เอกสารประกอบการสอนรายวิชา
477-201 การเขียนโปรแกรมคอมพิวเตอร์
การเขียนโปรแกรมภาษา Python เบื้องต้น
(Basic Python Programming)

ดร. จันทวรรณ ปิยะวัฒน์

สาขาวิชาระบบสารสนเทศทางธุรกิจ

ภาควิชาบริหารธุรกิจ คณะวิทยาการจัดการ

มหาวิทยาลัยสงขลานครินทร์

ภาคการศึกษาที่ 1/2562

คำนำ

เอกสารประกอบการสอนเล่มนี้ จัดทำขึ้นสำหรับการสอนรายวิชา 477-201 การเขียนโปรแกรม คอมพิวเตอร์ (Computer Programming) ในภาคการศึกษาที่ 1 ปีการศึกษา 2562 ซึ่งเป็นรายวิชาบังคับของนักศึกษาหลักสูตรระบบสารสนเทศทางธุรกิจ ชั้นปีที่ 2 ภาควิชาบริหารธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ จำนวน 3 หน่วยกิต 3(2-2-5) เป็นการสอนทฤษฎี 2 ชั่วโมงต่อสัปดาห์ ปฏิบัติ 2 ชั่วโมงต่อสัปดาห์ และนักศึกษาควรศึกษาค้นคว้าด้วยตัวเอง 5 ชั่วโมงต่อสัปดาห์

วิชา 477-201 การเขียนโปรแกรมคอมพิวเตอร์ มีจุดมุ่งหมายให้นักศึกษาได้มีพื้นฐาน ความรู้ความเข้าใจในหลักการเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นด้วยภาษา Python ส่วนประกอบ ต่างๆของโปรแกรมคอมพิวเตอร์และภาษา Python สามารถเขียนโปรแกรมคอมพิวเตอร์พื้นฐาน ด้วยภาษา Python ได้ตามการวิเคราะห์และออกแบบขั้นตอนการทำงานของโปรแกรมอย่างมีระบบ สามารถเขียนโปรแกรมแบบมีเงื่อนไขเพื่อการตัดสินใจ เขียนคำสั่งเพื่อให้โปรแกรมทำงานวนซ้ำได้ และเข้าใจการใช้งานโมดูลส่วนเสริมต่างๆ ของโปรแกรมภาษา Python เพื่อนำความรู้เหล่านี้ไปใช้ ในการเขียนโปรแกรมระดับในระดับที่ยากขึ้น ซึ่งได้แก่ การเขียนโปรแกรมแบบฟังก์ชันและ การเขียนโปรแกรมเชิงวัตถุได้

หนังสือเล่มนี้ได้จัดแบ่งเนื้อหาออกเป็น 11 บท ในแต่ละบทจะมีแบบฝึกหัดท้ายบทเพื่อให้ผู้เรียน ได้ลองวิเคราะห์และออกแบบแนวทางแก้ไขปัญหาและพัฒนาออกมาเป็นโปรแกรมด้วยภาษา Python ที่ได้เรียนรู้ไปแล้วได้ ทั้งนี้ผู้จัดทำหวังเป็นอย่างยิ่งว่าเอกสารประกอบการสอนฉบับนี้จะให้ความรู้และเป็นประโยชน์แก่ผู้เรียนและผู้อ่านทุกๆ ท่าน เพื่อสร้างความรู้ความเข้าใจในการฝึกเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นให้ดียิ่งขึ้น หากมีข้อเสนอแนะประการใด ผู้จัดทำขอรับไว้ด้วยความขอบพระคุณยิ่ง

ดร.จันทวรรณ ปิยะวัฒน์

สาขาวิชาระบบสารสนเทศทางธุรกิจ

ภาควิชาบริหารธุรกิจ คณะวิทยาการจัดการ

มหาวิทยาลัยสงขลานครินทร์

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1

แผนการสอน

1.1 คำอธิบายรายวิชาและวัตถุประสงค์ที่ระบุไว้ในหลักสูตร

แนวความคิดเรื่องการเขียนโปรแกรม ขั้นตอนวิธีในการแก้ไขปัญหา การสร้างคำสั่งสำหรับเขียนขั้นตอนวิธีการ เขียนผังงาน นิพจน์ คำสั่งในการเขียนโปรแกรม หลักไวยากรณ์ของภาษาโปรแกรมระดับสูง การเขียนโปรแกรมสมัยใหม่ การทดสอบ การแก้ไขโปรแกรม การติดตั้ง และการเขียนเอกสารประกอบโปรแกรม

Concept of programming, algorithm to solve the problem, flowchart, expression and instruction, high-level language syntax, modern programming, testing, debugging, installation and software documentation

1.2 วัตถุประสงค์ของวิชา

มีจุดมุ่งหมายให้นักศึกษาได้มีพื้นฐานความรู้ความเข้าใจในหลักการเขียนโปรแกรมคอมพิวเตอร์เบื้องต้นด้วยภาษา Python ส่วนประกอบต่างๆ ของโปรแกรมคอมพิวเตอร์และภาษา Python สามารถเขียนโปรแกรมคอมพิวเตอร์อย่างง่ายด้วยภาษา Python ได้ตามการวิเคราะห์และออกแบบขั้นตอนการทำงานของโปรแกรมอย่างมีระบบ และมีความรู้ความเข้าใจในการเขียนโปรแกรมแบบมีเงื่อนไขเพื่อการตัดสินใจ การเขียนคำสั่งเพื่อการทำงานซ้ำ และโมดูลส่วนเสริมต่างๆ ของโปรแกรมภาษา Python เพื่อเรียนรู้เรื่องการเขียนโปรแกรมแบบฟังก์ชันและการเขียนโปรแกรมเชิงวัตถุได้

1.3 เนื้อหาวิชา

1.3.1 สัปดาห์ที่ 1

สัปดาห์ที่ 1

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

เค้าโครงวิชา

- วัตถุประสงค์รายวิชา
- รายละเอียดเนื้อหาวิชา
- การวัดผลและการประเมินผล
- เงื่อนไขและข้อตกลงอื่น
- วิธีการเรียนการสอน
- เว็บไซต์และหนังสืออ่านประกอบ

ระบบจัดการการเรียนรู้ (ClassStart.org)

- ระบบในภาพรวม
- การสมัครสมาชิก
- การเข้าห้องเรียนออนไลน์ของรายวิชา
- การใช้งานระบบ
- การเข้าอ่านเอกสารการสอนและคลิป
- การส่งแบบฝึกหัดทางออนไลน์
- การทำข้อสอบออนไลน์
- การตรวจสอบคะแนนเก็บ

- การบันทึกการเรียนรู้ (Reflections)
- การสื่อสารออนไลน์

เว็บไซต์ Code.org

- การสมัครสมาชิก
- ฝึกการเขียนโปรแกรมง่าย ๆ (Game-based Learning) แบบ Block-based Programming

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- บรรยาย
- ปฏิบัติการใช้ระบบ ClassStart.org
- ปฏิบัติการเขียนโปรแกรมทางออนไลน์ที่ Code.org

1.3.2 สัปดาห์ที่ 2

สัปดาห์ที่ 2

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- X

1.3.3 สัปดาห์ที่ 3

สัปดาห์ที่ 3

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.4 สัปดาห์ที่ 4

สัปดาห์ที่ 4

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.5 สัปดาห์ที่ 5

สัปดาห์ที่ 5

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.6 สัปดาห์ที่ 6

สัปดาห์ที่ 6

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.7 สัปดาห์ที่ 7

สัปดาห์ที่ 7

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.8 สัปดาห์ที่ 8

สัปดาห์ที่ 8

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.9 สัปดาห์ที่ 9

สัปดาห์ที่ 9

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.10 สัปดาห์ที่ 10

สัปดาห์ที่ 10

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.11 สัปดาห์ที่ 11

สัปดาห์ที่ 11

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.12 สัปดาห์ที่ 12

สัปดาห์ที่ 12

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.13 สัปดาห์ที่ 13

สัปดาห์ที่ 13

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.14 สัปดาห์ที่ 14

สัปดาห์ที่ 14

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

1.3.15 สัปดาห์ที่ 15

สัปดาห์ที่ 15

ผู้สอน จันทวรรณ ปิยะวัฒน์

จำนวนชั่วโมงบรรยาย 2

จำนวนชั่วโมงปฏิบัติ 2

หัวข้อ/รายละเอียด

กิจกรรมการเรียนการสอน/สื่อที่ใช้

- x

บทที่ 2

ความรู้เบื้องต้นเกี่ยวกับ Python

2.1 Python คืออะไร

ในปี ค.ศ. 1980 Mr. Guido van Rossum ได้พัฒนาภาษาโปรแกรมมิ่งขึ้นมาและให้ชื่อว่าภาษา Python และเผยแพร่ให้ใช้งานสู่สาธารณชนในปี ค.ศ. 1991 (Guido, 2019) Python เป็นภาษาโปรแกรมคอมพิวเตอร์ระดับสูงซึ่งไวยากรณ์ของภาษาระดับสูงนี้จะใกล้เคียงคำในภาษาอังกฤษทั่วไป (Downey, 2015) Python ถูกใช้ในการสร้างโมบายแอพพลิเคชั่น เว็บไซต์ เว็บแอพพลิเคชั่น ออนไลน์เซอร์วิส รวมทั้งใช้ใน การวิเคราะห์ข้อมูลและคำนวณ ทางคณิตศาสตร์และวิทยาศาสตร์อย่างแพร่หลาย ตัวอย่าง ออนไลน์เซอร์วิสที่พัฒนาขึ้นด้วยภาษา Python ได้แก่ Instagram, Uber, Pinterest, Reddit, Spotify และ Dropbox (Shuup, 2019) โดยในระยะหลายปี ที่ผ่านมานี้ Python ได้รับความนิยมสูงขึ้นเรื่อยๆ โดยในเดือนมิถุนายน 2562 ดัชนีความนิยมภาษาโปรแกรมมิ่ง TIOBE ได้แสดงให้เห็นว่า Python เป็นภาษาโปรแกรมมิ่ง ที่ได้รับความนิยมเป็นอันดับที่ 3 เทียบกับภาษาโปรแกรมมิ่งอื่นๆ และมีความนิยมสูงสุดในรอบ 19 ปี (TIOBE, 2019)

รูปที่ 2.1: TIOBE Index for Python ในปี พ.ศ. 2562

หากเปรียบเทียบกับภาษาโปรแกรมมิ่งอื่น ๆ แล้ว Python มีไวยากรณ์ภาษา (Syntax) ที่สามารถอ่านง่าย เข้าใจได้ง่าย และเรียนรู้ง่าย Python จึงเป็นภาษาที่เหมาะสมสำหรับการสอนการเขียนโปรแกรมโดยเฉพาะอย่างยิ่งในระดับเบื้องต้น อีกทั้งยังเป็นภาษาที่ยืดหยุ่นสามารถพัฒนาได้บน ระบบปฏิบัติการที่หลากหลาย อาทิ Windows, Linux, OS/2, MacOS, iOS และ Android นอกจากนี้ โปรแกรมเมอร์ทั่วโลกได้พัฒนาไลบรารี (Libraries) ขึ้นมาจำนวนมากสำหรับต่อยอด การทำงานของภาษา Python พื้นฐาน เช่น Django, Numpy, Pandas, Matplotlib, Flask, Web2py เป็นต้น (Foundation, 2019)

2.2 Python ทำงานอย่างไร

ภาษาโปรแกรมมิ่งระดับสูงจะต้องถูกโปรแกรมแปลภาษา เช่น คอมไพเลอร์ (Compiler) หรือ อินเทอร์พรีเตอร์ (Interpreter) ทำการแปลภาษาระดับสูงให้กลายเป็นภาษาเครื่องที่คอมพิวเตอร์เข้าใจก่อน (Lutz, 2013) ภาษาตระกูลที่ต้องใช้ Compiler เพื่อแปลงเป็นภาษาคอมพิวเตอร์ซึ่งเป็นภาษาที่มนุษย์อ่านไม่ออกแล้วจึงจะทำงานได้ เช่น ภาษา Java ภาษา C หรือภาษา C++ ภาษาพวกนี้จะได้โปรแกรมที่ทำงานรวดเร็วมาก แต่ก็ยากที่จะเรียนรู้ในช่วงการฝึกฝนการเขียน Programming ใหม่ๆ (Barry, 2016)

แต่สำหรับภาษา Python เมื่อได้ Source code ที่เป็นนามสกุลไฟล์ `.py` แล้ว โปรแกรมจะถูกคอมไพล์โดยคอมไพเลอร์ของ Python เพื่อแปลคำสั่ง Python ให้เป็นคำสั่งแบบ Bytecode และบันทึกไว้ในไฟล์นามสกุล `.pyc` ต่อมาเมื่อผู้ใช้ต้องการ Run ไฟล์นี้ อินเทอร์พรีเตอร์ (Interpreter) ก็จะแปลง Bytecode เป็นภาษาเครื่องสำหรับการดำเนินการโดยตรงบนฮาร์ดแวร์ (Beazley & Jones, 2013) อาจเรียกได้ว่า Python เป็นภาษาลูกครึ่งและเรียนรู้ได้ง่าย เหตุผลที่ Python ทำการคอมไพล์เป็น Bytecode เป็นรหัสกลางไว้ก่อนนั้น นั่นก็เพราะ Python ถูกออกแบบมาให้เป็นภาษาการเขียนโปรแกรมที่ไม่ขึ้นกับแพลตฟอร์ม ซึ่งหมายความว่ามีการเขียนโปรแกรมหนึ่งครั้ง แต่สามารถเรียกใช้งานบนอุปกรณ์ใดก็ได้ แต่จะต้องติดตั้ง Python เวอร์ชันที่เหมาะสม

2.3 อัลกอริทึม (Algorithm) และ ผังงาน (Flowchart)

อัลกอริทึม (Algorithm) หมายถึง กระบวนการทีละขั้นตอนเพื่อแก้ไขปัญหาที่กำหนดอย่างชัดเจน โดยทั่วไปจะมีสามขั้นตอนหลัก คือ มีการนำเข้าข้อมูลหรืออินพุต แล้วนำมาประมวลผล และแสดงผลลัพธ์ออกมา (Bouras, 2019) ตัวอย่างเช่น โจทย์ให้หาค่าเฉลี่ยของตัวเลขที่รับมาจากผู้ใช้จำนวนสามค่า จะสามารถเขียนเป็นขั้นตอนได้ดังนี้คือ

ขั้นตอนที่หนึ่ง การนำเข้าข้อมูล

1. แจ้งให้ผู้ใช้ป้อนหมายเลขที่หนึ่ง
2. แจ้งให้ผู้ใช้ป้อนหมายเลขที่สอง
3. แจ้งให้ผู้ใช้ป้อนหมายเลขที่สาม ขั้นตอนที่สอง การประมวลผลข้อมูล
4. คำนวณผลรวมของเลขทั้งสามจำนวน

5. หารผลรวมด้วยสาม ขั้นตอนที่สาม การแสดงผลลัพธ์
6. แสดงผลลัพธ์ออกทางหน้าจอ

ส่วนผังงาน (Flowchart) เป็นการนำเสนอการไหลของอัลกอริทึมในรูปแบบของสัญลักษณ์จากคำสั่งหนึ่งไปยังอีกต่อไปจนถึงจุดสิ้นสุดของอัลกอริทึม (Cormen, Leiserson, Rivest, & Stein, 2009) สัญลักษณ์ที่ใช้บ่อยสำหรับผังงานมีดังต่อไปนี้

รูปที่ 2.2: สัญลักษณ์ผังงาน (Flowchart)

ตัวอย่างการเขียนผังงานจากอัลกอริทึมด้านบนสามารถเขียนได้ดังนี้

รูปที่ 2.3: ตัวอย่างการเขียนผังงาน

2.4 การติดตั้งโปรแกรม Python Runtime

การติดตั้งโปรแกรม Python Runtime คือการติดตั้งโปรแกรมที่ทำให้นักพัฒนาซอฟต์แวร์สามารถใช้งานโปรแกรมที่เขียนขึ้นด้วยภาษา Python เองได้ ให้เข้าที่เว็บไซต์ `https://www.python.org/` (Foundation, 2019) ไปที่ Download for Windows แล้วเลือก Python 3.7.0 แล้วทำการติดตั้งให้เรียบร้อยลงในเครื่อง และให้เลือก Add Python 3.7 To Path เพื่อที่จะสามารถใช้ Python ได้ที่ Command Line หลังจากนั้นจะเห็นได้ว่าที่สตาร์ทเมนูโปรแกรม Python 3.7 จะถูกสร้างขึ้น ในโฟลเดอร์นี้จะมีโปรแกรมชื่อว่า Idle ซึ่งเป็น Integrated Development Environment หรือ เครื่องมือที่ช่วยในการพัฒนาโปรแกรมที่ใช้งานง่ายๆ เหมาะแก่การเขียนโปรแกรมเบื้องต้น โดยจะมีทั้ง Text Editor และ Interactive Shell เวลาใช้งานควรเปิดไว้ 2 หน้าต่าง ด้านซ้ายมือเป็น Source code ด้านขวามือเป็น Python Shell เพื่อใช้ดูผลลัพธ์ในการ Run โปรแกรมที่เขียนขึ้น

รูปที่ 2.4: การ Download Python 3.7

รูปที่ 2.5: เลือก Add Python 3.7 to PATH

รูปที่ 2.6: โฟลเดอร์ Python 3.7

รูปที่ 2.7: ตัวอย่างหน้าโปรแกรม Idle หน้า Editor

รูปที่ 2.8: ตัวอย่างหน้าโปรแกรม Idle หน้า Python Shell

บทที่ 3

ส่วนประกอบต่าง ๆ ของภาษา Python

3.1 ตัวแปร (Variables)

ตัวแปร (Variables) คือชื่อที่กำหนดขึ้นสำหรับใช้เก็บค่าในหน่วยความจำของเครื่องคอมพิวเตอร์

3.2 การตั้งชื่อตัวแปร

การตั้งชื่อตัวแปรมีเงื่อนไขดังนี้

1. ให้ขึ้นต้นด้วยอักษรตัวภาษาอังกฤษตัวใหญ่หรือตัวเล็กตั้งแต่ Aa ถึง Zz เท่านั้น
2. ประกอบด้วยตัวอักษรหรือตัวเลข 0 ถึงเลข 9 หรือตัวขีดล่าง Underscore (_) แต่ห้ามมีช่องว่าง
3. ตัวเลข 0-9 จะนำหน้าชื่อตัวแปรไม่ได้
4. ตัวพิมพ์เล็กและตัวพิมพ์ใหญ่เป็นตัวแปรคนละตัวกัน (Case-Sensitive) เช่น Name ไม่ใช่ตัวแปรเดียวกันกับ name
5. ใช้ใส่เครื่องหมาย = ในการตั้งตัวแปรหรือให้ค่าแก่ตัวแปร
6. การตั้งชื่อตัวแปรควรตั้งอย่างสมเหตุสมผล
7. ภาษา Python จะมีคำที่ถูกสงวนไว้ในการเขียนโปรแกรม หรือ Keywords ซึ่งห้ามนำมาใช้ในการตั้งชื่อตัวแปร ชื่อฟังก์ชัน หรือ ชื่อคลาส

มีดังต่อไปนี้ (Lutz, 2014)

ตารางที่ 3.1: คำสงวนในภาษา Python

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	break
except	in	raise		

รูปที่ 3.1: การตั้งค่าตัวแปร

```
>>> a = 1
>>> a
1
>>> b = 2
>>> b
2
>>> a + b
3
>>> vat = 7
>>> vat
7
```

ตัวแปรจะชี้ไปที่หน่วยความจำในเครื่องคอมพิวเตอร์ซึ่งเก็บค่าของตัวแปรหรือ Value นั้นๆ อยู่ ฉะนั้นเมื่อเราพิมพ์ `a` ดังในตัวอย่าง คอมพิวเตอร์จึงแสดงเลข 1 ออกมา นอกจากนี้พื้นที่ที่เก็บค่านั้นนั้นจะมีที่อยู่อยู่บนหน่วยความจำมีหมายเลขประจำตำแหน่งอีกด้วย โดยใช้คำสั่ง `id()` เพื่อแสดงเลขประจำตำแหน่ง

รูปที่ 3.2: เลขประจำตัวตำแหน่งของตัวแปร

```
>>> a
1
>>> id(a)
1538021648
```

3.3 ประเภทของข้อมูล (Types)

สิ่งที่อยู่ในหน่วยความจำมีประเภทของข้อมูลหรือ Type อยู่ด้วย โดยใช้คำสั่ง `type()` เพื่อดูประเภทของข้อมูล ในภาษา Python มีประเภทของข้อมูลหลายๆ แบบ (Ramalho, 2015)

1. none คือ Nothing ไม่มีอะไร
2. int หรือ Integer คือตัวเลข เช่น 50 หรือ 630 เป็นต้น
3. bool หรือ Boolean คือค่าถูกผิด เช่น True หรือ False เป็นต้น
4. float หรือ floating Point คือจำนวนทศนิยม เช่น 5.6 หรือ 4.23 เป็นต้น
5. str หรือ String หรือข้อความซึ่งจะอยู่ภายใต้เครื่องหมาย “ (ฟันหนู) หรือ ‘ (ฝนทอง) เช่น “This is my dog.” หรือ ‘Jantawan’

รูปที่ 3.3: ประเภทของข้อมูล

```
>>> a = 1
>>> a
1
>>> type(a)
<class 'int'>
>>> firstname = 'Jantawan'
>>> firstname
'Jantawan'
>>> lastname = 'Piyawat'
>>> lastname
'Piyawat'
>>> id(firstname)
67626832
>>> type(firstname)
<class 'str'>
```

รูปที่ 3.4: ประเภทของข้อมูล

```
>>> n = None
>>> n
>>> id(n)
263420692
>>> type(n)
<class 'NoneType'>
>>> yes = True
>>> no = False
>>> type(yes)
<class 'bool'>
>>> degree = 1.1
>>> id(degree)
72213072
>>> type(degree)
<class 'float'>
```

3.4 เครื่องหมายสำหรับการคำนวณ

3.4.1 การคำนวณทางคณิตศาสตร์ (Arithmetic Operators)

เครื่องหมายสำหรับการคำนวณเรียกว่า Arithmetic Operators เช่น เครื่องหมายบวก ลบ คูณ หาร การยกกำลัง การหารเอาเศษ การหารเอาจำนวนเต็ม เป็นต้น การคำนวณทางคณิตศาสตร์แบบซับซ้อนจะต้องมีลำดับในการคำนวณซึ่งเหมือนกับการคำนวณคณิตศาสตร์ทั่วไป คือ ในการแก้สมการทางคณิตศาสตร์จะต้องทำในวงเล็บก่อน ตามด้วยเลขยกกำลัง แล้วจึงตามด้วย คูณหรือหารโดยคำนวณจากซ้ายไปขวา แล้วตามด้วยบวกหรือลบโดยคำนวณจากซ้ายไปขวาเช่นกัน โดยให้จำคำว่า PEMDAS ซึ่งเป็นตัวอักษร ภาษาอังกฤษตัวแรกของคำว่า Parentheses (วงเล็บ), Exponents (ยกกำลัง), Multiply (คูณ), Divide (หาร), Add (บวก), และ Subtract (ลบ)

รูปที่ 3.5: ตัวอย่างคำนวณทางคณิตศาสตร์

```
>>> a
1
>>> b
2
>>> a + b
3
>>> a - b
-1
>>> b - a
1
>>> c = a - b
>>> c
-1
>>> a * b
2
>>> a * c
-1
>>> b * b
4
>>> b / a
2.0
>>> b
2
>>> b ** 2
4
```

3.4.2 รูปแบบการเขียนการคำนวณทางคณิตศาสตร์

3.4.3 การจัดการข้อความด้วยเครื่องหมายทางคณิตศาสตร์

เครื่องหมายที่เป็นการคำนวณทางคณิตศาสตร์เมื่อถูกนำมาใช้กับข้อความ (String) จะเป็นอีกความหมายหนึ่ง เช่น การใช้เครื่องหมายบวกเชื่อมต่อระหว่างสตริง 2 ตัว หรือ การใช้เครื่องหมายดอกจันเป็นการเพิ่มสตริงเดียวกันตามจำนวนครั้งของการคูณ

รูปที่ 3.6: การจัดการข้อความด้วยเครื่องหมายทางคณิตศาสตร์

```
>>> firstname
'Jantawan'
>>> lastname
'Piyawat'
>>> firstname + lastname
'JantawanPiyawat'
>>> firstname + '  ' + lastname
'Jantawan Piyawat'
>>> firstname * 3
'JantawanJantawanJantawan'
```

3.5 Expressions และ Statements

Expressions หมายถึงการใช้เครื่องหมายคำนวณและการใช้ตัวแปรและค่าของตัวแปรเพื่อหาผลลัพธ์ออกมา เอา Expression มาประกอบกันจะเรียกว่า Statement ดังนั้น Statement ก็คือคำสั่งเรียงต่อกันนั่นเอง เพื่อใช้ในการสั่งงานคอมพิวเตอร์ด้วยภาษาคอมพิวเตอร์

รูปที่ 3.7: Expressions

```
>>> 1+2
3
```

รูปที่ 3.8: Statements

```
>>> c = a + b
>>> c
3
>>> print('hello world.')
hello world.
```

3.6 การเขียนข้อความอธิบายโปรแกรมโดยการใช้ Comment

Comment คือสิ่งที่เราเขียนใน Source Code ของโปรแกรมแต่คอมพิวเตอร์ไม่ต้องแปลผล เพื่อใช้ในการเขียนข้อความประกอบคำอธิบายในการสื่อสารระหว่างโปรแกรมเมอร์ด้วยกัน หรือเป็นการเตือนความจำของโปรแกรมเมอร์เอง โดย Comment ในภาษา Python นำหน้าด้วยเครื่องหมายชาร์ป (#) แล้วหลังจากนั้นตามด้วยข้อความอะไรก็ได้ ถ้าจะเขียน Comment หลายๆ บรรทัดจะต้องใช้เครื่องหมายฟันหนู (") หรือฝนทอง (') 3 ตัว แล้วก็พิมพ์ข้อความแล้วจึงปิดด้วยฟันหนู (") หรือฝนทองทั้ง 3 ตัวอีกครั้ง (') ผลที่ได้จะมีเครื่องหมาย Backslash n (\n) หมายถึงการขึ้นบรรทัดใหม่

รูปที่ 3.9: การใช้ Comment ในบรรทัดเดียว

```
>>> print('Hello world!')
Hello world!
>>> # Hello how are you doing?
```

รูปที่ 3.10: การใช้ Comment ในหลายบรรทัด

```
>>> x = '''
hello
1
2
3
'''
>>> x
'\nhello\n1\n2\n3\n'
>>> print(x)
hello
1
2
3
```

3.7 Source code

ที่ผ่านมาเป็นการเขียนโปรแกรมแบบ Interactive คือเขียนบน Python Shell แล้วโปรแกรมจะแสดงผลออกมาได้เลย ซึ่งเรียกว่าการทำงานแบบ Interpreter เป็นการใส่คำสั่งไปที่ Prompt และ Python จะแสดงผลของคำสั่งนั้นออกมาเลย แต่ในความเป็นจริงแล้วจะเขียนโปรแกรมหลายๆ บรรทัดแล้วสั่งโปรแกรมทำงานทีเดียวพร้อมกัน เราจะเขียนไว้ในไฟล์นั้นเรียกว่า Source Code โดยที่ Source Code ของภาษา Python นามสกุลจะเป็น .py เวลาใช้ที่โปรแกรม Idle ให้กดที่เมนู File เลือก New เขียน Source Code แล้วให้กด Run ถ้าหากจะกดรันโปรแกรมอีกสักครั้งให้กด F5

Source code:

รูปที่ 3.11: ตัวอย่าง Python source code

```
x = 7
y = 6
if x == y: print('x and y are equal.')
else:
    if x < y: print('x is less than y.')
    else: print('x is greater than y.')
```

Result:

รูปที่ 3.12: ตัวอย่างผลลัพธ์ที่ได้จากการประมวลผล Source code

```
x is greater than y.
```

3.8 คำสั่ง `print` (ตัวแปรหรือข้อมูล)

print() เป็นฟังก์ชันที่ใช้ในการแสดงผลตัวแปรหรือข้อมูลออกทางหน้าจอ

รูปที่ 3.13: คำสั่ง print()

```
>>> print('Hello world!')
Hello world!
>>>
```

3.9 การใช้คำสั่ง input() รับค่าจากแป้นพิมพ์

คำสั่ง input(ข้อความ prompt) เป็นคำสั่งสำหรับรับข้อมูลจากผู้ใช้ด้วยการพิมพ์ผ่านแป้นพิมพ์

รูปที่ 3.14: คำสั่ง input()

```
>>> name=input('What is your name? ')
What is your name? Jantawan
>>> print('Hello, ', name, end='.')
Hello, Jantawan.
```

3.10 แบบฝึกหัด

1. จงหาเลขประจำตำแหน่งของข้อมูลต่อไปนี้
    - love = 2
    - mom = “Jan”
    - wed = True
    - fah = 39.2
2. จงหาประเภทของข้อมูลต่อไปนี้
    - `love = 2`
    - `mom = 'Jan'`
    - `wed = True`

- `fah = 39.2`
- `money = '22'`

3. จงแสดงผลต่อไปนี้
   - ตั้งค่าตัวแปร dog, cat
   - แสดงข้อความ I have 3 dogs and 2 cats.
4. จงรับค่าจากผู้ใช้และแสดงผลต่อไปนี้
   - ตั้งตัวแปร name
   - รับค่าด้วยข้อความว่า กรุณาใส่ชื่อของคุณ
   - แสดงข้อความ สวัสดีค่ะคุณ
5. จงคำนวณหาค่าตัวเลขต่อไปนี้
   - หาค่าพื้นที่สี่เหลี่ยม กว้าง 5 เมตร ยาว 3 เมตร
   - หาค่าพื้นที่สามเหลี่ยม สูง 5 เมตร ฐาน 3 เมตร
6. ให้ `a = 3, b = 4, c = 5` จงหาค่าต่อไปนี้
   - `a == a*1`
   - `a != b`
   - `a > b`
   - `b < c`
   - `a+1 >= c`
   - `c <= a+b`

บทที่ 4

ประโยคเงื่อนไขในภาษา Python (Conditional Statements)

4.1 การเปรียบเทียบค่า (Boolean Expressions)

Boolean Expressions คือ การดำเนินการเปรียบเทียบค่าเพื่อให้ได้ผลลัพธ์ออกมาเป็นถูก (True) หากเงื่อนไขเป็นจริง หรือผิด (False) หากเงื่อนไขเป็นเท็จ เช่น ค่าของ x มากกว่าค่าของ y ใช่หรือไม่ ซึ่งผลลัพธ์จะออกมาเป็นถูกหรือผิด

รูปที่ 4.1: การใช้สัญลักษณ์เปรียบเทียบค่า

```
>>> a = 5
>>> b = 6
>>> a == b
False
>>> 5 > 6
False
>>> 5 < 6
True
>>> 5 >= 6
False
```

4.2 ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators)

การเปรียบเทียบค่ามากกว่าหนึ่งครั้งเชื่อมต่อกันสามารถดำเนินการได้โดยใช้ตัวดำเนินการทาง ตรรกศาสตร์ (Logical Operators) ซึ่งได้แก่ และ (and) หรือ (or) ไม่ (not) เช่น a > 2 or c > b and c >2

ตัวอย่างการใช้ตัวดำเนินการทางตรรกศาสตร์ในภาษา Python

รูปที่ 4.2: ตัวอย่างการใช้ and or not

```
>>> a = -4
>>> b = -2
>>> a > 2 or c > b and c> 2
False
```

4.3 การใช้คำสั่ง if เพื่อเลือกเงื่อนไข

เงื่อนไขที่ใช้ในภาษา Python คือ if Statement สิ่งที่ตามหลัง if คือ Boolean Expression เรียกว่า Statement ใหญ่ และใน Statement ใหญ่ ก็มี Statement ย่อย การดูว่า Statement ย่อยอยู่ใน if Statement ใดให้ดูที่การย่อหน้าหรือ Indentation ในภาษา Python การย่อหน้าสำคัญมากจะเป็นการบอกว่าอะไรอยู่ภายในอะไร

รูปแบบของการใช้งานคำสั่ง if ในภาษา Python โดยถ้าหากเงื่อนไขเป็นจริง ตัวโปรแกรมจะประมวลผลในคำสั่ง if หลังเครื่องหมาย :

รูปที่ 4.3: รูปแบบของการใช้งานคำสั่ง if

```
if expresion:
    #statements
```

ตัวอย่างเช่น ให้แสดงข้อความว่า อายุต่ำกว่าเกณฑ์ ถ้าหากค่าอายุที่รับเข้ามาต่ำกว่า 18 ปี

Source code:

รูปที่ 4.4: Source code จากโจทย์ตัวอย่าง

```
age = int(input('Enter your age: `))
if age < 18:
   print('You are underage.')
```

Result:

รูปที่ 4.5: Result จากโจทย์ตัวอย่าง

```
Enter your age: 15
You are underage.
```

4.4 การใช้ if กับ else

โครงสร้างคำสั่ง if...else จะดำเนินในบล็อคคำสั่ง else ถ้าหากเงื่อนไขในคำสั่ง if นั้นเป็นเท็จ โดยมีรูปแบบการเขียนดังนี้

รูปที่ 4.6: รูปแบบของการใช้งานคำสั่ง if-else

```
if expression:
    # statements
else:
    # statements
```

Source code:

รูปที่ 4.7: Source code ตัวอย่างการใช้ if...else

```
x = 15
y = 6
if x > y: print('x is greater than y.')
else: print('x is less than or equal to y.')
```

Result:

รูปที่ 4.8: Result ตัวอย่างการใช้ if...else

```
x is greater than y.
```

4.5 Chained Expressions

การใช้ Chained Expressions คือ การใส่ elif ไปเรื่อยๆ และเงื่อนไขสุดท้ายจะต้องใช้ else โดยไม่ต้องการระบุ Boolean Expressions ใดๆ อีกแล้วหลังจากที่ใส่ else

Source code:

รูปที่ 4.9: Source code ตัวอย่างการเขียน Chained Expressions

```
x = 6
y = 6
if x > y: print('x is greater than y.')
elif x < y: print('x is less than y.')
else: print('x and y are equal.')
```

Result:

รูปที่ 4.10: Result ตัวอย่างการเขียน Chained Expressions

```
x and y are equal.
```

4.6 Nested Expressions

if มี Statement อยู่ข้างในได้ และ el se ก็มี Statement อยู่ข้างในได้เช่นกัน เรียกว่า Nested Expressions

Source code:

รูปที่ 4.11: Source code ตัวอย่างการเขียน Nested Expressions

```
x = 7
y = 6
if x == y: print('x and y are equal.')
else:
    if x < y: print('x is less than y.')
    else: print('x is greater than y.')
```

Result:

รูปที่ 4.12: Result ตัวอย่างการเขียน Nested Expressions

```
x is greater than y.
```

4.7 แบบฝึกหัด

1. จงเขียน code ต่อไปนี้
   - ให้ถามว่า Are you bored? และให้ตอบว่า y หรือ n
   - ถ้าตอบ y ให้พิมพ์ข้อความว่า Let's go outside.
2. จงเขียน code ต่อไปนี้
   - ตั้งค่าตัวแปร var รับค่าเป็นตัวเลขจำนวนเต็มจากผู้ใช้
   - ถ้า var > 100 ให้แสดงผลว่า "The value is over 100."
   - ถ้า เป็นกรณีอื่นๆ ให้แสดงผลว่า "The value is less than or equal 100."
3. จงเขียน code ต่อไปนี้
   - รับค่า a, b เป็นจำนวนเต็ม
   - แสดงผลว่า a > b หรือ a < b หรือ a = b

4. จงเขียน code ต่อไปนี้

   - รับค่า score เป็นจุดทศนิยม
   - ถ้าคะแนน 81-100 แสดงผลว่า เกรด A
   - ถ้าคะแนน 61-80 แสดงผลว่า เกรด B
   - ถ้าคะแนน 41-60 แสดงผลว่า เกรด C
   - ถ้าคะแนน 0-40 แสดงผลว่า เกรด F
   - เมื่อแสดงผลดังกล่าวแล้ว ให้แจ้งด้วยว่า “ตัดเกรดแล้ว”

บทที่ 5

การเขียนและใช้งานฟังก์ชัน (Functions)

5.1 การเรียกใช้ฟังก์ชัน

การใช้ฟังก์ชันในภาษา Python ก็เหมือนฟังก์ชันทางคณิตศาสตร์ คือ กำหนดชื่อฟังก์ชันตามด้วยสิ่งที่อยู่ในวงเล็บซึ่งเรียกว่า arguments ซึ่งอาจจะมีได้มากกว่า 1 และในภาษา Python มีการกำหนดฟังก์ชันมาให้เรียกใช้ได้เลยอยู่บ้างแล้ว เช่น type(42) คือ การแสดงค่าประเภทของเลข 42 หรือ id(42) คือการแสดงตำแหน่งที่อยู่ของเลข 42 ในหน่วยความจำ

รูปที่ 5.1:

```
>>> type(42)
<class 'int'>
>>> a = 1
>>> id(a)
1538021648
```

5.2 การเรียกใช้โมดูล (Modules)

โมดูล (Modules) คือ ฟังก์ชันที่รวมกันไว้เป็นหมวดหมู่ และสามารถดึงมาใช้ได้ในโปรแกรมได้ด้วยการ import เช่น import math และหากเรียกใช้ฟังก์ชัน dir(math) จะแสดงฟังก์ชันในโมดูล math ออกมา

รูปที่ 5.2:

```
>>> import math
>>> type(math)
<class 'module'>
>>> id(math)
56337632
```

การใช้งานโมดูลจะมีการใช้งานแบบ Dot Notation หากเห็นการเขียนโมดูล math ในลักษณะนี้ เช่น `math.pi` ตัว `pi` เรียกว่าเป็นตัวแปรที่อยู่ในโมดูล `math` ซึ่งไม่ใช่ฟังก์ชัน แต่ถ้าเขียน `math.pow(2,2)` คือ สองยกกำลังสอง ลักษณะนี้จะเป็นการเรียกใช้ฟังก์ชันที่อยู่ในโมดูล

รูปที่ 5.3:

```
>>> math.pi
3.141592653589793
>>> math.pow(2,2)
4.0
```

5.3 ฟังก์ชันซ้อน (Composition)

การใช้ฟังก์ชันไม่จำเป็นต้องใช้ฟังก์ชันแบบฟังก์ชันเดียว ฟังก์ชันใช้ฟังก์ชันซ้อนกันได้ คือ การรวมฟังก์ชันหลายๆ อันซ้อนกัน เรียกต่อๆ กันไปได้

รูปที่ 5.4:

```
>>> math.exp(math.log(3+2))
4.9999999999999
```

5.4 การสร้างฟังก์ชัน

การเขียนฟังก์ชันหรือสร้างฟังก์ชันขึ้นมาเองเพื่อทำงานเพื่อวัตถุประสงค์บางอย่าง ต้องใช้ def Statement ซึ่งย่อมาจาก define

รูปที่ 5.5:

```
def function_name(args...):
    # statements

def function_name(args...):
    # statements
    return value
```

เมื่อเขียนฟังก์ชันไว้ใน Source Code แล้วทำการ Run ฟังก์ชันนั้นจะถูก Define ไว้ในระบบแล้วถูกเรียกใช้ขึ้นมาได้เลย ด้วยการเรียกชื่อฟังก์ชันนั้นกี่ครั้งต่อกี่ครั้งก็ได้ โดยการเรียกใช้คือ เรียกชื่อฟังก์ชันนั้นตามด้วยวงเล็บซึ่งจะมี Arguments หรือไม่ก็แล้วแต่ฟังก์ชันที่กำหนดไว้

Source code:

รูปที่ 5.6:

```
def happy_birthday_song():
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to you.')
```

Result:

รูปที่ 5.7:

```
>>> happy_birthday_song()
Happy Birthday.
Happy Birthday.
Happy Birthday.
Happy Birthday to you.
```

5.5 พารามิเตอร์ของฟังก์ชัน (Parameters)

ในวงเล็บ () เป็นการกำหนด Argument ของฟังก์ชัน ซึ่งจะกลายเป็นพารามิเตอร์ (Parameters) หรือตัวแปรที่ใช้ในฟังก์ชันนั้นๆ เท่านั้น หรือเรียกว่า Local Variables

Source code:

รูปที่ 5.8:

```
x = 5
def happy_birthday_song(name):
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    print(x)
happy_birthday_song('Mike')
```

Result:

รูปที่ 5.9:

```
Happy Birthday.
Happy Birthday.
Happy Birthday.
Happy Birthday to Mike
5
```

ตัวแปรใดก็ตามที่จะใช้เป็น Local ให้ใส่คำว่า global ไปด้านหน้าตัวแปรที่ถูกเรียกใช้ในฟังก์ชัน อันที่จริงแล้วจะไม่เขียนคำว่า global ก็ได้หากชื่อตัวแปรไม่ซ้ำกันเลย

Source code:

รูปที่ 5.10:

```
x = 5
def happy_birthday_song(name):
    global x
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    print(x)
happy_birthday_song('Mike')
```

Result:

รูปที่ 5.11:

```
Happy Birthday.
Happy Birthday.
Happy Birthday.
Happy Birthday to Mike
5
```

5.6 ฟังก์ชัน return

ฟังก์ชันทุกอันจะต้อง return คือสิ้นสุดการทำงาน แต่ได้เว้นไว้ในฐานที่เข้าใจ เมื่อโปรแกรมไล่การทำงานมาถึงจุด return โปรแกรมจะหยุดทำงานทันทีทั้งๆ ที่ยังมีคำสั่งอื่นตามมาหลังจาก return อีกก็ตาม ฟังก์ชัน return ซึ่งไม่ได้ return ค่าอะไรออกมาเรียกว่า Void

Source code:

รูปที่ 5.12:

```
x = 5
def happy_birthday_song(name):
    print('Happy Birthday.')
    return
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    print(x)
happy_birthday_song('Mike')
```

Result:

รูปที่ 5.13:

```
Happy Birthday.
```

5.7 การคืนค่าจากฟังก์ชัน

ฟังก์ชันสามารถ return ค่าได้ด้วย ดังเช่นตัวอย่างการหาพื้นที่สี่เหลี่ยม โดยกำหนดฟังก์ชันชื่อ `rectangle_area(width,` และฟังก์ชันมีพารามิเตอร์สองตัวสำหรับความกว้างและความยาวของสี่เหลี่ยม และฟังก์ชันทำการ return ผลลัพธ์ที่เป็นพื้นที่กลับไปด้วยคำสั่ง return

Source code:

รูปที่ 5.14:

```
# void function definition
def happy_birthday_song(name):
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday.')
    print('Happy Birthday to ' , name)
    return name

def rectangle_area(width, height):
    return width * height

x = rectangle_area(4, 3)
print('The area of rectangle is', str(x))
```

Result:

รูปที่ 5.15:

```
The area of rectangle is 12
```

5.8 การเขียนโปรแกรมเชิงฟังก์ชัน (Functional Programming)

การเขียนโปรแกรมเชิงฟังก์ชัน (Functional Programming) เป็นรูปแบบการเขียนโปรแกรมที่เก่าแก่ที่สุดและเป็นรูปแบบที่กลับมาได้รับความนิยมมากในปัจจุบัน คือสร้างฟังก์ชันแล้วให้ฟังก์ชันทำงานร่วมกันโดยไม่มีการใช้ Global Variables เลย ฟังก์ชันหนึ่งทำงานส่งผลลัพธ์แก่อีกฟังก์ชันหนึ่งต่อๆ กันไปเรื่อยๆ ซึ่งภาษา Python สามารถใช้เขียนโปรแกรมแบบนี้ได้ ฟังก์ชันแล้ว return ค่าเป็นผลลัพธ์แก่อีกฟังก์ชันหนึ่งไปเรื่อยๆ

จากรูป `x=f(g(h(x)))` ทำงานเหมือนกันกับ `x = h(x)` แล้ว `x = g(x)` แล้ว `x = f(x)`

รูปที่ 5.16:

```
x = f(g(h(x)))
x = h(x)
x = g(x)
x = f(x)
```

5.9 แบบฝึกหัด

1. ตั้งชื่อฟังก์ชัน `hello` เพื่อแสดงผลว่า สวัสดีคุณ
2. สร้างฟังก์ชันคำนวณพื้นที่ รับค่า ความกว้าง และ ความยาว
3. สร้างฟังก์ชันชื่อ `maximal_2` รับ arguments 2 ค่า และ return ค่าที่มากที่สุดออกมา
4. สร้างฟังก์ชันชื่อ `maximal_3` รับ arguments 3 ค่า และ return ค่าที่มากที่สุดออกมา
5. สร้างฟังก์ชันรับ arguments 3 ค่า และ return ผลคูณออกมา
6. สร้างฟังก์ชันรับค่าตัวเลขในหน่วยเมตรต่อวินาที แล้ว return ผลในหน่วยกิโลเมตรต่อชั่วโมง
7. สร้างฟังก์ชันหาผลต่างของรายรับกับรายจ่าย และส่งผลกลับมา

บทที่ 6

การใช้ประโยคสั่งทำงานวนซ้ำ

6.1 ฟังก์ชัน `range()`

ฟังก์ชัน `range()` คือ ระยะตั้งแต่เริ่มต้นถึงก่อนระยะสิ้นสุด มักจะใช้ในการควบคุมการทำงานของโปรแกรมเป็นจำนวนรอบ มีวิธีการเขียนดังนี้ `range(start, end)`

6.2 คำสั่ง `for`

คำสั่ง **`for`** statement เป็นการทำงานซ้ำๆ ตามจำนวนครั้งที่ระบุไว้ เช่น การใช้ **`for`** statement ร่วมกัน `range()` โดยมีรูปแบบการเขียนดังนี้

รูปที่ 6.1:

```
for var in sequence:
    # statements
```

Source code:

รูปที่ 6.2:

```
for x in range(0,5): print(x)
```

Result:

รูปที่ 6.3:

```
>>>
0
1
2
3
4
>>>
```

6.3 คำสั่ง `while`

`while` Statement เป็นคำสั่งให้โปรแกรมทำงานวนซ้ำในขณะที่เงื่อนไขของการวนซ้ำนั้นยังคงเป็นจริงอยู่ และเมื่อเงื่อนไขเป็นเท็จจะสิ้นสุดการทำงานวนซ้ำนั้นทันที ดังนั้นจึงต้องมีตัวควบคุมในการเพิ่มค่าไปเรื่อยๆ จนเงื่อนไขเป็นเท็จ มีลักษณะการเขียนดังนี้

รูปที่ 6.4:

```
while expression:
    # satements
```

Source code:

รูปที่ 6.5:

```
x = 0
while x < 10:
    print(x)
    x = x + 1
```

Result:

รูปที่ 6.6:

```
0
1
2
3
4
5
6
7
8
9
```

6.4 คำสั่ง break

คำสั่ง **break** เพื่อให้หลุดการทำงานในลูป

Source code:

รูปที่ 6.7:

```
x = 0
while True:
    print(x)
    if x == 10: break
    x = x +1
```

Result:

รูปที่ 6.8:

```
0
1
2
3
4
5
6
7
8
9
10
```

6.5 ฟังก์ชันที่เรียกตัวเอง (Recursion)

Recursion คือการเรียกใช้ฟังก์ชันซ้อนฟังก์ชันนั้นๆ เอง หรือ ฟังก์ชันเรียกใช้ตัวมันเอง จากฟังก์ชัน `countdown()` ในตัวอย่าง เมื่อมีการเรียกฟังก์ชัน `countdown(5)` โปรแกรมจะทำงานลดหลั่นลงไปเรื่อยๆ คือตั้งแต่ 5, 4, 3, 2, 1 ตราบเท่าที่ n ยังมากกว่า 0 แต่เมื่อเงื่อนไขเป็นเท็จแล้ว โปรแกรมจะแสดงคำว่า Go! แทน

Source code:

รูปที่ 6.9:

```
def countdown(n):
    if x > 0:
        print(n)
        countdown(n-1)
    else:
        print('Go.')
```

Result:

รูปที่ 6.10:

```
>>> countdown(5)
5
4
3
2
1
Go.
```

6.6 แบบฝึกหัด

1. สร้างฟังก์ชันหาค่าของ $3^1 + 3^2 + 3^3 + 3^4 + 3^5$ โดยใช้ For loop
2. สร้างฟังก์ชันให้ผู้ใช้ป้อนค่า x แล้วนำค่า x มาคำนวณ $x^1 + x^2 + x^3 + x^4 + x^5$
3. สร้างฟังก์ชันให้ผู้ใช้ป้อนค่า n และให้แสดงเลขเริ่มที่ n โดยลดลงทีละหนึ่ง โดยใช้ while loop
4. สร้างฟังก์ชันให้ผู้ใช้ป้อนตัวเลข แล้วหาว่ามีเลขใดที่สามารถหารเลขที่ผู้ใช้ป้อนได้ลงตัว เช่น ผู้ใช้ป้อนตัวเลข 4 จะมี 1 2 4 ที่หารเลข 4 ลงตัว
5. สร้างฟังก์ชันคำนวณปีเกิด คศ. เป็น 12 ราศีปีนักษัตร
6. สร้างฟังก์ชันรับจำนวนเงินมาหนึ่งค่า แล้วแลกเปลี่ยนธนบัตร 100 บาท 50 บาท 20 บาท เหรียญ 10 บาท 5 บาท และ 1 บาท

บทที่ 7

การใช้งาน String

7.1 ความหมายของ String

String คือ ข้อความหรือตัวอักษรที่เรียงต่อๆ กันที่อยู่ในเครื่องหมายคำพูดแบบ Double Quotes เช่น “How are you?” หรือ Single Quotes เช่น ‘How are you?’

7.2 ฟังก์ชัน `len()`

มีฟังก์ชันสำหรับ String อยู่หลายฟังก์ชัน เช่น ฟังก์ชัน `len()` มีวัตถุประสงค์เพื่อหาความยาวของ String นั้น และสำหรับการระบุตำแหน่งของตัวอักษรแต่ละตัวใน String จะใช้สัญลักษณ์ก้ามปู `[ ]` โดยตัวชี้หรือ Index จะเริ่มต้นจาก 0 เช่น `fruit[0]`

รูปที่ 7.1:

```
>>> fruit = 'banana'
>>> fruit
'banana'
>>> len(fruit)
6
>>> type(fruit)
<class 'str'>
>>> fruit[0]
'b'
>>> fruit[1]
'a'
>>> fruit[2]
'n'
```

7.3 การเดินทางตามตัวชี้ของ String

การเดินทางไปเรื่อยๆ ตามตัวชี้ของ String โดยในตัวอย่างแรกจะเป็นการใช้ while ส่วนในตัวอย่างถัดมาจะใช้ตัวแปร string เป็น Iterator ซึ่งผลลัพธ์จะออกมาเหมือนกัน

Source code:

รูปที่ 7.2:

```
fruit = 'banana'
i = 0
while i < len(fruit):
    print(fruit[i])
    i += 1
for character in fruit: print(character)
```

Result:

รูปที่ 7.3:

```
>>>
b
a
n
a
n
a
b
a
n
a
n
a
>>>
```

7.4 การตัดคำใน String

การตัดคำใน String ด้วยตัวชี้ (Index) โดยการตัดคำเป็นส่วนย่อยๆ จะมีรูปแบบการเขียนเป็น [start:end] โดยที่ start เป็นตำแหน่งของ Index เริ่มต้นที่ต้องการ และ end นั้นเป็นตำแหน่งก่อนหน้าตำแหน่งสุดท้ายของตัวอักษรที่ต้องการ

รูปที่ 7.4:

```
>>> s = 'Monty Python'
>>> s
'Monty Python'
>>> len(s)
12
>>> s[0:1]
'M'
>>> s[0]
'M'
>>> s[0:2]
'Mo'
>>>
```

หากเขียนเป็น `[start:]` ระบุจุดเริ่มต้นที่ start ผลลัพธ์จะแสดงยาวไปจนถึงจุดสิ้นสุด และหากเขียนเป็น `[:end]` ผลลัพธ์ที่ได้จะแสดงอักษรตั้งแต่ตัวแรกหรือตัวชี้ที่ศูนย์ไปจนถึงตัวสิ้นสุดที่ระบุไว้

รูปที่ 7.5:

```
>>> s[1:]
'onty Python'
>>> s[:5]
'Monty'
>>>
```

7.5 โครงสร้างข้อมูลที่เปลี่ยนแปลงไม่ได้

String เป็นโครงสร้างข้อมูลที่เปลี่ยนแปลงไม่ได้ ดังนั้นถ้าหากต้องการสร้าง String ใหม่ก็ต้องสร้างเป็น Object ใหม่เท่านั้น

รูปที่ 7.6:

```
>>> s
'Monty Python'
>>> s[0] = 'J'
Traceback (most recent call last):
    File ''<pyshell#19>'', line 1, in <module>
        s[0] = 'J'
TypeError: 'str' object does not support item assignment
>>> id(s)
62723320
>>> a = 'J' + s[1:]
>>> a
'Jonty Python'
>>> id(a)
62737048
```

7.6 การค้นหาตัวอักษรใน String

การเขียนโปรแกรมเพื่อค้นหาตัวอักษรใน String แล้วส่งค่าออกมาเป็นค่าตัวชี้ตัวอักษรใน String สามารถเขียนเป็นตัวอย่างฟังก์ชันดังนี้ คือ

- ให้ฟังก์ชันชื่อว่า find มีการส่งค่า str เป็น string และค่า char เป็น character
- ตั้งตัวนับ i เริ่มต้นที่ 0
- ขณะที่ i ยังน้อยกว่าจำนวนตัวอักษรใน string ที่ชื่อว่า str ให้ดำเนินการดังนี้คือ
  - ตรวจสอบว่า ถ้าตัวชี้ของตัวอักษรเท่ากับตัวอักษรที่ต้องการแล้ว ให้ส่งค่าตัวนับออกมา
- เมื่อ while เป็นเท็จแล้ว หรือเมื่อค้นจนครบ character ใน string แล้วให้ส่งค่ากลับคือ -1

Source code:

รูปที่ 7.7:

```
def find(str, char):
    i = 0
    while i < len(str):
       if str[i] == char: return i
       i += 1
    return -1
```

Result:

รูปที่ 7.8:

```
>>> find('Mike', 'c')
-1
>>> find('Mike', 'e')
3
>>>
```

7.7 เมธอดของ String (String Methods)

การดำเนินการกับ String สามารถศึกษาฟังก์ชันได้ที่ `https://docs.python.org/2.4/lib/string-methods.html` เช่น `str.upper()` ใช้ทำงานเพื่อแปลงตัวอักษรภาษาอังกฤษเป็นตัวพิมพ์ใหญ่

รูปที่ 7.9:

```
>>> s = 'Monty Python'
>>> s
'Monty Python'
>>> s.count('t')
2
>>> s.capitalize()
'Monty python'
>>> s.upper()
'MONTY PYTHON'
```

7.8 in โอเปอร์เรเตอร์

ใช้ในการพิสูจน์ค่าแบบ Boolean Expression เช่น 't' in s แปลว่า ตัวอักษรตัว t อยู่ใน String ชื่อว่า s หรือไม่ หรือในทางตรงข้ามเพื่อการตรวจสอบว่าไม่มีหรือไม่ให้ใส่ not in

รูปที่ 7.10:

```
>>> 't' in s
True
```

7.9 การเปรียบเทียบ String

การเปรียบเทียบ String สามารถใช้สัญลักษณ์ (> , < , <= , <= , == , !=) เพื่อเปรียบเทียบค่าของ String สองชุด โดยดูผลลัพธ์ของการเปรียบเทียบค่าของ ASCII value นั้นๆ

รูปที่ 7.11:

```
>>> 'banana' == 'banana'
True
>>> 'banana' !=  'banana'
False
>>>  'banana' >  'banana'
False
>>>  'banana' <  'banana'
False
>>>  'banana' >  'Banana'
True
>>>  'banana' >  'bazooka'
False
```

7.10 การจัดวางรูปแบบของ String (String Formatting)

การเปลี่ยนการจัดวางรูปแบบของ String มีสองวิธีคือ แบบ Classic ซึ่งทำได้โดยใส่สัญลักษณ์ + แต่สัญลักษณ์นี้จะมีข้อจำกัดคือไม่สามารถแทรกข้อความระหว่างกันได้ แต่หากใช้สัญลักษณ์ **%** จะช่วยแก้ไขข้อจำกัดนี้ได้ เช่น **%s** สำหรับ string และ **%d** สำหรับตัวเลข

รูปที่ 7.12:

```
%>>> 'Hello' + 'Mike'
%'HelloMike'
%>>> 'Hello \%s' \%'Mike'
%'Hello Mike'
%>>> 'Hello `\%s\exclaim\exclaim' \%'Mike'
%'Hello Mike!!'
%>>>
```

รูปที่ 7.13:

```
%>>> 'Total is \%d baht' \% 12
%'Total is 12 baht'
%>>>
```

ส่วนการเปลี่ยนการจัดวางของ String แบบ Modern คือ ใช้ .format และระบุ Parameters ได้มากกว่า 1 ตัว อีกทั้งยังสามารถจัดเรียงลำดับ Parameters สลับก่อนหลังได้ตามสะดวก สามารถศึกษาเพิ่มเติมได้ที่ **https://docs.python.org/3/library/string.html#format-examples**

รูปที่ 7.14:

```
>>> x = 12
>>> 'Total is {0} baht' .format(x)
'Total is 12 baht'
>>> 'Total is {1} baht. Mr. {0}' .format('Mike', x)
'Total is 12 baht. Mr. Mike'
```

7.11 แบบฝึกหัด

1. เขียนฟังก์ชันต่อไปนี้
   - รับค่าข้อความ 'James had had had the cat.'
   - นับจำนวนคำว่า had
2. เขียนฟังก์ชันต่อไปนี้
   - รับค่าข้อความ 'I intend to live forever, or die trying.'
   - แทนค่าคำว่า to ด้วย three
3. เขียนฟังก์ชันต่อไปนี้
   - คำนวณความยาวของ string ที่รับมาจากผู้ใช้

4. เขียนฟังก์ชันต่อไปนี้

    - รับ string มาแล้วเปลี่ยนค่ากลับกันระหว่างตัวอักษรตัวแรกกับตัวสุดท้ายของ string นั้น

บทที่ 8

ลิสต์ (Lists)

8.1 ความหมายของลิสต์

ลิสต์ (List) เป็นโครงสร้างข้อมูลที่สำคัญมากของภาษา Python คล้ายคลึงกับ Array ในภาษาอื่นๆ คือ ชุดของข้อมูลที่เรียงลำดับต่อๆ กัน จะเป็นค่าของอะไรก็ได้ การสร้าง list โดยกำหนดสัญลักษณ์ `[]` เช่น `t = []` หรือ `t=list()`

รูปที่ 8.1:

```
>>> t = []
>>> t
[]
>>> type(t)
<class 'list'>
>>> id(t)
55105976
>>> t = list()
```

ตัวอย่างของการกำหนดค่าใน List เช่น `t = [1, 2, 3, 4]` เป็น list ของตัวเลข `t = [1, "yes", "no", 1.1]` เป็นลิสต์ผสมของตัวเลขและข้อความ และ `t = [1, 2, 3, ['yes', 'no'], []]` เป็นลิสต์ที่มีลิสต์เป็นองค์ประกอบอยู่ด้วย กล่าวได้ว่าเราสามารถเอาค่าของหลายๆ ประเภทมาอยู่รวมกันในลิสต์เดียวกันได้

รูปที่ 8.2:

```
>>> t = [1,2,3,4,5]
>>> len(t)
5
>>> x = [1,'Mike',1.1,True]
>>> x
[1,'Mike',1.1,True]
>>> len(x)
4
>>> y = [1,['a','b'],True]
>>> y
[1,['a','b'],True]
>>> len(y)
3
>>>
```

8.2 การเข้าถึงค่าในลิสต์

ค่าในลิสต์จะเรียงตามตัวชี้หรือ Index ที่เริ่มต้นที่ 0 เช่น `t=[0]` และหากมีลิสต์ซ้อนอยู่ในลิสต์จะสามารถแสดงตัวชี้ได้ด้วยการกำหนดตัวชี้หลักตามด้วยตัวชี้ย่อย เช่น `t=[1][0]`

รูปที่ 8.3:

```
>>> y = [1,['a','b'], True]
>>> y
[1,['a','b'], True]
>>> len(y)
3
>>> t
[1, 2, 3, 4, 5, 6]
>>> t[0]
1
>>> t[1]
2
>>> t[2]
3
>>> y[0]
1
>>> y[1]
['a', 'b']
>>> y[1][0]
'a'
```

8.3 การแบ่งข้อมูลในลิสต์ (List Slicing)

List slicing หรือการแบ่งข้อมูลในลิสต์เป็นชุดข้อมูลย่อยๆ จะเขียนในรูปแบบ `[a:b]` เมื่อ a เป็น Index เริ่มต้นและ b เป็น Index ก่อนสมาชิกตัวสุดท้ายที่ต้องการตัด

รูปที่ 8.4:

```
>>> t
[1, 2, 3, 4, 5, 6]
>>> t[1:3]
[2,3]
>>> t[2:]
[3, 4, 5, 6]
>>> t[:4]
[1, 2, 3, 4]
>>>
```

8.4 Lists เปลี่ยนแปลงค่าได้

ลิสต์สามารถเปลี่ยนแปลงค่าได้

รูปที่ 8.5:

```
>>> t
[True, 2, 3, 4, 5, 6]
>>> t[1] = 'Hello World'
>>> t
[True, 'Hello World', 3, 4, 5, 6]
>>> t[1]
'Hello World'
```

8.5 การใช้ in กับลิสต์

การดำเนินการด้วย in สามารถใช้กับลิสต์ได้

รูปที่ 8.6:

```
>>> t
[True, 2, 3, 4, 5, 6]
>>> t[1] = 'Hello World'
>>> t
[True, 'Hello World', 3, 4, 5, 6]
>>> t[1]
'Hello World'
>>> 3 in t
True
```

8.6 การเดินทางไปในลิสต์ (List Traversal)

8.6.1 การใช้ for loop และตัวดำเนินการ in ในการเดินทางไปในลิสต์

Source code:

รูปที่ 8.7:

```
fruits = ['banana', 'orange', 'mango']
for fruit in fruits: print(fruit)
```

Result:

รูปที่ 8.8:

```
>>>
banana
orange
mango
>>>
```

8.6.2 การใช้ฟังก์ชัน `range()` กับลิสต์

สำหรับ `range()` ของความยาวของตัวแปร จะถูกนำมาใช้ในการเดินทางด้วย index ในลิสต์

Source code:

รูปที่ 8.9:

```
for i in range(len(fruits)):
    print('{0} = {1}' .format(i, fruits[i]))
```

Result:

รูปที่ 8.10:

```
>>>
0 = banana
1 = orange
2 = mango
>>>
```

8.7 ตัวดำเนินการของลิสต์ (List Operators)

ถ้าต้องการเอาลิสต์มารวมกันให้ใช้เครื่องหมายบวก (+) ถ้าต้องการขยายลิสต์ให้มีค่าชุดเดิมเพิ่มเป็นกี่เท่าตัวให้ใช้เครื่องหมายดอกจัน (*)

รูปที่ 8.11:

```
>>> a = [1,2,3]
>>> b = [4,5,6]
>>> a
[1, 2, 3]
>>> b
[4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
>>> a * 2
[1, 2, 3, 1, 2, 3]
>>>
```

8.8 เมธอดของลิสต์ (List Methods)

การขยายลิสต์ด้วยอีกลิสต์หนึ่ง ให้ใช้คำสั่ง `list.extend()`

รูปที่ 8.12:

```
>>> a
[1, 2, 3]
>>> b
[4, 5, 6]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6]
```

การเพิ่มค่าในลิสต์ทำได้ด้วยคำสั่ง `list.append()` ค่าใหม่ที่ได้จะต่อท้ายตัวสุดท้ายในลิสต์

รูปที่ 8.13:

```
>>> a.append(7)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

การเพิ่มค่าในลิสต์โดยกำหนดตำแหน่งของ index ให้ใช้คำสั่ง `list.insert()`

รูปที่ 8.14:

```
>>> a.insert(0, 0)
>>> a
[0, 1, 2, 3, 4, 5, 6, 7]
```

การลบค่าในลิสต์ทำได้ด้วยคำสั่ง `list.remove()`

รูปที่ 8.15:

```
>>> a.remove(7)
>>> a
[0, 1, 2, 3, 4, 5, 6]
```

นอกจากนี้สามารถลบค่าในลิสต์ได้ด้วยคำสั่ง del โดยจะต้องระบุค่า Index ของตัวที่ต้องการลบ เช่น `del a[0]` เป็นการลบค่าลิสต์ที่มี index 0 หรือถ้าลบเป็นช่วงให้ระบุตำแหน่งเริ่มต้นที่จะลบจนถึงตำแหน่งตัวก่อนสุดท้ายที่จะลบ `del a[2:4]` จะลบตัวที่ 2 และ 3 ในลิสต์ และถ้าลบค่าในลิสต์ออกทั้งหมดให้ใช้คำสั่ง `del a[ : ]`

ศึกษาเรื่อง list methods เพิ่มเติมได้ที่ `https://docs.python.org/3/tutorial/datastructures.html`

8.9 Map, reduce, and filter

การสร้าง Map ฟังก์ชัน เป็นการทำให้ลิสต์หนึ่งเป็นอีกลิสต์หนึ่ง ให้ฟังก์ชันชื่อ `capitalize()` รับลิสต์ t มา แล้ว return ลิสต์ r แล้วทำการเดินทางในค่าแต่ละค่า เมื่อเจอค่าก็ให้ทำการประมวลผลเป็นตัวพิมพ์ใหญ่แล้วเก็บไว้ในลิสต์ r

Source code:

รูปที่ 8.16:

```
def capitalize(t):
    r = []
    for s in t:
        r.append(s.capitalize())
    return r
```

Result:

รูปที่ 8.17:

```
>>> capitalize(['a', 'b', 'c', 'd'])
['A', 'B', 'C', 'D']
>>>
```

การสร้าง Reduce ฟังก์ชันเป็นการประมวลผลลิสต์เพื่อการผลสรุป ให้ฟังก์ชันชื่อ `sum()` รับลิสต์ t มา

แล้ว return ค่า sum โดยกำหนดตัวแปรชื่อ sum ให้ค่าเป็น 0 แล้วเดินทางไปในลิสต์ทีละค่า แล้วนำค่าเมื่อบวกกัน เมื่อบวกจนครบทุกตัวแล้วให้ส่งค่ากลับมาเป็น sum

Source code:

รูปที่ 8.18:

```
def sum(t):
    sum = 0
    for x in t: sum += x
    return sum
```

Result:

รูปที่ 8.19:

```
>>> sum([1,2,3,4,5])
15
```

ฟังก์ชันอีกประเภทหนึ่งเรียกว่า filter ฟังก์ชัน คือการ search แบบรับลิสต์มาแล้ว return ค่า คือมีการค้นหาแล้วทำการประมวลผลค่านั้นๆ หรืออาจจะ return มาเป็นลิสต์ก็ได้ แล้วก็ทำการประมวลผลกับข้อมูลที่อยู่ในลิสต์นั้น เช่น ฟังก์ชันรับลิสต์ t เข้าไป แล้ว return ลิสต์ที่เป็น integer เท่านั้น โดยตั้งต้นสร้างลิสต์ r แล้วเดินทางไปในค่าแต่ละค่าของลิสต์ t ถ้าเจอว่าประเภทของค่าเป็น int ให้ทำการแทรกค่าในลิสต์ r เมื่อทำครบแล้วให้ส่งค่าลิสต์ r ออกมา

Source code:

รูปที่ 8.20:

```
def only_int(t):
    r = []
    for x in t:
        if type(x) == int: r.append(x)
    return r
```

Result:

รูปที่ 8.21:

```
>>> only_int([1,2,3,True,'hello',4,'abcdef',1.1)
[1, 2, 3, 4]
>>>
```

8.10 Lists กับ String

String เปลี่ยนแปลงค่าไม่ได้ แต่ List เปลี่ยนแปลงค่าได้ ทั้ง String และ List เป็นการเรียงลำดับและเปลี่ยนแปลงค่ากลับกันไปมาได้ด้วยการใช้ฟังก์ชันของลิสต์ เช่น `split()`, `join()`

รูปที่ 8.22:

```
>>> s = 'Mink is a cat.'
>>> s
'Mink is a cat.'
>>> t = s.split()
>>> t
['Mink', 'is', 'a', 'cat.']
>>> ' '.join(t)
'Mink is a cat.'
>>> p = '081-123-4567'
>>> p
'081-123-4567'
>>> t = p.split('-')
>>> t
['081', '123', '4567']
>>> '-'.join(t)
'081-123-4567'
```

8.11 Objects and values

ภาษา Python เพื่อประหยัดพื้นที่ในหน่วยความจำ สำหรับ String ซึ่งไม่สามารถเปลี่ยนแปลงค่าได้ หรือ Immutable Data Structure ภาษา Python จะชี้ชื่อตัวแปรไปที่ที่เดียวกันสำหรับค่าที่เหมือนกัน

รูปที่ 8.23:

```
>>> a = 'banana'
>>> b = 'banana'
>>> id(a)
67486272
>>> id(b)
67486272
>>> a is b
True
```

แต่สำหรับลิสต์ซึ่งเปลี่ยนแปลงค่าได้ หรือ Mutable Data Structure ภาษา Python จะเก็บไว้คนละที่ในหน่วยความจำ แต่สามารถสร้างชื่อตัวแปรต่อๆ กันมาได้ สำหรับตำแหน่งหนึ่งๆ เรียกว่า การทำ Aliasing

Objects อยู่ในหน่วยความจำมี Values แต่ไม่มีชื่อ แต่มี id หรือหมายเลขกำกับ สร้างตัวแปรเพื่อชี้ไปยัง Objects เหล่านั้น ตั้งตัวแปรหลายตัวหรือตัวเดียวก็ได้

รูปที่ 8.24:

```
>>> a = [1,2,3]
>>> b = [1,2,3]
>>> a is b
False
>>> id(a)
67462368
>>> id(b)
67464088
>>> c = a
>>> id(c)
67462368
>>> c is a
True
```

8.12 แบบฝึกหัด

1. กำหนดคะแนนของนักเรียน 5 คน เก็บไว้ใน list คือ 75 80 68 82 62 ต้องการหาคะแนนรวม คะแนนเฉลี่ย คะแนนมากสุด คะแนนน้อยสุด

2. ให้ข้อมูลเป็น list มีค่าคือ 3 4 12 31 ให้หาจำนวนสมาชิกใน list และพิมพ์สมาชิกทุกตัวตัวละบรรทัด

3. ให้ข้อมูลเป็น list มีค่าคือ 25 4 3 15 21 นำมาเรียงจากน้อยไปหามาก พร้อมหาผลคูณของสมาชิกทุกตัว

4. ให้ข้อมูลเป็น list มีค่าคือ 6 9 8 7 10 ให้ลบข้อมูลตัวแรกทิ้งแล้วเพิ่มข้อมูลตัวแรกไปที่ตำแหน่งสุดท้าย

5. จงสร้างข้อมูลเป็น list ชื่อ a มีค่าคือ 1-10 และ list ชื่อ b มีค่าคือ 11-20 โดยใช้ for และใช้ฟังก์ชัน append เพื่อเพิ่มสมาชิกใน list แล้วหาค่า a+b

บทที่ 9

ดิกชันนารี (Dictionary)

9.1 ความหมายของดิกชันนารี

ดิกชันนารี (Dictionary) คือประเภทข้อมูลที่เก็บข้อมูลในเป็นคู่ๆ ของ Key และ Value โดยที่ Key ใช้สำหรับเป็น Index ในการเข้าถึงข้อมูล Value ของ Key นั้นๆ การสร้าง Dictionary เปล่าจะเขียนอยู่ภายในวงเล็บปีกกา หรือ dict() ส่วนการใส่ค่าใน Dictionary จะเขียนอยู่ในวงเล็บปีกกา แต่ละคู่จะขึ้นต้น key ตามด้วย : แล้วตามด้วย value และคั่นคู่ด้วยเครื่องหมาย comma (,)

รูปที่ 9.1:

```
>>> stocks = {'scb':160, 'ptt':360}
>>> stocks
{'scb':160, 'ptt':360}
```

9.2 การอ่านค่าในดิกชันนารี

การอ่านค่าใน Dictionary ด้วยคำสั่ง for หรือการใช้ Dictionary เป็น Iterators เมื่อมีการประกาศค่าของ Dictionary แล้ว for จะวนอ่านค่าใน Dictionary ทีละตัว เช่น `for key in stocks: print(key, stocks[key])`

Source code:

รูปที่ 9.2:

```
stocks = {'scb':160, 'ptt':360}
for key in stocks:
    print('{0} = {1}' .format(key, stocks[key]))
```

Result:

รูปที่ 9.3:

```
ptt = 360
scb = 160
```

9.3 การหาค่าของคีย์ (Key) ใน Dictionary

การหาค่าของคีย์ในดิกชันนารีเมื่อรู้ value เช่น สร้างฟังก์ชัน `reverse_lookup()` มีการส่ง Parameters สองตัวคือ Dictionary กับ Value สำหรับ Key แต่ละตัวใน Dictionary นี้ ถ้า ค่าของ Dictionary เท่ากับ Value ที่ต้องการ ให้ส่งค่า Key กลับมา แต่ถ้าไม่มีก็จะไม่ต้องส่งอะไรออกมา

Source code:

รูปที่ 9.4:

```
def reverse_lookup(d,v):
    for key in d:
        if d[key] == v: return key
    return None
```

Result:

รูปที่ 9.5:

```
>>> reverse_lookup(stocks, 160)
'scb'
```

9.4 Dictionary และ List

keys() method จะแสดง keys ออกมา ส่วน values() method จะแสดง values ออกมา

รูปที่ 9.6:

```
>>> stocks.keys()
dict_keys(['ppt','scb'])
>>> stocks.values()
dict_values([360,160])
```

ดังนั้น stocks.keys() และ stocks.values() คือ ลิสต์สองตัว ตัวหนึ่งเป็น keys อีกตัวเป็น values ที่ map เข้าหากัน จำไว้ว่า ค่าของลิสต์หรือ dictionary สามารถเป็นได้ทั้งลิสต์และ dictionary ด้วย

9.5 ฟังก์ชันที่รับ Parameters ได้ไม่จำกัดจำนวน

เมื่อไรก็ตามที่ต้องการสร้างฟังก์ชันที่รับ Parameters ได้ไม่จำกัดจำนวนและบอก Keywords ได้ด้วย ให้ใส่เครื่องหมายดอกจัน (*) สองครั้งไว้หน้า Parameters หมายความว่าสิ่งที่ผ่านเข้ามาทาง Parameters จะเป็น Dictionary

Source code:

รูปที่ 9.7:

```
def printall(**kwargs):
    for key in kwargs:
        print(key + ' ' + str(kwargs[key]))
```

Result:

รูปที่ 9.8:

```
>>> printall(x=1, y=2, z=3)
y 2
z 3
x 1
```

แต่หากต้องการให้ Arguments หรือ Parameters ที่รับเข้ามาเป็น list ให้ใส่เครื่องหมายดอกจัน (*) หนึ่งครั้งไว้หน้า Parameters

Source code:

รูปที่ 9.9:

```
def printall(*args):
    for x in args:
        print(x)
```

Result:

รูปที่ 9.10:

```
>>> printall(1,2,3,4,5)
1
2
3
4
5
```

9.6 แบบฝึกหัด

1. ให้ dictionary มีค่าคือ 0: 10, 1: 20 เขียนโปรแกรมให้เพิ่มค่าอีกคู่เข้าไป คือ 2:30

2. ทำการรวมค่าทั้งหมดใน dictionary นี้ 'cats':100,'dogs':60,'pigs':300

3. ทำการเชื่อมต่อ dictionary ทั้ง 3 ชุดนี้ให้เป็นชุดเดียว

    - dic1=1:10, 2:20 dic2=3:30, 4:40 dic3=5:50,6:60

4. จงดำเนินการต่อไปนี้

    - สร้าง dictionary ชื่อ stock มีค่าคือ "banana": 40, "apple": 10, "orange": 15, "pear": 33
    - สร้าง dictionary ชื่อ prices มีค่าคือ "banana": 4, "apple": 2, "orange": 1.5, "pear": 3
    - ให้คำนวณว่าถ้าขายผลไม้ได้ทั้งหมดจะได้เงินเท่าไร

5. เขียนฟังก์ชันตรวจสอบว่ามีค่าตัวเลขที่รับมาให้ key ของ dictionary หรือไม่ โดยให้ d คือ dictionary มีค่าคือ 1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60

บทที่ 10

ทูเบิล (Tuple)

10.1 ความหมายของ Tuple

Tuple จะคล้ายกับ List แต่สิ่งที่แตกต่างกันคือ Tuple นั้นเป็นประเภทข้อมูลที่ไม่สามารถเปลี่ยนแปลงได้ เมื่อไม่ต้องการให้ส่วนใดส่วนหนึ่งของโปรแกรมเผลอไปเปลี่ยน Value ก็ควรใช้ Tuple การสร้าง Tuple นั้น จะอยู่ภายในวงเล็บ () และคั่นค่าแต่ละตัวด้วยเครื่องหมายคอมมา (,) ส่วนการเข้าถึงค่าใน Tuple ใช้ index เหมือนกับ list

รูปที่ 10.1:

```
>>> t = 'a', 'b', 'c'
>>> t
('a', 'b', 'c')
>>> t = 'a',
>>> t
('a',)
>>>
```

รูปที่ 10.2:

```
>>> t = ('a', 'b', 'c')
>>> t[0]
'a'
>>> t[1]
'b'
>>> t[1:]
('b', 'c')
>>> t[0] = 'z'
Traceback (most recent call last):
    File ''<pyshell#16>'', line 1, in <module>
        t[0] = 'z'
TypeError: 'tuple' object does not support item assignment
>>>
```

10.2 การสลับค่าของ Tuple

การสลับค่าของ Tuple สามารถเขียน a, b = b, a และสามารถกำหนดค่าจาก String มาเป็น Tuple ได้ด้วยคำสั่ง split() เช่น username, domain = 'support@classstart.org'.split('@')

รูปที่ 10.3:

```
>>> username, domain = 'support@classstart.org'.split('@')
>>> username
'support'
>>> domain
'classstart.org'
>>>
```

10.3 การเก็บค่าการดำเนินการใน Tuple

เราสามารถใช้ Tuple ในการเก็บค่าที่ได้จากการดำเนินการได้โดยตรง เช่น floor, remainder = divmod(7, 3) โดยที่ floor คือ ค่าจำนวนเต็มที่ได้จากการหาร ส่วน remainder คือค่าของเศษที่ได้จากการหาร

รูปที่ 10.4:

```
>>> t = divmod(7,3)
>>> t
(2, 1)
>>> floor, remainder = divmod(7,3)
>>> floor
2
>>> remainder
1
```

Source code:

รูปที่ 10.5:

```
def split_email(email):
    return email.split('@')
```

Result:

รูปที่ 10.6:

```
>>> username, domain = split_email('support@classstart.org')
>>> username
'support'
>>> domain
'classstart.org'
>>>
```

10.4 ฟังก์ชัน `list()` เปลี่ยน tuple ให้เป็น list

รูปที่ 10.7:

```
>>> t = (1,2,3,4,5)
>>> t
(1,2,3,4,5)
>>> type(t)
<class 'tuple'>
>>> mylist = list(t)
>>> mylist
[1,2,3,4,5]
>>>
```

10.5 Dictionary และ Tuple

เมธอด items() ของ Dictionary จะให้ค่าเป็น List ของ Tuples โดย Tuples แต่ละตัวคือ Key และ Value

รูปที่ 10.8:

```
>>> d = {'ppt' : 360, 'scb' : 160}
d
{'ppt' : 360, 'scb' : 160}
>>> d.items()
dict_items([('ppt', 360),('scb', 160)])
```

สรุปความแตกต่างในการใช้ Data Types คือถ้าหากต้องการลำดับของอักษร จะใช้ String ถ้าต้องการลำดับของค่าที่เปลี่ยนแปลงได้จะใช้ List ถ้าต้องการลำดับของค่าที่เปลี่ยนแปลงไม่ได้จะใช้ Tuple ถ้าต้องการคู่ลำดับของ Key กับ Value จะใช้ Dictionary

10.6 แบบฝึกหัด

1. จงเขียนโปรแกรมสร้าง Tuple มีค่าดังต่อไปนี้ (”tuple”, False, 3.2, 1) และแสดงค่าออกมา

2. จงเขียนโปรแกรมสร้าง Tuple มีค่าดังต่อไปนี้ 4 8 3 แล้วทำการแยกค่าแต่ละค่าให้เป็นตัวแปรแต่ละตัว แล้วให้คำนวณผลรวมของตัวแปรทั้งหมด

3. จงเขียนโปรแกรมสร้าง Tuple มีค่าดังต่อไปนี้ 4 6 2 8 3 1 แล้วให้แปลง Tuple เป็น List และการเพิ่มเลข 30 เข้าไป แล้วให้แปลง list กลับมาเป็น Tuple ทำการ Print Tuple นั้น

4. จงเขียนโปรแกรมแปลง Tuple ให้เป็น String โดยให้ Tuple มีค่าคือ (’e’, ’x’, ’e’, ’r’, ’c’, ’i’, ’s’, ’e’, ’s’)

5. จงเขียนโปรแกรมตรวจสอบว่ามีข้อมูลอยู่ใน Tuple หรือไม่ โดยให้ Tuple มีค่าคือ (”w”, 3, ”r”, ”e”, ”s”, ”o”, ”u”, ”r”, ”c”, ”e”)

บทที่ 11

บทที่ 10 การจัดการไฟล์ (Files)

11.1 ความหมายของไฟล์

ไฟล์คือพื้นที่เก็บข้อมูลบนคอมพิวเตอร์ ไฟล์มีหลายประเภทตามการใช้งาน เช่น ไฟล์ Microsoft Word ไฟล์เพลง และไฟล์ VDO เป็นต้น โดยทั่วไปไฟล์มี 2 ประเภท คือ Text files และ Binary files โดย Text files เป็นไฟล์ที่เก็บชุดข้อความซึ่งเปิดอ่านได้ ส่วน Binary files จะอยู่ในรูปแบบของ Binary form เพื่อให้คอมพิวเตอร์ทำงาน

การเขียนโปรแกรมเพื่อใช้งานไฟล์ เช่น อ่านไฟล์ เขียนไฟล์ สร้างไฟล์ และแก้ไขไฟล์ โดยในภาษา Python จะต้องเรียกใช้โมดูล os คือ ระบบปฏิบัติการ (Operating Systems) os module มีความสามารถหลายอย่าง ศึกษาเพิ่มเติมที่ `https://docs.python.org/3.5/library/os.html`

11.2 การทำงานกับ Directories

การเรียกใช้โมดูล os ต้องใช้คำสั่ง import os ก่อนแล้วจะสามารถใช้คำนั่งในโมดูลได้ เช่น os.getcwd() ใช้แสดง Directory ที่กำลังทำงานอยู่ ส่วนคำนั่ง os.chdir() คือการเปลี่ยนตำแหน่งการทำงานของ Directory

รูปที่ 11.1:

```
%>>> import os
%>>> os.getcwd()
%\rq{'c:\\Users\\\\janta\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python37-32'}
%>>> os.chdir('c:\\\\Users\\\\janta\\\\Desktop')
```

คำสั่งอื่นๆ เช่น

- `os.path.abspath('file.txt')`
- `os.path.exists('file.txt')`
- `os.path.isdir('file.txt')`
- `os.path.isdir(os.getcwd())`
- `os.listdir(os.getcwd())`

รูปที่ 11.2:

```
>>> os.path.isdir(os.getcwd())
True
>>> os.listdir(os.getcwd())
['DLLs', 'Doc', 'include', 'Lib', 'libs', 'LICENSE.txt', 'NEWS.txt', 'python.exe',
>>>
```

11.3 การเปิดไฟล์

ฟังก์ชัน`open()` มีไว้เพื่อเปิดไฟล์ ก่อนที่จะเริ่มทำงานกับไฟล์ทุกครั้งจะต้องทำการเปิดไฟล์ก่อน โดยมีรูปแบบการเขียนคือ `fout = open(filename, flag)` ซึ่ง Filename คือ ชื่อไฟล์ที่ต้องการเปิด ส่วน Flag คือ รูปแบบในการเปิดไฟล์ ซึ่งมีหลายแบบ เช่น r ใช้อ่านข้อมูลอย่างเดียว w ใช้เขียนข้อมูลลงในไฟล์ใหม่ที่ไม่ได้สร้างไว้ก่อนหน้า a ใช้เขียนต่อท้ายไฟล์เดิม เป็นต้น และทุกครั้งหลังใช้งานไฟล์เสร็จแล้ว จะต้องทำการปิดไฟล์ด้วยคำสั่ง `close()`

รูปที่ 11.3:

```
%import os
%os.chdir('c:\textbackslash\textbackslash Users\\janta\\Desktop')
%fout = open('file.txt', 'w')
%fout.write('The first line. \textbackslash n')
%fout.write('The second line. \textbackslash n')
%fout.write('The thrid line. \textbackslash n')
%fout.close()
```

11.4 การอ่านไฟล์

คำสั่ง `read()` จะทำการอ่านข้อมูลทั้งหมดในไฟล์เพียงครั้งเดียว และข้อมูลที่อ่านได้จะเป็น String ส่วนคำสั่ง `readline()` จะทำการอ่านข้อมูลทีละบรรทัดแล้วเก็บไว้เป็น String ทีละบรรทัดใน List

11.5 การจัดการข้อผิดพลาด (Error)

การเขียนโปรแกรมเพื่อให้สามารถจัดการเหตุการณ์ที่คาดไม่ถึงได้ อาทิ เปิดไฟล์ไม่ได้เพราะ Hard disk มีปัญหา หรือ Network มีปัญหา สามารถเขียนโปรแกรมให้แสดง Error message ออกมาได้ โดยใช้ Try and Except statements สำหรับ `FileNotFoundError` เช่น ในการเปิดไฟล์ที่ยังไม่ได้สร้างไฟล์ไว้ก่อนหน้า แล้วต้องการให้ Error message แสดงออกมาว่า File not found!

Source code:

รูปที่ 11.4:

```
try:
    fin = open('abcdef.txt')
except FileNotFoundError:
    print('File not found.')
```

Result:

รูปที่ 11.5:

```
>>>
File not found.
>>>
```

นอกจากนี้ยังมี finally statement จะทำงานท้ายสุดไม่ว่าจะประสบความสำเร็จหรือไม่ก็ตาม อ่านเพิ่มเติมเรื่อง eexception handling ได้ที่ `https://docs.python.org/3/library/exceptions.html#bltin-exceptions`

Source code:

รูปที่ 11.6:

```
try:
    fin = open('file.txt')
except:
    print('File not found.')
finally:
    print('Finally...everything is ok.')
```

Result:

รูปที่ 11.7:

```
>>>
Finally...everything is ok.
>>>
```

Source code:

รูปที่ 11.8:

```
def print_james(name):
    if name == 'James':
        print('Hello James.')
    else:
        raise Exception('Name is not James.')

print_james('Jake')
```

Result:

รูปที่ 11.9:

```
%Traceback (most recent call last):
 % File "C:\Users\janta\OneDrive\Documents\Python 1_63\01_Overview\code01.py", line
  %  print_james('Jake')
 % File "C:\Users\janta\OneDrive\Documents\Python 1_63\01_Overview\code01.py", line
  %  raise Exception('Name is not James.')
%Exception: Name is not James.
```

11.6 ฐานข้อมูลแบบ Key-Value

Databases คือ ไฟล์แบบ Binary ที่เก็บข้อมูลในเครื่องคอมพิวเตอร์ที่เก็บในลักษณะ Key และ Value เหมือน Dictionary โดยจะต้องมีการ import โมดูล dbm ก่อนซึ่งเป็นการจัดการเกี่ยวกับ Database และผลลัพธ์ที่ได้จากการอ่านไฟล์ Database จะมีการแสดงตัวอักษร b ไว้ด้านหน้าเพื่อแสดงความเป็น Binary

Source code:

รูปที่ 11.10:

```
import dbm
db = dbm.open('stocks.db', 'c')
db['ptt']='360'
db['scb']='160'
print(db['ptt'])
print(db['scb'])
db.close()
```

Result:

รูปที่ 11.11:

```
>>>
b'360'
b'160'
>>>
```

การ Pickling เซฟข้อมูลเป็นอย่างอื่นนอกจาก text ดังนั้นให้แปลงอย่างอื่นให้เป็น text แล้วแปลงเป็น object เพื่อดึงกลับขึ้นมา

เริ่มต้นด้วยการ import pickle เช่น ให้ list ชื่อว่า t1 แล้วทำการ dumps(t1) ให้กลายเป็น string ตอนนี้ก็จะสามารถเอา string ไปเก็บในไฟล์ข้อมูลหรือเก็บใน key-value database ก็ได้ และเมื่อต้องการเรียกนำกลับมาใช้ในสภาพเดิมให้ใช้คำสั่ง loads(s)

รูปที่ 11.12:

```
>>> import pickle
>>> t = [1,2,3,4,5,6,7]
>>> t
[1,2,3,4,5,6,7]
>>> t1 = t
>>> t1
[1,2,3,4,5,6,7]
>>> s = pickle.dumps(t1)
>>> s
% b'\x80\x04\x95\x13\x00\x00\x00\x00\x00\x00\x00]\x94(K\x01K\x02K\x03K\x04K\x05K\x
>>> t2 = pickle.loads(s)
>>> t2
[1, 2, 3, 4, 5, 6, 7]
>>> id(t1)
66507840
>>> id(t2)
63748456
```

11.7 การให้ Python เรียกใช้โปรแกรมอื่น

การให้ Python เรียกใช้โปรแกรมอื่นได้ (Piping) โดยใช้โมดูล os และใช้ฟังก์ชัน popen() อาทิ ตั้งตัวแปรชื่อ cmd เรียกใช้คำสั่ง dir ของ Windows OS ด้วยคำสั่ง popen แล้วนำผลจากคำสั่งนั้นมาเก็บเป็น String ไว้ในตัวแปรชื่อ result

Source code:

รูปที่ 11.13:

```
import os
os.chdir('c:\\users\\janta\\Desktop')
cmd = 'dir'
fp = os.popen(cmd)
result = fp.read()
fp.close()
print(result)
```

Result:

รูปที่ 11.14:

```
Volume in drive C has no label.
Volume Serial Number is 2E09-DC35
Directory of c:\users\janta\Desktop
07/05/2020  01:34 PM    <DIR>          .
07/05/2020  01:34 PM    <DIR>          ..
07/03/2020  05:46 PM            52,422 560gra01.png
07/05/2020  01:34 PM            29,887 5f83b4e260d668989b715e8bf0e3e288.jpg
07/02/2020  07:04 PM           299,334 latexsheet-a4.pdf
07/03/2020  06:22 PM        14,832,375 Manual_MS-Teams_Full.pdf
07/03/2020  06:11 PM             2,394 Microsoft Teams.lnk
               5 File(s)     15,216,412 bytes
               2 Dir(s)  90,572,718,080 bytes free
```

11.8 แบบฝึกหัด

1. เขียนฟังก์ชันอ่าน text file ทั้งไฟล์
2. เขียนฟังก์ชันอ่านไฟล์ n บรรทัดแรกใน text file
3. เขียนฟังก์ชันอ่านไฟล์ทีละบรรทัดแล้วเก็บไว้ใน list
4. สร้าง text file ขึ้นมา แล้วเพิ่มบรรทัดใหม่หนึ่งบรรทัดว่า Welcome to my class. แล้ว print เนื้อหาออกมาทั้งหมด

บทที่ 12

Object-Oriented Programming (OOP)

12.1 ความหมายของ OOP (Object-Oriented Programming)

OOP (Object-Oriented Programming) หมายถึงการเขียนโปรแกรมเชิงวัตถุหรือเขียนโปรแกรมแบบออบเจ็กต์ ซึ่งมีวิธีการของการจัดโครงสร้างโปรแกรมเพื่อให้คุณสมบัติและพฤติกรรมหรือการกระทำรวมอยู่ในแต่ละวัตถุ (Lutz, 2011) ตัวอย่างเช่น วัตถุอาจเป็นตัวแทนของบุคคลที่มีคุณสมบัติ คือ ชื่อ นามสกุล อายุ ที่อยู่ และมีพฤติกรรม เช่น การเดิน การพูด การหายใจ และการวิ่ง เป็นต้น

12.2 คลาส (Classes) และ ออบเจ็กต์ (Objects)

คลาส (Classes) เปรียบเสมือนแบบพิมพ์เขียวหรือแม่พิมพ์ที่ใช้สร้างออบเจ็กต์ (Objects) ในคลาสจะกำหนดรูปแบบของข้อมูลหรือแอตทริบิวต์หรือตัวแปร (Attributes/Properties/Characteristics) และเมธอด (Methods/Behaviors) ตัวอย่างเช่น คลาสชื่อ pet มีแอตทริบิวต์คือ legs ส่วน wolf คือ ออบเจ็กต์ที่สร้างขึ้นมาจากคลาส pet

รูปที่ 12.1:

```
class Pet: # define a class pet
    legs = 0  # a property for the class
    wolf = Pet() # create an object of the class
    wolf.legs # access to a property of the object
```

12.3 การสร้างคลาส

การสร้างคลาสในภาษา Python ต้องใช้คำสั่ง class ตามด้วยชื่อคลาส เช่น

รูปที่ 12.2:

```
class Pet: #define a class pet
    legs = 0 #a property for the class
```

12.4 การสร้างออบเจ็กต์

หลังจากที่สร้างคลาสแล้วก็จะสามารถสร้างตัวแปรออบเจ็กต์จากคลาสได้ เช่น `wolf = Pet()` โดยออบเจ็กต์นี้จะมีแอตทริบิวต์คือ legs และสามารถเข้าถึงแอตทริบิวต์ของออบเจ็กต์ได้โดยใช้เครื่องหมายจุด (.)

รูปที่ 12.3:

```
wolf = Pet() # create an object of the class
print(wolf.legs) # access to a property of the object
```

12.5 ฟังก์ชัน `__init__()`

ฟังก์ชัน `__init__()` เรียกว่าเป็นคอนสตรัคเตอร์ (Constructor) ซึ่งถูกเรียกใช้อัตโนมัติทุกครั้ง ในการกำหนดค่าให้แก่แอตทริบิวต์ของออบเจ็กต์และการดำเนินการต่างๆที่จำเป็นเมื่อต้องสร้างออบเจ็กต์ขึ้นมา และสำหรับพารามิเตอร์ self มีไว้เพื่อการเข้าถึงตัวแปรต่างๆ ที่เป็นของคลาส โดยที่ไม่จำเป็นต้องใช้คำว่า self ก็ได้ สามารถตั้งเป็นคำอื่นได้

รูปที่ 12.4:

```
class Person:
    name = None
    gender = None

    #Define the constructor
    def __init__(self, n, g):
        self.name = n
        self.gender = g
#Main code starts here
p1 = Person('Mike', 'male') # create object p1
p2 = Person('Jan', 'female') # create object p2
```

12.6 การสร้างเมธอดของออบเจ็กต์

เมธอดในออบเจ็กต์ก็คือฟังก์ชันที่เป็นของออบเจ็กต์นั้น ในตัวอย่างเป็นการสร้างเมธอด foo เพื่อทำการ print คำว่า Hello แล้วตามด้วยชื่อ เมื่อออบเจ็กต์จะเรียกใช้เมธอดก็ให้เขียนชื่อออบเจ็กต์นั้นตามด้วยจุดแล้วจึงตามด้วยชื่อเมธอด

Source code:

รูปที่ 12.5:

```
class Person:
    name = None
    gender = None
    def __init__(self, n, g):
        self.name = n
        self.gender = g
    def foo(self):
        print('Hello' , self.name)

#Main code starts here
p1 = Person('Mike', 'male')
p2 = Person('Jan', 'female')
p1.foo()
p2.foo()
```

Result:

รูปที่ 12.6:

```
Hello Mike
Hello Jan
```

12.7 การแก้ไขค่าของแอตทริบิวต์ของออบเจ็กต์

การแก้ไขค่าของแอตทริบิวต์ของออบเจ็กต์สามารถทำได้เหมือนการกำหนดค่าของตัวแปร

Source code:

รูปที่ 12.7:

```
>>> p1.name = 'Ton-mike'
>>> p1.foo()
Hello Ton-mike
```

12.8 การลบแอตทริบิวต์ของออบเจ็กต์

การลบแอตทริบิวต์ของออบเจ็กต์ให้ใช้คำสั่ง del ตามด้วยแอตทริบิวต์ของออบเจ็กต์นั้น

รูปที่ 12.8:

```
>>> del p1.name
>>> p1.foo()
Hello None
>>>
```

12.9 การลบออบเจ็กต์

การลบออบเจ็กต์ทำได้โดยใช้คำสั่ง del ตามด้วยชื่อของออบเจ็กต์ได้เลย

รูปที่ 12.9:

```
>>> del p1
>>> p1
Traceback (most recent call last):
  File "<pyshell\#29>", line 1, in <module>
    p1
NameError: name 'p1' is not defined
>>>
```

12.10 การสืบทอดคลาส (Class Inheritance)

การสืบทอดคลาส (Class Inheritance) คือการที่คลาสหนึ่งสามารถสืบทอดเมธอดและแอตทริบิวต์ ของคลาสนั้นไปยังคลาสอื่นได้ เรียกคลาสที่เป็นฐานหรือให้การสืบทอดนั้นว่าคลาสแม่ (Parent class) และ เรียกคลาสที่ได้รับการสืบทอด ว่าคลาสลูก (Child class) ตัวอย่าง คลาสแม่คือ SchoolMember ได้มีการ สืบทอดเมธอดและ แอตทริบิวต์ ไปยังคลาสลูกคือคลาส Teacher หลังจากนั้นคลาสลูกก็จะสามารถใช้เมธอด และแอตทริบิวต์ของคลาสแม่ได้

Source code:

รูปที่ 12.10:

```
#Define the class Teacher. It inherits from class SchoolMember
class Teacher(SchoolMember):
    def __init__(self, name, age, salary):
        SchoolMember.__init__(self, name, age)
        self.salary = salary

#Main code
teacher1 = Teacher('Mr. Mike Piyawat', 47, 50000)
teacher2 = Teacher('Mrs. Jane Doe', 55, 120000)

print(teacher1.name)
print(teacher1.age)
print(teacher1.salary)
print('\n')
print(teacher2.name)
print(teacher2.age)
print(teacher2.salary)
```

Result:

รูปที่ 12.11:

```
>>>
Mr. Mike Piyawat
47
50000


Mrs. Jane Doe
55
120000
>>>
```

12.11 แบบฝึกหัด

1. เขียนคลาสชื่อว่า Pet ประกอบด้วย

- คอนสตรัคเตอร์ (Constructor)
- แอตทริบิวต์ genre
- แอตทริบิวต์ legs
- เมธอด **`start_running`** แสดงผลทางหน้าจอว่า “Start running”
- เมธอด **`stop_running`** แสดงผลทางหน้าจอว่า “Stop running”

2. จากข้อ 1 ให้เขียนโปรแกรมเพื่อสร้างออบเจ็กต์จากคลาส Pet สองออบเจ็กต์ แล้วเรียกใช้เมธอดที่มี

บรรณานุกรม

Barry, P. (2016). *Head first python: A brain-friendly guide*. Sebastopol, CA, USA: O'Reilly Media.

Beazley, D., & Jones, B. K. (2013). *Python cookbook*. Sebastopol, CA, USA: O'Reilly Media.

Bouras, A. S. (2019). *Python and algorithmic thinking for the complete beginner (2nd edition): Learn to think like a programmer*. Independently published.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. Cambridge, MA, USA: The MIT Press.

Downey, A. B. (2015). *Think python: How to think like a computer scientist*. Sebastopol, CA, USA: O'Reilly Media.

Foundation, P. S. (2019, January). *Python.* Retrieved from `https://www.python.org/`

Guido, V. R. (2019, January). *Guido van rossum - personal home page*. Retrieved from `https://gvanrossum.github.io//help.html`

Lubanovic, B. (2015). *Introducing python: Modern computing in simple packages*. Sebastopol, CA, USA: O'Reilly Media.

Lutz, M. (2011). *Programming python: Powerful object-oriented programming*. Sebastopol, CA, USA: O'Reilly Media.

Lutz, M. (2013). *Learning python*. Sebastopol, CA, USA: O'Reilly Media.

Lutz, M. (2014). *Python pocket reference: Python in your pocket*. Sebastopol, CA, USA: O'Reilly Media.

Ramalho, L. (2015). *Fluent python: Clear, concise, and effective programming*. Sebastopol, CA, USA: O'Reilly Media.

Shuup. (2019, April). *25 of the most popular python and django websites.* Retrieved from `https://www.shuup.com/django/25-of-the-most-popular-python-and-django-websites/`

TIOBE. (2019, August). *The python programming language.* Retrieved from `https://www.tiobe.com/tiobe-index/python/`