PYTHON
For Data Analytics

ไพทอนสำหรับการวิเคราะห์ข้อมูล

รศ. ดร. ประเสริฐ คณาวัฒนไชย
อ. นิภาศรี วงศ์สิริเดช

ไพทอนสำหรับการวิเคราะห์ข้อมูล

Python for Data Analytics

ผู้แต่ง ประเสริฐ คณาวัฒนไชย และ นิภาศรี วงศ์สิริเดช

พิมพ์ครั้งที่ 1

ISBN e-book 978-616-631-460-1

ราคา 300 บาท

จัดทำโดย ประเสริฐ คณาวัฒนไชย และ นิภาศรี วงศ์สิริเดช

สั่งซื้อได้ที่ ศูนย์หนังสือจุฬาลงกรณ์มหาวิทยาลัย

ถนนพญาไท เขตปทุมวัน กรุงเทพฯ 10330

โทร. 0-2218-9872 โทรสาร 0-2254-9495

Call Center โทร. 0-2255-4433

http://www.chulabook.com

ดาวน์โหลดข้อมูลที่ใช้ในตำราได้ที่: https://github.com/prasertcbs/pyda

ข้อคิดเห็นหรือข้อเสนอแนะ สามารถส่งมาได้ที่อีเมล prasert.k@chula.ac.th

คำนำ

ในยุคที่ข้อมูลกลายเป็นทรัพยากรสำคัญที่ขับเคลื่อนการตัดสินใจในทุกวงการ การตัดสินใจที่ดีไม่อาจอาศัยเพียงสัญชาตญาณหรือประสบการณ์อีกต่อไป แต่ต้องอาศัย “พลังของข้อมูล” เพื่อมองเห็นภาพรวม เข้าใจแนวโน้ม และคาดการณ์อนาคต ความสามารถในการวิเคราะห์และตีความข้อมูลได้อย่างมีประสิทธิภาพจึงกลายเป็นทักษะสำคัญของคนทำงานยุคดิจิทัล ไม่ว่าจะอยู่ในสายอาชีพใดก็ตาม

ตำราเล่มนี้เหมาะสำหรับ

- ผู้เริ่มต้นที่อยากเข้าใจการวิเคราะห์ข้อมูลด้วยเครื่องมือสมัยใหม่
- ผู้เริ่มต้นที่ต้องการเรียนรู้การวิเคราะห์ข้อมูลด้วย Python และไลบรารี Pandas ตั้งแต่ขั้นพื้นฐาน
- นักเรียน นักศึกษา หรือผู้สนใจที่ต้องการเรียนรู้ทักษะด้านการวิเคราะห์ข้อมูล
- ผู้มีประสบการณ์ด้านอื่นที่ต้องการต่อยอดทักษะด้วยเครื่องมือวิเคราะห์ข้อมูลสมัยใหม่

ผู้อ่านจะได้เรียนรู้แบบ “ค่อยเป็นค่อยไป” ตั้งแต่พื้นฐานของภาษา Python จนถึงการใช้ Pandas เพื่อจัดการ แปลง และวิเคราะห์ข้อมูลทางธุรกิจจริง ตั้งแต่ขั้นตอนการนำเข้าข้อมูล การทำความสะอาด การสำรวจและวิเคราะห์ข้อมูลในหลากหลายแง่มุม

ผู้เรียนจะได้ฝึกฝนผ่านกรณีศึกษาจากธุรกิจร้านจำหน่ายเครื่องดื่มและเบเกอรี ซึ่งสะท้อนสถานการณ์ทางธุรกิจจริงที่พบได้บ่อย พร้อมโค้ดตัวอย่างที่สามารถทดลอง ปรับแก้ และนำไปประยุกต์ใช้กับข้อมูลของตนเองได้ทันที

ผู้เขียนหวังว่าทุกบทเรียนในตำราเล่มนี้จะช่วยให้ท่านมองเห็นคุณค่าของข้อมูลที่ซ่อนอยู่ในธุรกิจของท่าน และเป็นก้าวแรกที่มั่นคงบนเส้นทางสู่การเป็นนักวิเคราะห์ข้อมูลที่มีความสามารถในยุคดิจิทัล

ขอขอบคุณทุกท่านที่ให้ความสนใจและเลือกใช้ตำราเล่มนี้เป็นส่วนหนึ่งในการพัฒนาตนเอง

รศ. ดร. ประเสริฐ คณาวัฒนไชย

อ. นิภาศรี วงศ์สิริเดช

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย

มีนาคม 2569

สารบัญย่อ

สารบัญ

1. รู้จักภาษา Python

วัตถุประสงค์การเรียนรู้

1. รู้จักภาษา Python
2. ประโยชน์ของภาษา Python
3. เข้าใจการทำงานเบื้องต้นของภาษา Python

ภาษา Python

ภาษา Python เป็นภาษาโปรแกรมที่สร้างขึ้นโดย Guido van Rossum ในช่วงปลายทศวรรษ 1980 ขณะที่เขาทำงานที่สถาบันวิจัย Centrum Wiskunde & Informatica (CWI) ในประเทศเนเธอร์แลนด์ และได้เปิดตัวต่อสาธารณชนครั้งแรกในปี 1991 ในปัจจุบัน (ปี 2026) ภาษา Python ได้พัฒนามาถึงเวอร์ชัน 3.14 ซึ่งทำงานได้บนระบบปฏิบัติการต่าง ๆ ไม่ว่าจะเป็น Windows, macOS และ Linux

มาดูเหตุผลหลัก ๆ ที่ทำให้ Python ได้รับความนิยมอย่างสูง ซึ่งสามารถสรุปได้ดังนี้

1. ง่ายต่อการเรียนรู้และใช้งาน เนื่องจากมีไวยากรณ์ที่เรียบง่ายทำให้อ่านและเขียนโค้ดได้ง่ายแม้ไม่มีพื้นฐานมาก่อน
2. ใช้งานได้หลากหลายด้าน ไม่ว่าจะเป็นทางด้านวิทยาศาสตร์ข้อมูล (Data Science) พัฒนาเว็บไซต์ และ AI
3. เป็น Open Source และ Cross-Platform และมีชุมชนนักพัฒนาจากทั่วโลกช่วยพัฒนาความสามารถของภาษา รวมถึงไลบรารีต่าง ๆ ไม่ว่าจะเป็นด้าน Machine Learning/AI การสร้างเว็บไซต์ และอื่น ๆ ซึ่งจะช่วยลดระยะเวลาในการพัฒนาแอปพลิเคชัน
4. ความต้องการในตลาดแรงงานสูง

ตัวอย่างการนำไปใช้จริง

- Spotify: ใช้ Python วิเคราะห์พฤติกรรมผู้ใช้และแนะนำเพลง
- NASA: ใช้คำนวณและประมวลผลข้อมูลทางวิทยาศาสตร์
- Netflix: ใช้จัดการระบบ Recommendation และ Automation

> ⓘ การติดตั้ง Python บน Windows
>
> การติดตั้ง Python ทำได้หลายวิธี โดยส่วนตัวของผู้เขียนชอบวิธีการติดตั้งผ่าน miniconda โดยดาวน์โหลดได้ที่ https://docs.conda.io/en/main/miniconda.html
>
> 📺 การติดตั้ง Python บน Windows

โปรแกรมแรก

หลังจากที่ทำการติดตั้ง Python แล้ว เราสามารถเรียกใช้งาน Python ผ่าน Command Prompt หรือ PowerShell ในระบบปฏิบัติการ Windows ส่วนการทำงานบนระบบปฏิบัติการ macOS จะใช้ผ่าน Terminal

โปรแกรม Text Editor ที่ได้รับความนิยมสูงสุดในการเขียนโค้ด คือ Visual Studio Code ซึ่งสามารถใช้ในการเขียน ทดสอบการทำงาน รวมถึงหาและแก้ไขข้อผิดพลาดของโปรแกรมที่พัฒนาโดยใช้ภาษา Python ได้อย่างมีประสิทธิภาพ

โดยปกติเราจะเก็บโค้ดไว้ในไฟล์ที่มีนามสกุล .py เช่น promotion.py การเก็บโค้ดไว้ในไฟล์จะช่วยให้เราสามารถเก็บรวบรวมโค้ดเพื่อนำไปใช้ซ้ำและแชร์กับผู้อื่นได้อย่างสะดวกยิ่งขึ้น นอกจากนี้ยังช่วยให้การพัฒนาโปรแกรมง่ายขึ้น เนื่องจากเราสามารถแยกโค้ดเป็นไฟล์ย่อย ๆ ได้ตามความเหมาะสม เพื่อให้โค้ดมีความชัดเจนและง่ายต่อการบริหารจัดการ รวมถึงการแก้ไขเพิ่มเติมความสามารถในการทำงานในภายหลัง

> ⓘ IDE คืออะไร
>
> **IDE (Integrated Development Environment)** คือโปรแกรมที่มีความสามารถในการสนับสนุนและช่วยในกระบวนการพัฒนาซอฟต์แวร์หรือแอปพลิเคชัน โดยเฉพาะอย่างยิ่งในการเขียนและแก้ไขโค้ด มักมีคุณสมบัติที่อำนวยความสะดวกในการเขียนโค้ด เช่น code completion การจัดรูปแบบโค้ด (code formatting) การตรวจจับข้อผิดพลาด (error detection) การทำ version control และอื่น ๆ
>
> Visual Studio Code หรือเรียกย่อ ๆ ว่า VS Code เป็นตัว IDE ที่นิยมใช้กันอย่างแพร่หลายในการเขียนโค้ดของ Python เนื่องจากเป็น IDE ที่ให้ติดตั้งและใช้งานได้ฟรี ทำงานได้กับระบบปฏิบัติการหลัก ๆ ไม่ว่าจะเป็น Windows, macOS และ Linux รวมถึงมีชุมชนที่ช่วยเขียนส่วนต่อขยายความสามารถ (extensions) ให้กับ VS Code เป็นจำนวนมาก ทำให้การเขียนโค้ดโดยใช้ VS Code เป็นไปอย่างมีประสิทธิภาพและรวดเร็วยิ่งขึ้น
>
> ดาวน์โหลดและติดตั้ง Visual Studio Code ได้ที่ https://code.visualstudio.com/download
>
> 📺 การติดตั้ง Visual Studio Code บน Windows

ตัวอย่างของโค้ดเบื้องต้นเพื่อแสดงข้อความที่ต้องการออกทางหน้าจอ พร้อมผลลัพธ์ที่ได้จากการทำงาน

```
print("Hello, World!")
print("ใคร ๆ ก็เขียนโปรแกรมได้")
```

```
Hello, World!
ใคร ๆ ก็เขียนโปรแกรมได้
```

ลองมาดูตัวอย่างโค้ดสำหรับการคำนวณเบื้องต้น เช่น นำตัวเลขสองตัวมาคูณกัน

```
print(20 * 5)
```

```
100
```

หากมองว่าตัวเลข 20 และ 5 ที่นำมาคูณกันนั้นเปรียบเสมือนความกว้างและความยาวของสี่เหลี่ยมผืนผ้า เราสามารถแสดงผลโดยเพิ่มคำอธิบายเพื่อสื่อความหมายของค่าที่ได้จากการคำนวณ

```
print("พื้นที่ = ", 20 * 5, "ตารางเมตร")
```

```
พื้นที่ = 100 ตารางเมตร
```

นอกจากนี้ เรายังสามารถเก็บค่าความกว้างและความยาวไว้ในตัวแปรก่อนแล้วจึงนำมาคำนวณ เช่น ใช้ตัวแปร width และ length เพื่อทำการเก็บค่าความกว้างและยาว แล้วนำมาคำนวณหาพื้นที่ภายหลัง ซึ่งจะทำให้เห็นรายละเอียดและเข้าใจการทำงานของโค้ดได้ดียิ่งขึ้น

```
width = 20
length = 5
print("พื้นที่ = ", width * length, "ตารางเมตร")
```

```
พื้นที่ = 100 ตารางเมตร
```

เนื่องจากภาษา Python มองตัวอักษรพิมพ์ใหญ่และพิมพ์เล็กว่ามีความแตกต่างกัน (case sensitive) เช่น ชื่อตัวแปร width และ Width จะถือว่าเป็นคนละตัวกัน หากเราใช้คำสั่ง Width * length จะทำให้เกิดข้อผิดพลาด เนื่องจากตัวแปร Width ไม่ได้ถูกกำหนดค่าไว้ก่อน

```
width = 20
length = 5
print("พื้นที่ = ", Width * length, "ตารางเมตร")
```

```
NameError                                 Traceback (most recent call last)
Cell In[5], line 3
      1 width = 20
      2 length = 5
----> 3 print("พื้นที่ = ", Width * length, "ตารางเมตร")

NameError: name 'Width' is not defined
```

Python เป็นภาษาแบบ case sensitive

ภาษา Python เป็นภาษาแบบ case sensitive เหมือนกับภาษา C, C++, JavaScript นั่นคือ ตัวอักษรพิมพ์ใหญ่และพิมพ์เล็กมีความแตกต่างกัน

1. ชื่อตัวแปร

```
width = 20
Width = 18
WIDTH = 25
```

ในที่นี้ตัวแปร width, Width และ WIDTH จะเป็นคนละตัวกันในภาษา Python

2. ชื่อคำสั่ง

```
age = 18
IF age >= 18:
    print('You are an adult')
```

ในที่นี้คำสั่ง IF จะถูกตีความว่าเป็นคำสั่งที่ไม่ถูกต้องเนื่องจาก Python จะไม่รู้จักคำสั่ง IF ซึ่ง Python จะต้องการให้ใช้คำสั่ง if แทน

```
Cell In[7], line 2
    IF age >= 18:
       ^
```

SyntaxError: invalid syntax

3. ชื่อฟังก์ชัน

```
print('Hello') # คำสั่ง print ที่ถูกต้อง
PRINT('Hello')
```

หากมีการเรียกใช้คำสั่ง PRINT() จะทำให้เกิดข้อผิดพลาดเนื่องจาก Python จะไม่รู้จักคำสั่ง PRINT() ซึ่ง Python มองว่าเป็นคำสั่งที่แตกต่างจาก print()

```
NameError                                 Traceback (most recent call last)
Cell In[6], line 2
      1 print('Hello') # คำสั่ง print ที่ถูกต้อง
----> 2 PRINT('Hello')
```

NameError: name 'PRINT' is not defined

```
def calc_vat():
    pass
def calc_VAT():
    pass
```

สำหรับชื่อฟังก์ชัน calc_vat() และ calc_VAT() ก็จะถือเป็นคนละฟังก์ชันเช่นกัน

การใช้ indentation

ในภาษา Python มีการใช้ indentation คือ การเยื้องส่วนของโค้ดด้วยการเว้นวรรคโดยใช้ช่องว่าง (space) หรือแท็บ (tab) เพื่อกำหนดขอบเขตของโค้ดบล็อก (code block) หากไม่มีการเว้นวรรคที่ถูกต้อง จะเกิด IndentationError ส่งผลให้โค้ดทำงานไม่ได้

ในภาษา Python มักนิยมใช้ช่องว่าง (space) 4 ช่อง ในการจัดกลุ่มของโค้ดบล็อก ซึ่งจำนวนช่องว่างที่ใช้เว้นวรรคสำหรับโค้ดบล็อกเดียวกันจะต้องเท่ากัน

ลองมาดูตัวอย่างของการใช้ indentation ในโค้ดภาษา Python เพื่อจัดโค้ดบล็อกที่เกี่ยวข้องกันไว้ด้วยกัน

ใช้กับเงื่อนไข (if statement)

```
x = 10
if x > 5:
    print(x)
    print("ค่า x มากกว่า 5")   # เว้นวรรค 4 ช่อง
```

```
10
ค่า x มากกว่า 5
```

สังเกตว่าโค้ดในบรรทัดที่ 3 และ 4 จะมีการเยื้องเข้าไป ซึ่งโค้ดเหล่านี้จะทำงานในกรณีที่ x มีค่ามากกว่า 5 เท่านั้น

ตัวอย่างการใช้ Indentation ที่ผิด

```
x = 10
if x > 5:
print(x) # ❌ จะเกิด IndentationError เพราะไม่เว้นวรรค
print("x มากกว่า 5")
```

```
Cell In[4], line 3
    print(x)
    ^
IndentationError: expected an indented block after 'if' statement on line 2
```

ในตัวอย่างนี้จะเห็นได้ว่าเกิด IndentationError เนื่องจากไม่มีการเยื้องโค้ดในบรรทัดที่ 3 และ 4 ซึ่งต้องการให้ทำงานในกรณีที่ x มีค่ามากกว่า 5

ฟังก์ชัน print()

ฟังก์ชัน print() ใช้สำหรับแสดงผลบนหน้าจอ ถือเป็นคำสั่งที่มีการใช้งานมากที่สุดคำสั่งหนึ่ง มีรูปแบบการใช้งานหลัก ๆ ดังนี้

รูปแบบ	คำอธิบาย	ตัวอย่างการใช้งาน
`print(value)`	พิมพ์ค่า `value` และขึ้นบรรทัดใหม่	`print("Hello, World!")`
`print(value1, value2, ..., sep=' ', end='\n')`	พิมพ์ค่า `value1, value2, …` โดยใช้ตัวคั่น `sep` และสิ้นสุดด้วย `end`	`print("พื้นที่", 20 * 5, "ตารางเมตร", sep=" ", end="\n")`
`print(value1, value2, ..., end='')`	พิมพ์ค่า `value1, value2, …` โดยไม่ขึ้นบรรทัดใหม่	`print("พื้นที่", 20 * 5, "ตารางเมตร", sep=" ", end="")`

มาดูตัวอย่างการใช้งานฟังก์ชัน print() ในรูปแบบที่แตกต่างกัน

```
# แสดงข้อความ "Hello, World!" บนหน้าจอ และขึ้นบรรทัดใหม่ตอนจบ
print("Hello, World!")

# แสดงรายการเครื่องดื่ม หากไม่ได้ระบุ sep ก็จะคั่นแต่ละคำด้วยช่องว่าง (default separator)
และขึ้นบรรทัดใหม่ตอนจบ (end='\n' เป็น default จะไม่ระบุก็ได้)
print("mocha", "latte", "espresso", end='\n') # \n คือการกำหนดให้ขึ้นบรรทัดใหม่

# แสดงรายการเครื่องดื่ม โดยใช้ ">" เป็นตัวคั่นระหว่างคำ และขึ้นบรรทัดใหม่ตอนจบ
print("mocha", "latte", "espresso", sep=">")

# คำนวณพื้นที่และแสดงผลลัพธ์พร้อมข้อความอธิบายให้อยู่ในบรรทัดเดียวกัน
โดยแยกแต่ละค่าที่แสดงด้วยช่องว่าง และขึ้นบรรทัดใหม่ตอนจบ
print("area = ", 5 * 10, "sq.m.")

# แสดงรายการเครื่องดื่มชา โดยคั่นด้วยช่องว่าง และไม่ขึ้นบรรทัดใหม่ในตอนจบ (end="")
print("ชาเขียว", "ชาไทย", "ชามะลิ", sep=" ", end="")

# ต่อจากบรรทัดก่อนหน้าโดยแสดงคำว่า "ชากุหลาบ" และขึ้นบรรทัดใหม่ตอนจบ
print("ชากุหลาบ")

# แสดงคำว่า "ชาดำเย็น" ในบรรทัดถัดไป
print("ชาดำเย็น")
```

```
Hello, World!
mocha latte espresso
mocha>latte>espresso
area =  50 sq.m.
ชาเขียว ชาไทย ชามะลิชากุหลาบ
ชาดำเย็น
```

คำอธิบายโค้ด (comment)

การเขียนคำอธิบายเพิ่มเติมในโค้ดหรือคอมเมนต์ (comment) เป็นการอธิบายแนวคิด วิธีการทำงานของโค้ด ซึ่งจะช่วยให้ผู้พัฒนารวมถึงบุคคลอื่นที่อ่านหรือต้องการแก้ไขโค้ดในอนาคตเข้าใจโครงสร้างและวิธีการทำงานได้ดียิ่งขึ้น

ตัว Python Interpreter จะไม่ประมวลผลคำสั่งใด ๆ ที่อยู่ในคอมเมนต์ ดังนั้นคำสั่งที่เป็นคอมเมนต์จะไม่มีผลต่อการทำงานของโค้ด

ใน Python เราสามารถเขียนคอมเมนต์ได้ 3 วิธี

1. คอมเมนต์แบบบรรทัดเดียว ทำได้โดยใช้เครื่องหมาย # นำหน้าคำอธิบายโค้ด เช่น

```
# นี่คือคอมเมนต์บรรทัดเดียว
# another single line comment

# อัตราภาษีมูลค่าเพิ่ม 7%
vat = .07

# ราคาสินค้าก่อนรวมภาษีมูลค่าเพิ่ม
price_before_vat = 100
```

2. คอมเมนต์ต่อท้ายบรรทัดของโค้ด เราสามารถเขียนคอมเมนต์ต่อท้ายบรรทัดของโค้ดได้ โดยใช้เครื่องหมาย # หลังจากโค้ด เช่น

```
vat = .07 # อัตราภาษีมูลค่าเพิ่ม 7%
price_before_vat = 100 # ราคาสินค้าก่อนรวมภาษีมูลค่าเพิ่ม
```

3. คอมเมนต์แบบหลายบรรทัด เราสามารถเขียนคอมเมนต์หลายบรรทัดโดยใช้เครื่องหมาย ''' (triple single quote) หรือ """ (triple double quote) คลุมข้อความที่เป็นคอมเมนต์ เช่น

```
'''
นี่คือคอมเมนต์หลายบรรทัด ด้วย triple single quote
บรรทัดที่ 2
บรรทัดที่ 3
'''
c = 27
f = 1.8 * c + 32
print(f)
```

```
80.6
```

หรือใช้ """ (triple double quote)

```
"""
นี่คือคอมเมนต์หลายบรรทัด ด้วย triple double quote
บรรทัดที่ 2
บรรทัดที่ 3
"""
c = 27
f = 1.8 * c + 32
print(f)
```

```
80.6
```

คำสงวน (reserved keywords) ของภาษา Python

คำสงวน (reserved keywords) คือคำที่มีการกำหนดให้มีหน้าที่และความหมายเฉพาะเจาะจงอยู่แล้วในภาษาคอมพิวเตอร์ จึงไม่สามารถนำมาใช้เป็นชื่อตัวแปรหรือชื่อฟังก์ชันได้

คำสงวน	คำอธิบาย
`False`	ค่าความจริงเป็นเท็จ
`True`	ค่าความจริงเป็นจริง
`None`	ค่าว่าง
`and`	ตัวดำเนินการ และ
`or`	ตัวดำเนินการ หรือ
`not`	ตัวดำเนินการ ไม่
`in`	ตรวจสอบค่าในลิสต์หรือตัวแปร
`is`	ตรวจสอบความเหมือนกันของอ็อบเจกต์
`if`	เงื่อนไข
`else`	เงื่อนไขอื่น
`elif`	เงื่อนไขอื่น ๆ
`for`	การวนซ้ำ `for`
`while`	การวนซ้ำ `while`
`break`	หยุดการวนซ้ำ
`continue`	ดำเนินการต่อไป
`def`	การกำหนดฟังก์ชัน
`return`	คืนค่า
`class`	การกำหนดคลาส
`try`	การทดสอบ
`except`	การจัดการข้อผิดพลาด
`finally`	การทำงานท้ายสุด
`import`	การนำเข้าโมดูล
`from`	การนำเข้าบางส่วนจากโมดูล
`as`	การกำหนดชื่อใหม่ให้กับโมดูล
`lambda`	ฟังก์ชันแบบไม่มีชื่อ
`pass`	การผ่าน

หากมีการนำคำสงวนเหล่านี้มาตั้งเป็นชื่อตัวแปรหรือฟังก์ชันจะเกิดข้อผิดพลาดทางไวยากรณ์ (syntax error) เช่น

```
if = 10  # จะเกิด SyntaxError เนื่องจาก if เป็นคำสงวน
```

```
Cell In[12], line 1
    if = 10
       ^
SyntaxError: invalid syntax
```

สรุป

ภาษา Python ถูกออกแบบให้เป็นภาษาที่อ่านและเรียนรู้เข้าใจได้ง่าย เป็น open source ที่มีชุมชนนักพัฒนาจำนวนมากช่วยในการพัฒนาความสามารถของภาษา และโมดูลการทำงานต่าง ๆ อย่างต่อเนื่อง ตัวภาษาเองมีความสามารถสูง เป็นภาษาอเนกประสงค์ และมีโมดูลสำหรับงานด้านต่าง ๆ อยู่มากมาย ทำให้สามารถนำไปประยุกต์ใช้กับงานได้หลากหลาย เช่น ทำการวิเคราะห์ข้อมูล, AI, Machine Learning, Web application ฯลฯ ทำให้ได้รับความนิยมอย่างมากในปัจจุบัน

แบบฝึกหัด

1. ทำไม Python ถึงได้รับความนิยมมากกว่าภาษาโปรแกรมอื่น ๆ
2. อธิบายถึงการใช้ indentation ในภาษา Python
3. อธิบายความแตกต่างระหว่างพารามิเตอร์ sep และ end ที่อยู่ในฟังก์ชัน print() พร้อมยกตัวอย่างประกอบให้เห็นภาพ

2. ตัวแปร (variable)

วัตถุประสงค์การเรียนรู้

1. เข้าใจตัวแปรและประเภทของตัวแปรพื้นฐานใน Python
2. สามารถใช้งานตัวแปรประเภท str, int, float และ bool ได้อย่างถูกต้อง
3. แปลงประเภทข้อมูลระหว่างกันได้
4. ประยุกต์ใช้ตัวแปรในการเก็บข้อมูลทางธุรกิจได้

บทนำ

ในการวิเคราะห์และประมวลผลข้อมูลด้วย Python ตัวแปร (Variables) ถือเป็นพื้นฐานสำคัญที่จะช่วยให้เราจัดการข้อมูลได้อย่างมีประสิทธิภาพ หน้าที่หลักของตัวแปรคือการจัดเก็บข้อมูลหรือค่าภายในโปรแกรม โดยค่าที่เก็บสามารถเปลี่ยนแปลงได้ในระหว่างการทำงานโปรแกรม เราสามารถใช้ตัวแปรในการจัดเก็บข้อมูลต่าง ๆ เช่น ชื่อ ที่อยู่ ราคา กำไร ฯลฯ

Python มีประเภทข้อมูลพื้นฐานที่สำคัญ 4 ประเภท ได้แก่

1. String (str) สำหรับเก็บข้อความ
2. Integer (int) สำหรับเก็บตัวเลขจำนวนเต็ม
3. Float (float) สำหรับเก็บตัวเลขแบบมีทศนิยม
4. Boolean (bool) เป็นชนิดข้อมูลสำหรับแทน “ค่าความจริง” ของนิพจน์ตรรกะ โดยมีค่าได้เพียงสองค่า ได้แก่ True และ False

ข้อมูลประเภทตัวเลข

Python รองรับข้อมูลตัวเลขหลายประเภท ซึ่งแต่ละประเภทมีคุณสมบัติและการใช้งานที่แตกต่างกัน ดังนี้

- เลขจำนวนเต็ม (Integers) เช่น ข้อมูลจำนวนแก้วที่ลูกค้าสั่ง คะแนนสะสม
- เลขที่มีทศนิยม (Floating-Point Numbers) เช่น อัตราภาษีมูลค่าเพิ่ม อัตราดอกเบี้ย

เพื่อให้เห็นภาพชัดขึ้น มาดูตัวอย่างการกำหนดตัวแปรเพื่อเก็บข้อมูลคะแนนที่ได้รับเมื่อสั่งซื้อสินค้าในร้าน โดยกำหนดให้ออเดอร์นี้เป็นการสั่งซื้อเครื่องดื่มสองแก้ว โดยที่เครื่องดื่มแต่ละแก้วคิดเป็นคะแนน 10 คะแนน และปัจจุบันลูกค้าคนนี้มีคะแนนสะสมอยู่แล้ว 50 คะแนน

```
# กำหนดตัวแปร
cups_ordered = 2 # จำนวนแก้วที่สั่ง
```

```
points_per_cup = 10 # คะแนนที่จะได้รับต่อแก้ว
current_points = 50 # คะแนนสะสมปัจจุบันของลูกค้า
# คำนวณคะแนนที่จะได้รับและคะแนนสะสมใหม่
total_points = current_points + (cups_ordered * points_per_cup)
print("คะแนนสะสม", total_points, "คะแนน")
```

```
คะแนนสะสม 70 คะแนน
```

ตัวอย่างถัดมาเป็นการคำนวณราคาสินค้าก่อนและหลังภาษีมูลค่าเพิ่ม (VAT)

```
price_before_vat = 250  # integer
print(
    "ประเภทตัวแปร price_before_vat คือ", type(price_before_vat),
    "มีค่าเท่ากับ", price_before_vat,
)
vat_rate = 0.07  # float
print("ประเภทตัวแปร vat_rate คือ", type(vat_rate), "มีค่าเท่ากับ", vat_rate)
price_after_vat = price_before_vat * (1 + vat_rate)
print(
    "ประเภทตัวแปร price_after_vat คือ", type(price_after_vat),
    "มีค่าเท่ากับ", price_after_vat,
)
```

```
ประเภทตัวแปร price_before_vat คือ <class 'int'> มีค่าเท่ากับ 250
ประเภทตัวแปร vat_rate คือ <class 'float'> มีค่าเท่ากับ 0.07
ประเภทตัวแปร price_after_vat คือ <class 'float'> มีค่าเท่ากับ 267.5
```

ในตัวอย่างข้างต้นได้มีการใช้ฟังก์ชัน type() เพื่อแสดงประเภทของตัวแปร จะเห็นได้ว่าตัวแปร price_before_vat เป็นแบบ int ส่วน vat_rate เป็น float

สรุปตัวดำเนินการทางคณิตศาสตร์

สมมติให้ a = 7 และ b = 3

การคำนวณ	ตัวอย่าง	ผลลัพธ์
บวก (+)	`a + b`	`10`
ลบ (-)	`a - b`	`4`
คูณ (*)	`a * b`	`21`
หาร (/)	`a / b`	`2.3333333333333335`
หารเอาส่วน (//)	`a // b`	`2`
หารเอาเศษ (%)	`a % b`	`1`
ยกกำลัง (**)	`a ** b`	`343`

มาดูตัวอย่างวิธีการคำนวณโพรโมชันมา 4 จ่าย 3 โดยในที่นี้สมมติให้ราคาต่อหัวคือ 100 บาท และมีจำนวนลูกค้าในกลุ่มเดียวกัน 5 คน

```
come_x = 4  # มา 4 คน
pay_y = 3  # จ่าย 3 คน
per_head = 100  # ราคาปกติต่อหัว
pax = 5  # จำนวนคนทั้งหมด

amount = (
    (pax // come_x) # ลูกค้า (pax) 5 คน สามารถแบ่งได้ 1 กลุ่ม (come_x = 4 คน) และเหลืออีก 1 คน
    *(pay_y * per_head) # ในที่นี้มีกลุ่ม 4 คน 1 กลุ่มที่จ่ายในราคา 3 คน
    +(pax % come_x * per_head) # เศษเหลือ 1 คนจะต้องจ่ายราคาเต็ม
    )

print(f"amount = {amount}")
```

```
amount = 400
```

ตัวแปรข้อความ (String)

String หรือ str เป็นประเภทข้อมูลที่ใช้เก็บข้อความ อย่างเช่นหากเราต้องการเก็บข้อมูลต่าง ๆ ภายในร้านกาแฟ เราสามารถเก็บข้อมูลเหล่านี้เป็น str ได้ เช่น

- ชื่อเมนู
- ประเภทเครื่องดื่ม
- ชื่อลูกค้า
- วิธีการชำระเงิน

วิธีการกำหนดค่าให้กับตัวแปร str ทำได้โดยคลุมข้อความไว้ภายใต้เครื่องหมาย ' (single quote) หรือ " (double quote) ดังตัวอย่าง

```
# กำหนดตัวแปรเพื่อเก็บค่า string
menu_name_th = "มัฟฟิน" # ตัวแปรเก็บชื่อเมนูภาษาไทย
menu_name_en = 'Muffin' # ตัวแปรเก็บชื่อเมนูภาษาอังกฤษ
category = "bakery" # ตัวแปรเก็บประเภทของเมนู

print(menu_name_th, menu_name_en, category)
```

```
มัฟฟิน Muffin bakery
```

หากเราต้องการเก็บข้อความที่มีเครื่องหมาย ' (single quote) เช่น cookie'n cream ในที่นี้จะต้องคลุมข้อความด้วย " (double quote) ดังนี้

```
item = "cookie'n cream"
print(item)
```

```
cookie'n cream
```

ในทางกลับกัน หากข้อความที่เราต้องการเก็บมี " (double quote) อยู่ เราจะต้องใช้ ' (single quote) ในการคลุมข้อความนั้นแทน

```
s = 'My favorite movie is "The Wizard of Oz".'
print(s)
```

```
My favorite movie is "The Wizard of Oz".
```

การใช้งานสตริง

ถัดมาลองมาดูตัวอย่างการประยุกต์ใช้งาน string ในรูปแบบต่าง ๆ เช่น การเชื่อม string การหาความยาวของ string การเปรียบเทียบและเปลี่ยนแปลงตัวพิมพ์ของ string เพื่อให้โปรแกรมสามารถแสดงผลลัพธ์ในรูปแบบที่ต้องการได้

การเชื่อมต่อสตริงหลายตัวเข้าด้วยกัน

เราสามารถเชื่อม string เข้าด้วยกันโดยการใช้ตัวดำเนินการ + อย่างเช่นในตัวอย่างนี้ที่แสดงการเชื่อมชื่อเมนูภาษาไทยและภาษาอังกฤษ โดยให้ชื่อภาษาไทยอยู่ภายในวงเล็บด้านหลังชื่อภาษาอังกฤษ

```
menu_name_th = "มัฟฟิน"
menu_name_en = "Muffin"

print(f"ประเภทของตัวแปร menu_name_th คือ {type(menu_name_th)}")
print(f"ประเภทของตัวแปร menu_name_en คือ {type(menu_name_en)}")

# การเชื่อมข้อความ (String Concatenation) ด้วย +
full_menu = menu_name_en + " (" + menu_name_th + ")"
print(f"{full_menu = }")
```

```
ประเภทของตัวแปร menu_name_th คือ <class 'str'>
ประเภทของตัวแปร menu_name_en คือ <class 'str'>
full_menu = 'Muffin (มัฟฟิน)'
```

อีกหนึ่งวิธีในการเชื่อมชื่อเมนูภาษาไทยและภาษาอังกฤษคือการใช้ f-string (formatted string literal) ซึ่งเป็นการเติมตัวอักษร f ไว้ด้านหน้า quote และใส่ตัวแปร รวมถึงนิพจน์ที่ต้องการแสดงค่าไว้ระหว่าง {} ดังตัวอย่าง

```
# การใช้ f-string สำหรับแสดงผลข้อความ
print(f"{menu_name_en} ({menu_name_th})")

# การใช้ f-string สำหรับแสดงนิพจน์ (ค่าที่ได้จากการคำนวณ)
unit_price = 20
qty = 5
print(f"ราคารวม = {unit_price * qty} บาท")
```

```
Muffin (มัฟฟิน)
ราคารวม = 100 บาท
```

การหาความยาวของสตริง

ใช้ len() เพื่อหาความยาวหรือจำนวนตัวอักษรของ string เช่น หาความยาวของหมายเลขโทรศัพท์เคลื่อนที่ เพื่อตรวจสอบว่ามีความยาวถูกต้องตามที่กำหนดหรือไม่

```
mobile = "0941234567"
print(f"{len(mobile)=}")
```

```
len(mobile)=10
```

การเปรียบเทียบสตริงว่าเหมือนกันหรือไม่

ในการเปรียบเทียบ string ทำได้โดยการใช้ตัวดำเนินการ == โดยตัวอย่างนี้จะแสดงการเปรียบเทียบชื่อเมนูที่มีการสะกดตัวพิมพ์ใหญ่และตัวพิมพ์เล็กต่างกัน

```
menu1 = "cappuccino"
menu2 = "Cappuccino"

print(f"{menu1 == menu2 = }")
```

```
menu1 == menu2 = False
```

จะเห็นได้ว่าในการเปรียบเทียบข้อความ ตัวอักษรพิมพ์ใหญ่และพิมพ์เล็กจะถือว่าแตกต่างกัน (case sensitive) ดังนั้น menu1 จึงไม่เท่ากับ menu2 และทำให้ผลลัพธ์การเปรียบเทียบเป็น False

การแปลงสตริงเป็นตัวพิมพ์ใหญ่หรือพิมพ์เล็ก

มาลองดูเทคนิคที่มักพบในการเปรียบเทียบ string โดยไม่สนใจตัวพิมพ์ใหญ่หรือพิมพ์เล็ก ทำได้โดยการแปลง string ที่ต้องการเปรียบเทียบให้เป็นตัวพิมพ์ใหญ่ด้วย .upper() หรือตัวพิมพ์เล็กด้วย .lower() แล้วจึงนำมาเปรียบเทียบกัน โดยตัวอย่างต่อไปนี้จะแสดงการใช้ .upper()

```
menu1 = "cappuccino"
menu2 = "Cappuccino"

print(f"{menu1=}, {menu1.upper()=}")
print(f"{menu2=}, {menu2.upper()=}")
print(f"{menu1.upper() == menu2.upper() = }")
```

```
menu1='cappuccino', menu1.upper()='CAPPUCCINO'
menu2='Cappuccino', menu2.upper()='CAPPUCCINO'
menu1.upper() == menu2.upper() = True
```

การลบช่องว่างที่อยู่ด้านหน้าและด้านหลังของสตริง

เราสามารถลบช่องว่างที่อยู่ใน string ได้โดยการใช้ .strip() ดังตัวอย่าง

```
menu_with_space = "  Americano  "
print(f"{menu_with_space = }, {len(menu_with_space)=} ตัวอักษร")
clean_menu = menu_with_space.strip()
print(f"{clean_menu      = }, {len(clean_menu)=} ตัวอักษร")
```

```
menu_with_space = '  Americano  ', len(menu_with_space)=13 ตัวอักษร
clean_menu      = 'Americano', len(clean_menu)=9 ตัวอักษร
```

การแยกสตริงออกจากกันโดยใช้ตัวคั่น

เราสามารถแยก string หนึ่งตัวออกเป็นหลาย ๆ ส่วน เช่น หากเรามี string ที่เก็บรายการสั่งเครื่องดื่ม 3 รายการโดยแต่ละรายการถูกคั่นด้วย , เราสามารถใช้ split(",") เพื่อแยกรายการแต่ละรายการออกจากกัน ดังตัวอย่าง

```
# การแยกข้อความ
order = "Espresso,Latte,Americano"
items = order.split(",") # ส่งค่ากลับมาเป็น list
print(f"ก่อนการ split: {order=}")
print(f"หลังการ split: {items=}")
```

```
ก่อนการ split: order='Espresso,Latte,Americano'
หลังการ split: items=['Espresso', 'Latte', 'Americano']
```

โดยผลลัพธ์ที่ได้จะอยู่ในรูปของ list ซึ่งจะมีการกล่าวถึงในบทถัด ๆ ไป

การเชื่อมสตริงที่อยู่ใน list เข้าด้วยกัน

ในกรณีที่เราต้องการเชื่อม string ที่อยู่ใน list เข้าด้วยกัน เราสามารถใช้ .join() ได้ เช่น หากเรามี list ที่เก็บชื่อเมนูเครื่องดื่ม เราสามารถเชื่อมชื่อเมนูเหล่านั้นเข้าด้วยกันโดยใช้ , เป็นตัวคั่น ดังตัวอย่าง

```
# การเชื่อมข้อความ
menu_list = ['Espresso', 'Latte', 'Americano'] # สร้าง list ของเมนูกาแฟ
menu_text = ",".join(menu_list) # เชื่อมสมาชิกแต่ละตัวใน menu_list ด้วย ,

print(f"ก่อนการ join: {menu_list=}")
print(f"หลังการ join: {menu_text=}")
```

```
ก่อนการ join: menu_list=['Espresso', 'Latte', 'Americano']
หลังการ join: menu_text='Espresso,Latte,Americano'
```

การแทนที่ข้อความเดิมด้วยข้อความใหม่

ในตัวอย่างนี้เราต้องการแทนที่คำนำหน้าชื่อ จาก นางสาว เป็น คุณ ทำได้โดยการใช้ .replace() โดยระบุข้อความเดิมที่ต้องการแทนที่และข้อความใหม่ที่ต้องการแทนที่ลงไป ดังนี้

```
name = "นางสาวสานฝัน ทองประกาย"
new_name = name.replace("นางสาว", "คุณ")
print(f"{name     = }")
print(f"{new_name = }")
```

```
name     = 'นางสาวสานฝัน ทองประกาย'
new_name = 'คุณสานฝัน ทองประกาย'
```

อีกตัวอย่างหนึ่งเป็นการแทนที่ช่องว่างระหว่างหมายเลขโทรศัพท์ด้วยค่าว่าง ("") ซึ่งสามารถทำได้ดังนี้

```
mobile_phone = "082 123 1680"
new_mobile_phone = mobile_phone.replace(" ", "") # แทนที่ช่องว่างด้วย ""
print(f"{mobile_phone     = }")
print(f"{new_mobile_phone = }")
```

```
mobile_phone     = '082 123 1680'
new_mobile_phone = '0821231680'
```

ตัวแปรตรรกะ (Boolean)

Boolean หรือ bool เป็นตัวแปรที่ใช้เก็บค่า จริง (True) หรือ เท็จ (False) โดยมักจะนำไปใช้ในการสร้างเงื่อนไขหรือลูป ซึ่งจะมีการกล่าวถึงในบทถัด ๆ ไป

```
is_gold_member = True # กำหนดให้ is_gold_member เป็น True
print(f"ประเภทของตัวแปร is_gold_member คือ {type(is_gold_member)}")
print(f"ค่าของตัวแปร is_gold_member คือ {is_gold_member}")
```

```
ประเภทของตัวแปร is_gold_member คือ <class 'bool'>
ค่าของตัวแปร is_gold_member คือ True
```

```
age = 50

is_senior = age > 60 # ถ้า age มากกว่า 60 ค่า is_senior จะเป็น True ไม่เช่นนั้นจะเป็น False
print(f"{type(is_senior)=}")
print(f"{is_senior=}")
```

```
type(is_senior)=<class 'bool'>
is_senior=False
```

การตั้งชื่อตัวแปร

หลักในการตั้งชื่อตัวแปรที่ดีมีดังนี้

1. ใช้ชื่อที่สื่อความหมาย: ตั้งชื่อตัวแปรให้สื่อความหมายเกี่ยวกับค่าหรือข้อมูลที่ตัวแปรเก็บเอาไว้ ชื่อควรอ่านง่ายและเข้าใจได้โดยไม่ต้องอธิบายเพิ่มเติม เช่น salary, sales, month
2. ใช้ตัวอักษรพิมพ์เล็ก: ในการตั้งชื่อตัวแปรควรใช้ตัวอักษรพิมพ์เล็กทั้งหมด เช่น price, loan, discount
3. ใช้ตัวอักษรและตัวเลข: ชื่อตัวแปรสามารถประกอบไปด้วยตัวอักษร (a-z, A-Z) ตัวเลข (0-9) และขีดล่าง (_) เช่น quarter_1, may_2023 แต่ Python ไม่อนุญาตให้ใช้สัญลักษณ์พิเศษ เช่น @, $, %, +, -, *, /, # มาเป็นส่วนหนึ่งของชื่อตัวแปร
4. ใช้ตัวขีดล่าง (_) เพื่อแบ่งคำ: เช่น my_variable, user_name, passport_number
5. ไม่ใช้คำสงวน (Reserved words): เช่น if, for, while, def เนื่องจาก Python ใช้คำเหล่านี้เป็นส่วนของไวยากรณ์ของภาษา
6. เลี่ยงการใช้ชื่อฟังก์ชัน: เช่น input, sum ซึ่งเป็นฟังก์ชันที่มีอยู่ใน Python จึงไม่ควรนำมาตั้งเป็นชื่อตัวแปร เช่น input = 5, sum = 482

แบบแผนในการตั้งชื่อ (Naming convention)

Naming convention หรือแบบแผนการตั้งชื่อในการเขียนโค้ด คือ กฎและรูปแบบที่ใช้ในการตั้งชื่อตัวแปร ฟังก์ชัน คลาส และองค์ประกอบอื่น ๆ ในโค้ดเพื่อให้การตั้งชื่อเป็นไปอย่างมีแบบแผนและเป็นมาตรฐาน

- **snake_case**: ใช้ตัวอักษรพิมพ์เล็กและขีดล่าง (_) เพื่อแยกคำ เช่น menu_name, total_amount, price_before_discount
- **camelCase**: ใช้ตัวอักษรพิมพ์เล็กเริ่มต้นคำแรกและตัวอักษรพิมพ์ใหญ่เริ่มต้นคำที่สองและต่อไป เช่น menuName, totalAmount, priceBeforeDiscount
- **PascalCase**: ใช้ตัวอักษรพิมพ์ใหญ่เริ่มต้นคำแต่ละคำ เช่น MenuName, TotalAmount, PriceBeforeDiscount
- **kebab-case**: ใช้ตัวอักษรพิมพ์เล็กและขีดกลาง (-) เพื่อแยกคำ เช่น menu-name, total-amount, price-before-discount

การเลือกใช้ Naming convention

สำหรับ Naming convention ที่นิยมใช้กันใน Python มีดังนี้

- ตัวแปรและฟังก์ชัน: ใช้การตั้งชื่อแบบ snake_case เช่น student_name, top_5_menus, read_excel(), plot_histogram()
- คลาส: ใช้ PascalCase โดยใช้ตัวอักษรพิมพ์ใหญ่เริ่มต้นคำแต่ละคำ เช่น Customer, Product, DateTime, BarChart, CellFormat
- ค่าคงที่: ใช้ตัวอักษรพิมพ์ใหญ่ทั้งหมด (A-Z) และขีดล่าง (_) เพื่อแยกคำ เช่น PI, VAT_RATE, MAX_VALUE

การเปลี่ยนประเภทของตัวแปร

ในบางกรณีเราจำเป็นต้องเปลี่ยนประเภทของข้อมูลเพื่อให้สามารถนำไปใช้คำนวณได้อย่างถูกต้อง โดยส่วนมากแล้วจะเป็นการเปลี่ยนประเภทข้อมูลระหว่าง string และ integer

เพื่อให้เห็นชัดถึงความสำคัญของการแปลงประเภทของข้อมูล มาดูที่ตัวอย่างการคำนวณราคารวมกัน

```
price_int = 42  # ราคา เก็บเป็น integer
price_str = "42"  # ราคา เก็บเป็น string
quantity = 4

print(
    f"คำนวณราคารวมด้วยราคาที่ถูกเก็บเป็น integer: {price_int=}, {quantity=}, {price
_int * quantity=}"
```