SE-ED COMPUTER BOOK developer

การเขียนโปรแกรมด้วย Python ฉบับพื้นฐาน

เพิ่มเติมและปรับปรุง
เนื้อหาใหม่ทั้งหมด
พร้อมตัวอย่างโค้ด
จำนวนมาก

- เริ่มต้นการเขียนโปรแกรม ภาษาไพธอนตั้งแต่ขั้นพื้นฐาน
- จัดเรียงเนื้อหาจากง่ายไปหายาก เน้นการใช้รูปภาพประกอบคำอธิบาย
- มีตัวอย่างโค้ดที่หลากหลายรูปแบบ สามารถศึกษาเรียนรู้ได้ด้วยตนเอง

DOWNLOAD
ดาวน์โหลดโค้ดหนังสือเล่มนี้ได้ที่
http://www.developerthai.com

บัญชา ปะสีละเตสัง
ผู้เขียนหนังสือขายดีระดับ Best Seller
ด้าน Programming

ซีเอ็ด

การเขียนโปรแกรมด้วย Python ฉบับพื้นฐาน

โดย บัญชา ปะสีละเตสัง

สงวนลิขสิทธิ์ตามกฎหมาย โดย บัญชา ปะสีละเตสัง © พ.ศ. 2565
ห้ามคัดลอก ลอกเลียน ดัดแปลง ทำซ้ำ จัดพิมพ์ หรือกระทำอื่นใด โดยวิธีการใดๆ ในรูปแบบใดๆ
ไม่ว่าส่วนหนึ่งส่วนใดของหนังสือเล่มนี้ เพื่อเผยแพร่ในสื่อทุกประเภท หรือเพื่อวัตถุประสงค์ใดๆ
นอกจากจะได้รับอนุญาต

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ

บัญชา ปะสีละเตสัง.

การเขียนโปรแกรมด้วย Python ฉบับพื้นฐาน.--กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2565.
448 หน้า.
1. ไพธอน (ภาษาคอมพิวเตอร์). 2. การเขียนโปรแกรม (คอมพิวเตอร์).
I. ชื่อเรื่อง.

005.133

Barcode (e-book) : 9786160845170

ผลิตและจัดจำหน่ายโดย

บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)
SE-EDUCATION PUBLIC COMPANY LIMITED

เลขที่ 1858/87-90 ถนนเทพรัตน แขวงบางนาใต้ เขตบางนา กรุงเทพฯ 10260
โทรศัพท์ 0-2826-8000

[หากมีคำแนะนำหรือติชม สามารถติดต่อได้ที่ comment@se-ed.com]

คำนำ

ไพธอน (Python) เป็นหนึ่งในภาษาคอมพิวเตอร์ที่มีจำนวนผู้ใช้งานสูงสุดในปัจจุบัน เนื่องจากลักษณะโครงสร้างที่ไม่ซับซ้อน เรียนรู้ง่าย และความสามารถอันโดดเด่นในอีกหลายๆ ด้าน ซึ่งนอกจากจะเหมาะกับผู้เริ่มต้นศึกษาด้านการเขียนโปรแกรมแล้ว ไพธอนยังเป็นภาษาที่ถูกนำไปใช้ในงานแขนงต่างๆ อีกหลายอย่าง เช่น Data Science, Machine Learning, AI, Web Application, Graphics, Game รวมถึงงานทางด้านฮาร์ดแวร์และอื่นๆ อีกมากมาย ดังนั้น การเขียนโปรแกรมด้วยไพธอน จึงเป็นสิ่งสำคัญอีกอย่างหนึ่งสำหรับผู้ที่ต้องการก้าวสู่สายงานด้านการพัฒนาซอฟต์แวร์ระดับมืออาชีพควรต้องเรียนรู้เอาไว้

หนังสือเล่มนี้ได้รวบรวมเนื้อหาที่จำเป็นต้องรู้ในขั้นพื้นฐานทั้งหมด พร้อมตัวอย่างประกอบเพื่อเสริมความเข้าใจอีกเป็นจำนวนมาก โดยใช้เครื่องมือยอดนิยมอย่าง Visual Studio Code ร่วมกับ Jupyter Notebook ซึ่งผู้เริ่มต้นศึกษาการเขียนโปรแกรม สามารถเรียนรู้ด้วยตนเอง รวมถึงเป็นแนวทางสำหรับศึกษาเพิ่มเติมจากแหล่งข้อมูลอื่นๆ ต่อไปได้

บัญชา ปะสีละเตสัง

banchar_pa@yahoo.com
facebook.com/DeveloperThai

PYTHON
if
elif
else:
is
in
while
for
def
return

สารบัญ

บทที่ 1 Python, VS Code และ Jupyter 15

เกี่ยวกับภาษา Python 15

การติดตั้งภาษา Python 16

การใช้งาน Visual Studio Code (VS Code) 18

- การติดตั้ง VS Code 19
- หน้าจอ Welcome (Get Started) 20
- การเลือกรูปแบบสี (Theme) 21
- การปรับเปลี่ยนฟอนต์ 21

การเขียนโค้ด Python บน VS Code โดยตรง 23

Jupyter สำหรับ VS Code 25

การสร้างไฟล์ของ Jupyter 27

การใช้เครื่องมือของ Jupyter 29

- การเขียนโค้ดและการรัน 29
- การเพิ่มและลบเซลล์ 29

การนำโค้ดของหนังสือมาใช้งาน 30

บทที่ 2 พื้นฐานการเขียนโค้ด Python 33

คีย์เวิร์ดของภาษา Python 33

ลักษณะของตัวแปร 35

การกำหนดค่าให้กับตัวแปร 36
การกำหนดข้อมูลชนิดตัวเลข 37
การกำหนดข้อมูลชนิดสตริง 39
การแสดงผลด้วยฟังก์ชัน print() 41
- การใช้ฟังก์ชัน print() แบบพื้นฐาน 41
- การใช้ฟังก์ชัน print() ที่ซับซ้อนขึ้น 43

ลักษณะพื้นฐานของสตริงที่ควรรู้เพิ่มเติม 44
- การเชื่อมต่อสตริง 44
- การเขียนอักขระพิเศษบางตัวในสตริง 45
- การแทรกข้อมูลในสตริง 47

การเขียนคำอธิบายโค้ด 49
- คำอธิบายแบบบรรทัดเดียว 49
- คำอธิบายแบบหลายบรรทัด 49

การรับข้อมูลทางคีย์บอร์ดด้วยฟังก์ชัน input() 50

บทที่ 3 ข้อมูลตัวเลขและการคำนวณ 57

เครื่องหมายสำหรับการกำหนดค่า 57
เครื่องหมายสำหรับการคำนวณทางคณิตศาสตร์ 59
เครื่องหมายสำหรับการคำนวณและกำหนดค่า 69
ลำดับการประมวลผลของเครื่องหมาย 72
ฟังก์ชันเกี่ยวกับตัวเลข 73
- ฟังก์ชันที่มีอยู่แล้วในไพธอน (Built-in Function) 74
- ฟังก์ชันของโมดูล math 74

การสร้างเลขสุ่ม 76
การจัดรูปแบบของตัวเลข 79
- การจัดรูปแบบด้วยวิธี F-String 79
- การจัดรูปแบบด้วยฟังก์ชัน format() 80

การแปลงสูตรคณิตศาสตร์เป็นโค้ด Python 82

บทที่ 4 การเปรียบเทียบและกำหนดเงื่อนไข 85

ข้อมูลชนิดจริงเท็จ (Boolean) 85
เครื่องหมายสำหรับการเปรียบเทียบ 86
ลักษณะพื้นฐานของคำสั่ง if 89
การกำหนดเงื่อนไขด้วย Comparison Operator 92
การใช้คำสั่ง if-else 95
การใช้คำสั่ง if-elif 97
การใช้คำสั่ง if-elif-else 100
การเปรียบเทียบทางตรรกะ 101
การกำหนดหลายเงื่อนไขด้วย Logical Operator 103
การเปรียบเทียบช่วงข้อมูลแบบต่อเนื่องกัน 106
การใช้ if-else แบบ Conditional Expression 107
การกำหนดค่าตัวแปรตามเงื่อนไขโดยไม่ใช้ if 107
การหยุดประมวลผลเมื่อเกิดข้อผิดพลาด 108

บทที่ 5 การทำซ้ำแบบวนรอบ 111

ฟังก์ชัน range( ) 111
การใช้ลูปแบบ for-in 115
การใช้ลูปแบบ while 119
การใช้ลูปซ้อนกัน 122
การใช้คำสั่ง continue 125
การใช้คำสั่ง break 126

บทที่ 6 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 1 129

บทที่ 7 โครงสร้างข้อมูลแบบรายการ 151

รายการข้อมูลแบบ List 151
- การสร้าง List 152
- การอ้างถึงสมาชิกและใช้ลูป for-in ร่วมกับ List 152
- การใช้ตัวดำเนินการร่วมกับ List 155
- ฟังก์ชันที่สามารถใช้ร่วมกับ List 157

- ฟังก์ชันของ List 159

รายการข้อมูลแบบ Tuple 163
- การสร้าง Tuple 163
- การเข้าถึงสมาชิกของ Tuple 164
- การใช้ตัวดำเนินการและฟังก์ชันร่วมกับ Tuple 165
- ฟังก์ชันของ Tuple 165

รายการข้อมูลแบบ Set 168
- การสร้าง Set 168
- การเข้าถึงและจัดการ Set 169
- ฟังก์ชันและปฏิบัติการเกี่ยวกับ Set 170

รายการข้อมูลแบบ Dictionary 172
- การสร้าง Dictionary 172
- การเข้าถึงและเพิ่มสมาชิกของ Dictionary 173
- การใช้ตัวดำเนินการและฟังก์ชันร่วมกับ Dictionary 173
- ฟังก์ชันของ Dictionary 174

รายการข้อมูลแบบ 2 มิติ 176

บทที่ 8 การใช้สตริงเพิ่มเติม 181

ทบทวนลักษณะของข้อมูลชนิด String 181

ลำดับอักขระในสตริง 182

จัดการสตริงด้วยตัวดำเนินการและฟังก์ชัน 183

ฟังก์ชันของออบเจ็กต์สตริง 188
- การตรวจสอบสตริง 188
- การค้นหาและแทนที่ภายในสตริง 191
- การแยกและตัดช่องว่างในสตริง 193
- การเปลี่ยนลักษณะตัวพิมพ์ 194
- การจัดแนวขอบของสตริง 195

บทที่ 9 ข้อมูลประเภทวันเวลา 199

ข้อมูลประเภทวันเวลา 199

การกำหนดวันเวลาอ้างอิง 200

การอ่านค่าจากส่วนของวันเวลา 202
การจัดรูปแบบวันเวลา 204
การกำหนดวันเวลาอ้างอิงในแบบสตริง 206
การเปรียบเทียบวันเวลา 207
การหาระยะห่างของวันเวลา 207
การเพิ่มหรือลดวันเวลา 208

บทที่ 10 การสร้างและใช้งานฟังก์ชัน 213

ลักษณะของฟังก์ชัน 213
ฟังก์ชันแบบไม่ส่งค่ากลับ 215
- การสร้างฟังก์ชันแบบไม่ส่งค่ากลับ 216
- การเรียกใช้ฟังก์ชันแบบไม่ส่งค่ากลับ 216

ฟังก์ชันแบบส่งค่ากลับ 217
- การสร้างฟังก์ชันแบบส่งค่ากลับ 218
- การเรียกฟังก์ชันแบบส่งค่ากลับ 219
- การส่งคืนผลลัพธ์มากกว่า 1 ค่า 221

วิธีหยุดการทำงานของฟังก์ชัน 223
พารามิเตอร์และอาร์กิวเมนต์ 224
- Positional Argument 225
- Keyword Argument (Label) 225
- Default Parameter 227
- Variadic Parameter (*args และ **kwargs) 227

ตัวแปรแบบ Global และ Local 230
ฟังก์ชันแบบ Lambda 232

บทที่ 11 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 2 237

บทที่ 12 การป้องกันข้อผิดพลาด 259

ลักษณะของ Exception 259
การใช้คำสั่ง try-except และ else 260
การใช้คำสั่ง try ร่วมกับ finally 263

การระบุชนิดของข้อผิดพลาด 265
- การตรวจสอบข้อผิดพลาดชนิดเดียว 267
- การตรวจสอบข้อผิดพลาดหลายชนิด 268

การใช้คำสั่ง raise 270
การใช้คำสั่ง assert 271

บทที่ 13 การอ่านและเขียนไฟล์ 273

การกำหนดตำแหน่งไฟล์ 273
- การสร้างไฟล์ทดสอบ 273
- การระบุตำแหน่งไฟล์ในโค้ด Python 274

การเปิดไฟล์และตรวจสอบ Exception 275
การอ่านไฟล์ 277
การเขียนไฟล์ 280
ไฟล์แบบไบนารี 284
ไฟล์แบบ CSV 286
การจัดการไฟล์และไดเร็กทอรี 288
- การตรวจสอบเกี่ยวกับไฟล์และไดเร็กทอรี 288
- การกระทำกับไฟล์และไดเร็กทอรี 289

บทที่ 14 การสร้างคลาสและโมดูล 291

Class และ Instance 291
Method 293
Attribute และ Initializer 296
Attribute ที่ใช้ร่วมกัน 299
Static Method 300
การสืบทอดระหว่างคลาส 302
การสร้างและใช้งานโมดูล 304
- การสร้างไฟล์ของโมดูล 304
- แนวทางการกำหนดโค้ดของโมดูล 304
- การใช้โมดูล 306

บทที่ 15 การสร้างส่วนติดต่อกับผู้ใช้ (1) 309

พื้นฐานเกี่ยวกับ Tkinter 309
การจัดโครงร่างแบบ place 311
การจัดโครงร่างแบบ pack 313
การจัดโครงร่างแบบ grid 315
การจัดโครงร่างซ้อนกันโดยใช้ Frame 318
การกำหนดสีและฟอนต์ 319
- การกำหนดสี 319
- การกำหนดฟอนต์ 321
- การกำหนดออปชันให้มีผลกับหลายวิดเจ็ต 321

องค์ประกอบพื้นฐานของ Widget 322
- ออปชันพื้นฐานของ Widget 322
- เมธอดพื้นฐานของ Widget 323

Button และออปชัน command 324
การกำหนดออปชัน command ในแบบ Lambda 326
Label และ Message 328
Entry และ ScrolledText 329
- วิดเจ็ต Entry 329
- วิดเจ็ต ScrolledText 333

บทที่ 16 การสร้างส่วนติดต่อกับผู้ใช้ (2) 335

SimpleDialog 335
MessageBox 336
LabelFrame 339
Checkbutton 340
Radiobutton 342
Combobox 343
Listbox 345
PhotoImage และ Canvas 348
Menubutton 350
Notebook 351

บทที่ 17 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 3 355

บทที่ 18 จัดการฐานข้อมูลด้วย Python (1) 391

ฐานข้อมูล SQLite 391
การติดตั้ง DB Browser for SQLite 392
ฐานข้อมูลและตาราง 393
- การสร้างไฟล์ฐานข้อมูล 394
- ข้อกำหนดพื้นฐานของตาราง 394
- การสร้างตาราง 395
- การเพิ่มข้อมูลสำหรับทดสอบ 396
- การทดสอบคำสั่ง SQL ใน SQLite Browser 398
- คำสั่ง SELECT-FROM สำหรับการเลือกข้อมูล 398
- การกำหนดเงื่อนไขด้วย WHERE 399
- การเรียงลำดับด้วย ORDER BY 400

การใช้ Python ร่วมกับ SQLite 401
- การเชื่อมต่อ Python กับไฟล์ฐานข้อมูล 401
- ออบเจ็กต์ Cursor 402

การอ่านข้อมูลจากตาราง 403
- การเขียนโค้ด Python สำหรับอ่านข้อมูล 403
- การใช้เมธอด fetchall() 404
- การใช้เมธอด fetchone() 406
- การใช้เมธอด fetchmany() 406

การแสดงผลแบบตารางด้วยโมดูล tabulate 407
- การใช้โมดูล tabulate ในเบื้องต้น 407
- การแสดงผลลัพธ์จากฐานข้อมูลด้วย tabulate 409

บทที่ 19 จัดการฐานข้อมูลด้วย Python (2) 411

การกำหนดพารามิเตอร์ สำหรับ SQL 411
การเพิ่มข้อมูล 412
- คำสั่ง SQL สำหรับการเพิ่มข้อมูล 413
- การเขียนโค้ด Python สำหรับเพิ่มข้อมูล 415

การแก้ไขและลบข้อมูล 417
- คำสั่ง SQL สำหรับการแก้ไขข้อมูล 417
- คำสั่ง SQL สำหรับการลบข้อมูล 417
- การเขียนโค้ด Python สำหรับแก้ไขและลบข้อมูล 418

การแสดงข้อมูลแบบตารางด้วยวิดเจ็ต Treeview 419
- การใช้งาน Treeview ในเบื้องต้น 419
- การปรับแต่ง Treeview 422
- การแสดงผลลัพธ์จากฐานข้อมูลด้วย Treeview 424

บทที่ 20 รวมตัวอย่างโค้ดเพิ่มเติม ชุดที่ 4 427

PYTHON
if
elif
else:
is
in
while
for
def
return

Python, VS Code และ Jupyter

1

Python เป็นหนึ่งในภาษาคอมพิวเตอร์ที่ได้รับความนิยมสูงสุดในปัจจุบัน ซึ่งนอกจากจะง่ายต่อการเรียนรู้แล้ว ยังสามารถนำไปประยุกต์ใช้งานได้หลากหลายรูปแบบ โดยเนื้อหาในบทนี้จะกล่าวถึงการติดตั้งเครื่องมือสำหรับการเขียนโค้ด พร้อมทั้งการตั้งค่าที่จำเป็นต้องทราบในเบื้องต้น ซึ่งผู้อ่านควรใช้งานให้เกิดความคุ้นเคยในระดับหนึ่งก่อน ทั้งนี้ก็เพื่อประโยชน์ต่อการเรียนรู้ในบทต่อ ๆ ไป

เกี่ยวกับภาษา Python

ภาษาไพธอน (Python Language) สร้างขึ้นเมื่อปี ค.ศ. 1994 โดยโปรแกรมเมอร์ชาวดัตช์ที่ชื่อ Guido van Rossum ด้วยพื้นฐานจากหลายภาษารวมกัน เช่น C, ABC, Modular-3, Algol-68, SmallTalk, Unix Shell เป็นต้น เพื่อให้เป็นภาษาคอมพิวเตอร์ที่เรียนรู้ได้ง่าย ไม่มีกฎเกณฑ์หรือหลักไวยากรณ์ที่ซับซ้อน และสามารถนำไปใช้บนระบบปฏิบัติการที่แตกต่างกันได้ ปัจจุบันไพธอนจัดอยู่ในกลุ่ม Top-5 ของภาษาคอมพิวเตอร์ที่มีผู้ใช้งานมากที่สุด จึงเป็นสิ่งที่ยืนยันถึงความสำเร็จได้เป็นอย่างดี และนอกจากนี้ก็ยังมีลักษณะที่น่าสนใจอื่น ๆ คือ

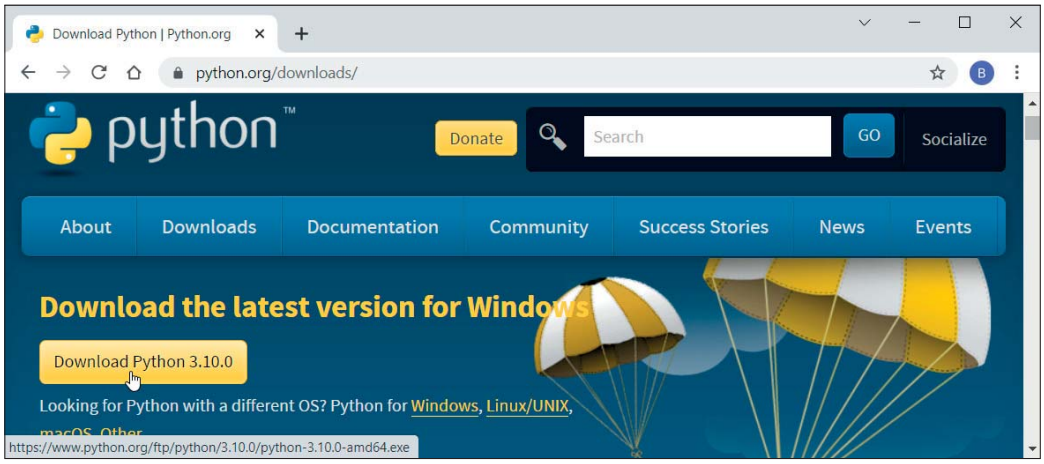
- **Free -** สามารถนำมาใช้งานได้ฟรี โดยไม่มีค่าใช้จ่ายหรือเงื่อนไขใด ๆ
- **Easy to Use & Learn -** มีโครงสร้างทางภาษาที่เรียบง่าย และสามารถเรียนรู้ได้เร็ว จึงเหมาะอย่างยิ่งสำหรับผู้เริ่มต้นศึกษาการเขียนโปรแกรม

- **General Purpose -** ไพธอนเป็นภาษาแบบเอนกประสงค์ที่เราสามารถใช้ในการโปรแกรมได้หลากหลายรูปแบบ นับตั้งแต่แอปพลิเคชันทั่วไป เว็บแอปพลิเคชัน การเขียนโปรแกรมสำหรับเครือข่าย หรือการคำนวณทางวิทยาศาสตร์ เป็นต้น
- **Portable -** สามารถนำโปรแกรมที่เขียนด้วยไพธอนบนระบบปฏิบัติการหนึ่ง ไปใช้บนอีกระบบหนึ่งได้ เช่น เราอาจเขียนโปรแกรมบน Windows แล้วนำไปใช้งานบนเครื่อง Mac หรือ Linux เป็นต้น หรือเรียกว่าการทำงานข้ามแพลตฟอร์ม (Platform) นั่นเอง
- **Open Source -** ไพธอนเป็นภาษาแบบเปิดเผย Source Code ภายใต้ลิขสิทธิ์ของ Python Software Foundation ดังนั้น จึงมีกลุ่มอาสาสมัครเข้าร่วมพัฒนาอย่างมากมาย ช่วยให้ภาษามีวิวัฒนาการอย่างต่อเนื่อง
- **Functional & OOP -** สามารถเลือกเขียนโปรแกรมได้ทั้งแบบ Functional และแบบ Object-Oriented
- **GUI Programming -** ไพธอนสนับสนุนการเขียนโปรแกรมแบบ Graphic User Interface เพื่อติดต่อและโต้ตอบกับผู้ใช้แบบกราฟิก ซึ่งสามารถนำไปใช้งานบนระบบปฏิบัติการที่แตกต่างกันได้
- **Databases -** ไพธอนมีไลบรารีที่สนับสนุนการเชื่อมต่อกับโปรแกรมฐานข้อมูลหลักๆ เกือบทั้งหมด
- **Automatic Memory Management -** ภาษาไพธอนมีกลไกจัดการหน่วยความจำแบบอัตโนมัติ จึงช่วยลดข้อผิดพลาดจากการเกิดปัญหา Memory Leak และส่งผลให้โปรแกรมทำงานได้อย่างคงทน
- **Large Community & Support -** ภาษาไพธอนมีชุมชนหรือกลุ่มผู้ใช้ขนาดใหญ่ ที่พร้อมให้การสนับสนุนและช่วยเหลือเมื่อเราเกิดปัญหา
- **Support Libraries -** นอกจากไลบรารีแบบ Built-in ที่มากับภาษาไพธอนเองแล้ว เราสามารถนำไลบรารีจากภายนอก หรือ Third-Party Libraries เข้ามาเสริมการทำงานได้อีกด้วย จึงเกิดทางเลือกในการพัฒนาแอปพลิเคชันที่หลากหลายยิ่งขึ้น

การติดตั้งภาษา Python

ก่อนที่เราจะสามารถทดสอบการทำงานต่างๆ ได้ จำเป็นต้องติดตั้งตัวประมวลผลของภาษาไพธอนลงไปก่อน โดยมีขั้นตอนที่สำคัญดังนี้

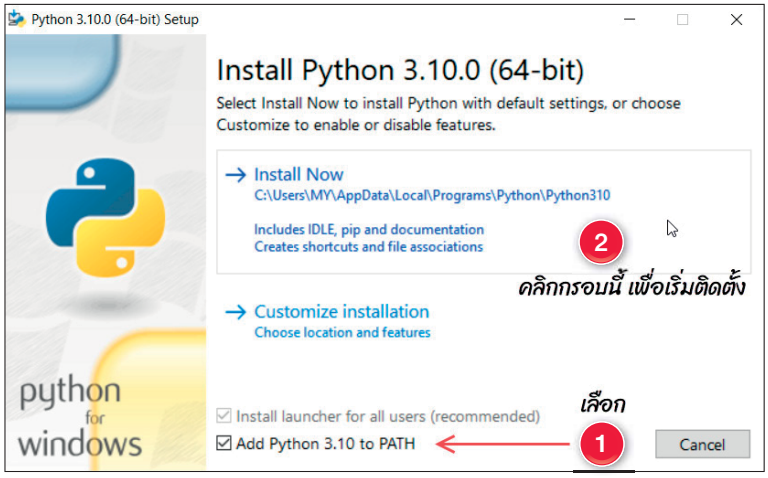
1. เราสามารถดาวน์โหลดชุดติดตั้งได้จากเว็บไซต์ **https://www.python.org/downloads**



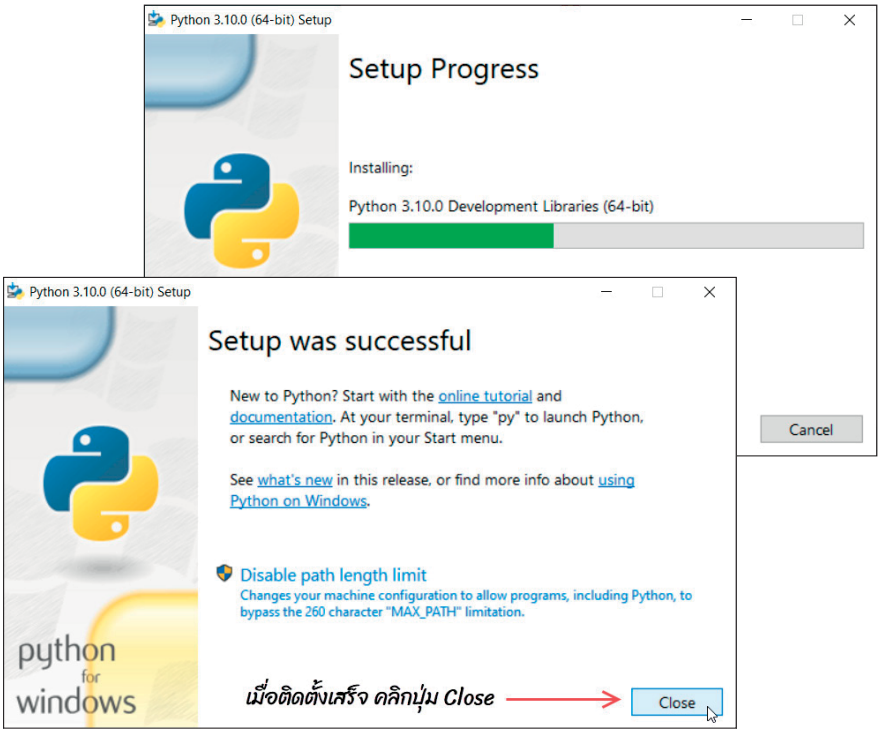
2. หลังจากดาวน์โหลดเสร็จ ก็ดับเบิลคลิกไฟล์ที่ได้มา เมื่อปรากฏหน้าจอที่แสดงฟีเจอร์ (Feature) ในการติดตั้ง สิ่งที่ควรเลือกคือ

   1) Add Python X.XX (หมายเลขเวอร์ชัน) to PATH

   2) คลิกที่กรอบ Install Now เพื่อเลือกติดตั้งองค์ประกอบที่จำเป็นต่อการใช้งานทั่วๆ ไปทั้งหมด



3. หลังจากนั้น จะเข้าสู่ขั้นตอนการติดตั้ง ก็ให้เรารอจนกว่าจะเสร็จสิ้น



หลังจากติดตั้งภาษาไพธอนลงไปแล้ว สิ่งที่เราต้องพิจารณาในลำดับถัดไปคือ การเลือกเครื่องมือในการเขียนโค้ด ซึ่งมีตัวเลือกค่อนข้างมาก และต่างก็มีข้อดีข้อเสียที่แตกต่างกัน สำหรับในหนังสือเล่มนี้ ผู้เขียนจะเลือกใช้ Jupyter บน VS Code เป็นหลัก ดังรายละเอียดที่จะกล่าวถึงในหัวข้อต่อ ๆ ไป

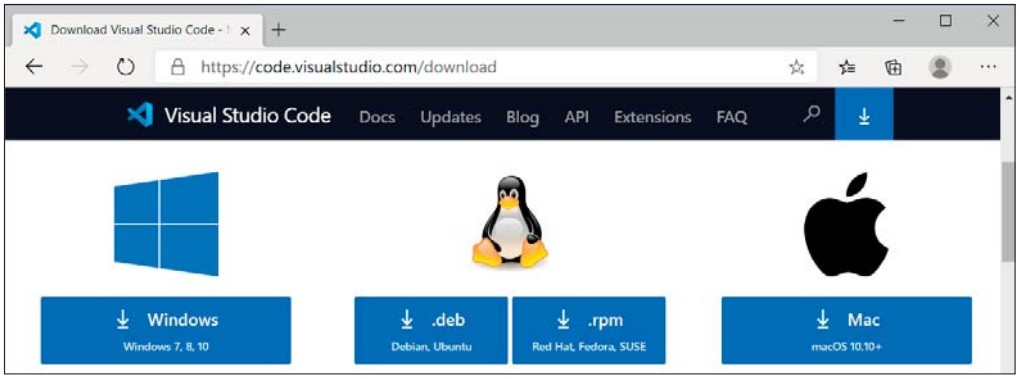
การใช้งาน Visual Studio Code (VS Code)

Visual Studio Code หรือโดยทั่วไปเรามักเรียกสั้น ๆ ว่า VS Code เป็นโปรแกรมประเภท IDE (Integrated Development Environment) ที่ Microsoft สร้างขึ้นมาสำหรับใช้ในการเขียนโค้ดทั่ว ๆ ไป และได้รับความนิยมสูงสุดในปัจจุบัน เนื่องจากมีลักษณะที่โดดเด่นหลายประการ เช่น ให้ใช้งานฟรี รองรับการเขียนโค้ดได้หลายภาษา และที่สำคัญคือ มีส่วนเสริมการทำงานให้เลือกใช้มากมาย แม้ในหนังสือเล่มนี้จะเน้นการใช้ Jupyter เป็นหลัก แต่ก็ต้องทำงานอยู่บนพื้นฐานของ VS Code (ใช้งานร่วมกัน) ดังนั้น จึงจำเป็นต้องรู้จักวิธีการใช้งานบางอย่างเกี่ยวกับ VS Code เอาไว้ล่วงหน้า โดยสิ่งที่เราควรรู้ในเบื้องต้นมีดังนี้

การติดตั้ง VS Code

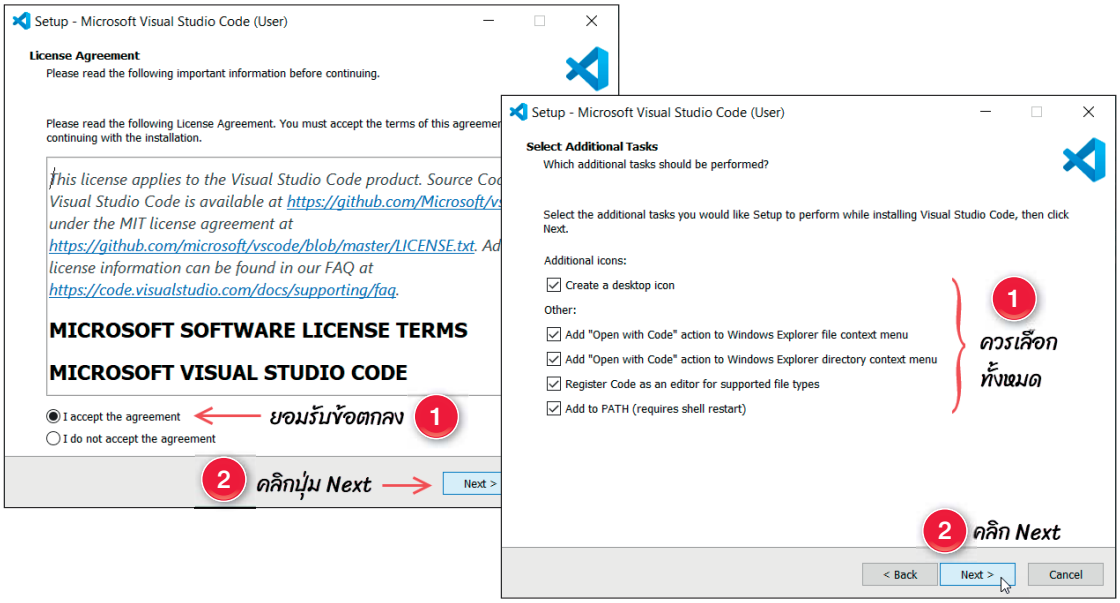
สำหรับการติดตั้ง VS Code ลงในเครื่องที่เราจะใช้งาน มีขั้นตอนโดยสังเขปคือ

1. สามารถดาวน์โหลดชุดติดตั้งได้ที่ **https://code.visualstudio.com/download**



2. หลังจากดาวน์โหลดเสร็จ ให้ดับเบิลคลิกไฟล์ติดตั้งที่ได้มา จากนั้นในหน้าจอถัดไป ก็ยอมรับเงื่อนไขแล้วคลิกปุ่ม Next (ดังภาพถัดไป ด้านซ้ายมือ)

3. ขั้นตอนต่อไป จะมีออปชันให้เลือก ก็แนะนำให้เลือกทั้งหมด แล้วคลิกปุ่ม Next

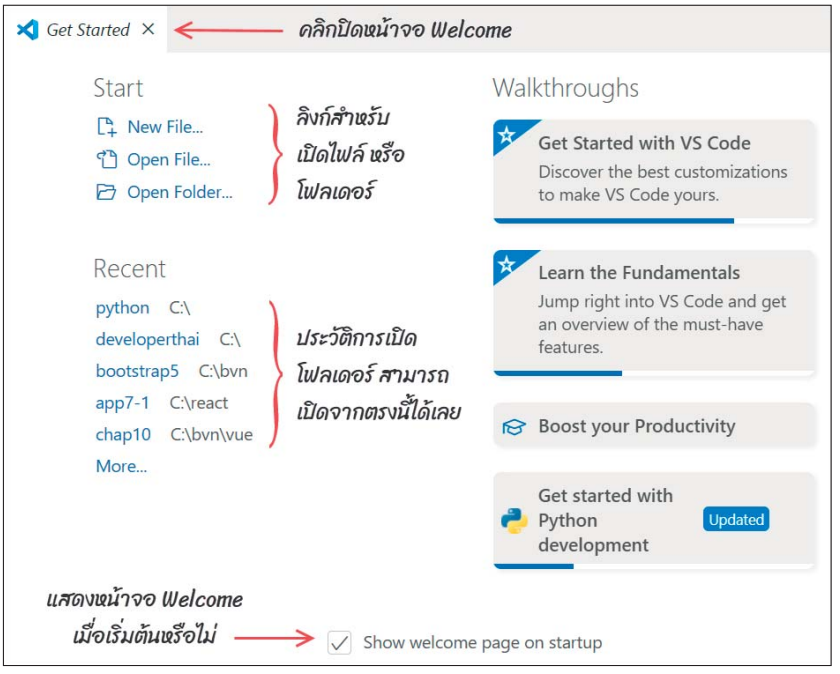


4. ต่อจากนั้น คลิกปุ่ม Install เพื่อเริ่มการติดตั้งไฟล์ จากนั้นก็รอจนกว่าจะแล้วเสร็จ

หน้าจอ Welcome (Get Started)

เมื่อเราเปิดเข้าสู่ VS Code ตามปกติ จะปรากฏหน้าจอ Welcome (หรือ Get Started) เป็นอันดับแรก ซึ่งลักษณะที่สำคัญเกี่ยวกับหน้าจอนี้คือ

- อาจมีการแจ้งข้อมูลข่าวสารที่น่าสนใจเกี่ยวกับ VS Code
- มีลิงก์สำหรับให้เราคลิกเพื่อเปิดไฟล์หรือโฟลเดอร์เป้าหมายที่ต้องการได้อีกหนึ่งวิธี นอกเหนือจากการเลือกที่เมนู
- มีลิงก์ที่แสดงประวัติการเปิดโฟลเดอร์ที่เราเข้าสู่ VS Code ในครั้งก่อน ๆ ซึ่งหากมีโฟลเดอร์เป้าหมายที่เราต้องการรวมอยู่ด้วย ก็สามารถคลิกเปิดจากตรงนี้ได้ทันที



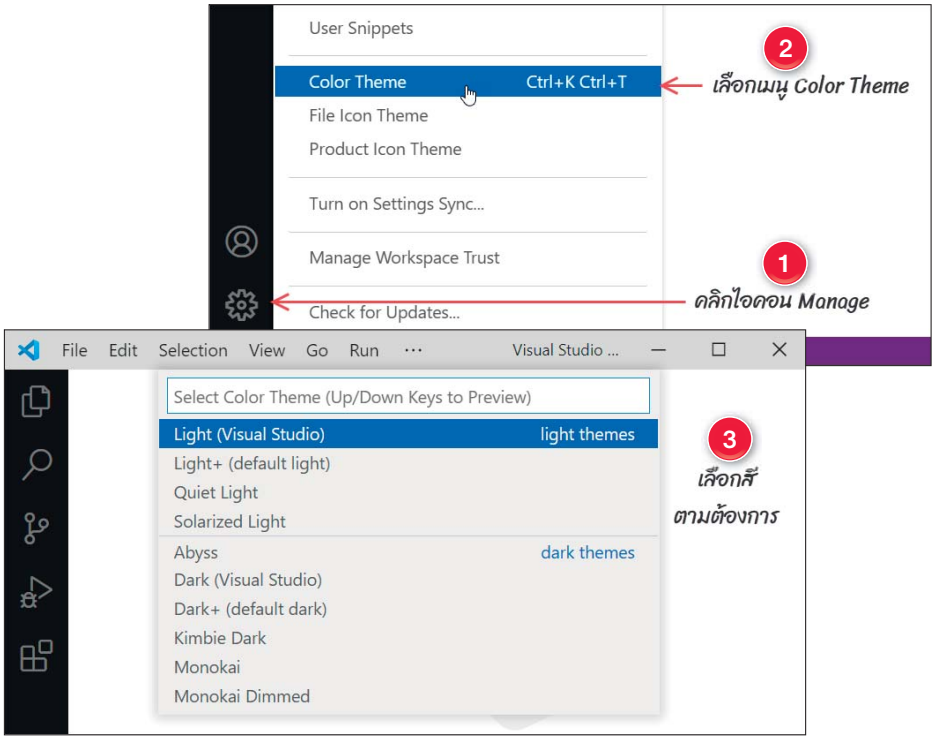
หากเราต้องการปิดหน้าจอนี้ ก็คลิกปุ่ม X ที่แท็บด้านบน หรือหากไม่ต้องการให้ปรากฏหน้า Welcome (Get Started) ต่อไปอีก ก็ให้เราเอาเครื่องหมายถูกตรงตัวเลือก Show welcome page on startup ที่บริเวณขอบด้านล่างออก หรือหลังจากปิดไปแล้ว หากเราต้องการให้หน้าจอ Welcome กลับมาแสดงอีกครั้ง ก็ให้เลือกที่เมนู **Help > Get Started**

การเลือกรูปแบบสี (Theme)

ตามปกติ VS Code จะเลือกธีม (Theme) เป็นสีดำ (Dark) เอาไว้ล่วงหน้า ซึ่งหากเราชอบแบบนี้ ก็ไม่จำเป็นต้องปรับเปลี่ยนก็ได้ แต่ถ้าเราต้องการสีสันในรูปแบบอื่น ก็สามารถเปลี่ยนแปลงได้ ซึ่ง VS Code มีตัวเลือกอยู่จำนวนหนึ่ง โดยใช้ขั้นตอนดังนี้

1. ที่มุมล่างขวาของโปรแกรม VS Code ให้คลิกที่ไอคอน Manage (รูปเฟือง)
2. เลือกที่เมนู Color Theme

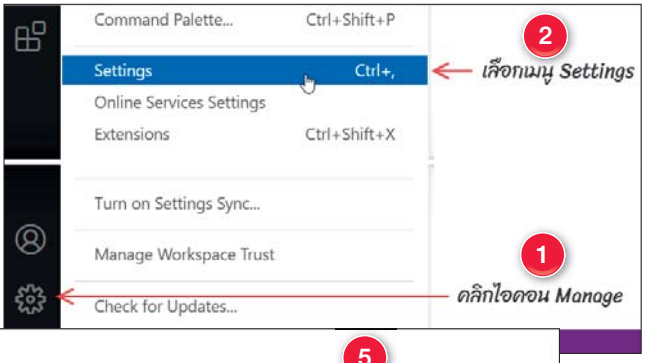
จากนั้นจะปรากฏรายชื่อสีให้เลือก ถ้าเรายังไม่ทราบว่าแต่ละสีมีลักษณะเป็นยังไง ก็อาจลองเลือกดูก่อนก็ได้ ถ้าไม่ถูกใจก็ค่อยเลือกสีใหม่ สำหรับที่ใช้ประกอบในหนังสือเล่มนี้ ผู้เขียนจะเลือกธีมสีแบบ Light (Visual Studio)

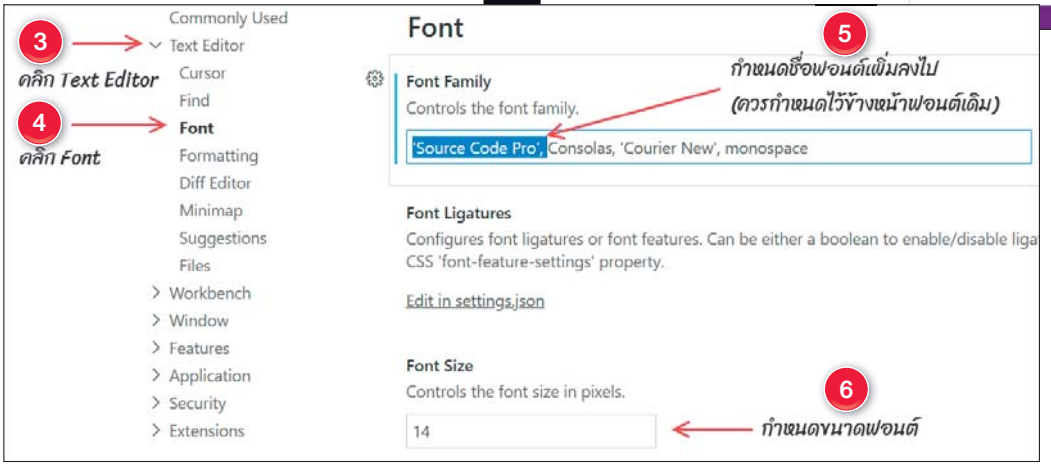


การปรับเปลี่ยนฟอนต์

ฟอนต์ (Font: ในที่นี้คือตัวอักษรที่ใช้เขียนโค้ดและแสดงผลลัพธ์) ที่ VS Code กำหนดเอาไว้ล่วงหน้า แม้จะเป็นรูปแบบมาตรฐานที่นิยมใช้งานกันทั่วไป แต่หากต้องการใช้ฟอนต์ที่เราถูกใจ ก็สามารถเปลี่ยนแปลงได้ด้วยขั้นตอนดังต่อไปนี้

1. ที่มุมล่างขวาของโปรแกรม VS Code ให้คลิกที่ไอคอน Manage (รูปเฟือง)
2. เลือกที่เมนู Settings
3. คลิกแท็บ Text Editor
4. คลิกแท็ก Font
5. กำหนดชื่อฟอนต์ลงในช่อง Font Family ซึ่งตามปกติ จะมีฟอนต์เดิมที่ VS Code กำหนดไว้ให้ล่วงหน้า หากเราต้องการเปลี่ยนไปใช้ชนิดอื่น ควรเพิ่มชื่อฟอนต์ไว้ข้างหน้าฟอนต์เดิม (ชื่อที่อยู่ก่อน จะถูกพิจารณาก่อน และไม่แนะนำให้ลบชื่อฟอนต์เดิมออก) โดยคั่นด้วยเครื่องหมาย , และถ้าชื่อฟอนต์ชนิดนั้นมีช่องว่าง ให้เขียนไว้ในเครื่องหมาย ' ' ซึ่งชื่อฟอนต์ที่ระบุ ต้องถูกติดตั้งเอาไว้ในเครื่องนั้นอยู่ก่อนแล้ว (การติดตั้งฟอนต์ ให้คัดลอกไฟล์ของฟอนต์ไปไว้ที่ C:\Windows\Fonts แล้วเริ่มโปรแกรมที่จะใช้ฟอนต์ใหม่)
6. ถ้าต้องการกำหนดขนาด ก็ระบุตัวเลขลงในช่อง Font Size และเมื่อปิดหน้าจอ Setting ฟอนต์ที่ตั้งค่าเอาไว้ ก็จะมีผลทันที



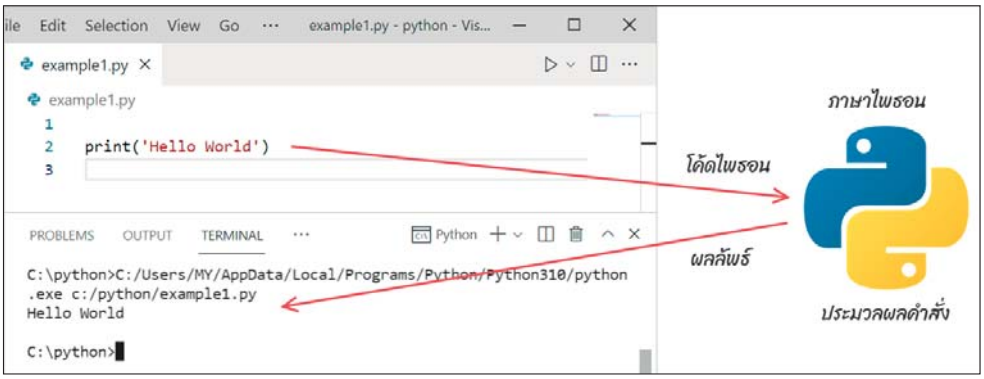


นอกจากนี้ ยังมีเทคนิคเพิ่มเติมอีกอย่างคือ การย่อหรือขยาย (Zoom) ขนาดของฟอนต์ในโปรแกรม VS Code ให้ใหญ่ขึ้นหรือเล็กลง ซึ่งกรณีนี้ จะมีผลกับทั้งโค้ด เมนู และข้อความอื่นๆ ทั้งหมดบน VS Code โดยใช้วิธีการง่ายๆ คือ

- หากต้องการขยายขนาดเพิ่มขึ้น (Zoom In: คล้ายกับการเข้าไปมองใกล้ๆ ซึ่งจะเห็นขนาดที่ใหญ่ขึ้น) ให้กดคีย์บอร์ดปุ่ม <Ctrl => หรือเลือกเมนู **View > Appearance > Zoom In** ซึ่งหากทำซ้ำเช่นนี้อีก ขนาดของฟอนต์จะเพิ่มขึ้น 1 ระดับไปเรื่อยๆ
- หากต้องการลดขนาดให้เล็กลง (Zoom Out: คล้ายกับการออกมามองไกลๆ ซึ่งเราจะเห็นขนาดที่เล็กลง) ให้กดคีย์บอร์ดปุ่ม <Ctrl -> หรือเลือกเมนู **View > Appearance > Zoom Out** ซึ่งหากทำซ้ำเช่นนี้อีก ขนาดของฟอนต์จะลดลง 1 ระดับไปเรื่อยๆ

การเขียนโค้ด Python บน VS Code โดยตรง

แม้ในหนังสือเล่มนี้จะเน้นการเขียนโค้ดบน Jupyter ในรูปแบบส่วนเสริมของ VS Code ดังที่จะกล่าวถึงในหัวข้อถัดไป แต่เราก็ควรมีพื้นฐานการเขียนโค้ดบน Editor ของ VS Code เอาไว้บ้าง ซึ่งอาจต้องนำไปใช้งานในบางโอกาส ดังนั้น จึงจะแนะนำแนวทางไว้พอสังเขป แต่อย่างไรก็ตาม VS Code มีเพียงหน้าที่เกี่ยวกับการเขียนโค้ดเท่านั้น แต่การประมวลผลคำสั่ง (Compile) เป็นหน้าที่ของตัวแปลภาษาไพธอน ด้วยเหตุนี้เราจำเป็นต้องติดตั้งภาษาไพธอนเอาไว้ก่อนแล้ว จึงจะทดสอบการทำงานได้ และหากเราติดตั้งภาษาไพธอนดังที่กล่าวไปในหัวข้อก่อนนี้ ก็สามารถใช้งานได้เลย



จากภาพ เป็นหลักการโดยสังเขปในการทำงานของภาษาไพธอน กล่าวคือ เราจะเขียนโค้ดบน Editor ของ VS Code จากนั้น เมื่อสั่งรัน (Run) โค้ดจะถูกส่งไปประมวลผลยังตัวแปลภาษาไพธอน แล้วผลลัพธ์ที่ได้จะถูกส่งกลับมาแสดงผลที่ส่วน Terminal ของ VS Code

สำหรับขั้นตอนการเขียนโค้ดบน VS Code มีแนวทางโดยสังเขปดังนี้

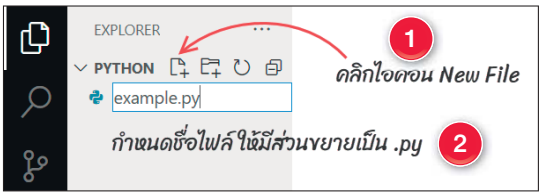
1. ตามหลักการของ VS Code เราควรสร้างโฟลเดอร์สำหรับเก็บไฟล์ไว้ต่างหาก ซึ่งในที่นี้ผู้เขียนจะสร้างโฟลเดอร์ชื่อ python ไว้ที่ตำแหน่ง **c:\python**

2. เมื่อเปิดเข้าสู่ VS Code ให้เปิดโฟลเดอร์ที่ใช้เก็บโค้ดไพธอนที่สร้างเอาไว้ เช่น เลือกที่เมนู File > Open Folder จากนั้น ก็เลือกโฟลเดอร์ดังกล่าว (ในที่นี้คือ c:\python)

3. หลังจากนั้น ให้เราสร้างไฟล์เพิ่มเข้ามาในโฟลเดอร์ ดังนี้

   1) คลิกที่ไอคอน New File

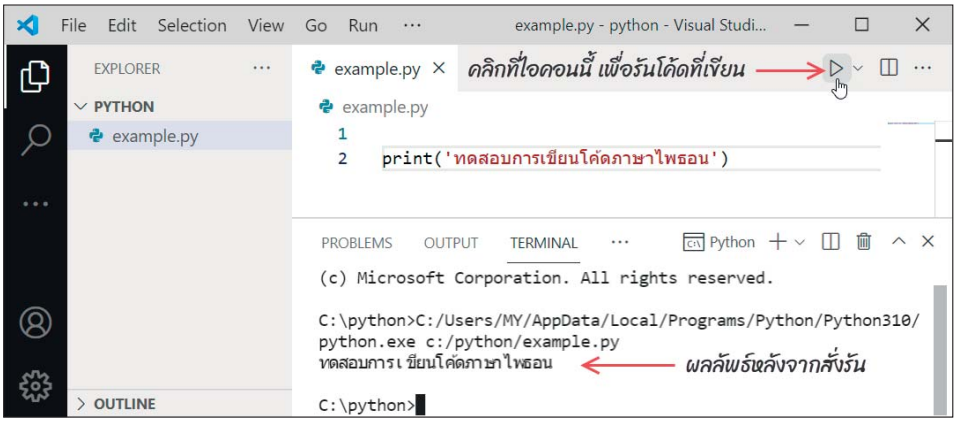
   2) กำหนดชื่อไฟล์ลงไป โดยต้องมีส่วนขยายเป็น **.py** เท่านั้น



4. ต่อไป ก็เขียนโค้ดภาษาไพธอนลงใน Editor ได้เลย แล้วบันทึกการเปลี่ยนแปลง เช่น กด <Ctrl + S> หรือเลือกจากเมนู File > Save

5. หากต้องการรัน ให้คลิกที่ปุ่มไอคอนหัวลูกศรตรงมุมขวาบน ซึ่งไอคอนนี้จะปรากฏเมื่อเราอยู่ในมุมมองการเขียนโค้ดภาษาไพธอนเท่านั้น

6. หลังการสั่งรัน หากโค้ดไม่มีข้อผิดพลาด ก็จะถูกส่งไปประมวลผลยังตัวแปลภาษาไพธอนแล้วผลลัพธ์ที่ได้ก็จะปรากฏที่ Terminal ด้านล่าง ดังหลักการที่กล่าวมาแล้ว



หากเราจะทดสอบในกรณีอื่นๆ ถ้าไม่จำเป็นต้องเก็บโค้ดเดิมเอาไว้ ก็สามารถลบทิ้ง แล้วเขียนโค้ดใหม่ลงไปแทนได้เลย แต่หากต้องการเก็บโค้ดเดิมเอาไว้ เราต้องสร้างไฟล์ใหม่เพิ่มเข้ามาในโฟลเดอร์ด้วยหลักการเดิมทั้งหมด แล้วเมื่อเลิกใช้งาน ก็ควรปิดโฟลเดอร์โดยเลือกที่เมนู File > Close Folder และในการใช้งาน VS Code คราวต่อไป ก็อาจทำแบบใดแบบหนึ่งดังนี้

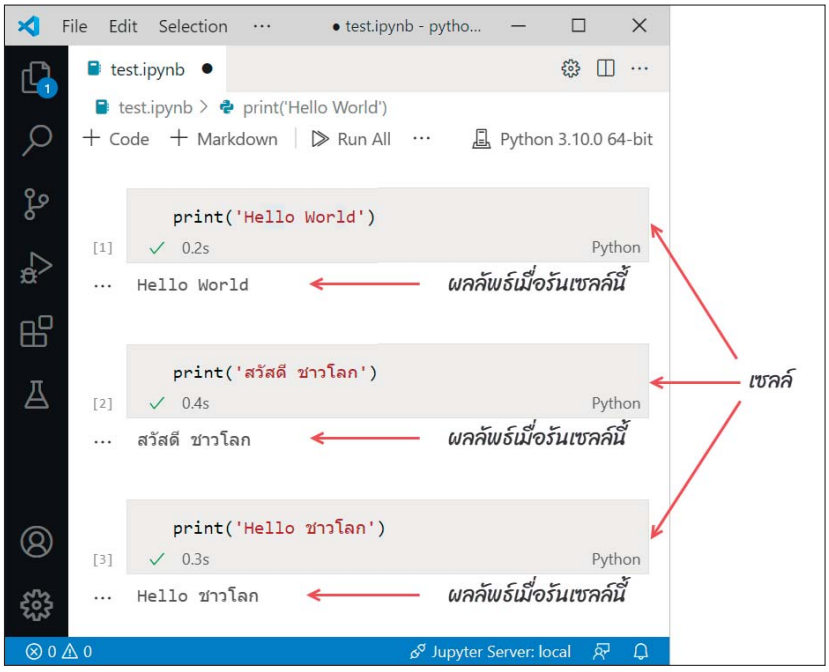
- **วิธีที่ 1** เลือกเปิดโฟลเดอร์ที่เก็บไฟล์ของโค้ดเอาไว้ เช่น เลือกเมนู File > Open Folder จากนั้นเลือกโฟลเดอร์ที่จัดเก็บโค้ดไพธอนเอาไว้
- **วิธีที่ 2** อาจเลือกเปิดไฟล์แบบเจาะจงก็ได้ เช่น เลือกเมนู File > Open File จากนั้นเลือกไฟล์ที่จัดเก็บโค้ดไพธอนเอาไว้ (.py) แล้วเมื่อเลิกใช้งาน ก็ควรปิดไฟล์โดยเลือกที่เมนู File > Close Editor

ในการเปิดใช้งาน VS Code ครั้งต่อๆ ไป ถ้ามีรายชื่อโฟลเดอร์หรือไฟล์เป้าหมาย ปรากฏบนหน้าจอ Welcome (Get Started) ก็สามารถคลิกที่ลิงก์เพื่อเปิดจากตรงนั้นเลยก็ได้

Jupyter สำหรับ VS Code

กรณีที่เราเริ่มต้นศึกษาการเขียนโค้ด อาจมีเนื้อหาปลีกย่อยหลายอย่างที่เราจำเป็นต้องทดสอบอยู่ตลอด ซึ่งหากเราต้องการนำโค้ดกลับมาใช้งานใหม่ ก็ต้องจัดเก็บโค้ดเหล่านั้นแยกไว้คนละไฟล์ ดังนั้น ถ้าเราทดสอบ 100 ตัวอย่าง ก็ต้องสร้าง 100 ไฟล์ เป็นต้น นี่จึงกลายเป็นความยุ่งยากอีกอย่างหนึ่งของการเรียนรู้ในเบื้องต้น

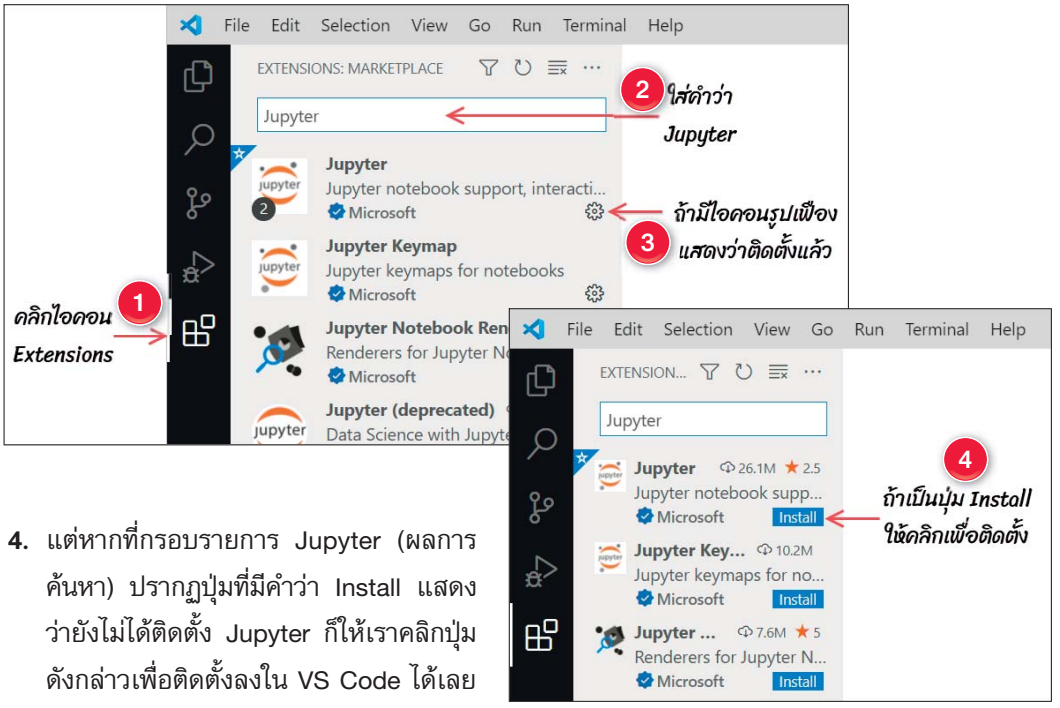
จากปัญหาดังที่กล่าวมา เราอาจเปลี่ยนมาเขียนโค้ดด้วยเครื่องมืออีกแบบหนึ่ง ซึ่งสามารถจัดเก็บโค้ดการทดสอบปลีกย่อยหลายๆ อัน รวมไว้ในไฟล์เดียวกันได้ โดยเราเรียกเครื่องมือนี้ว่า **Jupyter Notebook** (อาจเรียกสั้นๆ ว่า Jupyter) ซึ่งมีลักษณะที่น่าสนใจดังนี้



- Jupyter จะแบ่งพื้นที่สำหรับการเขียนโค้ดออกเป็นช่อง ๆ ซึ่งเรียกว่าเซลล์ (Cell)
- ในไฟล์เดียวกัน สามารถสร้างได้มากกว่า 1 เซลล์
- เราสามารถเขียนโค้ดเพื่อทดสอบในแต่ละเรื่องโดยแยกไว้คนละเซลล์ และสามารถรันเพื่อดูผลลัพธ์เฉพาะโค้ดที่อยู่ในเซลล์ใดเซลล์หนึ่งได้เลย

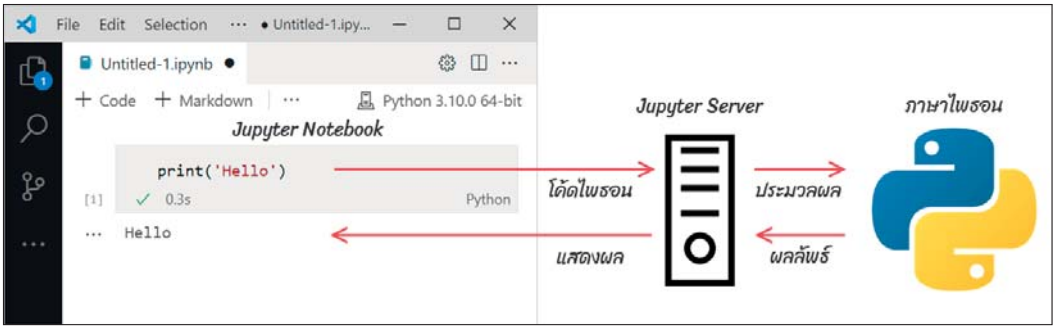
ตามปกติ เราสามารถใช้ Jupyter ได้หลายแนวทาง เช่น ติดตั้งผ่านชุด Anaconda หรือใช้งานผ่าน Google Colab เป็นต้น แต่วิธีเหล่านี้ จะใช้งานในแบบเว็บแอปพลิเคชัน ซึ่งอาจมีขั้นตอนที่ยุ่งยากเล็กน้อยสำหรับผู้เริ่มต้น และนอกจากวิธีการดังกล่าว ก็ยังมีทางเลือกให้เราสามารถนำ Jupyter มาใช้ร่วมกับ VS Code ในรูปแบบของส่วนเสริม (Extension) ได้เช่นกัน โดยเราต้องติดตั้งเพิ่มเติมลง VS Code ดังขั้นตอนต่อไปนี้

1. ที่ VS Code ให้คลิกที่ไอคอน Extension ที่แถบด้านซ้าย
2. พิมพ์คำว่า Jupyter ลงในช่องค้นหา (ขณะนั้นต้องเชื่อมต่ออินเทอร์เน็ต)
3. ตามปกติ ผลการค้นหาจะมีรายการที่ชื่อ Jupyter (ที่ผ่านการตรวจสอบจาก Microsoft แล้ว) และถ้ามีไอคอนรูปเฟืองปรากฏที่กรอบรายการดังกล่าว แสดงว่า Jupyter ถูกติดตั้งเอาไว้แล้ว ซึ่งเราไม่จำเป็นต้องทำอะไรต่อ สามารถคลิกที่ไปไอคอน Extension (เช่นเดียวกับข้อ 1) เพื่อปิดหน้าจอการค้นหาส่วนเสริมได้เลย



4. แต่หากที่กรอบรายการ Jupyter (ผลการค้นหา) ปรากฏปุ่มที่มีคำว่า Install แสดงว่ายังไม่ได้ติดตั้ง Jupyter ก็ให้เราคลิกปุ่มดังกล่าวเพื่อติดตั้งลงใน VS Code ได้เลย

อย่างไรก็ตาม Jupyter ก็เป็นเพียงเครื่องมือในการเขียนโค้ดเช่นเดียวกับ VS Code แต่หน้าที่ในการประมวลผลคำสั่งก็เป็นของตัวแปลภาษาไพธอนอีกเช่นกัน ดังนั้น เราต้องติดตั้งภาษาไพธอนเอาไว้ก่อนแล้วจึงจะรันทดสอบ Jupyter ได้ โดยลักษณะการทำงานร่วมกันระหว่าง VS Code กับ Jupyter และ Python เป็นดังภาพ



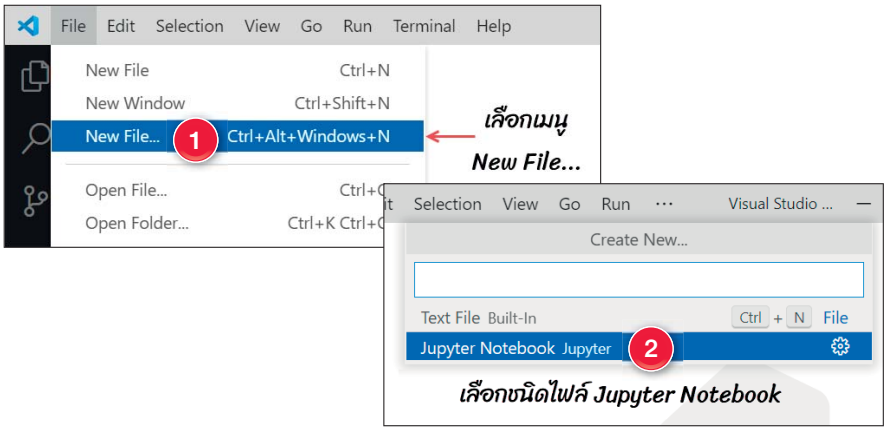
การสร้างไฟล์ของ Jupyter

การใช้งาน Jupyter ในแบบส่วนเสริมของ VS Code นั้น เราต้องสร้างและเปิดไฟล์ที่เป็นรูปแบบของ Jupyter เครื่องมือต่างๆ จึงจะปรากฏให้เห็น โดยไฟล์ของ Jupyter นั้นต้องมีส่วนขยายเป็น **.ipynb** (มาจาก Interactive Python Notebook ซึ่งเป็นชื่อเดิมของ Jupyter) ซึ่งแนวทางการสร้างไฟล์อาจทำแบบใดแบบหนึ่งคือ

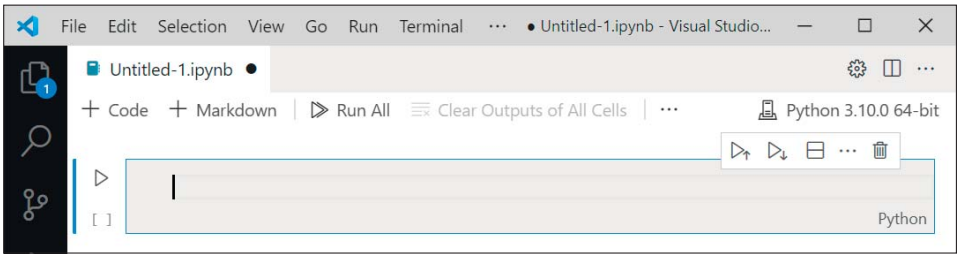
- **วิธีที่ 1** เปิดเข้าสู่ VS Code แล้วทำตามขั้นตอนดังนี้

  1) เลือกเมนู File > New File... (ต้องเป็นเมนู New File อันที่สอง ไม่ใช่อันบนสุด) หรือคลิกที่ลิงก์ New File... บนหน้าจอ Welcome (Get Started) ก็ได้

  2) เลือกชนิดไฟล์เป็น Jupyter Notebook



3) หลังจากนั้น จะเข้าสู่มุมมองของ Jupyter ซึ่งไฟล์จะถูกกำหนดชื่อเป็น Untitled-x.ipynb ไว้ล่วงหน้า



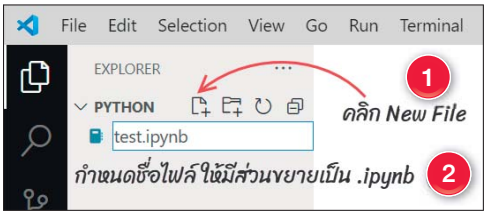
4) หากต้องการบันทึกไฟล์ ให้เลือกเมนู File > Save (หรือ Save As) แล้วกำหนดชื่อใหม่ลงไป โดยส่วนขยายของไฟล์ต้องเป็น .ipynb เท่านั้น เช่น test.ipynb และเลือกจัดเก็บไว้ที่ใดก็ได้ ซึ่งตามการอ้างอิงในหนังสือเล่มนี้ผู้เขียนจะเก็บไว้ที่ C:\python

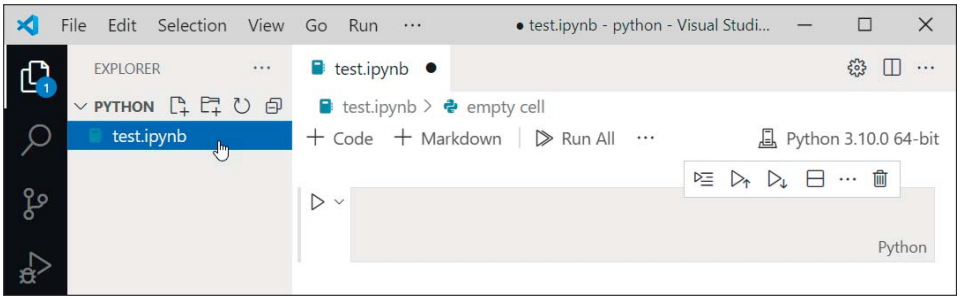
- **วิธีที่ 2** เข้าสู่ VS Code แล้วเปิดโฟลเดอร์ที่จะจัดเก็บไฟล์ Jupyter (ถ้ายังไม่มีโฟลเดอร์ดังกล่าวต้องสร้างขึ้นมาก่อน) เช่นในหนังสือเล่มนี้ ผู้เขียนจะเปิดโฟลเดอร์ที่ตำแหน่ง C:\python หลังจากนั้นให้ทำดังนี้

1) ให้คลิกไอคอน New File

2) กำหนดชื่อไฟล์ตามต้องการ โดยให้มีส่วนขยายเป็น .ipynb โดยไฟล์จะถูกจัดเก็บลงในโฟลเดอร์ที่เรากำลังเปิดในขณะนั้น

3) เมื่อเราคลิกเปิดไฟล์ที่มีส่วนขยายเป็น .ipynb ส่วน Editor ของ VS Code ก็จะแสดงเครื่องมือในมุมมองของ Jupyter โดยอัตโนมัติ





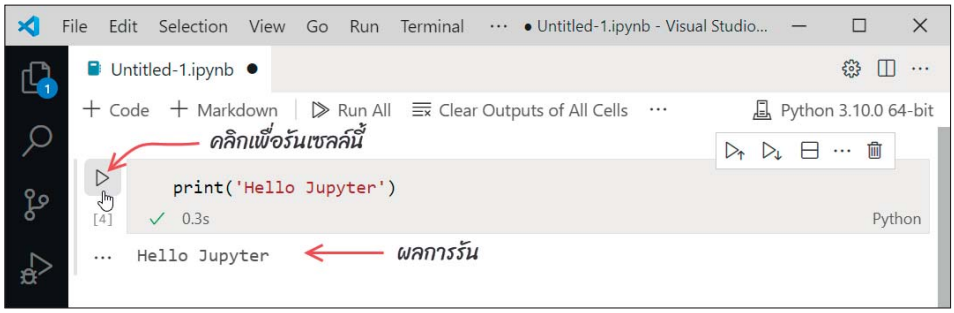
การใช้งานในครั้งต่อไป เราอาจเปิดไฟล์ .ipynb โดยตรง เช่น เลือกเมนู File > Open File หรือจะเปิดโฟลเดอร์ที่จัดเก็บไฟล์เอาไว้ เช่น เลือกเมนู File > Open Folder แล้วค่อยเปิดไฟล์ .ipynb ภายใน VS Code ก็ได้

การใช้เครื่องมือของ Jupyter

เราต้องเปิดไฟล์ ที่มีส่วนขยายเป็น .ipynb เครื่องมือของ Jupyter จึงจะปรากฏให้เห็น ดังที่ได้กล่าวไปในหัวข้อที่ผ่านมา สำหรับแนวทางการใช้เครื่องมือที่สำคัญของ Jupyter มีดังนี้

การเขียนโค้ดและการรัน

เมื่อเราสร้างหรือเปิดไฟล์ของ Jupyter ต่อไปก็สามารถเขียนโค้ดภาษาไพธอนลงในเซลล์ได้ตามปกติ และหากต้องการรัน ให้คลิกไอคอนหัวลูกศรที่ขอบด้านซ้ายของเซลล์นั้น แล้วผลลัพธ์ที่ได้ จะถูกนำมาแสดงที่ด้านล่างของเซลล์

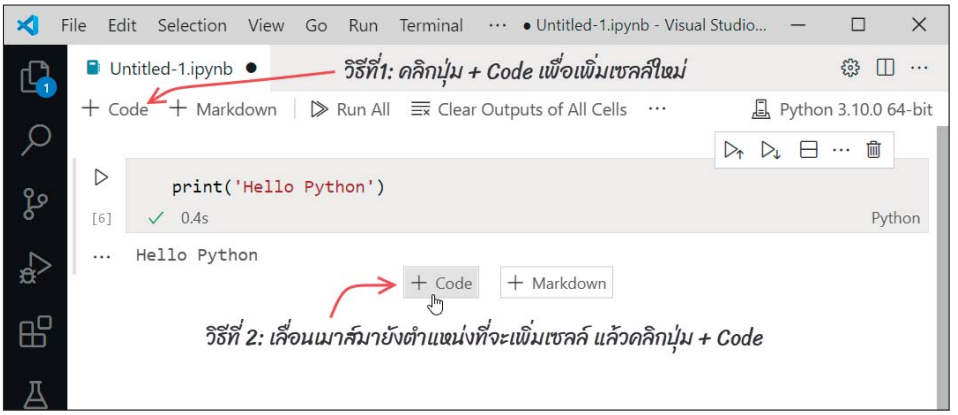


นอกจากนี้ ยังมีทางเลือกอื่นๆ ในการรัน เช่น กดคีย์บอร์ดปุ่ม <Ctrl + Alt + Enter> เพื่อรันเซลล์ที่ถูกเลือกในขณะนั้น (ต้องคลิกที่เซลล์ให้อยู่ในสถานะ Active ซึ่งเส้นขอบจะเป็นสีฟ้า)

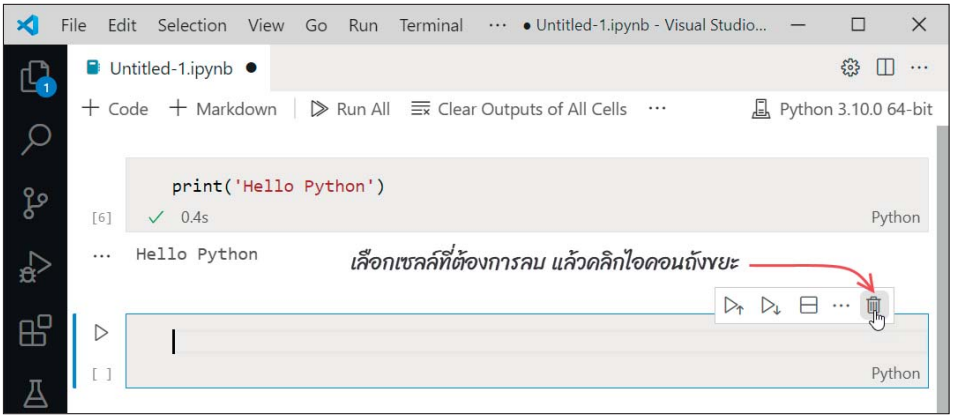
การเพิ่มและลบเซลล์

หากเราต้องการเพิ่มเซลล์ใหม่เพื่อทดสอบโค้ดของเรื่องอื่นๆ ความจริงก็มีหลายทางเลือกที่ทำได้ แต่วิธีที่น่าจะสะดวกที่สุดอาจเลือกอย่างใดอย่างหนึ่งดังนี้

- **วิธีที่ 1** คลิกที่ปุ่ม + Code ที่แถบทูลบาร์ด้านบน (ซ้ายสุด) ซึ่งในกรณีนี้จะเป็นการเพิ่มเซลล์ใหม่ต่อจากเซลล์ที่ถูกเลือกในขณะนั้น หรือถ้าไม่เลือกเซลล์ใดเลยก็จะเพิ่มต่อจากเซลล์แรก
- **วิธีที่ 2** เมื่อต้องการเพิ่มเซลล์ที่ตำแหน่งใด ให้เลื่อนเมาส์มายังบริเวณกึ่งกลางหน้าจอของตำแหน่งนั้น เมื่อปรากฏปุ่ม + Code ก็ให้คลิกปุ่มดังกล่าว ซึ่งเซลล์จะถูกเพิ่มที่นั่น



หากเราต้องการลบเซลล์ใดทิ้งไป ให้เลือก (คลิก) ที่เซลล์นั้นก่อน เพื่อให้อยู่ในสถานะ Active ซึ่งจะปรากฏแถบทูลบาร์ที่มุมขวาบน แล้วคลิกปุ่มไอคอนถังขยะ



การนำโค้ดของหนังสือมาใช้งาน

เราได้ทราบไปแล้วว่า ภายในไฟล์ .ipynb ของ Jupyter อาจมีได้มากกว่า 1 เซลล์ แต่หากในไฟล์เดียวกันมีจำนวนเซลล์มากเกินไป ก็อาจยุ่งยากต่อการเลื่อนค้นหาเมื่อต้องการทดสอบในภายหลัง ดังนั้น ในหนังสือเล่มนี้จึงจะแยกโค้ดของแต่ละบทให้อยู่คนละไฟล์ เช่น

chapter2.ipynb สำหรับโค้ดตัวอย่างทั้งหมดของบทที่ 2
chapter3.ipynb สำหรับโค้ดตัวอย่างทั้งหมดของบทที่ 3
...

ในแต่ละไฟล์นั้น จะประกอบไปด้วยเซลล์ย่อยๆ สำหรับตัวอย่างต่างๆ ที่มีในบทนั้นๆ ซึ่งจะเขียนกำกับเอาไว้ที่บรรทัดแรกของแต่ละเซลล์ และจะตรงกับที่ระบุไว้ในหนังสือ เช่น

สำหรับโค้ดของหนังสือเล่มนี้ ผู้อ่านสามารถดาวน์โหลดได้ที่ **http://www.developerthai.com** ทั้งนี้หากเราจะทดสอบการทำงาน ให้เข้าสู่ VS Code แล้วเปิดโฟลเดอร์ python จากนั้นค่อยคลิกเปิดไฟล์ของบทที่ต้องการ ต่อไปก็สามารถรันทดสอบตามหลักการที่ได้กล่าวมาแล้วทั้งหมด

PYTHON
if
elif
else:
is
in
while
for
def
return

พื้นฐานการเขียนโค้ด Python

2

ภาษาไพธอนมีองค์ประกอบและข้อกำหนดที่สำคัญหลายอย่างซึ่งเราควรรู้จักในเบื้องต้น โดยสิ่งที่จะกล่าวถึงในบทนี้ เช่น การประกาศและกำหนดค่าตัวแปร การแสดงผล ข้อมูลชนิดตัวเลขและสตริง รวมถึงการรับข้อมูลทางคีย์บอร์ด ซึ่งสิ่งเหล่านี้ล้วนเป็นพื้นฐานสำคัญของภาษาไพธอนที่เราต้องใช้งานกันไปตลอด

คีย์เวิร์ดของภาษา Python

คีย์เวิร์ด (Keyword) หรือ Reserved Words คือคำที่ถูกสงวนไว้เป็นคำสั่งสำหรับควบคุมการทำงาน ซึ่งเราไม่สามารถนำคำเหล่านี้ไปตั้งเป็นชื่อตัวแปรหรือฟังก์ชันได้ โดยคีย์เวิร์ดทั้งหมดของไพธอนมีดังนี้

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

คำว่า True, False และ None ต้องเขียนขึ้นต้นด้วยตัวพิมพ์ใหญ่ เพราะไม่ได้ใช้เพื่อเป็นคำสั่ง แต่ใช้แทนค่าของข้อมูล ส่วนคำอื่นๆ ต้องเขียนเป็นตัวพิมพ์เล็กทั้งหมด อย่างไรก็ตาม ในอนาคตอาจมีการเพิ่มคีย์เวิร์ดเข้ามาใหม่ก็เป็นได้ ซึ่งเราสามารถเขียนโค้ดเพื่อแสดงคีย์เวิร์ดทั้งหมดของเวอร์ชันที่ใช้งานอยู่ในขณะนั้น เช่น อาจเขียนคำสั่งลงในเซลล์ของ Jupyter ดังนี้

ตัวอย่าง 2-1 แสดงคีย์เวิร์ดทั้งหมดของภาษาไพธอน

```
import keyword
print(keyword.kwlist)
```

```
#ตัวอย่าง 2-1

import keyword

print(keyword.kwlist)
```

✓ 0.4s Python

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

ถ้าเราไม่แน่ใจว่าคำนั้นเป็นคีย์เวิร์ดของไพธอนหรือไม่ ก็สามารถตรวจสอบด้วยคำสั่งดังนี้

```
keyword.iskeyword("คำที่จะตรวจสอบ")
```

ถ้าเป็นคีย์เวิร์ดจะได้คำว่า True กลับคืนมา แต่ถ้าไม่ใช่จะได้คำว่า False เช่น

```
#ตัวอย่าง 2-2

import keyword

p = keyword.iskeyword("python")
print(p)

a = keyword.iskeyword("and")
print(a)
```

✓ 0.4s Python

```
False
True
```

ตัวอย่าง 2-2 การตรวจสอบว่าเป็นคีย์เวิร์ดของภาษาไพธอนหรือไม่

```
import keyword

p = keyword.iskeyword("python")
print(p)

a = keyword.iskeyword("and")
print(a)
```

ลักษณะของตัวแปร

ตัวแปร (Variable) หมายถึงคำที่เราใช้สำหรับอ้างอิงถึงข้อมูลต่างๆ ทั้งการจัดเก็บและการนำข้อมูลไปใช้งาน จะต้องทำผ่านตัวแปร ซึ่งการตั้งชื่อตัวแปรในไพธอน มีข้อกำหนดดังนี้

- ต้องประกอบด้วยตัว a - z หรือ A - Z หรือ Underscore (_) หรือตัวเลข 0 - 9 เท่านั้น
- ห้ามขึ้นต้น (อักขระตัวแรก) ด้วยตัวเลข แต่ตัวต่อๆ ไปสามารถใช้เป็นตัวเลขได้
- ห้ามมีช่องว่างหรืออักขระอื่นใดนอกเหนือจาก 2 ข้อที่กล่าวมา
- การเขียนด้วยตัวพิมพ์เล็ก-ใหญ่ต่างกัน เช่น abc, ABC, Abc ถือว่าไม่เหมือนกัน
- ต้องไม่ซ้ำกับคีย์เวิร์ดของภาษาไพธอน
- ในภาษาไพธอน เรานิยมตั้งชื่อตัวแปรให้ขึ้นต้นด้วยตัวพิมพ์เล็ก แต่ถ้าเป็นการนำหลายๆ คำมารวมกันเป็นชื่อตัวแปร ก็อาจใช้รูปแบบอย่างใดอย่างหนึ่งคือ
  - คั่นแต่ละคำด้วยเครื่องหมาย Underscore (_) เช่น max_value, is_first_time
  - หรืออาจเขียนติดกัน โดยให้อักขระตัวแรกของแต่ละคำเป็นตัวพิมพ์ใหญ่ ยกเว้นคำแรก เช่น numProducts, isValidInput, numDaysOfMonth
- ตัวอย่าง การตั้งชื่อตัวแปรที่ถูกต้อง
  - x
  - value
  - firstname
  - pricePerUnit
  - num1
  - grand_total
  - freeDiskSpace
  - string2number
  - v3_beta2
  - core_i_7
  - _x x_
  - _x_

- ลักษณะการตั้งชื่อตัวแปรที่ ไม่ถูกต้อง
  - price$ เพราะมีอักขระ $
  - 3x เพราะเริ่มต้นด้วยตัวเลข
  - class เพราะซ้ำกับคีย์เวิร์ด
  - lucky number เพราะมีช่องว่าง

การกำหนดค่าให้กับตัวแปร

ในภาษาไพธอน เราจะประกาศตัวแปรพร้อมกับกำหนดค่าอย่างใดอย่างหนึ่งให้กับมันทันที โดยไม่ต้องระบุชนิดข้อมูล และไม่ใช้คีย์เวิร์ดอื่นใดเพื่อบ่งชี้ว่าเป็นตัวแปรเหมือนที่ต้องทำในภาษาอื่น ๆ ซึ่งรูปแบบโดยทั่วไปของการสร้างและกำหนดค่าตัวแปรในภาษาไพธอนคือ

```
ชื่อตัวแปร = ค่าที่กำหนด
```

สำหรับชื่อตัวแปร ก็ใช้หลักการตามที่ได้กล่าวไว้ในหัวข้อที่แล้ว ส่วนการกำหนดค่าให้กับตัวแปร มีหลักการพื้นฐานทั่วไปคือ

- เราต้องประกาศตัวแปรพร้อมกำหนดค่าอย่างใดอย่างหนึ่งให้มัน ในลักษณะดังนี้

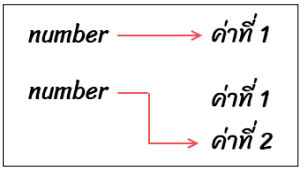
x = ค่าที่กำหนด	กำหนดค่าให้กับมันทันทีที่ประกาศตัวแปร
y = ค่าที่กำหนด	กำหนดค่าให้กับมันทันทีที่ประกาศตัวแปร
a = x	นำค่าของตัวแปร x มากำหนดให้แก่ตัวแปร a (ต้องมี x อยู่ก่อนแล้ว)
b = x + y	ตัวแปร b มีค่าเท่ากับผลบวกของ x กับ y (ต้องมี x และ y อยู่ก่อนแล้ว)

- หากประกาศเพียงชื่อ หรือประกาศแล้วไปกำหนดค่าภายหลัง **จะเกิดข้อผิดพลาด**

x	**ผิด** เพราะกำหนดแค่ชื่อตัวแปร
a a = ค่าที่กำหนด	**ผิด** เพราะตอนแรกระบุแค่ชื่อตัวแปร แม้จะกำหนดค่าในภายหลังก็ตาม

- ตัวแปรที่กำหนดค่าให้กับมันเอาไว้แล้ว สามารถเปลี่ยนแปลงค่าในภายหลังได้ เช่น

```
number = ค่าที่ 1
number = ค่าที่ 2
number = ค่าที่ 3
```



- ตามปกตินั้น ค่าที่กำหนดให้แก่ตัวแปร รวมทั้งเครื่องหมาย = ต้องเขียนในบรรทัดเดียวกับตัวแปร แต่ถ้าจำเป็นต้องเลื่อนค่าที่จะกำหนดให้กับมัน หรือเครื่องหมาย = ไปไว้ที่บรรทัดถัดไป ให้วางเครื่องหมาย Backslash (\) ต่อท้ายตัวแปร หรือต่อท้ายเครื่องหมาย = จากนั้นก็นำส่วนที่เหลือไปเขียนที่บรรทัดถัดไปได้เลย เช่น ลองเปรียบเทียบลักษณะการเขียนโค้ดที่ผิดและถูกต้องดังต่อไปนี้

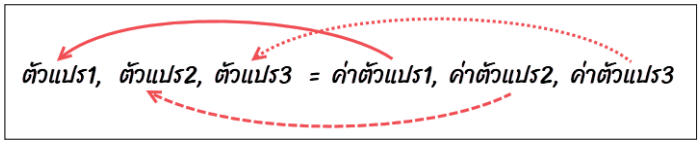
  - ลักษณะการเขียนที่ **ผิด**

```
number =
      ค่าที่กำหนด
```

  - ลักษณะการเขียนที่ **ถูกต้อง**

```
number = \
      ค่าที่กำหนด
```

- นอกจากนี้ เราสามารถกำหนดค่าให้แก่ตัวแปรหลายตัวๆ พร้อมกัน โดยการเขียนคำสั่งในบรรทัดเดียวกันด้วยรูปแบบดังต่อไปนี้




```
ตัวแปร1, ตัวแปร2, ตัวแปร3, ... = ค่าตัวแปร1, ค่าตัวแปร2, ค่าตัวแปร3, ...
```

การกำหนดข้อมูลชนิดตัวเลข

การกำหนดข้อมูลชนิดตัวเลขให้กับตัวแปร มีหลักการขั้นพื้นฐานคือ

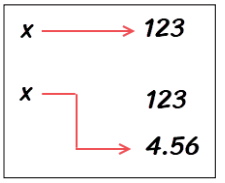
- การกำหนดค่าที่เป็นข้อมูลชนิดตัวเลขไม่ว่าจะเป็นจำนวนเต็มหรือทศนิยมก็ตาม สามารถระบุตัวเลขลงไปได้เลย เพราะตามปกติเราจะเขียนตัวเลขติดกันอยู่แล้ว เช่น

x = 123	จำนวนเต็ม
y = 456.789	เลขทศนิยม
a = -10	จำนวนเต็มติดลบ

b = -20.25	เลขทศนิยมและติดลบ
c, d = 5, 15	c = 5 และ d = 15

- เนื่องจากภาษาไพธอนไม่ยึดติดกับชนิดข้อมูล ดังนั้น เราสามารถเปลี่ยนค่าของตัวแปรจากจำนวนเต็มเป็นทศนิยม หรือจากทศนิยมเป็นจำนวนเต็ม โดยไม่เกิดข้อผิดพลาด เช่น

x = 123 x = 4.56	เดิมเป็นจำนวนเต็ม แล้วเปลี่ยนเป็นเลขทศนิยม
y = 7.11 y = 101	เดิมเป็นเลขทศนิยม แล้วเปลี่ยนเป็นจำนวนเต็ม



- การเขียนตัวเลขนั้น เรา **ไม่** สามารถใช้เครื่องหมาย , เพื่อคั่นหลักพันได้ แต่ในภาษาไพธอน อนุญาตให้เราใส่เครื่องหมาย Underscore (_) แทรกลงไประหว่างตัวเลขเพื่อให้อ่านง่ายขึ้น โดยสามารถนำค่านั้นไปคำนวณได้ตามปกติ ซึ่งข้อกำหนดที่สำคัญคือ
  - สามารถเขียน _ แทรกไว้ระหว่างตัวเลขได้ทั้งส่วนจำนวนเต็มและทศนิยม
  - ไม่จำเป็นต้องใช้คั่นหลักพัน แต่คั่นระหว่างตัวเลขที่หลักใดๆ ก็ได้
  - ห้ามเขียนเครื่องหมาย _ ต่อเนื่องกันตั้งแต่ 2 อันขึ้นไป
  - ห้ามใช้เป็นตัวขึ้นต้นและตัวสุดท้าย ทั้งส่วนจำนวนเต็มและทศนิยม
  - สามารถนำตัวเลขนั้นไปคำนวณได้ตามปกติ
  - ถ้าเราแสดงตัวเลขนั้นออกไป เช่น ใช้ฟังก์ชัน print() จะไม่มีเครื่องหมาย _ ปรากฏให้เห็น

x = 1,234	**ผิด** (ใช้เครื่องหมาย , คั่นหลักพันไม่ได้)
a = 1_234_567 print(a) print(a + 1)	**ถูก** เขียนแบบนี้ได้ 1234567 1234568
b = 1_2_3_4	**ถูก** เขียนแบบนี้ได้ (ไม่จำเป็นต้องใช้คั่นหลักพัน)
c = 1__2___3	**ผิด** (ห้ามเขียน _ ต่อเนื่องกันตั้งแต่ 2 อันขึ้นไป)
d = _1_2_3	**ผิด** (ห้ามใช้ _ เป็นตัวขึ้นต้น)
e = 1_2_3_	**ผิด** (ห้ามใช้ _ เป็นตัวสุดท้าย)

f = 123_456.78_9_0 print(f)	**ถูก** เขียนแบบนี้ได้ 123456.789
g = 12._3_4_5	**ผิด** (เพราะส่วนทศนิยมขึ้นต้นด้วย _)
h = 123.456_	**ผิด** (เพราะส่วนทศนิยมลงท้ายด้วย _)

สำหรับรายละเอียดเพิ่มเติมของข้อมูลชนิดตัวเลข จะกล่าวถึงอีกครั้งในบทที่ 3

การกำหนดข้อมูลชนิดสตริง

สตริง (String) เป็นการนำอักขระแต่ละตัวมาวางเรียงต่อกันให้กลายเป็นข้อความ ซึ่งเราต้องกำหนดจุดเริ่มต้นและสิ้นสุดของมันด้วยเครื่องหมายคำพูด (Quotes) แบบใดแบบหนึ่งคือ

- Single Quotes (' ')
- Double Quotes (" ")
- Single Triple Quotes (''' ''') หรือ Double Triple Quotes (""" """) หรือเรียกรวมๆ ว่า Triple Quotes

```
title = "Python Programming"
birthday = 'Happy Birthday'
christmas = '''Merry Christmas'''
new_year = """สวัสดี ปีใหม่"""
firstname, lastname = 'Tom', "Jerry"
```

สตริงที่กำหนดด้วยเครื่องหมาย " " หรือ ' ' อันเดียวกัน (ไม่รวม Triple Quotes) ต้องเขียนในบรรทัดเดียวกันให้ครบทั้งหมด หากแยกสตริงที่อยู่ในเครื่องหมาย " " หรือ ' ' อันเดียวกัน ไปขึ้นบรรทัดใหม่ จะเกิดข้อผิดพลาด เช่น กรณีต่อไปนี้

a = "Python Programming"	**ผิด** เพราะแยกสตริงที่อยู่ในเครื่องหมาย " " อันเดียวกัน ไปขึ้นบรรทัดใหม่
b = 'ไพธอนไม่ใช่งู แต่เป็นภาษาคอมพิวเตอร์'	**ผิด** เพราะแยกสตริงที่อยู่ในเครื่องหมาย ' ' อันเดียวกัน ไปขึ้นบรรทัดใหม่

อย่างไรก็ตาม หากเป็นสตริงที่ยาวมาก จนไม่สามารถเขียนให้ครบในบรรทัดเดียวกันได้ ก็มีทางเลือกให้เราสามารถแยกสตริงไปเขียนไว้คนละบรรทัดได้ โดยใช้วิธีใดวิธีหนึ่งคือ

- **วิธีที่ 1** หากกำหนดสตริงด้วยเครื่องหมาย " " หรือ ' ' ให้แทรกเครื่องหมาย Backslash (\) ตรงจุดที่จะแยกสตริงไปขึ้นบรรทัดใหม่ เช่น

a = "Python \ Programming"	**ถูก** หลังจากแทรกเครื่องหมาย \ สามารถแยกสตริงอยู่ในเครื่องหมาย " " อันเดียวกัน ไปขึ้นบรรทัดใหม่ได้
b = 'ไพธอนไม่ใช่งู \ แต่เป็นภาษาคอมพิวเตอร์'	**ถูก** เพราะแทรกเครื่องหมาย \ ตรงจุดที่จะขึ้นบรรทัดใหม่แล้ว

- **วิธีที่ 2** หากกำหนดด้วย Triple Quotes ไม่ว่าจะเป็น """ """ หรือ ''' ''' ก็ตาม สามารถแยกสตริงไปขึ้นบรรทัดใหม่ได้เลย โดยไม่จำเป็นต้องแทรกเครื่องหมาย \ เช่น

a = """Python Programming"""	**ถูก** สามารถแยกสตริงอยู่ในเครื่องหมาย """ """ อันเดียวกัน ไปขึ้นบรรทัดใหม่ได้เลย
b = '''ไพธอนไม่ใช่งู แต่เป็นภาษาคอมพิวเตอร์'''	**ถูก** การใช้เครื่องหมาย ''' ''' สามารถแยกสตริงไปขึ้นบรรทัดใหม่ได้เลย

การแทรกเครื่องหมาย \ ในสตริงดังที่กล่าวมา เป็นเพียงการแยกบรรทัดในขั้นตอนการเขียนโค้ดเท่านั้น ซึ่งเมื่อนำสตริงไปใช้งานอื่นต่อไป เครื่องหมาย \ จะไม่รวมอยู่ในสตริง จึงสามารถใช้งานได้เช่นเดียวกับสตริงทั่วไป แต่ก็มีปัญหาอย่างหนึ่งที่เราควรคำนึงถึง นั่นก็คือ หากเราแยกสตริงไปขึ้นบรรทัดใหม่ ไม่ว่าจะเป็นการแทรกเครื่องหมาย \ ไว้ในสตริง หรือการใช้ Triple Quotes ก็ตาม หากเยื้องสตริงเข้าไป เมื่อเราแสดงผล จะเกิดช่องว่างระหว่างสตริงตามระยะที่เยื้องเข้าไป เช่น

ตัวอย่าง 2-3 ระยะเยื้องที่เกิดจากการแทรกเครื่องหมาย \

```
count = \
'One, \
    Two, \
     Three, \
        Four'

print(count)
```

ถ้าเราไม่ต้องการให้มีช่องว่างดังที่กล่าวมา อาจใช้วิธีการแบ่งสตริงออกเป็นส่วนย่อย ๆ ให้อยู่ในเครื่องหมาย " " หรือ ' ' คนละอัน แล้วค่อยนำสตริงเหล่านั้นมาเชื่อมต่อเข้าด้วยกัน ดังที่จะกล่าวถึงในหัวข้อต่อไป

สำหรับการกำหนดสตริงด้วยเครื่องหมาย " " หรือ ' ' นั้น ตามปกติเราจะเลือกใช้อะไรก็ได้ หรือบางครั้งอาจใช้ร่วมกัน เช่น ถ้าในสตริงนั้นมีเครื่องหมาย " เป็นส่วนหนึ่งของมันด้วย เราก็ครอบสตริงทั้งหมด (ชั้นนอก) ด้วย ' ' หรือในทางตรงข้าม ถ้าในสตริงนั้นมีเครื่องหมาย ' เป็นส่วนหนึ่งของมันด้วย เราก็ครอบสตริงทั้งหมดด้วย " " เช่น ลองเปรียบเทียบกรณีต่อไปนี้

msg = 'Let's go'	**ผิด** เพราะใช้เครื่องหมาย ' ซ้อนกัน
msg = "Let's go"	**ถูก**
str = "She said "Hello""	**ผิด** เพราะใช้เครื่องหมาย " ซ้อนกัน
str = 'She said "Hello"'	**ถูก**

อย่างไรก็ตาม สำหรับปัญหาการเขียนเครื่องหมาย " หรือ ' ไว้ในสตริง เราอาจแก้ไขโดยการวางเครื่องหมาย Backslash (\) ไว้หน้า " หรือ ' ที่เป็นส่วนหนึ่งของสตริง ซึ่งวิธีนี้เราสามารถกำหนดสตริงด้วย " " หรือ ' ' เพียงอย่างใดอย่างหนึ่ง โดยไม่ต้องใช้สลับกันก็ได้ เช่น

msg = 'Let\'s go'	หากแสดงผลจะกลายเป็น Let's go
str = "She said \"Hello\""	หากแสดงผลจะกลายเป็น She said "Hello"

สำหรับรายละเอียดเกี่ยวกับข้อมูลชนิดสตริงยังมีอีกมาก ซึ่งเราจะได้เรียนรู้เพิ่มเติมทั้งในบทนี้และบทต่อๆ ไป

การแสดงผลด้วยฟังก์ชัน print()

เมื่อเราต้องการแสดงข้อความออกไปที่หน้าจอ จะกำหนดด้วยฟังก์ชัน print( ) พร้อมระบุข้อมูลที่ต้องการแสดงไว้ในวงเล็บ ซึ่งที่ผ่านมานั้น แม้เราจะใช้ฟังก์ชันนี้มาบ้างแล้ว แต่ยังมีรายละเอียดอีกหลายอย่างที่ควรรู้เพิ่มเติมดังต่อไปนี้

การใช้ฟังก์ชัน print() แบบพื้นฐาน

รูปแบบพื้นฐานที่สุดของฟังก์ชัน print( ) มีลักษณะดังนี้

```
    print(ข้อมูล)
หรือ print(ข้อมูล1, ข้อมูล2, ข้อมูล3, ...)
```

- สำหรับข้อมูลที่จะแสดงผล หากเป็นข้อความ (สตริง) ให้เขียนไว้ในเครื่องหมายคำพูด (Quotes) แบบใดแบบหนึ่งดังที่กล่าวมาแล้ว
- หากเป็นตัวเลข สามารถเขียนลงไปในวงเล็บได้โดยตรง
- หากจะแสดงค่าของตัวแปร ก็ระบุชื่อตัวแปรนั้นลงในวงเล็บได้เลย
- หากเรากำหนดแค่ print() โดยไม่ระบุข้อมูลที่จะแสดง จะเป็นการแทรกบรรทัดว่าง และถ้าต้องการแทรกหลายบรรทัด ก็ระบุ print() ตามจำนวนบรรทัดที่ต้องการ

ตัวอย่าง 2-4 การใช้คำสั่ง print() เพื่อแสดงข้อมูลชนิดต่างๆ

```
print('Hello')
print('สวัสดี')
print(123)
print(456.789)
print()

language = 'Python'
version = 4.0
print(language)
print(version)
```

```
Hello
สวัสดี
123
456.789

Python
4.0
```

- สำหรับรูปแบบ print(ข้อมูล1, ข้อมูล2, ข้อมูล3, ...) เป็นการแสดงข้อมูลมากกว่า 1 อย่างแบบต่อเนื่องในบรรทัดเดียวกัน โดยใช้ฟังก์ชัน print() เพียงครั้งเดียว ซึ่งข้อมูลเหล่านั้นอาจเป็นสตริง ตัวเลข หรือค่าจากตัวแปรก็ได้ และข้อมูลแต่ละค่าจะถูกเว้นช่องว่างให้โดยอัตโนมัติ เช่น

ตัวอย่าง 2-5 การแสดงข้อมูลหลายค่าในคำสั่ง print() เดียวกัน

```
print('Python', 'Programming', 'for', 'Beginner')

a = 'หนึ่ง,'
b = 'สอง,'
c = 'สาม,'
print(a, b, c, 'สี่')

name = 'Guido van Rossum'
year = 1994
print(name,
      'สร้างภาษาไพธอนในปี ค.ศ.',
      year)
```

```
Python Programming for Beginner
หนึ่ง, สอง, สาม, สี่
Guido van Rossum สร้างภาษาไพธอนในปี ค.ศ. 1994
```

การใช้ฟังก์ชัน print() ที่ซับซ้อนขึ้น

เราสามารถกำหนดลักษณะการแสดงผลเพิ่มเติมให้กับฟังก์ชัน print() โดยใช้รูปแบบอย่างใดอย่างหนึ่งดังนี้

```
print(ข้อมูล1, ข้อมูล2, ข้อมูล3, ..., sep=...)
print(ข้อมูล1, ข้อมูล2, ข้อมูล3, ..., end=...)
print(ข้อมูล1, ข้อมูล2, ข้อมูล3, ..., sep=..., end=...)
```

- สำหรับข้อมูล 1, 2, 3, ... ก็เหมือนกับรูปแบบในหัวข้อที่แล้ว คือจะมีจำนวนเท่าไหร่ก็ได้
- sep และ end เป็นการระบุออปชันเพิ่มเติมให้กับฟังก์ชัน โดยลักษณะเช่นนี้ในภาษาไพธอนจะเรียกว่า Keyword Argument (หรือเรียกสั้นๆ ว่า คีย์เวิร์ด) แต่เป็นคนละกรณีกับ Keyword หรือ Reserved Word ที่เป็นคำสงวนของไพธอน ซึ่งเราจะได้ศึกษารายละเอียดเรื่องนี้ในบทที่ 10
- **sep** (มาจาก separator) คือสิ่งที่จะใช้คั่นระหว่างข้อมูลแต่ละอัน โดยค่าที่กำหนดให้แก่ sep ต้องเป็นสตริงเท่านั้น (อยู่ในเครื่องหมายคำพูดแบบใดแบบหนึ่ง) ซึ่งตามปกติ (ไม่ระบุ sep) จะเป็นช่องว่าง 1 ช่อง
- **end** คือสิ่งที่จะเขียนต่อท้ายสตริง และต้องกำหนดในแบบสตริงเช่นเดียวกับ sep โดยตามปกติ (ถ้าไม่ระบุ end) จะเป็น '\n' ซึ่งเป็นสัญลักษณ์สำหรับการขึ้นบรรทัดใหม่ (รายละเอียดอยู่ในหัวข้อถัดไป) แต่ถ้าต้องการให้ฟังก์ชัน print() ถัดไป แสดงผลในบรรทัดเดียวกับฟังก์ชัน print() ปัจจุบัน ให้กำหนดค่าเป็น end=''
- อาจกำหนด sep และ end เพียงอย่างใดอย่างหนึ่ง หรือกำหนดทั้งคู่ หรือไม่มีเลยก็ได้

ตัวอย่าง 2-6 การใช้ออปชัน sep และ end ของคำสั่ง print()

```
a = 'One'
b = 'Two'
c = 'Three'
d = 'Four'
print(a, b, c, d, sep=',')

a = 'www'
b = 'developerthai'
c = 'com'
print(a, b, c, sep='.')
```

```
d = 1
m = 'กรกฎาคม'
y = 2565
print('ออกเดินทางในวันที่: ', end='')
print(d, m, y, sep=' ')

h = 12
m = 34
print('เวลา: ', end='')
print(h, m, sep=':')
```

```
One, Two, Three, Four
www.developerthai.com
ออกเดินทางในวันที่: 1 กรกฎาคม 2565
เวลา: 12:34
```

ลักษณะพื้นฐานของสตริงที่ควรรู้เพิ่มเติม

เนื่องจากสตริง เป็นชนิดข้อมูลพื้นฐานสำคัญที่เราต้องนำไปใช้กันอยู่ตลอด ดังนั้นจึงมีรายละเอียดปลีกย่อยค่อนข้างมาก โดยในหัวข้อนี้ จะกล่าวถึงลักษณะพื้นฐานบางอย่างเกี่ยวกับสตริงที่เราควรรู้จักเพิ่มเติม ซึ่งต้องใช้ร่วมกับเนื้อหาของเรื่องต่อ ๆ ไป ส่วนรูปแบบการใช้งานอื่น ๆ จะกล่าวถึงอีกครั้งในบทที่ 8

การเชื่อมต่อสตริง

การเชื่อมต่อสตริง (String Concatenation) เป็นการนำสตริงย่อย ๆ มารวมเข้าด้วยกัน โดยใช้เครื่องหมาย + เพื่อให้กลายเป็นสตริงอันเดียวกัน รวมถึงการนำค่าจากตัวแปรมารวมกับสตริงอีกด้วย เช่น

ตัวอย่าง 2-7 การเชื่อมต่อสตริงด้วยเครื่องหมาย +

```
str1 = 'ขอเชิญหมายเลข ' + ' สามสิบ ' + ' ที่ช่องบริการ' + ' ห้า ' + 'ค่ะ'
print(str1)

name = 'Guido van Rossum'
str2 = 'ผู้สร้างภาษาไพธอนคือ ' + name + ' เมื่อปี 1994'
print(str2)

name = "Malee"
country = "Thailand"
print("My name is " + name +
      " from " + country)
```

```
ขอเชิญหมายเลข สามสิบ ที่ช่องบริการ ห้า ค่ะ
ผู้สร้างภาษาไพธอนคือ Guido van Rossum เมื่อปี 1994
My name is Malee from Thailand
```

อย่างไรก็ตาม ในภาษาไพธอนนั้น ไม่สามารถใช้เครื่องหมาย + เพื่อเชื่อมต่อระหว่างสตริงกับตัวเลขโดยตรงได้ เช่น กรณีต่อไปนี้ **จะเกิดข้อผิดพลาดทั้งหมด**

```
v = 4.0
s = 'Python' + v

str = 'ขอเชิญหมายเลข  ' + 30 + ' ที่ช่องบริการ ' + 5 + ' ค่ะ'
```

วิธีแก้ไขคือ ต้องเขียนตัวเลขในแบบสตริงไว้ในเครื่องหมายคำพูดแบบใดแบบหนึ่ง หรืออาจใช้ฟังก์ชัน str() เพื่อแปลงตัวเลขให้เป็นสตริง จากนั้นค่อยนำมารวมกัน เช่น

ตัวอย่าง 2-8 การเชื่อมต่อสตริงกับตัวเลข

```
v = '4.0'
s = 'Python ' + v
print(s)

msg = "ขอเชิญหมายเลข " + str(30) + " ที่ช่องบริการ " + str(5) + " ค่ะ"
print(msg)

print('ระยะทาง ' + str(1) + ' ไมล์เท่ากับ '+ str(1.6) + ' กิโลเมตร')
```

```
Python 4.0
ขอเชิญหมายเลข 30 ที่ช่องบริการ 5 ค่ะ
ระยะทาง 1 ไมล์เท่ากับ 1.6 กิโลเมตร
```

และจากลักษณะที่เคยกล่าวถึงมาแล้วคือ สตริงที่อยู่ในเครื่องหมาย " " หรือ ' ' อันเดียวกัน ต้องเขียนไว้ในบรรทัดเดียวกันเสมอ แต่ในกรณีที่สตริงนั้นยาวมาก ซึ่งไม่สามารถเขียนไว้ในบรรทัดเดียวกันได้ทั้งหมด เราก็อาจแบ่งเป็นสตริงย่อยๆ แล้วนำมารวมกันด้วยเครื่องหมาย + ได้อีกวิธีหนึ่ง แต่ต้องวางเครื่องหมาย \ ไว้หลังเครื่องหมายบวกตามเดิม เช่น

```
long_string = 'ในกรณีที่สตริงนั้นยาวมาก และไม่สามารถเขียน' + \
              'ไว้ในบรรทัดเดียวกันได้ทั้งหมด เราก็อาจแบ่งเป็นสตริงย่อยๆ ' + \
              'ให้อยู่คนละบรรทัด แล้วนำมารวมกันด้วยเครื่องหมายบวก'
```

การเขียนอักขระพิเศษบางตัวในสตริง

เนื่องจากในภาษาไพธอน เราต้องกำหนดสตริงด้วยเครื่องหมาย " หรือไม่ก็ ' ซึ่งหากจะเขียนอักขระบางตัวลงในสตริง ต้องใส่เครื่องหมาย \ นำหน้า และนอกจากนี้ถ้าเรานำเครื่องหมาย \ ไปวางไว้หน้าอักขระบางตัวก็จะมีผลต่อการแสดงสตริงนั้นด้วย โดยอักขระพิเศษเกี่ยวกับสตริงที่เราควรรู้จักในเบื้องต้นมีดังนี้