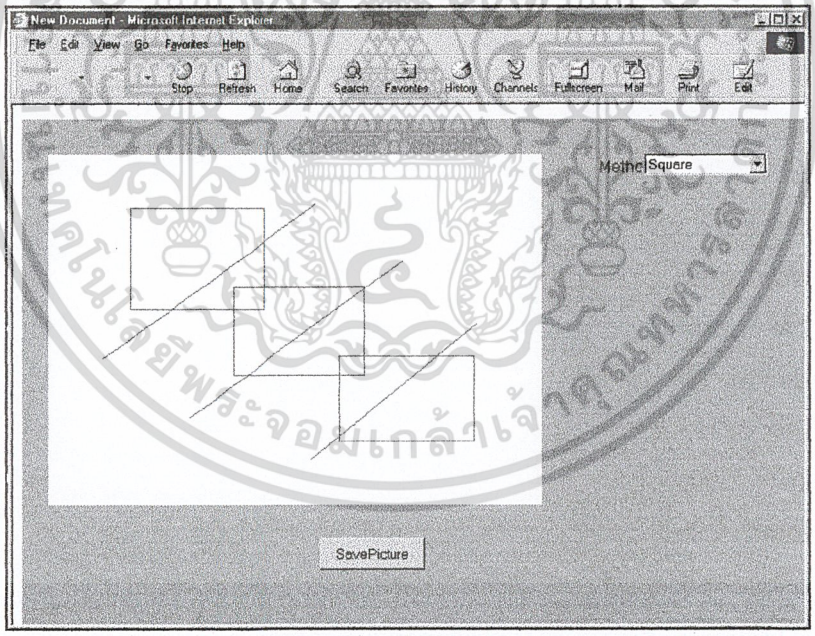



*รูปที่ 7.5 ผลการวาดเส้นตรงบน Tk*




*รูปที่ 7.6 ผลการวาดเส้นตรงบนแอพเพล็ต*

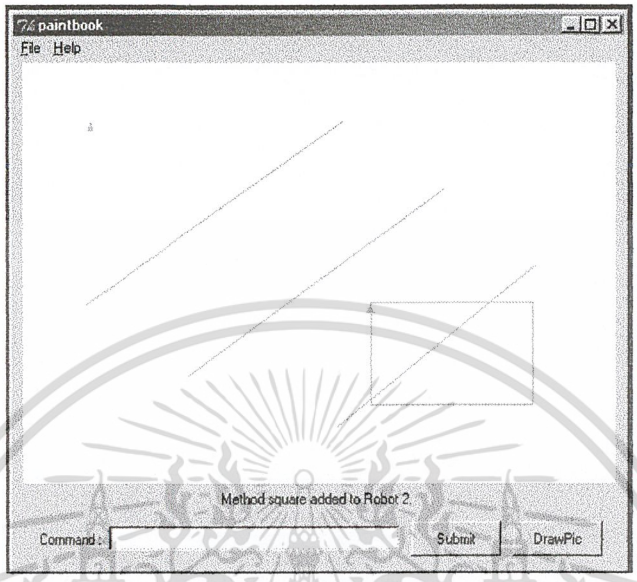
9. หลังจากนั้นเราจะทำการใส่คำสั่งให้เอเย่นต์ทำการวาดรูปเส้นตรงพร้อมทั้งรูปสี่เหลี่ยม 3 รูป ดังแสดงในรูปที่ 7.7



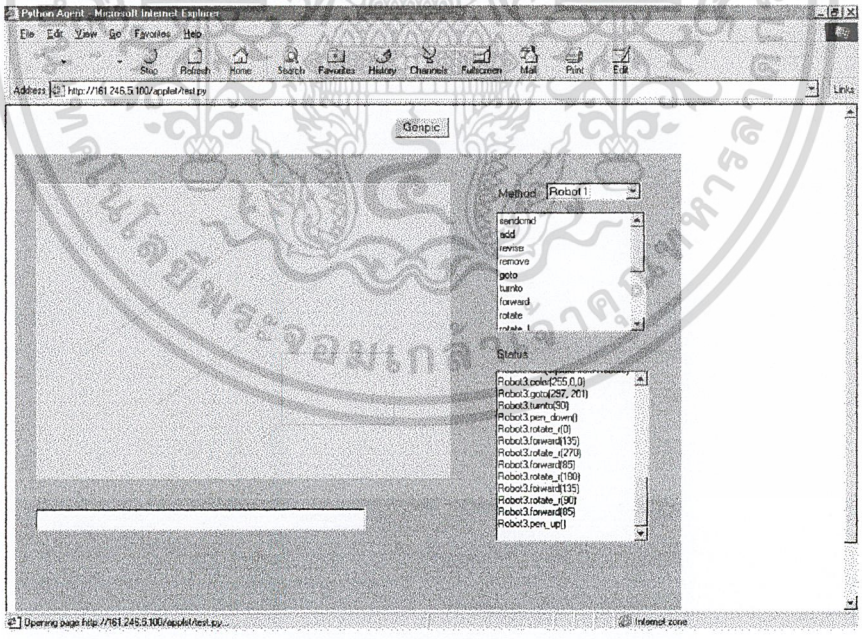

*รูปที่ 7.7 แสดงการใส่คำสั่งวาดรูปเส้นตรงและสี่เหลี่ยม*

10. เอเย่นต์จะทำการวาดรูปเส้นตรง 3 รูป และรูปสี่เหลี่ยม 1 รูปลงบนบอร์ดทั้งที่เป็นทีเคอินเทอเฟสและส่วนที่เป็นแอพเพล็ต ดังแสดงในรูปที่ 7.8 และ 7.9 จะเห็นว่ารูปที่วาดมีรูปเส้นตรงอยู่ครบเนื่องจากเอเย่นต์ได้ทำการเปลี่ยนแปลงกระบวนการทำงานโดยใช้หลักวิวัฒนาการทำการเพิ่มกระบวนการทำงานในการวาดเส้นตรงเรียบร้อยแล้ว ทำให้เอเย่นต์ทุกตัวสามารถวาดรูปเส้นตรงได้ แต่มีเพียง

เอเย่นต์ตัวที่ 3 เท่านั้นที่มีความสามารถในการวาดรูปสี่เหลี่ยม ดังนั้นรูปที่วาดออกมาจึงมีรูปสี่เหลี่ยมอยู่เพียงรูปเดียวซึ่งเกิดจากการวาดของเอเย่นต์ตัวที่ 3 ส่วนเอเย่นต์ตัวอื่นๆ จะทำการเปลี่ยนแปลงกระบวนการทำงานโดยการเพิ่มกระบวนการวาดรูปสี่เหลี่ยมลงในกระบวนการใหม่ที่ได้ทำการเปลี่ยนแปลงแล้ว

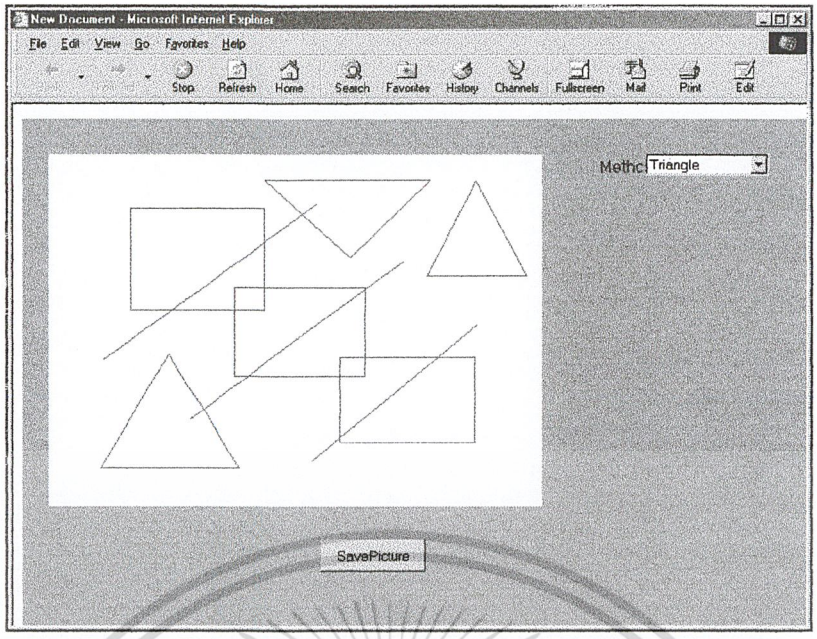



*รูปที่ 7.8 ผลการวาดรูปสี่เหลี่ยมบน Tk*




*รูปที่ 7.9 ผลการวาดรูปสี่เหลี่ยมบนแอพเพล็ต*

11. เมื่อทำการสั่งงานเอเย่นต์ให้ทำการวาดเช่นนี้เรื่อยๆ จะทำให้เอเย่นต์ทุกตัวมีกระบวนการทำงานที่เหมือนกันและสามารถทำการวาดรูปได้ตามที่ผู้ใช้งานกำหนดทุกรูปคือ เส้นตรง รูปสามเหลี่ยม และรูปสี่เหลี่ยม ดังแสดงในรูปที่ 7.10

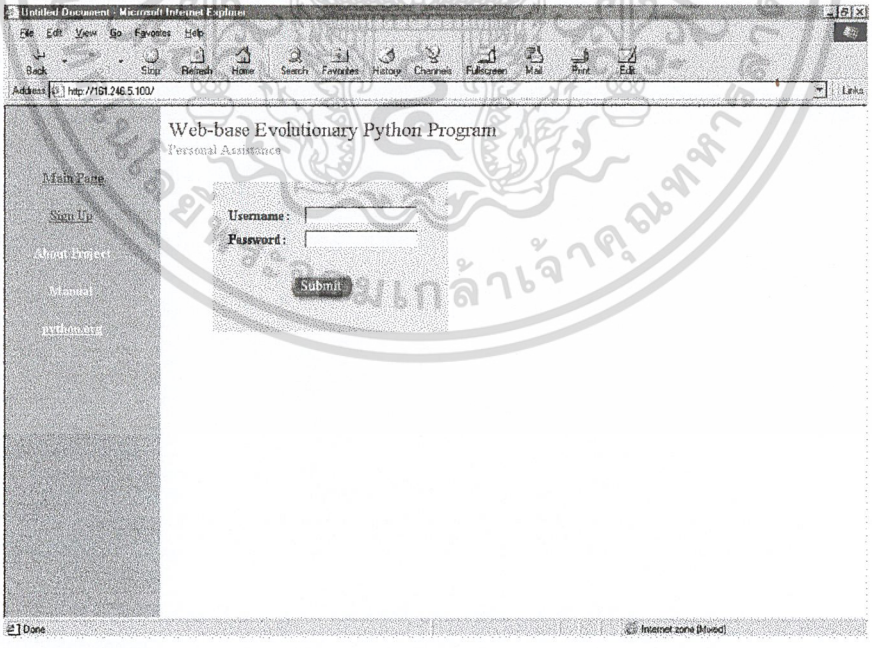



*รูปที่ 7.10 ผลจากการทำงานเมื่อเอเย่นต์มีการวิวัฒนาการครบทุกกระบวนการทำงาน*

**7.2 การทำงานของโปรแกรมเพอซันแนลแอซซิสแทนต์**

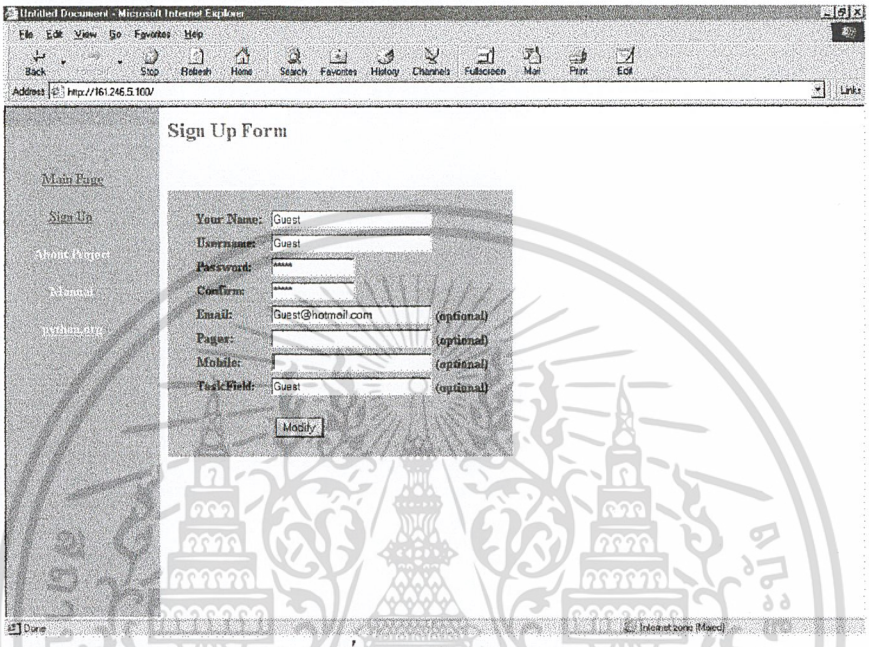
เมื่อผู้ใช้งานโปรแกรมทำการเปิดเว็บเพจหลักของโปรแกรม บราวเซอร์จะทำการร้องขอเซิร์ฟเวอร์เพื่อเรียกสคริปต์ maincgi.py ให้ทำงานซึ่งมีการทำงานดังนี้

1. โปรแกรมแสดงหน้าต่าง login ดังแสดงในรูปที่ 7.11



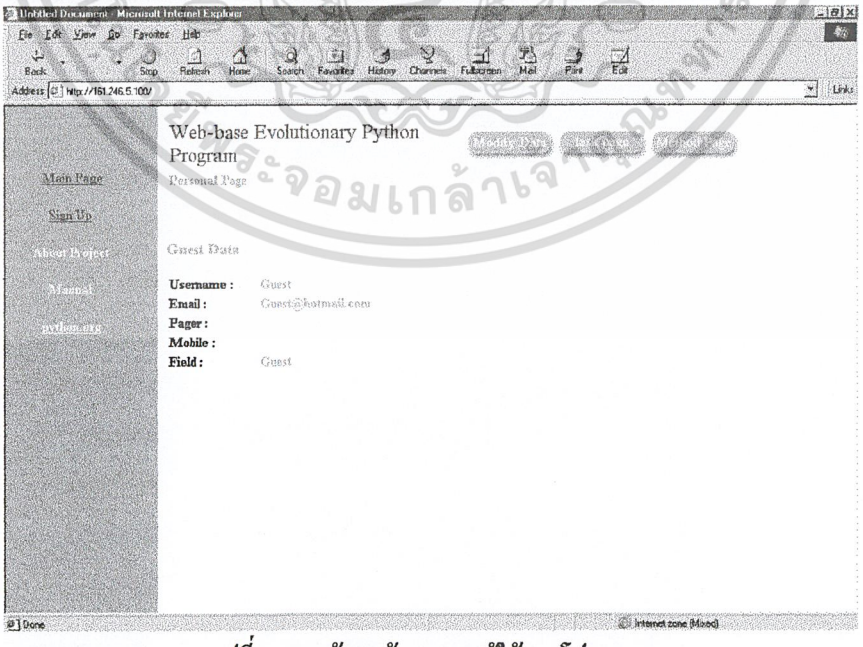

*รูปที่ 7.11 หน้าจอ login*

2. ผู้ใช้งานทำการกรอกข้อมูลของตนเองลงในหน้า signup ดังแสดงในรูปที่ 7.12 ซึ่งมีข้อมูลที่ต้องกรอก คือ ชื่อผู้ใช้งาน และ พาสเวิร์ด นอกจากนั้นยังใส่ข้อมูลอื่นๆ เช่น อีเมล์ เบอร์เพจ และเบอร์โทรศัพท์มือถือเพื่อใช้ในการแจ้งเตือน นอกจากนั้นยังสามารถระบุฟิลด์เพิ่มเติมสำหรับเก็บรายละเอียดของการนัดหมายของผู้ใช้งานได้ เมื่อผู้ใช้งานทำการกรอกข้อมูลและทำการยืนยันเรียบร้อยโปรแกรมจะทำการสร้างไดเรคทอรีเพื่อเก็บข้อมูลของผู้ใช้งาน รวมทั้งโค้ดในส่วนของผู้ใช้งาน เช่นกระบวนการทำงานของเอเย่นต์ลงในไดเรคทอรีของผู้ใช้งานรายนั้น



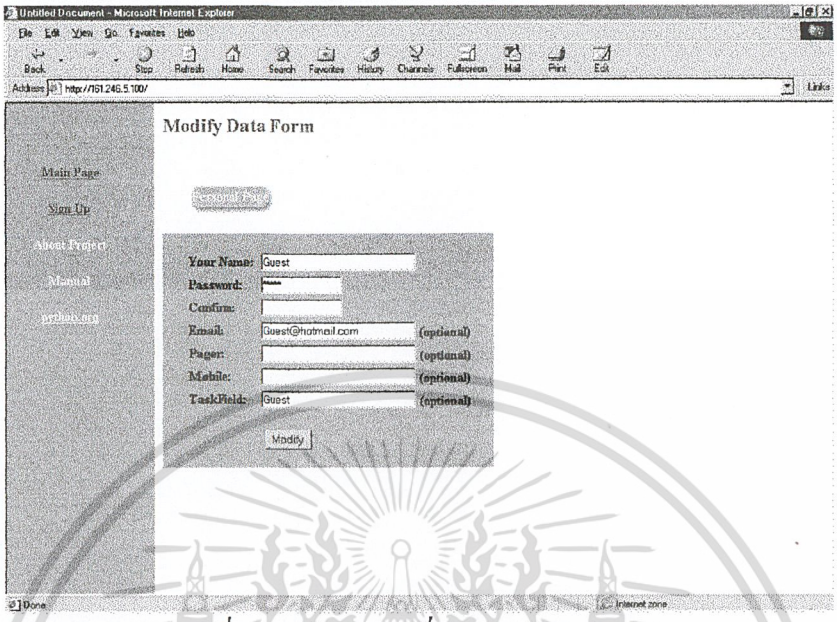

*รูปที่ 7.12 หน้าจอ signup*

3. หลังจากที่ผู้ใช้งานโปรแกรมทำการ login เข้าใช้งานโปรแกรม จะแสดงหน้าจอรายละเอียดของผู้ใช้งานและจุดเชื่อมต่อไปยังส่วนอื่นๆของโปรแกรม ดังแสดงในรูปที่ 7.13



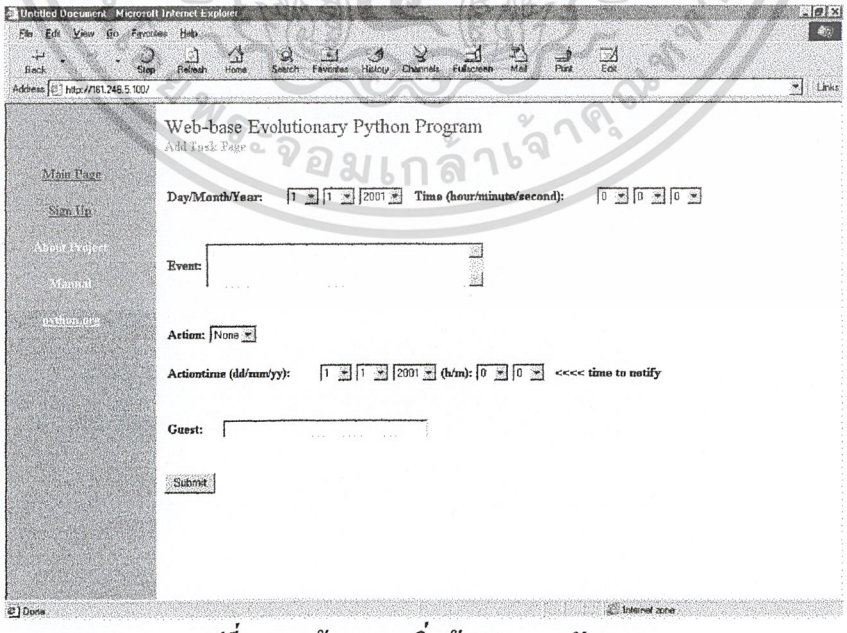

*รูปที่ 7.13 หน้าจอข้อมูลของผู้ใช้งานโปรแกรม*

4. ถ้าผู้ใช้งานโปรแกรมต้องการเปลี่ยนแปลงข้อมูลส่วนตัวสามารถทำได้โดยการแก้ไขในหน้าจอการเปลี่ยนแปลงข้อมูลส่วนตัวดังแสดงในรูปที่ 7.14 ซึ่งโปรแกรมจะทำการแก้ไขข้อมูลของผู้ใช้งานโปรแกรม โดยการทำงานเช่นเดียวกับขั้นตอนการ signup



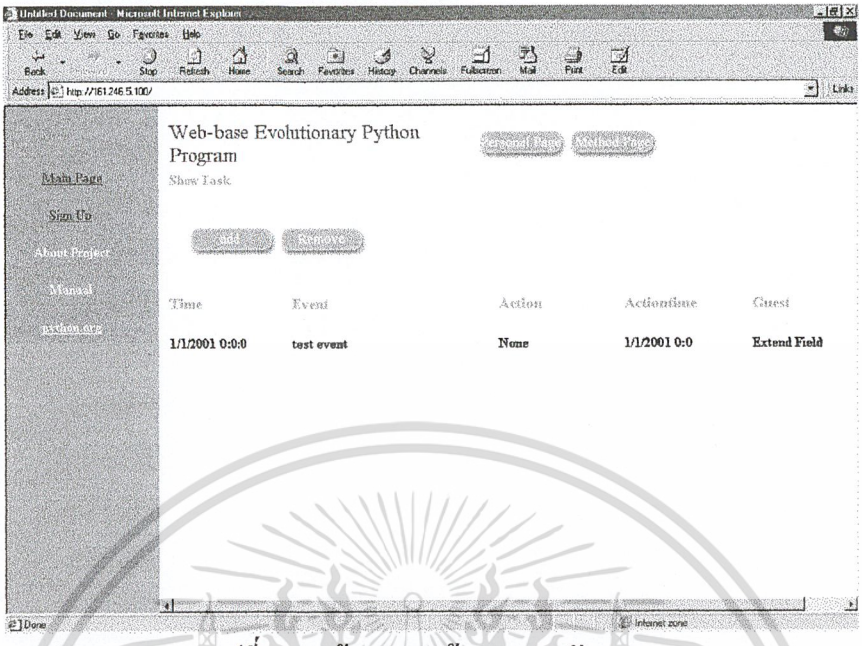

*รูปที่ 7.14 หน้าจอการเปลี่ยนแปลงข้อมูลส่วนตัว*

5. ผู้ใช้งานโปรแกรมสามารถเพิ่มรายการนัดหมายลงในตารางนัดหมายได้ โดยการกรอกรายละเอียดของเหตุการณ์นัดหมายลงในหน้าการเพิ่มข้อมูลตารางนัดหมายที่มีรายละเอียดคือ เวลาในการนัดหมาย รายละเอียดของการนัดหมาย กระบวนการในการแจ้งเตือนของเอเย่นต์ที่ต้องการใช้ในการแจ้งเตือน เวลาที่ต้องการให้แจ้งเตือน และฟิลด์สำหรับกรอกรายละเอียดอื่นๆที่ผู้ใช้งานต้องการ ดังแสดงในรูปที่ 7.15 ซึ่งโปรแกรมจะทำการเก็บข้อมูลการนัดหมายไว้ในไฟล์ task.txt ในไดเรคทอรีของผู้ใช้งาน



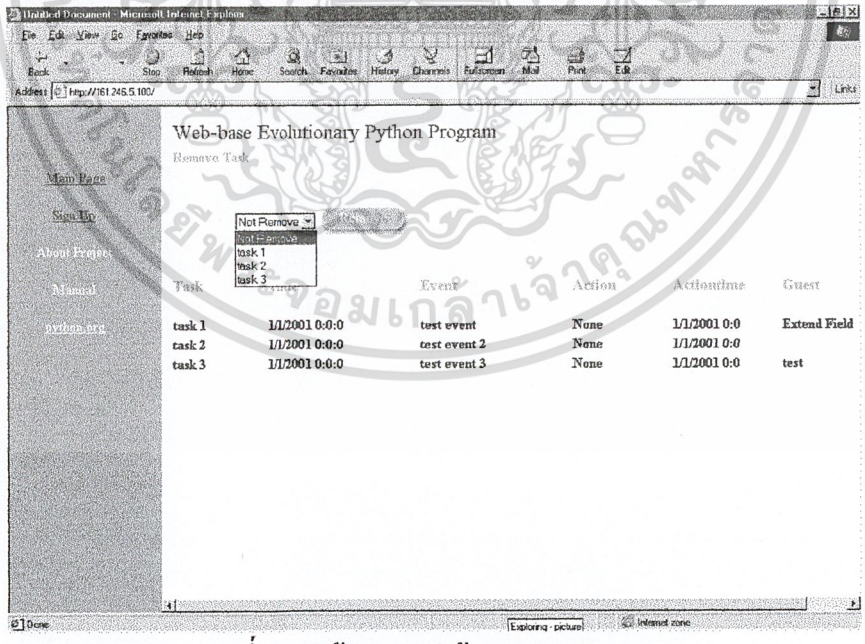

*รูปที่ 7.15 หน้าจอการเพิ่มข้อมูลตารางนัดหมาย*

6. ผู้ใช้งานโปรแกรมสามารถดูตารางนัดหมายซึ่งแสดงรายละเอียดของการนัดหมายต่างๆของผู้ใช้งานได้ในหน้าแสดงข้อมูลตารางนัดหมายดังแสดงในรูปที่ 7.16




*รูปที่ 7.16 หน้าจอแสดงข้อมูลตารางนัดหมาย*

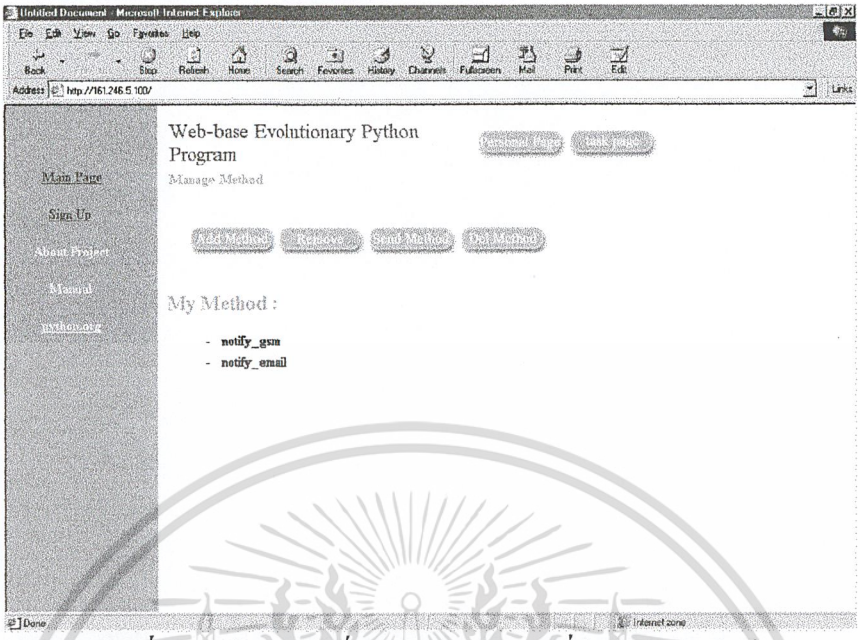
7. ผู้ใช้งานโปรแกรมสามารถทำการลบรายการในตารางนัดหมายได้ในหน้าการลบข้อมูลตารางนัดหมายดังแสดงในรูปที่ 7.17 ซึ่งมีรายละเอียดของตารางนัดหมายและผู้ใช้งานสามารถเลือกรายการที่ต้องการลบออกจากตารางนัดหมายได้ทันที โดยเมื่อผู้ใช้งานทำการลบรายการออกจากตารางนัดหมายโปรแกรมจะทำการลบข้อมูลการนัดหมายออกจากไฟล์ task.txt




*รูปที่ 7.17 หน้าจอการลบข้อมูลตารางนัดหมาย*

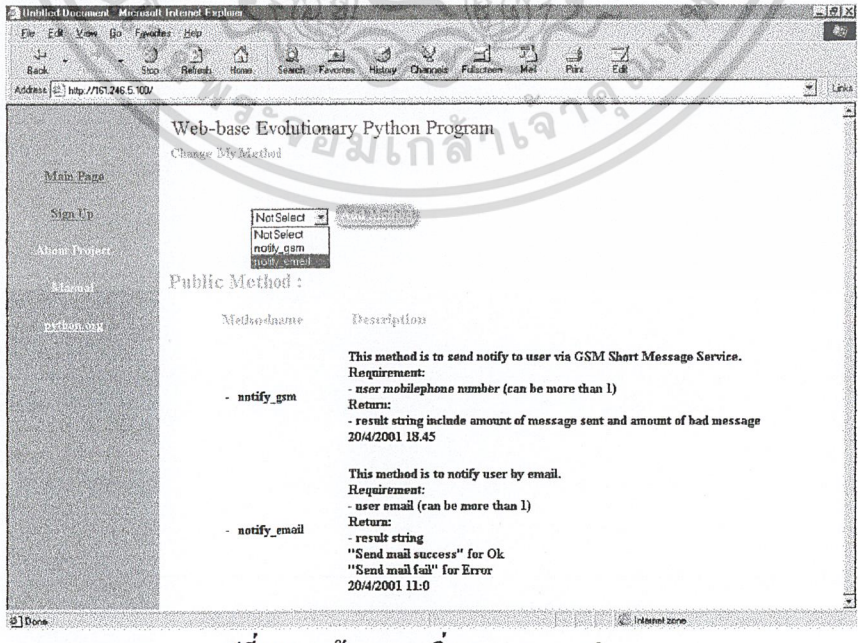
8. สำหรับกระบวนการทำงานของเอเย่นต์ผู้ใช้งานสามารถจัดการกับกระบวนการทำงานของเอเย่นต์ได้ โดยในหน้าแสดงชื่อกระบวนการทำงานที่มีอยู่ภายในเอเย่นต์ดังแสดงในรูปที่ 7.18 จะแสดงชื่อของ

กระบวนการทำงานที่มีอยู่ในขณะนั้นของเอเย่นต์ของผู้ใช้งานและจุดเชื่อมต่อไปยังส่วนที่ใช้ในการเพิ่ม ลด กระบวนการทำงานของเอเย่นต์ รวมถึงการเพิ่มกระบวนการทำงานลงในพับบลิคไลบรารี



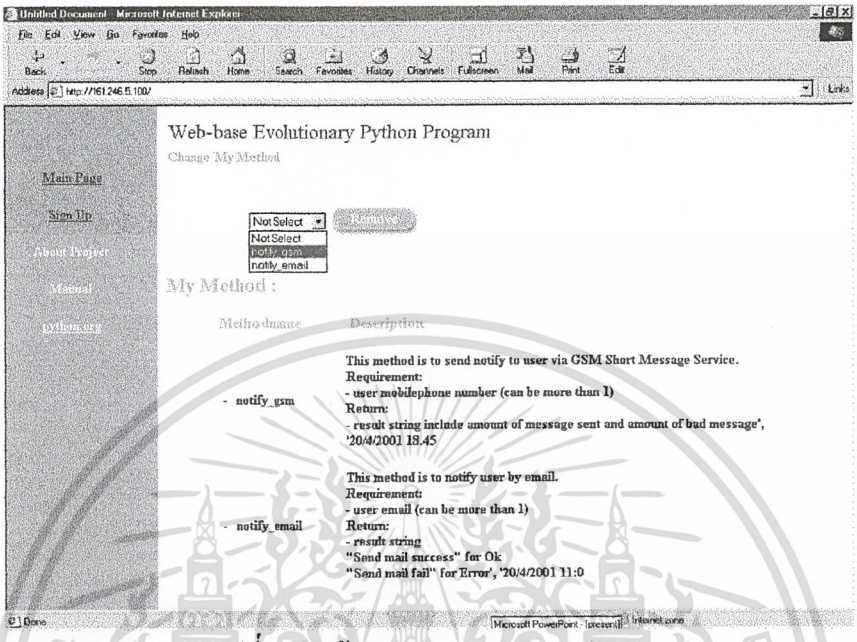

*รูปที่ 7.18 หน้าจอแสดงชื่อกระบวนการทำงานที่มีอยู่ภายในเอเย่นต์*

9. ผู้ใช้งานโปรแกรมสามารถทำการเพิ่มกระบวนการทำงานให้แก่เอเย่นต์โดยการเลือกกระบวนการทำงานในหน้าการเพิ่มกระบวนการทำงานดังแสดงในรูปที่ 7.19 ซึ่งจะแสดงชื่อกระบวนการทำงานที่มีอยู่ในพับบลิคไลบรารี รวมทั้งคำอธิบายการทำงานของกระบวนการทำงานนั้น เมื่อผู้ใช้ต้องการเพิ่มกระบวนการทำงานให้แก่เอเย่นต์ของตัวเอง ผู้ใช้งานสามารถเลือกกระบวนการทำงานจากเมนูบาร์เพื่อทำการเพิ่มกระบวนการทำงานนั้นให้กับเอเย่นต์ โดยโปรแกรมจะทำการคัดลอกกระบวนการทำงานลงในไฟล์ user_met.py ของผู้ใช้งานรายนั้นตามหลักการวิวัฒนาการคือทำการสร้างคลาสใหม่ที่มีการเพิ่มกระบวนการทำงานใหม่ลงไป



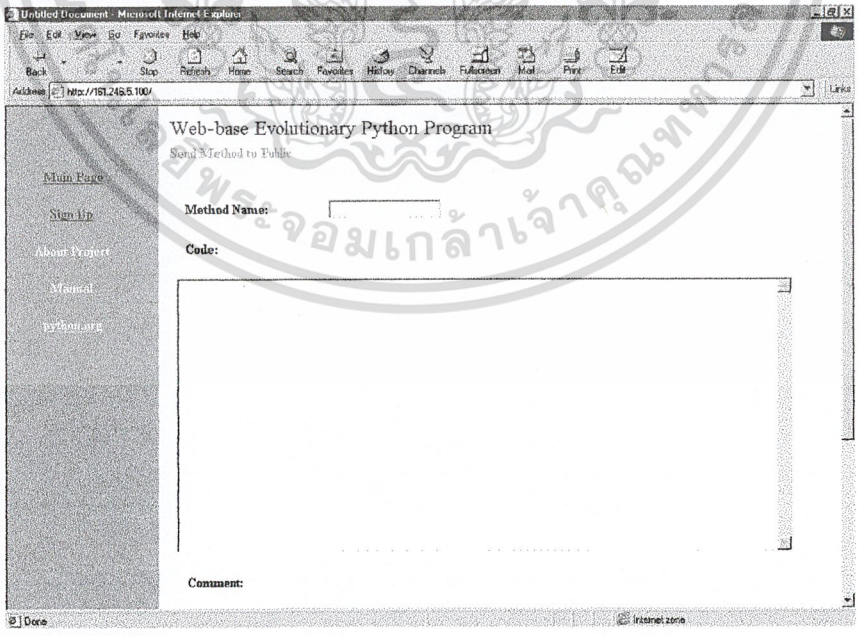

*รูปที่ 7.19 หน้าจอการเพิ่มกระบวนการทำงาน*

10. ผู้ใช้งานโปรแกรมสามารถทำการลบกระบวนการทำงานของเอเย่นต์ได้ในหน้าการลบกระบวนการทำงานดังแสดงในรูปที่ 7.20 ผู้ใช้งานสามารถเลือกกระบวนการทำงานจากเมนูบาร์เพื่อทำการลบกระบวนการทำงานนั้นออกจากเอเย่นต์ โดยโปรแกรมจะทำการเปลี่ยนแปลงกระบวนการทำงานตามหลักการวิวัฒนาการ



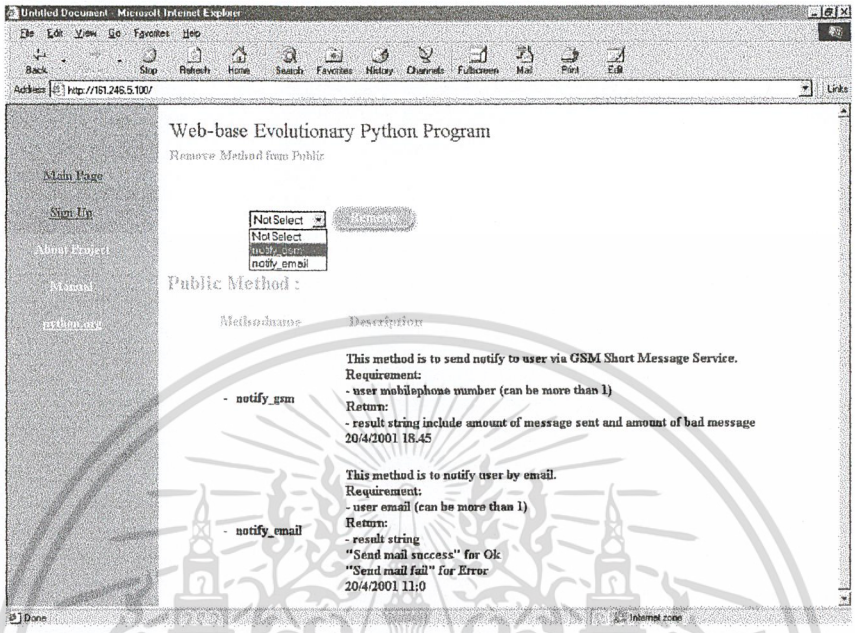

*รูปที่ 7.20 หน้าจอการลบกระบวนการทำงาน*

11. ผู้ใช้งานโปรแกรมสามารถเพิ่มกระบวนการทำงานลงในพับบลิคไลบรารีเพื่อให้ผู้ใช้งานโปรแกรมรายอื่นสามารถนำกระบวนการนั้นไปใช้งานได้ โดยการส่งโค้ดของกระบวนการทำงานไปยังเซิร์ฟเวอร์ในหน้าการเก็บกระบวนการทำงานลงในพับบลิคไลบรารีดังแสดงในรูปที่ 7.21




*รูปที่ 7.21 หน้าจอการเก็บกระบวนการทำงานลงในพับบลิคไลบรารี*

12. ผู้ใช้งานโปรแกรมสามารถลบกระบวนการทำงานออกจากพับบลิคไลบรารีได้ในหน้าการลบกระบวนการทำงานออกจากพับบลิคไลบรารีดังแสดงในรูปที่ 7.22 โดยผู้ใช้งานที่เป็นผู้เพิ่มกระบวนการนั้นเท่านั้นที่สามารถลบกระบวนการนั้นออกได้




*รูปที่ 7.22 หน้าจอการลบกระบวนการทำงานออกจากพับบลิคไลบรารี*

# บทที่ 8

# บทสรุปและวิจารณ์

## 8.1 บทสรุปและวิจารณ์

จากการศึกษาพัฒนาโปรแกรมเอเย่นต์ที่มีความสามารถในการเปลี่ยนกระบวนการทำงานและทำงานผ่านระบบเครือข่ายได้นั้น สามารถพัฒนาโปรแกรมให้ใช้งานทฤษฎีการเปลี่ยนกระบวนการแบบ Evolutionary ได้ โปรแกรมดังกล่าวคือโปรแกรมเอเย่นต์วาดรูป ที่มี Agent หลายๆ ตัว ทำงานในการวาดรูปร่วมกัน และสามารถแลกเปลี่ยนข้อมูลของกระบวนการทำงานได้ แต่โปรแกรมดังกล่าวยังไม่สามารถใช้งานให้เป็นประโยชน์ได้มากนัก จึงมีการพัฒนาโปรแกรมที่สอง คือโปรแกรมเพอซันแนลแอซซิสแทนต์ มีหน้าที่ในการจัดการกับตารางนัดหมายของผู้ใช้ โดยสามารถบันทึกตารางนัดหมาย, เรียกดูตารางนัดหมายพร้อมกับเปลี่ยนแปลงได้ นอกจากนี้ผู้ใช้ยังสามารถเพิ่มกระบวนการในการทำงานต่าง ๆ ได้ เช่น สามารถให้โปรแกรมแจ้งเตือนผ่านจดหมายอิเล็กทรอนิกส์ หรือระบบช็อตแมสเซจเซอร์วิส ทั้งหมดนี้ผู้ใช้สามารถใช้งานผ่านระบบเครือข่าย โปรแกรมที่ได้พัฒนาขึ้นมานี้มีข้อแตกต่างจากโปรแกรมทั่วไปคือเป็นโปรแกรมที่มีความยืดหยุ่นในการใช้งานและสามารถนำกระบวนการใหม่ๆที่ไม่มีอยู่ในตัวเองมาพัฒนาให้สามารถทำงานได้หลากหลายมากขึ้นตามความต้องการของผู้ใช้ ข้อเสียของโปรแกรมที่พัฒนานี้คือถึงแม้ว่าผู้ใช้สามารถเพิ่มกระบวนการต่าง ๆ ได้เองแต่ก็มีข้อจำกัดอยู่ที่ผู้พัฒนาต้องพัฒนากระบวนการทำงานให้สอดคล้องกับตัวแปรระบบ และสิ่งแวดล้อมของระบบที่โปรแกรมกำหนดไว้

## 8.2 ปัญหาที่เกิดขึ้น

1. ข้อมูลเกี่ยวกับภาษาไพธอนมีจำนวนน้อย รวมทั้งส่วนที่เป็นอินเตอร์เฟสติดต่อกับผู้ใช้งานใช้ Tk ซึ่งไม่มีเอกสารที่ใช้ในการศึกษา ทำให้การศึกษา และพัฒนาโปรแกรมเป็นไปได้ช้า
2. ไลบรารีที่สนับสนุนโปรแกรมมีจำนวนน้อย บางโมดูลยังมีปัญหาในการใช้งาน เช่น โมดูล Threading
3. ภาษาทำงานได้ช้าเนื่องจากเป็นภาษาที่เป็นอินเตอพรีเตอร์ นอกจากนั้นการใช้งาน Thread ในภาษาไพธอนไม่ดีนักจึงทำให้โปรแกรมทำงานได้ช้า

## 8.3 วิธีการแก้ปัญหา

ค้นหาเอกสารอ้างอิงของภาษาไพธอนจากอินเตอร์เน็ตในส่วนของกราฟฟิกยูสเซอร์อินเตอร์เฟส และเนื่องจากภาษาไพธอนเป็นอินเตอร์พรีเตอร์ จึงสามารถศึกษาจากโค้ดของภาษา ในไลบรารีต่าง ๆ ได้โดยตรง สำหรับการใช้งานที่มีการพัฒนาภาษาไพธอนร่วมกับภาษาอื่น ๆ บนวินโดวส์นั้นสามารถทำได้โดยภาษาไพธอนสามารถเรียกใช้งาน Component Object Model (COM) เพื่อใช้ติดต่อกับโปรแกรมอื่นนอกจากนี้โปรแกรมอื่น ๆ สามารถเรียกใช้โมดูลที่เขียนด้วยภาษาไพธอนผ่าน COM เช่นกัน

# ภาคผนวก
# การติดตั้งเว็บเซิร์ฟเวอร์สำหรับใช้งาน Mod_python

**ขั้นตอนการติดตั้งภาษาไพธอน**

1. ทำการดาวน์โหลดโปรแกรมติดตั้งไพธอนได้จาก www.python.org ซึ่งจะได้ไฟล์ py152.exe ซึ่งเป็นไฟล์ที่ใช้ติดตั้งภาษาไพธอนเวอร์ชั่น 1.5.2 บนระบบปฏิบัติการวินโดวส์ โดยเวอร์ชั่นล่าสุดเป็นเวอร์ชั่น 2.1
2. ทำการรันโปรแกรม py152.exe และทำการติดตั้งภาษาไพธอนลงในไดเรคทอรีที่ต้องการ

**ขั้นตอนการติดตั้งและใช้งาน Apache Web Server**

1. โครงงานนี้จะทำการติดตั้ง Apache บนระบบปฏิบัติการวินโดวส์ โดยทำการดาวน์โหลดโปรแกรมติดตั้ง Apache เซิร์ฟเวอร์จากเว็บ www.apache.org จะได้ไฟล์ apache_1_3_14_win32_r2.exe
2. ทำการรันโปรแกรม apache_1_3_14_win32_r2.exe เพื่อทำการติดตั้ง Apache ลงบนไดเรคทอรีที่ต้องการ ซึ่งโครงงานนี้ได้ทำการติดตั้งลงในไดเรคทอรี C:\ApacheGroup\Apache
3. เมื่อทำการทดสอบโปรแกรมในขณะที่คอมพิวเตอร์ไม่ได้ทำการติดตั้งอยู่ในระบบเครือข่ายสามารถใช้แอดเดรส 127.0.0.1 เพื่อทำการทดสอบโปรแกรม หรือระบุแอดเดรสโดยการแก้ไขข้อมูลในไฟล์ httpd.conf โดยใส่คำสั่ง Listen พร้อมทั้งแอดเดรสที่ต้องการและพอร์ตที่ใช้งานซึ่งปกติจะใช้งานที่พอร์ต 80 เช่น Listen 161.246.5.100:80
4. เมื่อต้องการใช้งาน Apache Web Server ทำได้โดยการเลือก start apache สามารถทดสอบการทำงานของ Apache โดยการเปิดบราวเซอร์ไปที่แอดเดรส 127.0.0.1 หรือ แอดเดรสที่ทำการกำหนดให้กับเซิร์ฟเวอร์ ซึ่งเซิร์ฟเวอร์จะทำการเปิดเว็บเพจ index.html ภายในไดเรคทอรี apache/htdocs โดยเว็บเพจ index.html จะแสดงสัญลักษณ์ของ Apache บนบราวเซอร์ เมื่อต้องการหยุดการทำงานทำได้โดยการเลือก stop apache

**ขั้นตอนการติดตั้ง Mod_python**

1. เราสามารถทำการดาวน์โหลดโปรแกรม Mod_python สำหรับทำการติดตั้งบน Apache ได้จาก www.mod_python.org ซึ่งจะต้องใช้ 2 ไฟล์คือ mod_python-2.7.1.tgz และ Mp152dll.zip ซึ่งจะเป็น zip file ให้ทำการ extract ไฟล์ mod_python-2.7.1.tgz ลงใน temporary folder ของ windows (C:\WINDOWS\TEMP) โดยโปรแกรม winzip จะแสดงข้อความ “Archive contains one file, should Winzip decompress it to a temporary folder?" ให้คลิก yes จะได้ไดเรคทอรี mod_python-2.7.1 ซึ่งภายในประกอบด้วยไดเรคทอรีต่างๆ เช่น Doc สำหรับเก็บเอกสารเอชทีเอ็มแอลอธิบายการใช้งาน และ lib ที่เก็บตัวโค้ด Mod_python

2. ทำการคัดลอกไดเรคทอรี mod_python (C:\WINDOWS\TEMP\mod_python-2.7.1\lib\python) ไปไว้ในไดเรคทอรีที่ใช้เก็บไลบารีของไพธอนเช่น C:\Program Files\Python\lib
3. ทำการ extract ไฟล์ 152dll.zip จะได้ไฟล์ mod_python.dll ให้ทำการคัดลอกไปไว้ในไดเรคทอรี C:\ApacheGroup\Apache\modules
4. ทำการแก้ไขไฟล์ httpd.conf ของ Apache โดยการใส่บรรทัดต่อไปนี้ลงในส่วน "Dynamic Shared Object (DSO) Support"

   LoadModule python_module modules/mod_python.dll
5. ทำการเพิ่มบรรทัดต่อไปนี้ลงในไฟล์ httpd.conf ในส่วนของ Script Alias and CGI

   ```
   <Directory "<Your Document Root>">
       AddHandler python-program .py
       PythonHandler test
       PythonDebug on
   </Directory>
   ```

   โดยในส่วน <Your Document Root> หมายถึงไดเรคทอรีที่เราต้องทำการจัดเก็บ CGI Script ภาษาไพธอน เช่น

   ```
   <Directory "C:\ApacheGroup\Apache\applet">
       AddHandler python-program .py
       PythonHandler test
       PythonDebug on
   </Directory>
   ```

   จะเป็นการกำหนดให้เมื่อมีการเรียกใช้ไฟล์ที่มีนามสกุล .py ภายในไดเรคทอรี applet เซิร์ฟเวอร์จะทำการเปิดไฟล์ test.py ขึ้นมาทำงานทันทีโดยไม่สนใจชื่อไฟล์และจะใช้ไพธอนในการรันโปรแกรม
6. เราสามารถทดสอบการทำงานของ Mod_python ได้โดยการสร้างไฟล์ test.py และนำไปเก็บไว้ในไดเรคทอรี C:\ApacheGroup\Apache\applet โดยในไฟล์ test.py ให้เขียนโปรแกรมดังนี้

   ```
   from mod_python import apache
   def handler(req):
       req.content_type = "text/plain"
       req.send_http_header()
       req.write("Hello World!")
       return apache.OK
   ```

   หลังจากนั้นทำการเปิดบราวเซอร์ไปที่ 127.0.0.1/applet/test.py จะมีข้อความ "Hello World!"
7. เราสามารถใช้งาน public handler ได้โดยการใส่บรรทัดต่อไปนี้ลงในไฟล์ httpd.conf ในส่วนของ Script Alias and CGI

   ```
   <Directory "C:/ApacheGroup/Apache/cgi-bin" >
   ```

```
    SetHandler python-program
    PythonHandler mod_python.publisher
</Directory>
```

เมื่อ C:/ApacheGroup/Apache/cgi-bin คือไดเรคทอรีที่ต้องการเรียกใช้ public handler

8. เราสามารถทำการทดสอบการทำงานของ public handler ได้โดยการสร้างไฟล์ hello.py ซึ่งมีโปรแกรมดังนี้

```
""" Publisher example """
def say_yes(req, what="Yes"):
    return "I am saying %s" % what
def say_no(req,what="No"):
    return "I am saying %s" % what
```

หลังจากนั้นทำการเปิดบราวเซอร์ไปที่ 127.0.0.1/cgi-bin/hello.py/say_yes จะมีข้อความ "I am saying Yes" แต่ถ้าเปิดที่ 127.0.0.1/cgi-bin/hello.py/say_no จะมีข้อความ "I am saying No"

# บรรณานุกรม

[1] ฉลองชัย จงประเสริฐพร และ วรรวิภา ท่าพระนาง : " CGI / WEB Programming ", บริษัท ซีเอ็ดยูเคชั่น จำกัด ( มหาชน )

[2] ทรงเกียรติ ภาวดี ( 2542 ) : " แกะรอย CGI ", บริษัท ซีเอ็ดยูเคชั่น จำกัด ( มหาชน )

[3] วีระศักดิ์ ซึงถาวร ( 1998 ) : " Fundamantal of JAVA Programming (Volume 1) ", SUM System Company Limited

[4] วีระศักดิ์ ซึงถาวร ( 2543 ) : " Fundamantal of JAVA Programming (Volume 2) ", บริษัท ซีเอ็ดยูเคชั่น จำกัด ( มหาชน )

[5] Bill McCarty and Luke Cassady – Dorian ( 1999 ) : " Java Distributed Object ", A Division of Macmillan Computer Publishing, Indiana

[6] Daryl Harms and Kenneth McDonald ( 2000 ) : " The Quick Python Book ", Manning Publications CO., USA

[7] Joseph P Bigus and Jennifer Bigus ( 1998 ) : " Constructing Intelligent Agents with Java ", Wiley Computer Publishing, USA

[8] Mark Hammond and Andy Robinson ( 2000 ) : " Python Programming on Win32 ", O'reilly & Associates Inc., CA

[9] Mark Lutz and David Ascher ( 1999 ) : " Learning Python ", O'reilly & Associates Inc., CA

[10] Mark Watson ( 1997 ) : " Intelligent Java Application for the Internet and Intranet ", Morgan Kaufmann Publishing Inc., CA

[11] Martin C Brown ( 2000 ) : " Python Annotated Archives ", Osborne / McGraw – Hill, USA

[12] Merlin Huges, Michel Shoffner and Derek Hammer ( 1999 ) : " Java Network Programming ", Manning Publication Co., USA