Think Beyond Develop

Think Beyond

#มือใหม่ Python เก่งได้ใน 30 วัน

- พื้นฐาน Python สำหรับนักเรียน นักศึกษา และโปรแกรมเมอร์
- ตัวอย่างโค้ดจริงกว่า 150 ตัวอย่าง พร้อมคำอธิบายโดยละเอียด
- วิธีดึงข้อมูลจากเว็บไซต์ สร้างเกม และเว็บแอปพลิเคชัน
- การนำ Python มาใช้กับ Data Science และ Data Visualization
- วิธีใช้งานโมดูลยอดนิยม เช่น turtle, tkinter, flask, Django, pandas และอื่น ๆ

จีราวุธ วารินทร์

SCAN FREE

ไฟล์ประกอบหนังสือ

จัดจำหน่ายโดย IDC PREMIER

คำนำ

Python คือ ภาษาคอมพิเตอร์ยุคใหม่ที่ถูกออกแบบมาให้ใช้งานได้ง่าย ยืดหยุ่น แต่เปี่ยมไปด้วยประสิทธิภาพ เมื่อดูสถิติการใช้งานภาษาคอมพิวเตอร์ทั่วโลก ปฏิเสธไม่ได้เลยว่า Python เป็นภาษาคอมพิวเตอร์ยอดนิยมอันดับหนึ่งในยุคปัจจุบัน

เพื่อให้ผู้อ่านสามารถเข้าใจภาษา Python ได้อย่างรวดเร็ว หนังสือเล่มนี้จึงแบ่งออกเป็น 5 ส่วน เริ่มจากพื้นฐาน Python จนไปถึงตัวอย่างการนำ Python ไปใช้งานในรูปแบบต่าง ๆ ได้

ส่วนแรก พื้นฐาน Python ที่จำเป็นต้องทราบทั้งหมด เช่น การประกาศตัวแปร การเลือกชนิดข้อมูลที่เหมาะสม วิธีคำนวณ การวนลูป การตัดสินใจ วิธีสร้างและใช้ฟังก์ชัน การกำหนดคลาส การสร้างอ็อบเจกต์ และอื่น ๆ อีกมากมาย หลังจากอ่านส่วนนี้จบจะมีความรู้พื้นฐานเกี่ยวกับ Python ครบทุกแง่มุมแล้ว

ส่วนที่ 2 ได้แสดงตัวอย่างการสร้างส่วนติดต่อกับผู้ใช้ โดยใช้โมดูล turtle และ tkinter เช่น การแสดงข้อความ การกำหนดปุ่ม การวาดรูป วิธีสร้างแบบฟอร์ม รวมถึงตัวอย่างการสร้างเกม หลังจากจบในส่วนนี้จะสามารถสร้างแอปพลิเคชันหรือเกมแบบต่าง ๆ

ส่วนที่ 3 เป็นการนำ Python ไปใช้ดึงข้อมูลจากเว็บไซต์ หรือที่เรียกว่า Web Scraping ซึ่งจะเริ่มด้วยการดึงข้อมูลจากเว็บในแบบง่าย ๆ โดยใช้โมดูล BeautifulSoup ตามด้วยการดึงข้อมูลที่ซับซ้อนขึ้นด้วยโมดูล Selenium

ส่วนที่ 4 เป็นตัวอย่างการนำ Python ไปสร้างเว็บแอปพลิเคชัน โดยใช้โมดูลยอดนิยมอย่าง Flask, Sqlite3 และ Django เมื่ออ่านส่วนนี้จบจะสามารถสร้างเว็บเซิร์ฟเวอร์ อ่านเขียนฐานข้อมูล และสร้างเว็บแอปพลิเคชันในแบบที่ต้องการได้ทันที

ส่วนสุดท้าย เป็นการนำ Python ไปประยุกต์ใช้งานด้านอื่น ๆ เช่น การใช้โมดูล pandas สำหรับจัดการและวิเคราะห์ข้อมูล และใช้โมดูล bokeh เพื่อนำข้อมูลที่ได้ไปพล็อตเป็นกราฟแบบต่าง ๆ

จากเนื้อหาทั้งหมดของหนังสือเล่มนี้จะเน้นทฤษฎีพอสังเขป แต่เน้นการยกตัวอย่างโค้ดจำนวนมากเป็นหลัก ผู้เขียนจึงมั่นใจเป็นอย่างยิ่งว่า ผู้อ่านจะมีความรู้เกี่ยวกับ Python ครบทุกด้าน ในระดับที่สามารถนำไปใช้งานจริงได้อย่างแน่นอน

จีราวุธ วารินทร์

jeerawuth@me.com

สารบัญ

Part 1 : Basic

บทที่ 1 รู้จักกับ Python 6
บทที่ 2 ติดตั้ง Python และการใช้ Python Shell 24
บทที่ 3 ตัวแปรและชนิดข้อมูล 42
บทที่ 4 การคำนวณค่าทางคณิตศาสตร์ 65
บทที่ 5 ฟังก์ชัน 76
บทที่ 6 การตัดสินใจ 100
บทที่ 7 การวนซ้ำ 117
บทที่ 8 Scope, Closure และ Decorator 139
บทที่ 9 พื้นฐานการใช้งาน Dictionary 158
บทที่ 10 List, Tuple และ Set 165
บทที่ 11 พื้นฐานการใช้งาน Error Handling 187
บทที่ 12 ตัวอย่างโปรแกรมเข้ารหัสในแบบซีซาร์ (Caesar Cipher) 195
บทที่ 13 การอ่านและเขียนไฟล์ 206
บทที่ 14 คลาสและอ็อบเจกต์ 229
บทที่ 15 การสืบทอดคลาส (Class Inheritance) 269
บทที่ 16 โมดูล (Module) และแพ็กเกจ (Packages) 286

Part 2 User Interface

บทที่ 17 วาดรูปด้วย Turtle Graphics 302
บทที่ 18 เกมหลบหลีกชิงธง 327
บทที่ 19 สร้างแบบฟอร์มโดยใช้ tkinter โมดูล 343
บทที่ 20 วาดรูปกราฟิกโดยใช้ tkinter 391

Part 3 Web Scraping

บทที่ 21 ดึงข้อมูลจากเว็บแบบง่าย ๆ ด้วย BeautifulSoup 408
บทที่ 22 Web Scraping แบบซับซ้อนด้วย Selenium 425

Part 4 Web Application

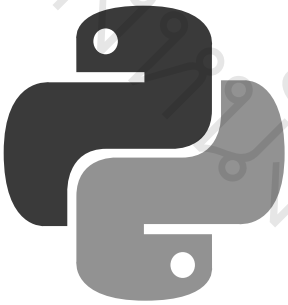
บทที่ 23 การใช้งาน Virtual Environment 444
บทที่ 24 สร้างเว็บแอปพลิเคชันด้วย Flask 454
บทที่ 25 อ่านและแก้ไขข้อมูลใน Google Sheet 506
บทที่ 26 พื้นฐานการใช้งาน SQLite ใน Python 527
บทที่ 27 พื้นฐานการใช้งาน Django 552
บทที่ 28 โปรเจกต์ข้อมูลสินค้า 590

Part 5 Data Science and Data Visualization

บทที่ 29 จัดการข้อมูลด้วย pandas 616
บทที่ 30 การพล็อตกราฟด้วย bokeh 634

Part 1

Basic



บทที่ 1
รู้จักกับ Python

Python (ไพทอน) เป็นภาษาคอมพิวเตอร์ยอดนิยมที่มีผู้ใช้งานทั่วโลก สามารถนำ Python มาสร้างเป็นแอปพลิเคชันหรือใช้งานได้ในหลายลักษณะ เช่น สร้างเว็บแอปพลิเคชัน นำมาใช้วิเคราะห์ข้อมูล ใช้ในงานด้านวิทยาศาสตร์ ใช้ในงานด้าน AI และ Machine Learning

ในบทแรกนี้จะมาทำความรู้จักกับภาษา Python และวิธีการทดสอบโค้ด Python ผ่านทางเว็บเบราว์เซอร์ ซึ่งวิธีนี้มีข้อดี คือ ไม่ต้องติดตั้งอะไรลงไปในเครื่องคอมพิวเตอร์ก็สามารถทดสอบหรือเรียนรู้การใช้ Python ได้ทันที

ภาษาคอมพิวเตอร์

การสั่งงานคอมพิวเตอร์เพื่อให้คอมพิวเตอร์ทำตามที่ต้องการจะต้องใช้คำสั่งที่คอมพิวเตอร์เข้าใจ ซึ่งเรียกว่า **ภาษาเครื่อง (Machine Code)** แต่ภาษาเครื่องเป็นรหัสตัวเลขฐานสองที่ซับซ้อน ทำให้ยากต่อการใช้งาน ดังนั้นจึงมีการพัฒนา**ภาษาคอมพิวเตอร์ (Computer Language)** ขึ้นมาเพื่อให้สามารถสั่งงานคอมพิวเตอร์ได้ง่ายขึ้น

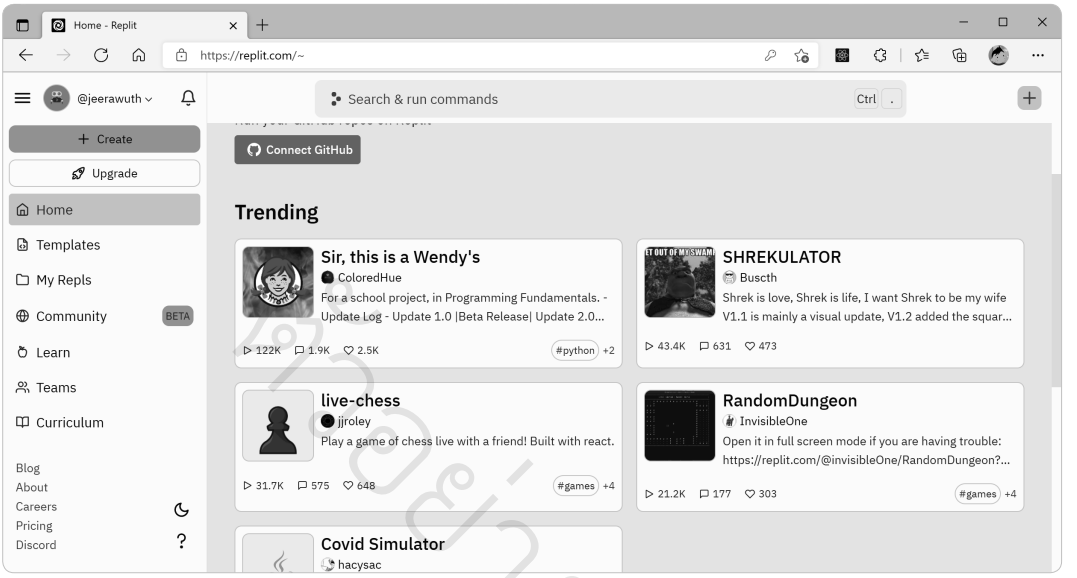
ภาษาคอมพิวเตอร์ที่ใช้ในปัจจุบันมีอยู่ด้วยกันมากมาย เช่น ภาษา C, Java, Swift และ Python โดยแต่ละภาษาคอมพิวเตอร์จะกำหนดกฎหรือโครงสร้างของภาษาที่แตกต่างกัน ผู้ใช้งานจะต้องทราบว่าการสั่งงานคอมพิวเตอร์จะต้องเขียนอย่างไร เช่น ในภาษา Python หากต้องการสั่งให้พิมพ์ข้อความออกมาที่หน้าจอก็จะใช้คำสั่ง print()

เมื่อต้องการสั่งงานคอมพิวเตอร์จะเขียนชุดคำสั่งโดยใช้ภาษาคอมพิวเตอร์ เช่น Java, Python, Swift โดยชุดคำสั่งที่เขียนขึ้นจากภาษาคอมพิวเตอร์จะถูกเรียกว่า **ซอร์สโค้ด (Source Code)** แต่เนื่องจากคอมพิวเตอร์เข้าใจแต่ภาษาเครื่องเท่านั้น ไม่เข้าใจซอร์สโค้ดที่มนุษย์เขียนขึ้น ดังนั้นจึงต้องมีตัวกลางเพื่อแปลงจากซอร์สโค้ดให้กลายเป็นภาษาเครื่อง เรียกว่า **ตัวแปลภาษา (Translator)**

ตัวแปลภาษา ทำหน้าที่เป็นเหมือนล่าม แปลความหมายของซอร์สโค้ดที่มนุษย์เขียน เพื่อบอกกับคอมพิวเตอร์ว่ามนุษย์ต้องการทำอะไร ตัวอย่างเช่น หากต้องการสั่งให้คอมพิวเตอร์คำนวณหาพื้นที่สามเหลี่ยม ก็จะต้องเขียนซอร์สโค้ดส่งไปให้ตัวแปลภาษาแปลความหมาย เมื่อคอมพิวเตอร์ได้รับคำสั่งก็จะคำนวณพื้นที่สามเหลี่ยมออกมาตามต้องการ

สำรวจหน้าจอโฮมของ Repl

หลังจากลงทะเบียนกับ Repl เรียบร้อย ให้กลับไปยัง https://replit.com อีกครั้ง เมื่อเข้าสู่ระบบก็จะพบหน้าจอโฮม (Home) ดังรูป



จากรูปหน้าจอหลักของ Repl จะมีอยู่ 2 ส่วน คือ

- ด้านซ้ายจะเป็นเมนูคำสั่ง เช่น เมนู Home ใช้แสดงหน้าจอโฮมที่เป็นหน้าจอหลัก เมนู Templates ใช้กำหนดภาษาเริ่มต้นสำหรับโปรเจกต์ใหม่ เมนู My Repls สำหรับแสดงข้อมูลเกี่ยวกับโค้ดที่ได้เขียนเอาไว้ หรือเมนู Learn สำหรับเรียนรู้วิธีใช้งาน Repl
- ด้านขวาจะแสดงเนื้อหาที่เกี่ยวข้องกับเมนูที่ได้เลือกไว้ เช่น เมื่อเลือกเมนู Learn ก็จะพบข้อมูลเกี่ยวกับวิธีใช้งาน หรือคำแนะนำเกี่ยวกับการใช้งาน Repl

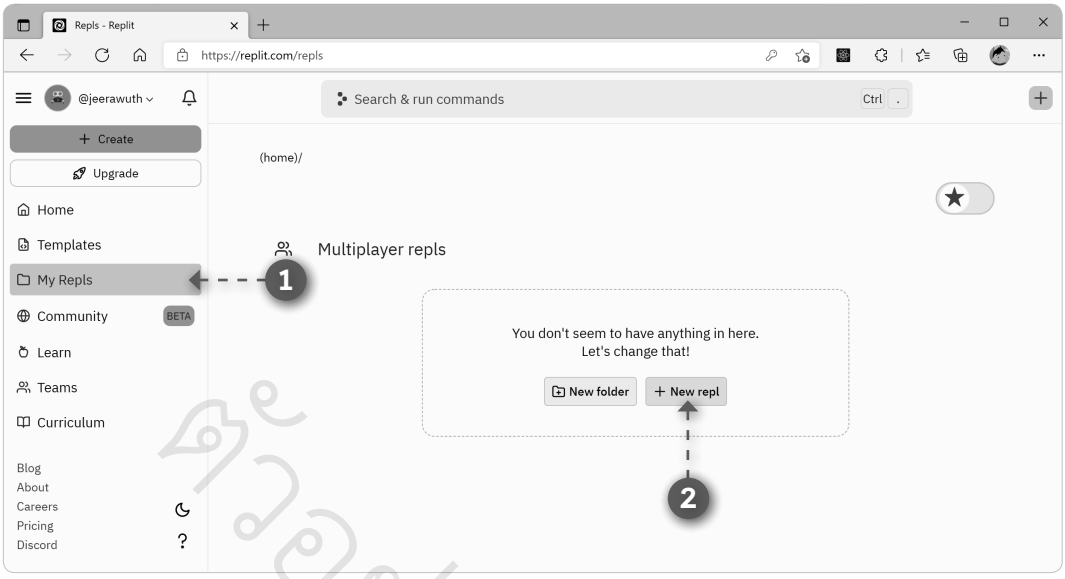
โค้ดแรกกับ Python

คำสั่งแรกของ Python ที่ต้องทราบ คือ คำสั่ง print ใช้เพื่อพิมพ์ข้อความออกมาที่หน้าจอ เช่น เมื่อต้องการพิมพ์ข้อความ "Hello Python" ก็ให้ใช้คำสั่ง print("Hello Python")

เริ่มต้นด้วยการสร้างไฟล์ขึ้นมาใหม่ จากนั้นเลือกภาษาที่ใช้งานเป็น Python และกรอกโค้ดคำสั่งลงไป ดังรายละเอียดต่อไปนี้

1. เปิดเบราว์เซอร์และไปยัง https://replit.com ให้เข้าสู่ระบบ จากนั้นคลิกที่เมนู My Repls เพื่อแสดงพื้นที่ส่วนตัว ซึ่งเก็บโค้ดที่ได้เขียนเอาไว้ (ในการใช้งานครั้งแรกจะเป็นพื้นที่ว่าง ๆ ที่ยังไม่มีข้อมูลอะไรอยู่)

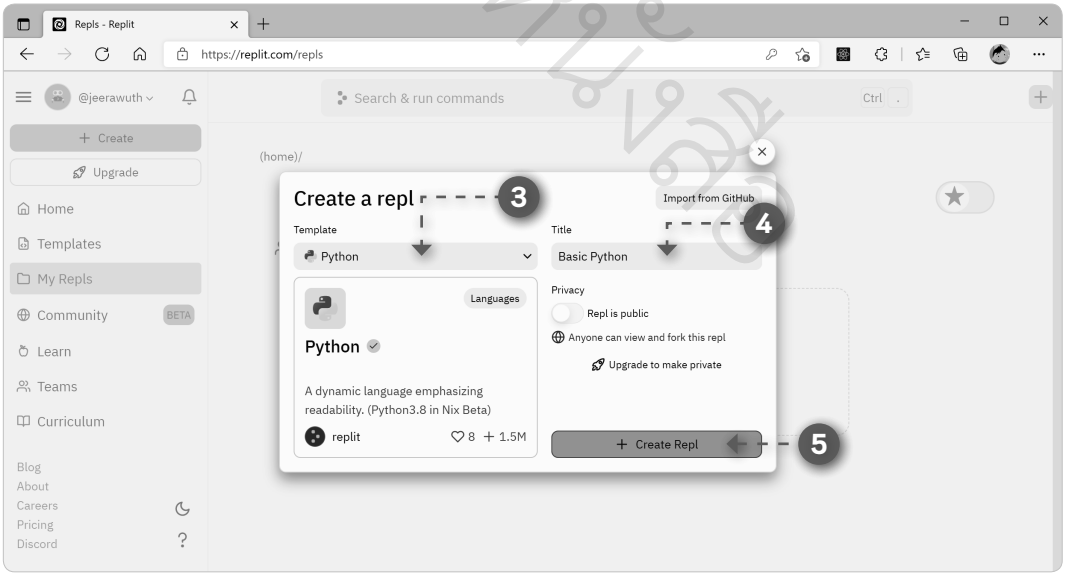
2. คลิกที่ปุ่ม + New repl



3. เลือกเทมเพลตที่จะใช้งาน ให้เลือกเป็น Python

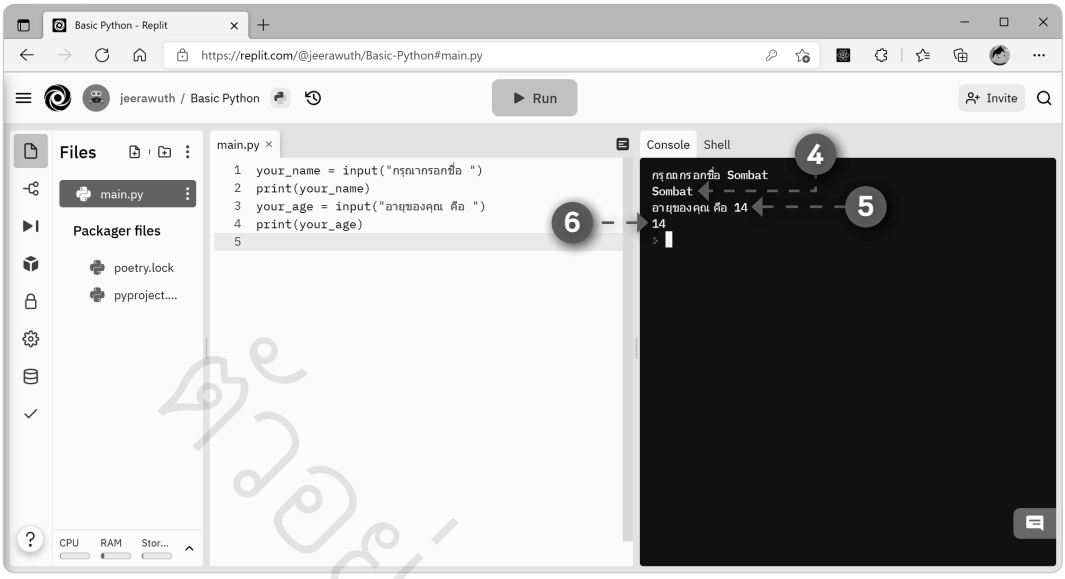
4. ที่ Title สามารถตั้งชื่อได้ตามต้องการ เช่น Basic Python

5. คลิกที่ปุ่ม + Create Repl



6. ระบบจะสร้างไฟล์ main.py มาให้อัตโนมัติ (นามสกุลไฟล์จะเป็น .py ซึ่งเป็นนามสกุลไฟล์ของ Python)

5. จะปรากฏข้อความ "อายุของคุณ คือ " ให้กรอกอายุ (เช่น 14) แล้วกด <Enter>
6. อายุที่ผู้ใช้กรอกจะถูกนำมาแสดงที่หน้าจอ (ในตัวอย่างคือเลข 14)



เชื่อมข้อความด้วยเครื่องหมายบวก (+)

เมื่อต้องการนำข้อความมาต่อกันสามารถใช้เครื่องหมายบวก (+) เชื่อมข้อความเข้าด้วยกัน

```
text1 + text2
```

- **text1** คือ ข้อความในส่วนแรก
- **text2** คือ ข้อความในส่วนที่สอง
- **text1** + **text2** คือ การนำข้อความจาก text1 มาเรียงต่อกับข้อความใน text2

ตัวอย่างการเชื่อมระหว่างข้อความ "My name is: " กับข้อความที่อยู่ในตัวแปร my_name

```
1 my_name = "Lisa"
2 print("My name is: " + my_name)
```

- บรรทัดที่ 1 ประกาศตัวแปร my_name เก็บข้อความ "Lisa"
- บรรทัดที่ 2 นำข้อความ "My name is: " มาต่อกับข้อความที่อยู่ในตัวแปร my_name

Note

ข้อความที่ใช้ในชุดคำสั่งของ Python จะต้องอยู่ภายใต้เครื่องหมาย " " เช่น "Hello Python" หรืออยู่ภายใต้เครื่องหมาย ' ' ก็ได้ เช่น 'Hello Python' ส่วนชื่อตัวแปรไม่ต้องอยู่ภายใต้เครื่องหมาย " " หรือ ' ' เช่น my_name

- บรรทัดที่ 2 ประกาศตัวแปร sentence_b เก็บข้อความ โดยใช้เครื่องหมาย ' ' ครอบข้อความ สังเกตว่าภายในข้อความมีการใช้ ' ซึ่งเป็นอักษรที่สงวนไว้ใช้กับคำสั่งของ Python จึงเปลี่ยนไปใช้ \' แทน '

ตัวอย่าง Escape Character รูปแบบต่าง ๆ ที่ใช้ใน Python

คำสั่ง	ผลลัพธ์
\"	แสดงเครื่องหมาย " (Double quote)
\'	แสดงเครื่องหมาย ' (Single quote)
\\	แสดงเครื่องหมาย \ (Backslash)
\n	สั่งให้ขึ้นบรรทัดใหม่ (New line)
\r	เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของบรรทัด (Carriage return)
\t	ขยับข้อความเป็นระยะ 1 แท็บ (เหมือนกับการกดคีย์ Tab)
\f	ไปยังบรรทัดถัดไป แต่เริ่มต้นโดยใช้เคอร์เซอร์ในตำแหน่งปัจจุบัน (Form Feed)
\b	ลบตัวอักษรทางด้านซ้าย 1 ตัวอักษร (เหมือนกับการกดคีย์ Backspace)

เพื่อให้เข้าใจการใช้งาน Escape Character ได้ง่ายยิ่งขึ้น จะยกตัวอย่างการใช้งาน Escape Character ไว้ในตัวอย่างต่อไปนี้ เช่น การใช้ \r, \t, \' และ \f

```
1 sentence_a = "123456789 \r000"
2 sentence_b = "Name:\tJeeawuth"
3 sentence_c = 'I\'ve been \fwaiting for you.'
4 print(sentence_a)
5 print(sentence_b)
6 print(sentence_c)
```

- บรรทัดที่ 1 ใช้เครื่องหมาย \r เพื่อขยับตำแหน่งเคอร์เซอร์ไปยังตำแหน่งเริ่มต้นของข้อความ จากนั้นก็ให้แสดงข้อความ 000 ดังนั้นผลลัพธ์ที่แสดงบนหน้าจอจึงเป็น 000456789
- บรรทัดที่ 2 ใช้เครื่องหมาย \t เพื่อขยับข้อความ Jeerawuth ไปทางขวาเป็นระยะ 1 แท็บ
- บรรทัดที่ 3 ใช้เครื่องหมาย \' เพื่อแสดง ' (Single quote) และใช้ \f เพื่อไปยังบรรทัดใหม่ แล้วแสดงข้อความ waiting for you. ในตำแหน่งปัจจุบัน

บทที่ 3
ตัวแปรและชนิดข้อมูล

ในบทนี้จะกล่าวถึงการประกาศและใช้งานตัวแปร รวมถึงพื้นฐานการใช้งานชนิดข้อมูลแบบต่าง ๆ ใน Python ได้แก่ ตัวเลขจำนวนเต็ม (int) ตัวเลขทศนิยม (float) ข้อความ (str) และค่าความจริงทางตรรกะ (bool)

ข้อมูล (Data) ที่อยู่ใน Python ไม่ว่าจะเป็นตัวเลขจำนวนเต็ม เลขทศนิยม ข้อความ หรือค่าความจริงจะอยู่ในรูปของอ็อบเจกต์ (Object) โดย Python จะกำหนดค่า (Value) และกำหนดชนิดข้อมูล (Data Type) เอาไว้ด้วยเสมอ เช่น

ข้อมูลทุก ๆ อย่างใน Python คือ อ็อบเจกต์ (Object)

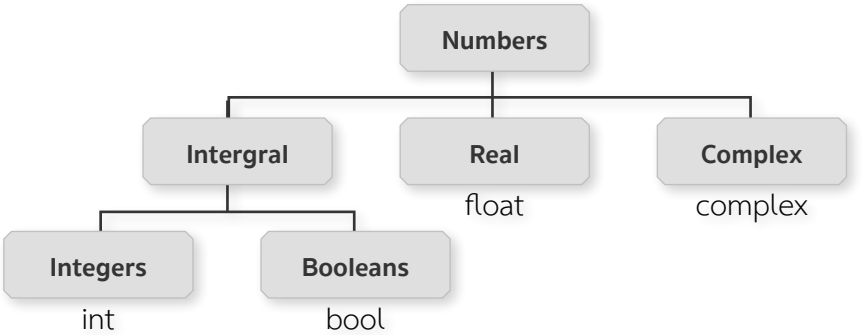
- เลขจำนวนเต็ม 10 ก็คืออ็อบเจกต์ที่มีค่า (Value) เท่ากับ 10 และมีชนิดข้อมูล (Data Type) เป็นแบบ int (Integer : จำนวนเต็ม)
- เลขทศนิยม 99.89 ก็คืออ็อบเจกต์ที่มีค่าเท่ากับ 99.89 และมีชนิดข้อมูลเป็นแบบ float (ตัวเลขทศนิยม)
- ข้อความ "Hello Python" ก็คืออ็อบเจกต์ที่มีค่าเท่ากับ "Hello Python" และมีชนิดข้อมูลเป็นแบบ str (String : ข้อความ)

ชนิดข้อมูล (Type หรือ Data Type)

ชนิดข้อมูล (Data Type) ใช้เพื่อบอกให้ทราบว่า ข้อมูลที่ใช้งานอยู่นั้นมีคุณสมบัติอย่างไร และสามารถทำอะไรได้บ้าง เช่น ถ้าเป็นข้อมูลชนิดตัวเลขก็จะสามารถนำมาคำนวณทางคณิตศาสตร์ บวก ลบ คูณ หารได้ แต่ถ้าเป็นข้อมูลที่เป็นข้อความก็จะนำมาคำนวณไม่ได้

ชนิดข้อมูลที่ติดมากับ Python จะถูกจัดกลุ่มและมีการกำหนดโครงสร้างเป็นลำดับชั้น เพื่อบอกถึงความสัมพันธ์ของชนิดข้อมูลในแต่ละชนิดดังนี้

- **None** คือ ชนิดข้อมูลที่บอกให้ทราบว่าอ็อบเจกต์ที่สนใจอยู่นั้นยังไม่มีการกำหนดค่าใด ๆ ลงไป (คล้ายกับ null ในภาษาคอมพิวเตอร์อื่น ๆ)
- **Numbers** คือ ชนิดข้อมูลที่กำหนดให้กับตัวเลขแบบต่าง ๆ โดยชนิดข้อมูลที่อยู่ในกลุ่มของ Numbers ได้แก่ int (เลขจำนวนเต็ม) float (เลขทศนิยม) complex (จำนวนเชิงซ้อน) และ bool (ค่าความจริง)



คำสงวนของ Python

คำสงวน (Reserve Word) หรือคีย์เวิร์ด (Keyword) คือ คำที่ถูกสงวนไว้ใช้กับคำสั่งของ Python โดยเฉพาะ ไม่สามารถนำคำเหล่านี้มาใช้งานส่วนตัวได้ เช่น ไม่สามารถนำคำสงวนมาใช้ตั้งชื่อตัวแปร

คำสงวนที่ใช้ใน Python สังเกตว่าจะเป็นอักษรตัวพิมพ์เล็กทั้งหมด (หากใช้อักษรตัวพิมพ์ใหญ่ไม่ถือว่าเป็นคำสงวน)

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

ตัวอย่าง 3.2
ใช้คำสงวนกับการประกาศตัวแปรไม่ได้

ตัวอย่างข้อผิดพลาดที่เกิดจากการนำเอาคำสงวนมาใช้งาน เช่น นำคำสงวนมาใช้ตั้งชื่อตัวแปร

```
1 greeting = "Hello World"
2 assert = "Goodbye"
3 print(assert)
```

- บรรทัดที่ 1 ประกาศตัวแปร greeting เพื่อใช้อ้างอิงไปยังข้อความ "Hello World"
- บรรทัดที่ 2 ประกาศตัวแปร assert เพื่อใช้อ้างอิงไปยังข้อความ "Goodbye" ซึ่งบรรทัดนี้จะเกิดข้อผิดพลาดขึ้น (Error) เนื่องจาก assert เป็นคำสงวนที่ถูกใช้โดย Python

หลักเกณฑ์การตั้งชื่อตัวแปรใน Python

การตั้งชื่อตัวแปรจะต้องเป็นไปตามข้อกำหนดของ Python โดยมีหลักเกณฑ์ดังนี้

- การตั้งชื่อตัวแปร หากใช้อักษรตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็กต่างกัน จะถือว่าเป็นตัวแปรคนละตัวกัน (Case Sensitive) เช่น ตัวแปรชื่อว่า my_data จะเป็นตัวแปรคนละตัวกับ MY_DATA และตัวแปร time ก็จะเป็นตัวแปรคนละตัวกับ Time
- ชื่อตัวแปรไม่สามารถขึ้นต้นด้วยตัวเลข เช่น ไม่สามารถตั้งชื่อตัวแปรเป็น 9_data หรือ 8_students ได้
- ชื่อตัวแปรจะขึ้นต้นด้วยตัวอักษรหรือจะใช้ขีดล่าง (_) เช่น all_students, data_9 หรือ _books
- ชื่อตัวแปรจะต้องไม่ตรงกับคำสงวนที่ใช้ใน Python เช่น ไม่สามารถตั้งชื่อตัวแปรเป็น if, while หรือ break

พื้นฐานเกี่ยวกับสตริง (String)

ข้อความที่ใช้ในโปรแกรม คือ ชุดของตัวอักษรที่เรียงต่อกัน โดยมีลำดับที่แน่นอน (Sequences) ซึ่งในภาษาคอมพิวเตอร์ส่วนใหญ่จะเรียกชุดของตัวอักษรเช่นนี้ว่า **สตริง (String)** (ภาษาไทยจะใช้คำว่า สายอักขระ)

▲ รูปแสดงชุดของตัวอักษรที่เรียงต่อกันเป็นสตริง

สามารถเลือกตัวอักษรที่อยู่ในสตริงมาใช้งาน โดยระบุตำแหน่งของตัวอักษรที่อยู่ในสตริง ซึ่งเรียกตำแหน่งที่ใช้อ้างอิงลำดับตัวอักษรในสตริงว่า **อินเด็กซ์ (Index)** โดยจะเริ่มจาก 0 นับจากซ้ายไปขวา ดังรูป

▲ รูปแสดงอินเด็กซ์ที่ใช้บอกตำแหน่งของแต่ละตัวอักษรในสตริง

จากรูป อักษรตัวแรก คือ H จะมีค่าอินเด็กซ์เป็น 0 อักษร W จะมีค่าอินเด็กซ์เท่ากับ 6 ส่วนตำแหน่งสุดท้ายของสตริงจะมีค่าอินเด็กซ์เป็น 10

ตัวอย่าง 3.6
การใช้อินเด็กซ์ในสตริง

การใช้อินเด็กซ์เพื่อเลือกอักษรที่ต้องการจากสตริง

```
1 greeting = "Hello World"
2 first_character = greeting[0]
3 last_character = greeting[10]
4 print(first_character)    # H
5 print(last_character)     # d
6 print("Goodbye World"[8]) # W
```

- บรรทัดที่ 1 ประกาศตัวแปร greeting เก็บสตริง (มีชนิดข้อมูลเป็นแบบ str)
- บรรทัดที่ 2 ประกาศตัวแปร first_character เก็บตัวอักษร โดยเข้าไปยังตัวแปร greeting แล้วเลือกตัวอักษรที่มีค่าอินเด็กซ์เท่ากับ 0
- บรรทัดที่ 3 ประกาศตัวแปร last_character เก็บตัวอักษร โดยเข้าไปยังตัวแปร greeting แล้วเลือกตัวอักษรที่มีค่าอินเด็กซ์เท่ากับ 10

บทที่ 11
พื้นฐานการใช้งาน Error Handling

การรันคำสั่งบางอย่างอาจมีข้อผิดพลาดเกิดขึ้น เนื่องจากเงื่อนไขหรือสถานะที่แตกต่างกัน ตัวอย่างเช่น การดาวน์โหลดไฟล์จากอินเทอร์เน็ต หากดาวน์โหลดไฟล์ได้สมบูรณ์ก็จะสามารถใช้งานไฟล์นี้ได้ เช่น บันทึกไฟล์ลงในเครื่อง หรือเปิดขึ้นมาแก้ไข แต่ถ้าดาวน์โหลดไฟล์ไม่สำเร็จ การบันทึกไฟล์หรือการกระทำใด ๆ กับไฟล์จะทำให้เกิดข้อผิดพลาดและโปรแกรมทำงานผิดปกติ หรือไม่สามารถทำงานต่อไปได้

ดังนั้น เพื่อป้องกันปัญหาที่อาจเกิดขึ้นในขณะรันโปรแกรม Python จึงกำหนด try-except สำหรับใช้ตรวจสอบข้อผิดพลาด และเลือกรันชุดคำสั่งที่เหมาะสม

การใช้ try-except ใน Python

try-except ใช้เพื่อตรวจสอบชุดคำสั่ง และจัดการกับข้อผิดพลาดที่อาจเกิดขึ้น ซึ่งรูปแบบพื้นฐานของการใช้ try-except จะมีลักษณะเป็นบล็อก ดังนี้

```
1 try:
2     # statement
3 except TypeOfError:
4     # statement
5 else:
6     # statement
7 finally:
8     # statement
```

- **try** ใช้กำหนดบล็อกสำหรับใช้รันโค้ดที่อาจมีข้อผิดพลาดเกิดขึ้นได้
- **except** ใช้กำหนดบล็อกคำสั่ง หากเกิดข้อผิดพลาดตามที่กำหนด ก็จะรันชุดคำสั่งในบล็อกนี้ สามารถกำหนด except ได้หลายบล็อกเพื่อตรวจสอบประเภทหรือชนิดของข้อผิดพลาดที่แตกต่างกัน
- **else** ใช้กำหนดบล็อกคำสั่ง หากคำสั่งในบล็อก try ไม่เกิดข้อผิดพลาดใด ๆ (บล็อก else นี้ถือว่าเป็นตัวเลือกจะกำหนดหรือไม่กำหนดก็ได้)
- **finally** ใช้กำหนดบล็อกคำสั่ง ซึ่งคำสั่งในบล็อกนี้จะถูกเรียกใช้งานเสมอ ไม่ว่าบล็อก try จะเกิดข้อผิดพลาดหรือไม่ (บล็อก finally นี้ถือว่าเป็นตัวเลือกจะกำหนดหรือไม่กำหนดก็ได้)

การใช้ try-except บล็อก มีข้อดี คือ แม้ว่าจะมีข้อผิดพลาดเกิดขึ้นก็สามารถเขียนโค้ดคำสั่ง เพื่อให้โปรแกรมสามารถทำงานต่อได้ เช่น เมื่อดึงข้อมูลจากฐานข้อมูลมาแสดงบนหน้าจอ หากไม่มีการเชื่อมต่ออินเทอร์เน็ตก็ให้แสดงข้อความบอกให้ผู้ใช้เชื่อมต่ออินเทอร์เน็ต หรือหากดึงข้อมูลจากฐานข้อมูลไม่สำเร็จก็จะแสดงข้อความบอกให้ผู้ใช้ลองใหม่อีกครั้ง

ชนิดของข้อผิดพลาดแบบต่าง ๆ

ข้อผิดพลาดที่เกิดขึ้นมีได้หลายแบบ ซึ่ง Python ได้เตรียมอ็อบเจกต์สำหรับเก็บข้อผิดพลาดในแบบต่าง ๆ เอาไว้ ข้อผิดพลาดที่มักเกิดขึ้นบ่อย มีดังนี้

- **SyntaxError** ข้อผิดพลาดที่เกิดจากการเขียนชุดคำสั่งซึ่งผิดจากไวยากรณ์ที่ Python กำหนด เช่น การไม่ใส่วงเล็บหลังชื่อฟังก์ชัน
- **IndentationError** ข้อผิดพลาดที่เกิดจากการจัดการเยื้องของโค้ดคำสั่งที่ไม่ถูกต้อง
- **TypeError** ข้อผิดพลาดที่เกิดจากการกำหนดชนิดข้อมูลไม่ถูกต้อง เช่น การนำข้อความมาใช้คำนวณกับตัวเลข
- **IndexError** ข้อผิดพลาดที่เกิดจากการใช้หมายเลขอินเด็กซ์ที่เกินขอบเขตของ List หรือ Tuple
- **KeyError** ข้อผิดพลาดที่เกิดจากการใช้ชื่อคีย์ของ Dictionary ที่ไม่ถูกต้อง
- **NameError** ข้อผิดพลาดที่เกิดจากการอ้างอิงชื่อตัวแปรที่ยังไม่ได้กำหนดค่า หรือไม่อยู่ในสโคปที่กำหนด
- **ValueError** ข้อผิดพลาดที่เกิดจากการผ่านค่าที่ไม่ถูกต้องไปยังฟังก์ชัน เช่น ฟังก์ชันต้องการอาร์กิวเมนต์เป็นแบบ int แต่ผ่านค่าในแบบ tuple
- **ZeroDivisionError** ข้อผิดพลาดที่เกิดจากการหารตัวเลขด้วย 0 (ศูนย์)
- **ImportError** ข้อผิดพลาดที่เกิดจากการอิมพอร์ตโมดูลไม่ถูกต้อง
- **FileNotFoundError** ข้อผิดพลาดที่เกิดจากการอ้างอิงไปยังไฟล์ไม่ถูกต้องหรือไม่มีอยู่จริง

ตัวอย่างข้อผิดพลาดที่มักพบบ่อย คือ IndexError ซึ่งเกิดจากการอ้างอิง Index ที่เกินขอบเขตของ List (หรือ Tuple) ดังตัวอย่าง

```
1 my_devices = [‘iPhone 14’, ‘iPad Pro’, ‘Macbook Pro 2023’]
2 current_device = my_devices[3]
```

- บรรทัดที่ 1 ประกาศตัวแปร my_devices เก็บรายชื่ออุปกรณ์
- บรรทัดที่ 2 เกิดข้อผิดพลาดชนิด IndexError เนื่องจากคำสั่ง my_devices[3] ใช้ค่าอินเด็กซ์เท่ากับ 3 ซึ่งเกินขอบเขตของ my_devices

ตัวอย่าง 11.2
จัดการกับข้อผิดพลาดพร้อมกันหลายแบบ

ตัวอย่างการใช้ try-except เพื่อตรวจสอบข้อผิดพลาดที่อาจเกิดขึ้นหลาย ๆ แบบ โดยกำหนดบล็อก except จำนวนหลายบล็อก เพื่อตรวจสอบข้อผิดพลาดที่แตกต่างกัน หากพบข้อผิดพลาดใดในบล็อก try ก็ออกจากบล็อก try ทันที (ไม่รันคำสั่งอื่น ๆ ที่เหลือภายในบล็อก try)

```
1 try:
2     print(data)
3     result = 'MY TEXT' + 1
4 except TypeError:
5     print('Type is not match')
6 except NameError:
7     print('Can not reach that name!!!')
8 finally:
9     print('This is always print!!!')
```

- บรรทัดที่ 1 กำหนดบล็อก try
- บรรทัดที่ 2 พิมพ์ค่าในตัวแปร data ออกไปยังหน้าต่างคอนโซล ซึ่งบรรทัดนี้ผิดพลาดเนื่องจากตัวแปร data ยังไม่ได้อ้างอิงไปยังค่าใด ๆ จึงไม่สามารถพิมพ์ข้อมูลออกมาได้ เมื่อพบข้อผิดพลาดก็จะออกจากบล็อก try ทันที (ไม่รันคำสั่งอื่น ๆ ภายในบล็อก try)
- บรรทัดที่ 3 นำข้อความ 'MY TEXT' ไปบวกกับตัวเลข 1 ซึ่งบรรทัดนี้ผิดพลาด เนื่องจากไม่สามารถนำข้อความที่เป็นสตริงไปบวกกับตัวเลขจำนวนเต็ม 1 ได้
- บรรทัดที่ 4 กำหนดบล็อก except และกำหนดชนิดของข้อผิดพลาดเป็น TypeError เพื่อตรวจสอบข้อผิดพลาดจากการใช้ชนิดข้อมูลผิด
- บรรทัดที่ 6 กำหนดบล็อก except เพิ่ม โดยกำหนดชนิดของข้อผิดพลาดเป็น NameError เพื่อตรวจสอบข้อผิดพลาดจากการอ้างอิงไปยังชื่อตัวแปรที่ไม่ถูกต้อง
- บรรทัดที่ 8 บล็อก finally สำหรับกำหนดชุดคำสั่งที่จะถูกรันเสมอ ไม่ว่าจะเกิดข้อผิดพลาดหรือไม่เกิดข้อผิดพลาดจากบล็อก try หรือไม่

จากตัวอย่าง เมื่อดูผลลัพธ์ที่หน้าต่างคอนโซลก็ จะมีข้อความอธิบายข้อผิดพลาดที่เกิดจากการอ้างอิงไปยังชื่อตัวแปรที่ผิดพลาด (NameError) และพบข้อความในส่วนของ finally ด้วย

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL
thinkbeyond@MacBook-Pro-khxng-ThinkBeyond firstProject % /usr/local/bin
/python3 /Users/thinkbeyond/dev/python_dev/firstProject/main.py
Can not reach that name!!!
This is always print!!!
thinkbeyond@MacBook-Pro-khxng-ThinkBeyond firstProject %
```

บทที่ 14
คลาสและอ็อบเจกต์

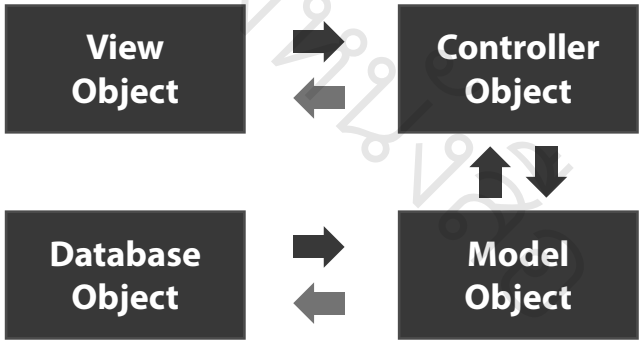
Python เป็นภาษาคอมพิวเตอร์ที่ถูกออกแบบมาสำหรับการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming: OOP) โดยเฉพาะ ดังนั้นส่วนประกอบต่าง ๆ ใน Python ไม่ว่าจะเป็นข้อมูล ชนิดข้อมูล ฟังก์ชัน คลาส ต่างก็เป็นอ็อบเจกต์ทั้งสิ้น

บทนี้จะเป็นพื้นฐานเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุในภาษา Python เริ่มตั้งแต่แนะนำวิธีสร้างคลาส (Class) การสร้างอ็อบเจกต์จากคลาส (Object) การกำหนดค่าให้กับแอตทริบิวต์ (Attribute) การกำหนดพรอป-เพอร์ตี (Property) รวมถึงวิธีประกาศและใช้งานเมธอด (Method) แบบต่าง ๆ เช่น Instance Method, Class Method และ Static Method

การเขียนโปรแกรมเชิงวัตถุ

การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming: OOP) เป็นวิธีเขียนโปรแกรมที่แบ่งส่วนประกอบของโปรแกรมออกเป็นส่วนประกอบย่อย ๆ โดยแต่ละส่วนจะเรียกว่า **อ็อบเจกต์** แต่ละอ็อบเจกต์มีคุณสมบัติและความสามารถตามที่กำหนดไว้

นักพัฒนาโปรแกรมจะสร้างส่วนประกอบย่อย ๆ (อ็อบเจกต์) เช่น อ็อบเจกต์สำหรับแสดงผล อ็อบเจกต์สำหรับติดต่อกับฐานข้อมูล อ็อบเจกต์สำหรับคำนวณค่า จากนั้นจึงนำเอาอ็อบเจกต์มาจับรวมกัน สร้างความสัมพันธ์ หรือเชื่อมโยงอ็อบเจกต์เหล่านั้นเข้าด้วยกัน จนกลายมาเป็นโปรแกรม

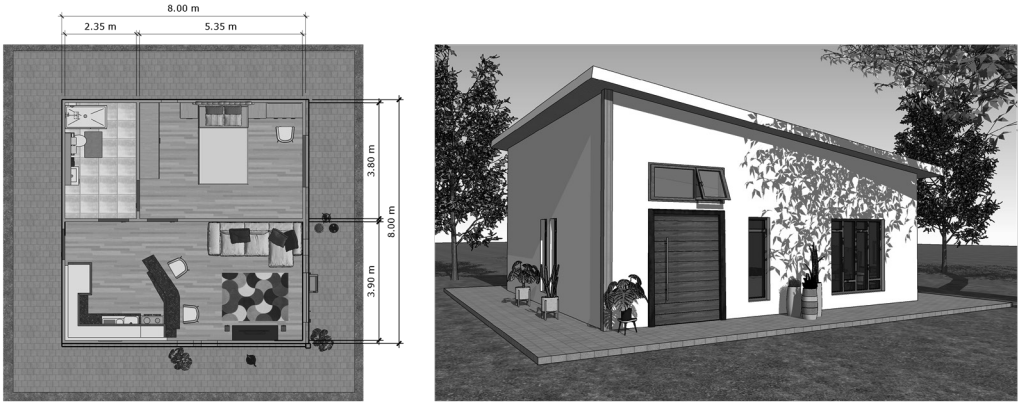


▲ รูปแสดงตัวอย่างการนำอ็อบเจกต์ต่าง ๆ มารวมกันเป็นโปรแกรม

คลาส (Class) และอ็อบเจกต์ (Object)

การเขียนโปรแกรมเชิงวัตถุจะใช้คลาส (Class) เป็นแม่แบบหรือเป็นเทมเพลตสำหรับสร้างอ็อบเจกต์ (Object) คลาสสามารถเปรียบได้กับแบบแปลนหรือพิมพ์เขียวของบ้าน ส่วนตัวบ้านก็คืออ็อบเจกต์ที่สร้างตามแบบแปลนที่ออกแบบไว้นั่นเอง

แบบแปลนบ้านหนึ่งชุดสามารถสร้างบ้านได้หลายหลัง เช่นเดียวกับ Class ที่สามารถใช้เป็นแม่แบบสำหรับสร้างอ็อบเจกต์ประเภทเดียวกันได้หลาย ๆ ตัว



▲ รูปแสดงตัวอย่างแปลนบ้าน (Class) และตัวอย่างบ้านที่สร้างจากแปลน (Object)

อ็อบเจกต์แต่ละตัวจะมีคุณสมบัติประจำตัวที่แตกต่างกัน เช่น อ็อบเจกต์ประตูก็จะมีคุณสมบัติประจำตัว เช่น มีกลอนประตู มีสีของประตู มีขนาดความกว้างความสูง เรียกคุณสมบัติประจำตัวของอ็อบเจกต์เช่นนี้ว่า **พรอปเพอร์ตี (Property)**

นอกจากนั้นอ็อบเจกต์ยังอาจมีความสามารถหรือพฤติกรรม (Behavior) ที่แตกต่างกัน เช่น อ็อบเจกต์ประตูจะมีความสามารถในการเปิดประตูและปิดประตู หากต้องการใช้ความสามารถเหล่านี้ของอ็อบเจกต์จะใช้งานผ่านฟังก์ชันที่เรียกว่า **เมธอด (Method)**

ความสัมพันธ์ระหว่าง class และ type

class เป็นแม่แบบสำหรับใช้สร้างอ็อบเจกต์ แต่เนื่องจากทุก ๆ ส่วนของ Python คืออ็อบเจกต์ ดังนั้น class ถือเป็นอ็อบเจกต์ชนิดหนึ่งที่ถูกสร้างมาจากคลาส type ดังรูป

จากรูป อ็อบเจกต์ถูกสร้างมาจาก class ส่วน class ก็ถูกสร้างมาจาก type อีกทอดหนึ่ง ด้วยเหตุนี้จึงมักเรียก class ว่า type

เมื่อใช้ฟังก์ชัน type() ตรวจสอบคลาส str, int, float ก็จะได้ผลลัพธ์เป็น type บอกให้ทราบว่าทุก ๆ class ที่ใช้ใน Python จะมีชนิดข้อมูลเป็นแบบ type นั่นเอง

```
1 print(type(str))    # <class 'type'>
2 print(type(int))    # <class 'type'>
3 print(type(float))  # <class 'type'>
```

- บรรทัดที่ 1 ใช้ฟังก์ชัน type ตรวจสอบคลาส str จะได้ผลลัพธ์เป็น <class ‘type’> ซึ่งบอกให้ทราบว่า คลาส str ถูกสร้างมาจากคลาส type
- บรรทัดที่ 2 ใช้ฟังก์ชัน type ตรวจสอบคลาส int จะได้ผลลัพธ์เป็น <class ‘type’> เช่นเดียวกัน
- บรรทัดที่ 3 ใช้ฟังก์ชัน type ตรวจสอบคลาส float จะได้ผลลัพธ์เป็น <class ‘type’>

การประกาศคลาส

การประกาศคลาสใน Python จะหมายถึง การสร้างชนิดข้อมูล (type) แบบใหม่ขึ้นมาใช้งาน

การประกาศคลาสจะใช้คีย์เวิร์ด class ตามด้วยชื่อคลาส และปิดท้ายด้วยเครื่องหมาย : จากนั้นให้ใส่ชุดคำสั่งภายในบล็อกของคลาส หากยังไม่ต้องการกำหนดรายละเอียดใด ๆ ไปยังคลาส ก็ให้ใส่คำสั่ง pass

```
1 class MyClass:
2     pass
```

- บรรทัดที่ 1 ประกาศ class โดยใช้ชื่อว่า MyClass
- บรรทัดที่ 2 ใส่คำสั่ง pass เนื่องจากยังไม่ต้องการลงรายละเอียดภายในคลาส

หลังจากประกาศคลาส เมื่อโค้ดนี้ถูกคอมไพล์ คลาสอ็อบเจกต์ก็จะถูกสร้างขึ้นและอยู่ในสโคประดับโมดูล ดังนั้นจึงเป็น Global Scope ที่ทุก ๆ แห่งภายในไฟล์สามารถเรียกใช้งานคลาสอ็อบเจกต์นี้ได้

เมื่อใช้ฟังก์ชัน type() ทดสอบคลาสที่เพิ่งสร้างขึ้นพบว่า คลาสมีชนิดข้อมูลเป็นแบบ type เช่นเดียวกับข้อมูลชนิดอื่น ๆ ของ Python

```
1 class MyClass:
2     pass
3
4 print(type(MyClass))    # <class ‘type’>
```

- บรรทัดที่ 4 เรียกฟังก์ชัน type() โดยผ่านค่า MyClass เข้าไป ผลลัพธ์ที่ได้ คือ ชนิดข้อมูลในแบบ type

อินสแตนส์ (instance)

คลาส (class) จะมีความสามารถอย่างหนึ่งที่ติดมาด้วย คือ คลาสสามารถถูกเรียกใช้งานได้ (เรียกว่า **Callable**) เมื่อมีการเรียกใช้งานคลาส ผลลัพธ์ที่ได้ คือ อ็อบเจกต์

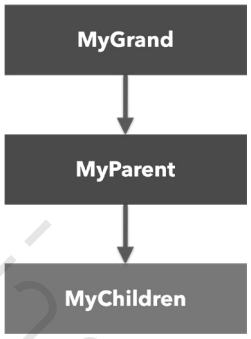
ตรวจสอบลำดับชั้นด้วยฟังก์ชัน issubclass()

ใช้ฟังก์ชัน issubclass() เพื่อตรวจสอบว่าคลาสที่สนใจอยู่ภายใต้คลาสอื่น ๆ หรือไม่ การใช้ฟังก์ชัน issubclass() จะตรวจสอบจากชื่อคลาส ดังนั้นจึงไม่ต้องสร้าง instance ขึ้นมาก่อน

```
issubclass(sub_name, class_name)
```

- **sub_name** คือ ชื่อคลาสที่ต้องการตรวจสอบ
- **class_name** คือ ชื่อของคลาสที่อยู่เหนือขึ้นไป

ตัวอย่างการตรวจสอบลำดับการจัดเรียงคลาสโดยใช้ฟังก์ชัน issubclass

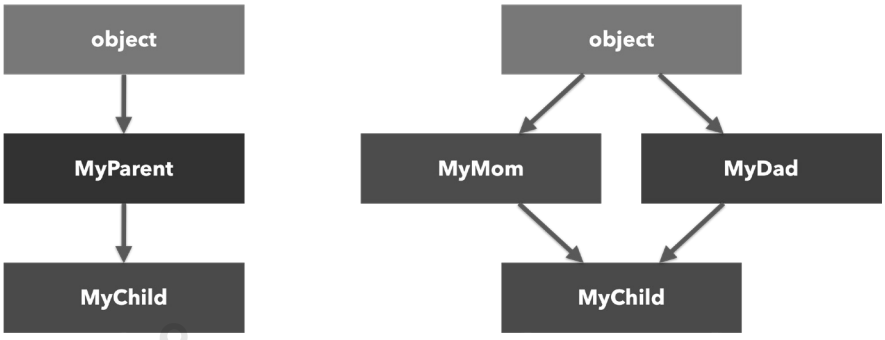



```
1  class MyGrand:
2      pass
3
4  class MyParent(MyGrand):
5      pass
6
7  class MyChildren(MyParent):
8      pass
9
10 check_1 = issubclass(MyChildren, MyParent)
11 check_2 = issubclass(MyChildren, MyGrand)
12 print(check_1)  # True
13 print(check_2)  # True
```

- บรรทัดที่ 10 ตรวจสอบว่า MyChildren เป็น subclass ของ MyParent หรือไม่
- บรรทัดที่ 11 ตรวจสอบว่า MyChildren เป็น subclass ของ MyGrand หรือไม่

การสืบทอดจากหลายคลาสพร้อมกัน (Multiple Inheritance)

Python รองรับการสืบทอดคลาสได้ 2 แบบ ได้แก่ Single Inheritance และ Multiple Inheritance



Single Inheritance คือ การสืบทอดคลาสจากคลาสเดียว ซึ่งทุกคลาสของ Python จะสืบทอดมาจากคลาส object เสมอ

Multiple Inheritance คือ การสืบทอดจากหลายคลาสพร้อมกัน แอตทริบิวต์หรือเมธอดต่าง ๆ จากทุก ๆ Parent จะถูกส่งต่อไปให้กับ Children ทั้งหมด

เมื่อต้องการสืบทอดจากหลาย ๆ คลาส ก็ให้ระบุชื่อคลาสที่ต้องการสืบทอดทั้งหมดใส่ไว้ภายในวงเล็บ ดังตัวอย่าง

```
1 class MyMom:
2     pass
3
4 class MyDad:
5     pass
6
7 class MyChild(MyMom, MyDad):
8     pass
```

- บรรทัดที่ 1 ประกาศคลาส MyMom ถ้าไม่กำหนดว่าสืบทอดมาจากคลาสไหน Python จะถือว่าสืบทอดมาจากคลาส object เสมอ
- บรรทัดที่ 4 ประกาศคลาส MyDad
- บรรทัดที่ 7 ประกาศคลาส MyChild โดยสืบทอดทั้งจากคลาส MyMom และ MyDad

การใช้ฟังก์ชัน issubclass() เพื่อตรวจสอบว่าคลาสปัจจุบันเป็น subclass ของคลาสที่สนใจหรือไม่นั้น สามารถตรวจสอบพร้อมกันหลายคลาสได้ ดังตัวอย่าง