

สารบัญ Guide book : Python

เรื่อง	หน้า
การติดตั้ง Python และ VS Code	1
การใช้งาน Terminal	2
ชนิดของตัวแปรและการประกาศตัวแปร	5
การรับค่า แสดงค่าและตัวดำเนินการ	6
การทำงานวนซ้ำแบบ while loop	7
การทำงานวนซ้ำแบบ for loop	8
การทำงานแบบเงื่อนไข	9
การใช้งาน List, Tuple, Set, Dictionary	10
การใช้งาน List ใน Python	11
การใช้งาน Tuple ใน Python	12
การใช้งาน Set ใน Python	12
การใช้งาน Dictionary ใน Python	13
การสร้างและใช้งานฟังก์ชัน	14
ตัวแปร Global และ Local	16
การ Import Library	17
แบบฝึกหัดและเฉลย	18

ไฟล์นี้เป็นเพียงการ Python Coding แบบเบสิกเท่านั้น
ไม่ได้มีการสอน OOP หรือการใช้คำสั่งจาก Library อื่น

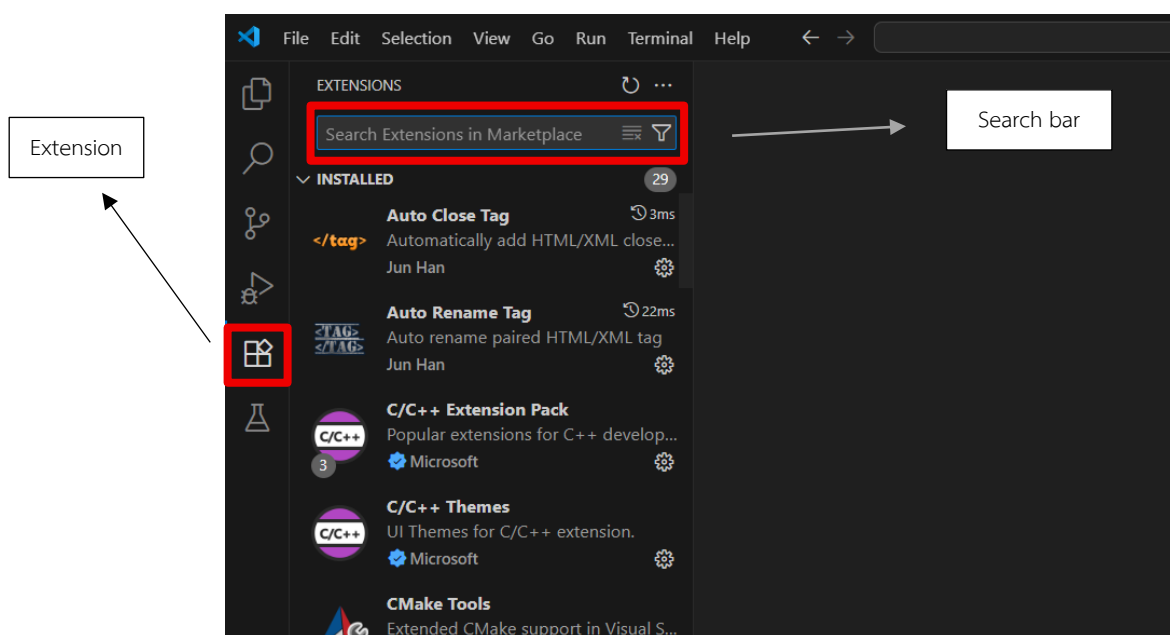
Chayanon Thuenwong

คู่มือ Python Programming เบื้องต้น

Guide book การเขียนโปรแกรมด้วยภาษา Python ผ่านโปรแกรม Visual Studio Code

การติดตั้ง Python และโปรแกรม Visual Studio Code

1. ติดตั้ง Python ผ่าน Python.org [<https://www.python.org/>]
2. ติดตั้ง Visual Studio ผ่าน visualstudio.com [<https://code.visualstudio.com/download>]
3. เข้า Visual Studio Code เพื่อติดตั้งเครื่องมือ Extension เพิ่มเติมสำหรับ Python programming



4. ที่ช่อง Search bar ให้พิมพ์ Python แล้วกด Install เพื่อลง Extension เพิ่มเติม
5. และติดตั้ง Extension “Code Runner” เพื่อใช้ในการรันโปรแกรม



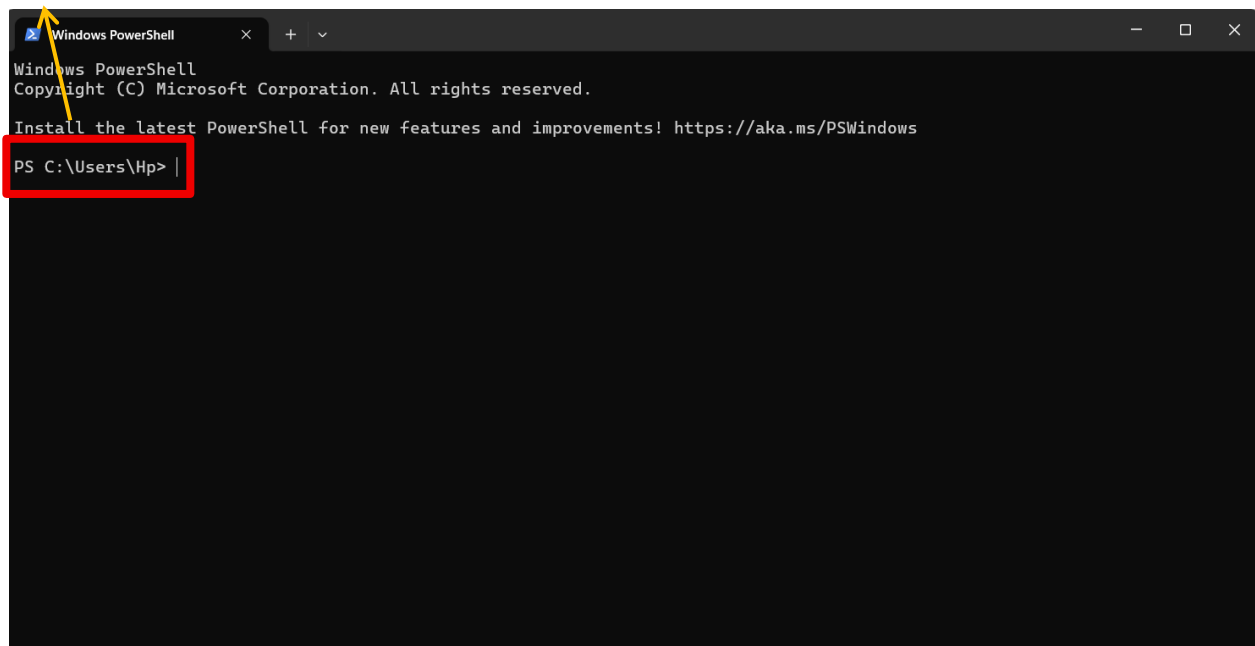
การใช้งาน VS code และการสร้างไฟล์ Python

1. การสร้างไฟล์ Python สามารถสร้างได้ง่ายโดยการสร้างไฟล์และตั้งชื่อ “ชื่อไฟล์.py” แต่ในคู่มือนี้จะสอนใช้งานการรันคำสั่งต่างๆ ผ่าน Terminal เพื่อฝึกการใช้งาน Terminal สำหรับงานในอนาคต

การใช้งาน Terminal ในการทำงาน

1. Terminal เปรียบเสมือนแผงควบคุมของคอมพิวเตอร์ โดยสามารถใช้คำสั่งต่าง ๆ ในการเรียกใช้งานข้อมูลในคอมพิวเตอร์ได้ Terminal ที่มีชื่อเสียง เช่น Command Prompt , Windows Powershell , Bash

ตำแหน่งไฟล์ที่ Terminal เข้าถึงอยู่



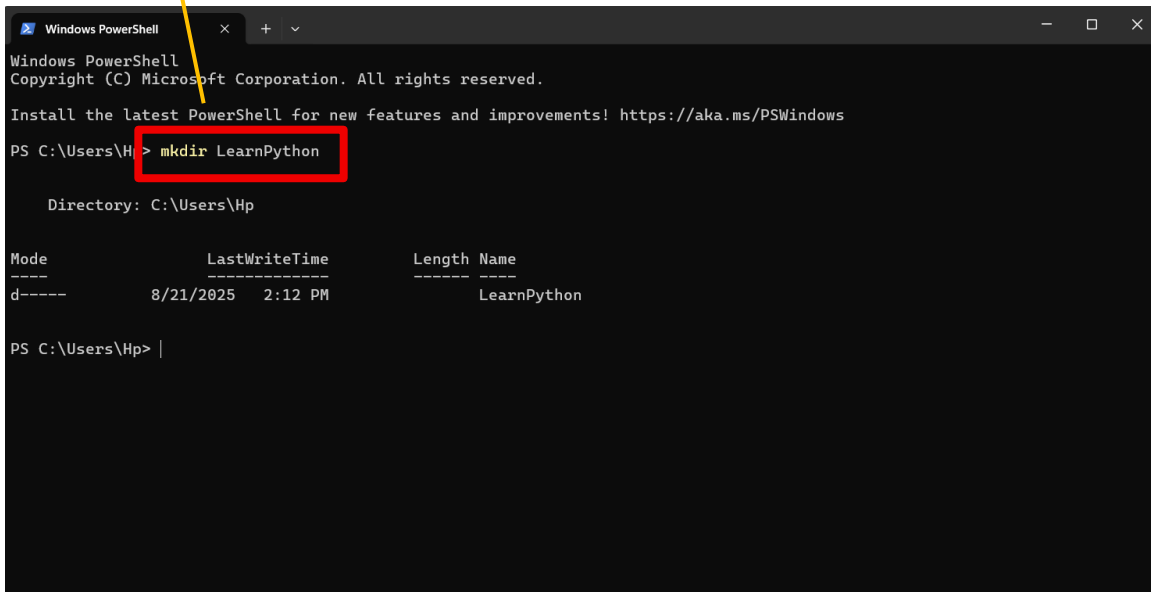
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hp> |
```

2. ใช้คำสั่ง `mkdir` ชื่อโฟลเดอร์ ในการสร้างโฟลเดอร์ใหม่ ในการเขียนโปรแกรม

คำสั่งในการสร้างโฟลเดอร์ใหม่



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

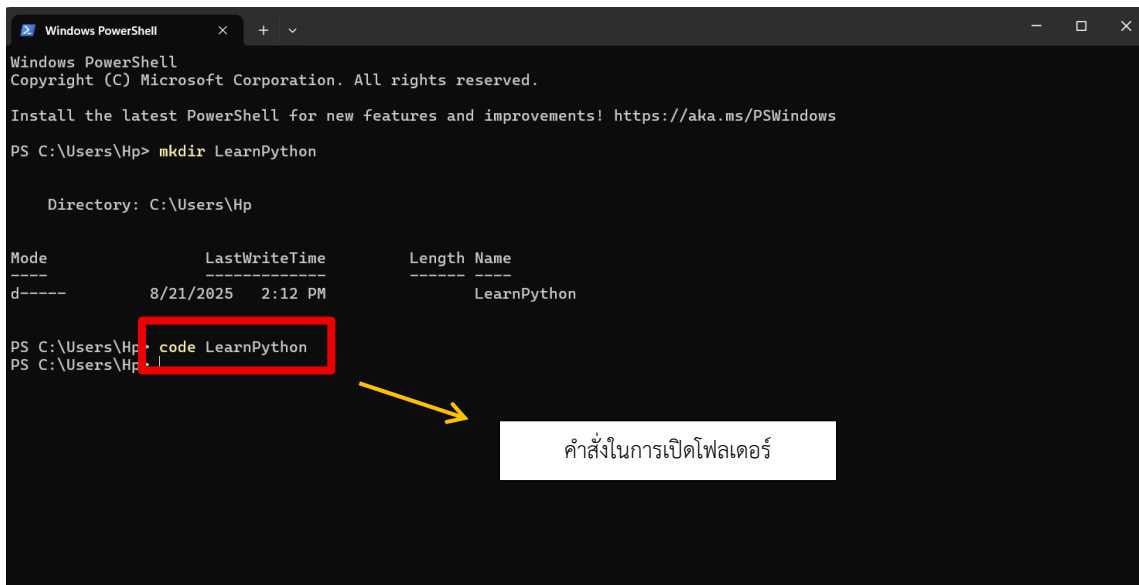
PS C:\Users\Hp> mkdir LearnPython

Directory: C:\Users\Hp

Mode                LastWriteTime         Length Name
----                -
d-----         8/21/2025   2:12 PM             LearnPython

PS C:\Users\Hp>
  
```

3. ใช้คำสั่ง `code` ชื่อโฟลเดอร์ ในการเปิดโฟลเดอร์เพื่อใช้ในการเขียนโค้ด หลังจากพิมพ์คำสั่งแล้ว Visual Studio จะเปิดขึ้นมาเอง



```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hp> mkdir LearnPython

Directory: C:\Users\Hp

Mode                LastWriteTime         Length Name
----                -
d-----         8/21/2025   2:12 PM             LearnPython

PS C:\Users\Hp> code LearnPython
PS C:\Users\Hp>
  
```

คำสั่งในการเปิดโฟลเดอร์

4. ใช้คำสั่ง `cd` ชื่อโฟลเดอร์ เพื่อให้ Terminal เข้าถึงตำแหน่งของโฟลเดอร์ที่เราสร้างเอาไว้

The screenshot shows a Windows PowerShell window. The user has created a directory named 'LearnPython' in the 'C:\Users\Hp' directory. The command prompt shows the directory listing for 'C:\Users\Hp', which includes the 'LearnPython' directory. The user then enters the command `cd .\LearnPython\`, which is highlighted with a red box. A yellow arrow points from this command to a white box containing the text 'เปลี่ยนตำแหน่งเข้าถึงข้อมูล' (Change access location).

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hp> mkdir LearnPython

Directory: C:\Users\Hp

Mode                LastWriteTime         Length Name
----                -
d-----          8/21/2025   2:12 PM             LearnPython

PS C:\Users\Hp> code LearnPython
PS C:\Users\Hp> cd .\LearnPython\
PS C:\Users\Hp\LearnPython>
  
```

5. ใช้คำสั่ง `echo > ชื่อไฟล์.py` ในการสร้างไฟล์ Python สำหรับเขียนโปรแกรม

The screenshot shows a Windows PowerShell window. The user has created a directory named 'LearnPython' in the 'C:\Users\Hp' directory. The command prompt shows the directory listing for 'C:\Users\Hp', which includes the 'LearnPython' directory. The user then enters the command `echo > Test.py`, which is highlighted with a red box. A yellow arrow points from this command to a white box containing the text 'คำสั่งในการสร้างไฟล์' (Command for creating file).

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Hp> mkdir LearnPython

Directory: C:\Users\Hp

Mode                LastWriteTime         Length Name
----                -
d-----          8/21/2025   2:12 PM             LearnPython

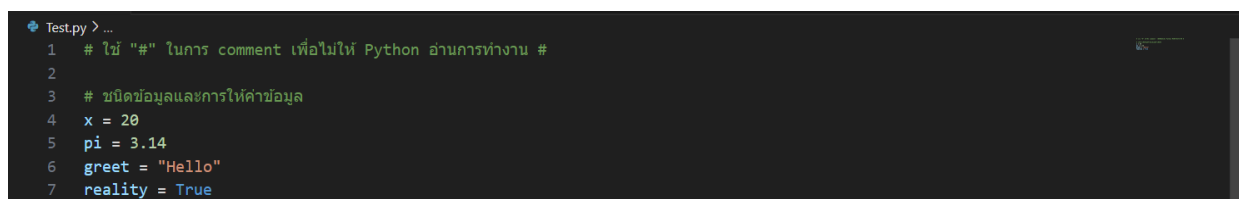
PS C:\Users\Hp> code LearnPython
PS C:\Users\Hp> cd .\LearnPython\
PS C:\Users\Hp\LearnPython> echo > Test.py

cmdlet Write-Output at command pipeline position 1
Supply values for the following parameters:
InputObject[0]:
PS C:\Users\Hp\LearnPython>
  
```

6. เสร็จสิ้นการสร้างไฟล์ กลับไปที่ Visual Studio Code เพื่อเขียนโค้ด

ชนิดของตัวแปรและการกำหนดตัวแปร

1. **ชนิดของตัวแปร** ในภาษา Python จะมีอยู่ 4 ชนิด ได้แก่
 - 1) int จะเป็นข้อมูลชนิดจำนวนเต็ม ทั้งจำนวนเต็มลบ เต็มศูนย์ เต็มบวก
 - 2) float จะเป็นข้อมูลชนิดจำนวนทศนิยม
 - 3) string จะเป็นข้อมูลชนิดข้อความตัวอักษร
 - 4) boolean จะเป็นข้อมูลที่เกี่ยวกับค่า True และ False
2. **การกำหนดตัวแปร** เป็นการตั้งชื่อตัวแปร แล้วกำหนดค่าให้ตัวแปรนั้น ๆ
 - 1) ตัวแปรชนิด int เช่น `x = 20`
 - 2) ตัวแปรชนิด float เช่น `pi = 3.14`
 - 3) ตัวแปรชนิด string ต้องมี “ ” หรือ ‘ ’ ในการกำหนดค่า เช่น `greet = "Hello"`
 - 4) ตัวแปรชนิด boolean เช่น `reality = True`
3. **การตั้งชื่อตัวแปร** มีนัยามในการตั้งชื่อเพื่อให้สามารถรันโปรแกรมได้
 - 1) ต้องประกอบด้วยตัวอักษรภาษาอังกฤษเล็กหรือใหญ่ ตัวเลข หรือ _ เท่านั้น
 - 2) ห้ามขึ้นต้นด้วยตัวเลข เช่น `123admin`
 - 3) ห้ามตั้งตามคำสงวน เช่น `if`, `for`,



```

Test.py > ...
1  # ใช้ "#" ในการ comment เพื่อไม่ให้ Python อ่านการทำงาน #
2
3  # ชนิดข้อมูลและการให้ค่าข้อมูล
4  x = 20
5  pi = 3.14
6  greet = "Hello"
7  reality = True
  
```

การใช้คำสั่งในการแสดงผลออกทางจอ

ใช้ฟังก์ชัน `print ()` ในการแสดงผลออกทางจอ ค่าที่อยู่ใน () สามารถเป็นได้ทั้งข้อความ ตัวแปรที่กำหนดค่าหรือรับค่าไว้แล้ว ค่าย้อนกลับของฟังก์ชัน

การใช้คำสั่งรับค่าข้อมูลมาเก็บในตัวแปร

ใช้คำสั่ง `input ()` โดยค่าที่อยู่ใน () คือค่าที่จะแสดงออกทางจอไปพร้อมกับการรับค่า โดยต้องมีการสร้างตัวแปรมารับค่า เช่น `name = input("ชื่อของคุณคือ : ")` มีข้อจำกัดเล็กน้อยดังนี้

1. ชนิดข้อมูลที่ถูกเก็บลงในตัวแปร จะมีชนิดข้อมูลเป็น String เสมอ
2. หากต้องการเปลี่ยนชนิดข้อมูล สามารถระบุชนิดข้อมูลไปด้านหน้าคำสั่งได้ เช่น
`Number = int(input("เลขของคุณ : "))` ชนิดข้อมูลที่ถูกเก็บจะเป็นจำนวนเต็ม

ตัวดำเนินการทางคณิตศาสตร์

เครื่องหมาย	ความหมาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ
**	ยกกำลัง

หากต้องการคำสั่งเพิ่มเติม เช่น รากที่สอง ตรีโกณ สามารถนำเข้า library จากภายนอกได้ (จะสอนในภายหลัง) การนำ `string + string` คือการเอาข้อความ 2 ข้อความมาต่อกัน เช่น `print("Hello" + "World")` เมื่อกดรันจะได้ Hello World แต่เมื่อนำ `int + int` จะเป็นการบวกเลขตามปกติ

ตัวดำเนินการเปรียบเทียบ

เครื่องหมาย	ความหมาย
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
!=	ไม่เท่ากับ
==	เท่ากับ

คำสั่งวนซ้ำแบบ while loop

1. การใช้คำสั่ง while เป็นการทำงานวนซ้ำแบบตรวจสอบเงื่อนไขก่อน แล้วจึงทำงานตามชุดคำสั่ง
2. ใช้คำสั่ง while เงื่อนไข: ในการทำงาน หากเงื่อนไขเป็นจริง ระบบจะทำงานตามชุดคำสั่ง

The image shows a Python IDE with a file named `Test.py`. The code in the editor is as follows:

```
1 i = 1
2 while i < 10:
3     print(i)
4     i = i + 1
```

A red box highlights the code, and a yellow arrow points to a text box that says "รูปแบบการใช้งานคำสั่ง" (Command usage format).

Below the code, the terminal output is shown:

```
PS C:\Users\Hp\LearnPython> python -u "c:\Users\Hp\LearnPython\Test.py"
1
2
3
4
5
6
7
8
9
PS C:\Users\Hp\LearnPython>
```

A red box highlights the output, and a yellow arrow points to a text box that says "ผลลัพธ์การทำงาน" (Execution result).

On the right side, there is a box titled "อธิบายการทำงาน" (Explanation of operation) with the following text:

Python จะตรวจสอบเงื่อนไขก่อน หากเงื่อนไขเป็นจริง จะทำงานตามคำสั่งด้านล่าง

1. $i < 10$ เป็นจริง เพราะกำหนดให้ $i = 1$
2. เมื่อเงื่อนไขเป็นจริง จึงทำงานคำสั่ง `print(i)`
3. กำหนดค่า i ใหม่ โดยให้ $+ 1$ เข้าไป
4. ทำแบบเดิมซ้ำจน $i = 10$
5. เมื่อ $i = 10$ ทำให้เงื่อนไขเป็นเท็จ
6. ระบบวนซ้ำจึงหยุดทำงาน

Below this, the code is updated to include a variable `x`:

```
1 x = "การพิมพ์เลขครั้งที่"
2 i = 1
3 while i <= 10:
4     print(x,i)
5     i+=1
```

The terminal output for this updated code is:

```
PS C:\Users\Hp\LearnPython> python -u "c:\Users\Hp\LearnPython\Test.py"
การพิมพ์ เลขครั้งที่ 1
การพิมพ์ เลขครั้งที่ 2
การพิมพ์ เลขครั้งที่ 3
การพิมพ์ เลขครั้งที่ 4
การพิมพ์ เลขครั้งที่ 5
การพิมพ์ เลขครั้งที่ 6
การพิมพ์ เลขครั้งที่ 7
การพิมพ์ เลขครั้งที่ 8
การพิมพ์ เลขครั้งที่ 9
การพิมพ์ เลขครั้งที่ 10
PS C:\Users\Hp\LearnPython>
```


คำสั่งวนซ้ำแบบ for loop

1. เป็นการวนซ้ำโดยระบบจะทำงานตามจำนวนรอบที่กำหนดไว้
2. ระบบจะบวกค่าตัวแปรขึ้นในการทำงานแต่ละรอบ ไม่ต้องเขียนคำสั่งเพิ่มค่าตัวแปร
3. ใช้คำสั่ง `for ตัวแปร in range(ค่าเริ่มต้นของตัวแปร, เงื่อนไข):`

The screenshot shows a Python IDE with a file named `Test.py`. The code in the editor is:

```
1 for i in range(1, 11):
2     print(i)
3
```

A red box highlights the code, and a yellow arrow points to a text box that says "รูปแบบการใช้งานคำสั่ง" (Usage of the command).

Below the code, the terminal output is shown, with a red box highlighting the numbers 1 through 10. A yellow arrow points to a text box that says "ผลลัพธ์การทำงาน" (Execution result).

On the right side, there is a text box titled "อธิบายการทำงาน" (Explanation of the operation) with the following text:

Python จะทำงานวนซ้ำตามจำนวนรอบที่กำหนดไว้

1. สร้างตัวแปร i โดยมีค่าเริ่มต้นเท่ากับ 1
2. ทำงานคำสั่ง `print (i)`
3. ทำงานวนซ้ำอีกครั้ง โดยค่า i เพิ่มขึ้นทีละ 1
4. ทำแบบเดิมจนถึงรอบที่ 10 ถือเป็นรอบสุดท้าย
5. ระบบจะหยุดในรอบที่ 11

การทำงานแบบเงื่อนไข if elif else

1. ใช้คำสั่ง *if* เงื่อนไข: ในการเริ่มใช้การทำงานเงื่อนไข หากคำสั่งเป็นจริงจะทำงานตามชุดคำสั่ง
2. หากมีเงื่อนไขหลายเงื่อนไข ใช้คำสั่ง *elif* เงื่อนไข: หลังคำสั่ง *if* เงื่อนไข:
3. คำสั่ง *else*: หากค่าอยู่นอกเหนือเงื่อนไข

The screenshot shows a Python IDE with a file named 'Test.py'. The code in the editor is as follows:

```
1 score = int(input("ป้อนคะแนนของคุณ : "))
2 if score < 50:
3     print("สอบตก 😞")
4 elif score >= 50:
5     print("สอบผ่าน 🎉")
```

A red box highlights the conditional logic block (lines 2-5). A yellow arrow points from this box to a white box containing the text 'รูปแบบการใช้งานคำสั่ง' (Usage of the command).

The terminal window at the bottom shows the execution of the script:

```
PS C:\Users\Hp\LearnPython> python -u "C:\Users\Hp\LearnPython\Test.py"
ป้อนคะแนนของคุณ : 62
สอบผ่าน 🎉
PS C:\Users\Hp\LearnPython>
```

การทำงานเงื่อนไขแบบ match case

1. ใช้คำสั่ง *match* ตัวแปร: ในการเลือกตัวแปรที่จะสร้างเงื่อนไข
2. ใช้คำสั่ง *case* เงื่อนไข: ในการสร้างเงื่อนไขและชุดคำสั่ง

The screenshot shows a Python IDE with a file named 'Test.py'. The code in the editor is:

```
score = int(input("เลือกบริการ [1-2] : "))
match score:
    case 1 : print("ถอนเงิน 🏧")
    case 2 : print("ฝากเงิน 🏧")
```

A red box highlights the code, and a yellow arrow points from it to a white box containing the text 'รูปแบบการใช้งานคำสั่ง' (Usage of the command). Below the code editor, the terminal shows the execution of the program:

```
PS C:\Users\Hp\LearnPython> python -u "c:\Users\Hp\LearnPython\Test.py"
เลือกบริการ [1-2] : 2
ถอนเงิน 🏧
PS C:\Users\Hp\LearnPython>
```

การใช้งาน List, Tuple, Set และ Dictionary

1. List เป็นการสร้างตัวแปรมาเก็บข้อมูลหลายๆ ค่า โดยข้อมูลสามารถซ้ำได้ แก้ไขได้
2. Tuple เป็นการสร้างตัวแปรมาเก็บข้อมูลหลายๆ ค่า โดยข้อมูลสามารถซ้ำได้ แต่แก้ไขไม่ได้
3. Set เป็นการสร้างตัวแปรมาเก็บข้อมูลหลายๆ ค่า โดยข้อมูลไม่สามารถซ้ำได้ แต่แก้ไขได้
4. Dictionary เป็นการสร้างตัวแปรมาเก็บข้อมูลแบบ Key – Value แก้ไขได้ ซ้ำได้

การใช้งาน List ใน Python

1. การสร้าง list ต้องมีการประกาศตัวแปร และให้ค่าโดยใช้ [] ครอบข้อมูล เช่น
`Car = ["Toyota", "Mitsubishi", "Ford", "Suzuki"]`
2. ตัวแปรเดียวแต่เก็บข้อมูลได้หลายค่า แก้ไขได้ ซ้ำกันได้



การเข้าถึงข้อมูลผ่าน Index

[" Toyota " , " Mitsubishi " , " Ford " , " Suzuki "]

Index ที่ 0 Index ที่ 1 Index ที่ 2 Index ที่ 3

Index คือการเข้าถึงข้อมูลผ่านลำดับข้อมูล โดยเริ่มนับจาก 0 จากทางซ้ายไปทางขวา สามารถเข้าถึงได้จาก ชื่อตัวแปร[index] เช่น `print(Car [1])` เมื่อรันจะได้ข้อมูลที่แสดงผลคือ Mitsubishi

การเข้าถึงข้อมูลผ่าน Index สามารถใช้ได้ทั้ง List, Tuple, Set, Dictionary แล้วยังใช้ในข้อมูลแบบ string ได้ด้วย เช่น "Python" จะมี Index ตั้งแต่ 0-5 หากใช้คำสั่ง `print(Python[0])` จะได้ "P" ออกมา

อีกคำสั่งที่เจอบ่อยคือ `len()` ย่อมาจาก length เป็นคำสั่งวัดจำนวน index ในตัวแปร หากใช้คำสั่ง `print(len(Car))` จะได้ 4 ออกมา หรือ `print(len(Python))` จะได้ 6 ออกมา

การใช้งาน Tuple ใน Python

1. การใช้งาน Tuple ต้องมีการประกาศตัวแปร และมี () ครอบชุดข้อมูลไว้ เช่น
`Car = ("Toyota", "Mitsubishi", "Ford", "Suzuki")`
2. การใช้งาน Tuple จะไม่สามารถเปลี่ยนแปลงข้อมูลได้

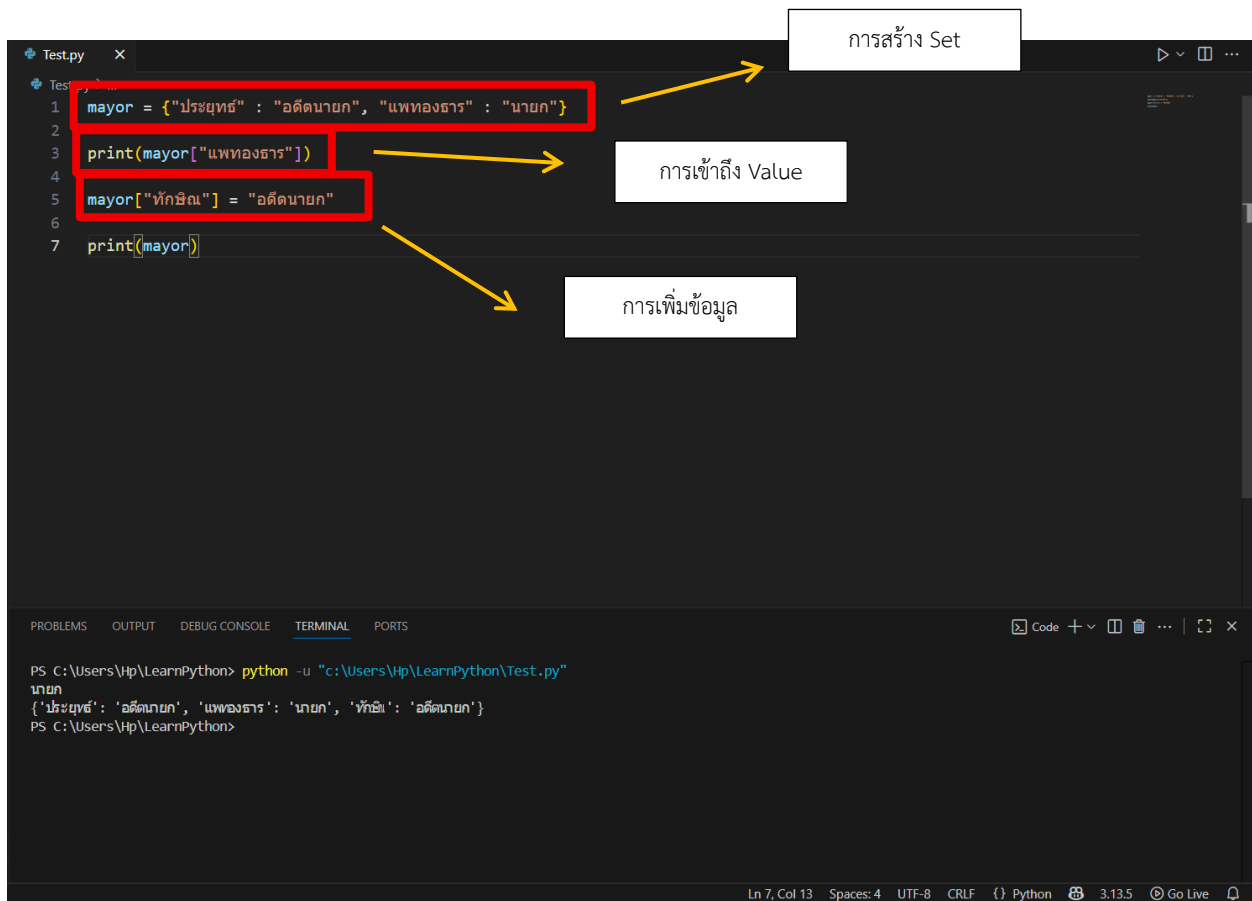
การใช้งาน Set ใน Python

1. การใช้งาน Set ต้องมีการประกาศตัวแปร และมี { } ครอบชุดข้อมูลไว้ เช่น
`Car = {"Toyota", "Mitsubishi", "Ford", "Suzuki"}`
2. สามารถแก้ไขข้อมูลได้ แต่ถ้ามีข้อมูลซ้ำกัน Python จะยัดให้มีข้อมูลเพียงตัวเดียว เช่น
`Car = {"Toyota", "Mitsubishi", "Ford", "Suzuki"}` หากเพิ่มข้อมูล "Toyota" ลงไป
 Python ก็ยังคงแสดงแค่ `Car = {"Toyota", "Mitsubishi", "Ford", "Suzuki"}` เหมือนเดิม



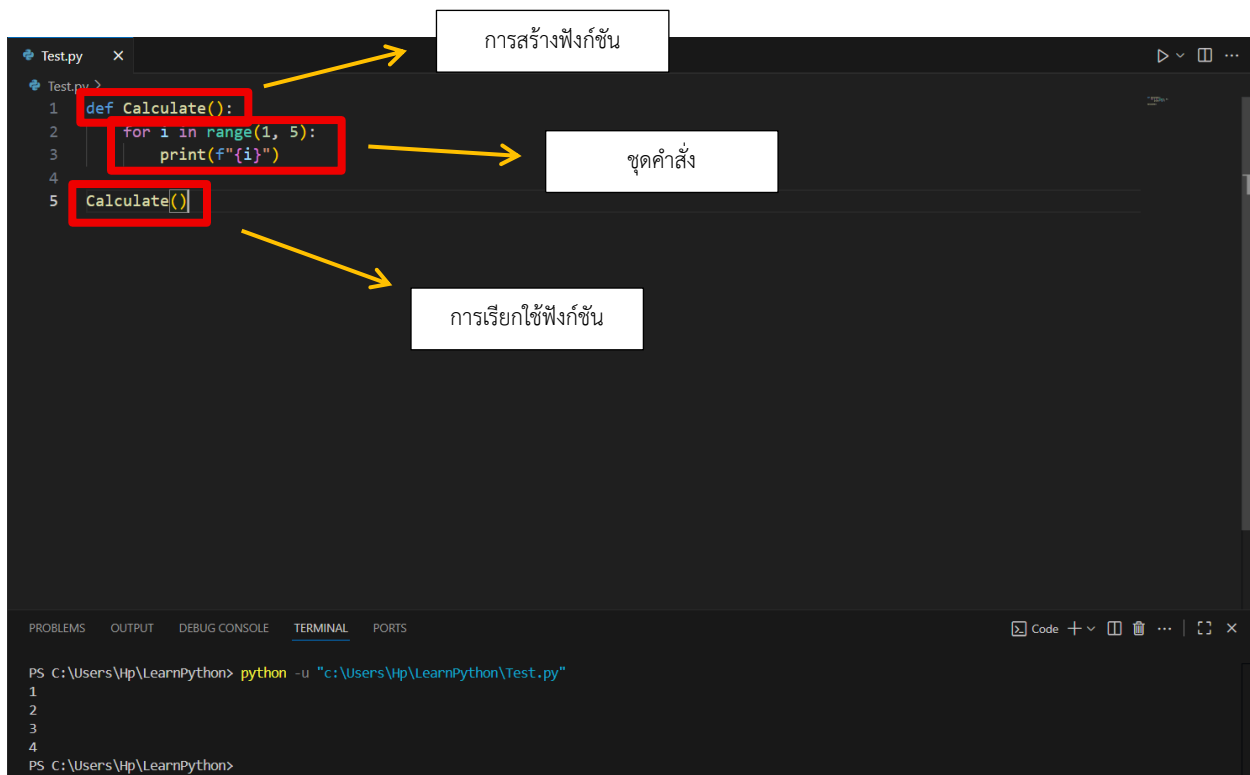
การสร้าง Dictionary ใน Python

1. การใช้งาน Dictionary ต้องมีการประกาศตัวแปร และใช้ { } ครอบชุดข้อมูล เช่น
`mayor = {"ประยุทธ์" : "อดีตนายก", "แพทองธาร" : "นายก"}`
2. ข้อมูลที่อยู่ใน Dictionary จะต้องมีการเก็บข้อมูลแบบ Key – Value เช่น
`{"ประยุทธ์" : "อดีตนายก"}` โดย “ประยุทธ์” จะทำหน้าที่เป็น Key และ “อดีตนายก” ทำหน้าที่ Value
3. หากเข้าถึง Key จะได้ Value ออกมา เช่น `print(mayor["แพทองธาร"])` จะได้ นายก ออกมา



การสร้างฟังก์ชันใน Python

1. การสร้างฟังก์ชัน เป็นการสร้างชุดคำสั่งเพื่อให้ง่ายต่อการเรียกใช้งาน
2. การสร้างชุดคำสั่งไว้ในฟังก์ชัน สามารถนำชุดคำสั่งไปใช้ใน Python ไฟล์อื่นได้ด้วย
3. การสร้างฟังก์ชันใช้คำสั่ง `def` ชื่อฟังก์ชัน (): เช่น `def calculated():`
4. การเรียกใช้งานฟังก์ชันใช้คำสั่ง ชื่อฟังก์ชัน() ได้เลย เช่น `calculated()`



*** เพิ่มเติม *** การใช้คำสั่ง `print()` แบบใช้ตัวแปรให้ง่ายขึ้น

1. `print("ชื่อของคุณ", name + "นามสกุลของคุณ", surname)` จากคำสั่งข้างต้น หากมีตัวแปรเยอะจะต้องเขียน , เยอะ และอาจจะงงได้
2. สามารถใช้ไวยากรณ์ `print(f"ชื่อของคุณ {name} นามสกุล {surname}")` แทนได้ โดยเติม `f` ข้างหน้า `"` และใช้ `{ }` ครอบตัวแปรได้เลย ก็ยังคงได้ผลลัพธ์เหมือนกัน

การใช้งานฟังก์ชันแบบมี Parameter และ Argument

1. เป็นการสร้างฟังก์ชันแบบมีค่าส่งเข้าไปในฟังก์ชัน
2. ตัวรับค่าในฟังก์ชัน เรียกว่า Parameter เช่น *def Calculated (number):*
โดย number เป็น Parameter
3. ตัวส่งค่าไปในฟังก์ชัน เรียกว่า Argument โดยใช้คู่กับการเรียกใช้ฟังก์ชัน เช่น *Calculated(n)*
โดย n เป็น Argument

Parameter

Argument

```

1 def Calculated(number):
2     area = (3.14**2) * number
3     return area
4
5 n = 2
6 print(Calculated(n))
  
```

อธิบาย

1. กำหนดให้ $n = 2$
2. เรียกใช้ฟังก์ชันโดยส่งค่า $n = 2$ ออกไปสู่ฟังก์ชัน
3. number รับค่า $n = 2$ เข้ามา ทำให้ number = 2 ด้วย
4. นำค่า number ไปใช้ในชุดคำสั่ง

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Hp\LearnPython> python -u "c:\Users\Hp\LearnPython\Test.py"

19.7192

PS C:\Users\Hp\LearnPython>

* คำสั่งเพิ่มเติม *

การใช้ *return* ตัวแปร มักพบเห็นได้มากในฟังก์ชัน การใช้ *return* เป็นการส่งค่าตัวแปรที่เลือกกลับไปยังจุดที่เรียกใช้ฟังก์ชัน

จากรูป *return area* นั้น Python จะส่งค่า *area* กลับไปยัง *Calculated(n)* ทำให้ *Calculated(n)* มีค่าเท่ากับ *area* เมื่อใช้คำสั่ง *print(Calculated(n))* จะเสมือนกับการเรียกใช้คำสั่ง *print(area)* เลย

ตัวแปรชนิด Global หรือ Local

1. ตัวแปร Global เป็นตัวแปรที่สามารถเรียกใช้งานได้ทุกที่ แต่ตัวแปร Local อาจจะเรียกใช้ได้แค่ฟังก์ชันนั้น ๆ
2. ตัวแปร Global มักจะเป็นตัวแปรที่ประกาศใช้ภายนอกฟังก์ชัน สามารถเข้าถึงได้จากทั้งในและนอกฟังก์ชัน
3. ตัวแปร Local มักจะเป็นตัวแปรที่ถูกประกาศใช้ภายในฟังก์ชัน ไม่สามารถเรียกใช้ได้จากนอกฟังก์ชัน เมื่อฟังก์ชันเรียกใช้แล้วก็จะถูกลบออกจากหน่วยความจำ

The screenshot shows a Python script in a dark-themed IDE. The code is as follows:

```

1 pi = 3.14
2 i = 10
3
4 def calculated(number):
5     n = number
6     area = pi * pi * n
7     return area
8
9 print(calculated(i))
10
11 print(n)
  
```

Annotations with yellow arrows point to specific lines of code:

- Line 1: `pi = 3.14` and Line 2: `i = 10` are grouped by a red box. An arrow points to a text box: "ประกาศตัวแปรนอกฟังก์ชัน จึงเป็นตัวแปร global".
- Line 5: `n = number` is highlighted with a red box. An arrow points to a text box: "ประกาศตัวแปร n ภายในฟังก์ชัน n จึงเป็นตัวแปร local".
- Line 11: `print(n)` is highlighted with a red box. An arrow points to a text box: "คำสั่งนี้จะรันไม่ได้ เพราะ n ถูกประกาศอยู่ในฟังก์ชัน Python จึงไม่รู้จักตัวแปร n เพราะ n เป็นตัวแปร local".

* การใช้คำสั่งเพิ่มเติมเกี่ยวกับ Global *

ตัวแปรชนิด global จะสามารถใช้ได้ทั้งในและนอกฟังก์ชัน แต่หากต้องการเปลี่ยนแปลงค่าของตัวแปร global จากภายในฟังก์ชัน จะไม่สามารถทำได้ เพราะ Python จะนึกว่าตัวแปรนั้นเป็นตัวแปรใหม่ จึงต้องมีการใช้ *global* ตัวแปร เพื่อที่จะบอก Python ว่าตัวแปรตัวนั้น เป็นตัวแปรเดียวกันกับตัวแปรนอกฟังก์ชัน

The screenshot shows a Python script in a dark-themed IDE. The code is as follows:

```

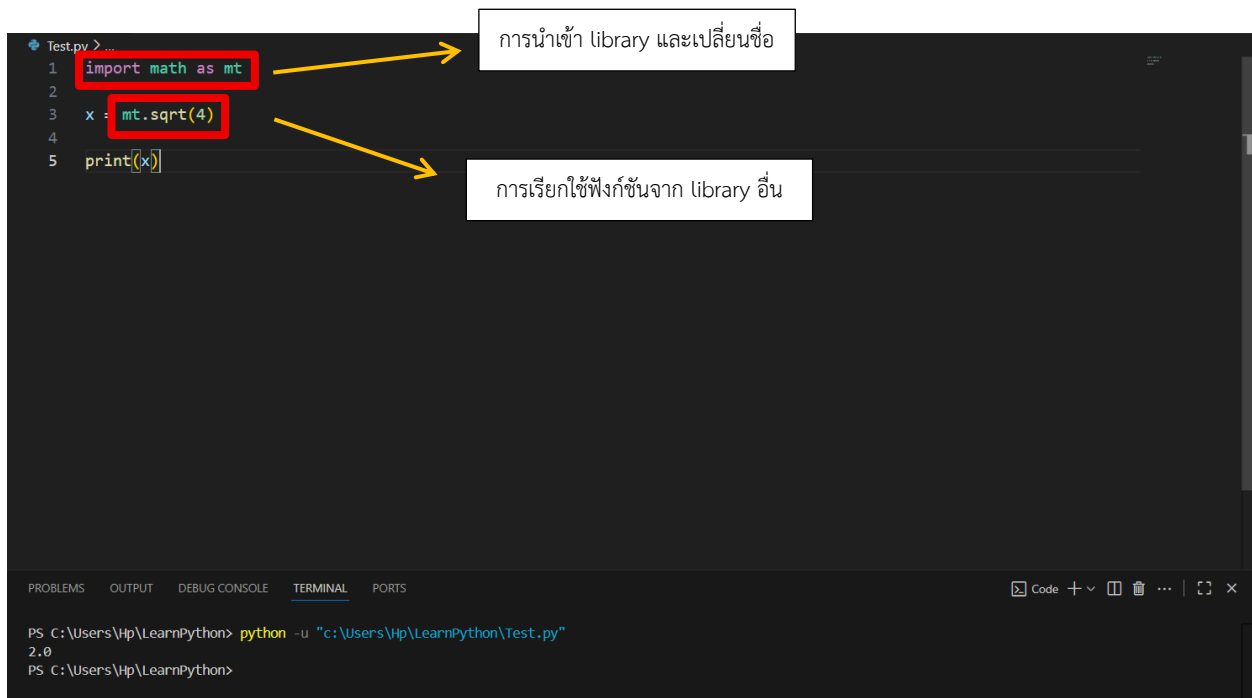
1 i = 0
2
3 def calculated(number):
4     global i
5     i = i + 1
6     return i
7
8 print(calculated(i))
9
10
11
  
```

An annotation with a yellow arrow points to line 4: `global i`. The text box says: "ใช้คำสั่ง global i เพราะฟังก์ชัน มีการเปลี่ยนแปลงค่าของ i ขึ้นทีละ 1".

At the bottom, the terminal output shows the command: `python -u "c:\Users\vip\LearnPython\Test.py"` and the output: `1`.

การ Import Library จากแหล่งอื่น

1. การ import library คือการนำเข้าฟังก์ชันจากไฟล์อื่นหรือแหล่งอื่น
2. การ import library มักใช้อยู่ส่วนหัวของไฟล์
3. ใช้คำสั่ง `import` ชื่อ `library` และเมื่อนำเข้ามาแล้วสามารถเปลี่ยนชื่อได้จากคำสั่ง `as` ชื่อใหม่ เช่น
`import math as mt` คือการนำเข้าฟังก์ชันของ `math` ด้วยชื่อใหม่คือ `mt`
4. การเรียกใช้ฟังก์ชันจาก library อื่นที่นำเข้ามาใช้งานทำได้โดย ชื่อ.ชื่อฟังก์ชัน() เช่น `mt.sqrt()`
หมายถึงเรียกใช้ library `math` และเรียกใช้ฟังก์ชัน `sqrt()`



แบบฝึกหัด การเขียนโปรแกรมด้วยภาษา Python

ข้อ	โจทย์
1	เขียนโปรแกรมรับเลข 2 ตัว แล้วหาผลรวม ผลต่าง ผลคูณและผลหาร
2	เขียนโปรแกรมรับค่าตัวเลขเรื่อย ๆ จนกว่าค่าที่รับจะติดลบ โปรแกรมจึงหยุด แล้วหาค่าเฉลี่ยของค่าที่รับเข้ามา
3	เขียนโปรแกรมรับค่าตัวเลข แล้วแสดงออกมาเป็นพีระมิด ตัวอย่าง รับตัวเลข 5 เข้ามา 1 12 123 1234 12345
4	เขียนโปรแกรมรับค่าตัวเลข แล้วหาผลรวมของเลขโดด ตัวอย่าง รับตัวเลข 572 เข้ามา $5 + 7 + 2 = 14$
5	เขียนโปรแกรมสร้างระบบสมัครสมาชิก และเข้าสู่ระบบ โดยมีการรับค่า Username และ Password
6	เขียนโปรแกรมจำลองตู้ ATM มีระบบฝากเงิน ถอนเงิน เช็คยอดเงินคงเหลือ
7	เขียนโปรแกรมรับค่าตัวเลข แล้วคำนวณหาค่าของแฟกทอเรียลจากตัวเลขที่รับมา

เฉลย แบบฝึกหัด การเขียนโปรแกรมด้วยภาษา Python

ข้อที่ 1 เขียนโปรแกรมรับเลข 2 ตัว แล้วหาผลรวม ผลต่าง ผลคูณและผลหาร

```
Test.py > ...
1 x = int(input("เลขตัวที่ 1 : "))
2 y = int(input("เลขตัวที่ 2 : "))
3 print(f"ผลรวม {x+y} ผลต่าง {x-y} ผลคูณ {x*y} ผลหาร {x/y}")
```

ข้อที่ 2 เขียนโปรแกรมรับค่าตัวเลขเรื่อย ๆ จนกว่าค่าที่รับจะติดลบ โปรแกรมจึงหยุด แล้วหาค่าเฉลี่ยของค่าที่รับเข้ามา

```
Test.py > ...
1 default = 0
2 i = 0
3 while True:
4     x = int(input("ป้อนตัวเลข : "))
5     if x > 0:
6         default += x
7         i += 1
8     elif x == 0:
9         continue
10    elif x < 0:
11        print(f"ค่าเฉลี่ยของคุณคือ {default/i}")
12
```

ข้อที่ 3 เขียนโปรแกรมรับค่าตัวเลข แล้วแสดงออกมาเป็นพีระมิด

```
Test.py X
Test.py > ...
1 num = int(input("ป้อนเลขของคุณ : "))
2 for i in range(1,num+1):
3     for j in range(1,i+1):
4         print(j, end="")
5     print("")
6
```

ข้อที่ 4 เขียนโปรแกรมรับค่าตัวเลข แล้วหาผลรวมของเลขโดด

```
Test.py X
Test.py > ...
1 default = 0
2 num = input("ป้อนเลขของคุณ : ")
3
4 for i in range(1, len(num) + 1):
5     default += int(num[i - 1])
6
7 print(default)
8
```

ข้อที่ 5 เขียนโปรแกรมสร้างระบบสมัครสมาชิก และเข้าสู่ระบบ โดยมีการรับค่า Username และ Password

```

Test.py > ...
1 def Login(Username, Password):
2     if Username in ID and Password == ID[Username]:
3         print("เข้าสู่ระบบสำเร็จสิ้น")
4     else:
5         print("ส่มเหลว! โปรดลองอีกครั้ง")
6
7
8 def Register(Username, Password):
9     ID[Username] = Password
10    print("สมัครสมาชิกสำเร็จ")
11
12
13 ID = {}
14 while True:
15     menu_list = int(
16         input("\nเลือกบริการของคุณ \n1.เข้าสู่ระบบ\n2.สมัครสมาชิก\n3.ออก\nกรณบริการของคุณ : ")
17     )
18
19     if menu_list == 1:
20         loginUsername = input("Username : ")
21         loginPassword = input("Password : ")
22         Login(loginUsername, loginPassword)
23     elif menu_list == 2:
24         registerUsername = input("Username : ")
25         registerPassword = input("Password : ")
26         Register(registerUsername, registerPassword)
27     elif menu_list == 3:
28         print("ขอบคุณที่ให้บริการ")
29         break
30     else:
31         print("โปรดป้อนค่าให้ถูกต้อง")
32

```

ข้อที่ 6 เขียนโปรแกรมจำลองตู้ ATM มีระบบฝากเงิน ถอนเงิน เช็ดยอดเงินคงเหลือ

```

Test.py X
Test.py > ...
1 def Deposit(money):
2     global count
3     count += money
4     print("ฝากเงินสำเร็จ")
5
6 def Withdraw(money):
7     global count
8     count -= money
9     print("ถอนเงินสำเร็จ")
10
11 count = 0
12 while True:
13     menu_list = int(
14         input("\nเลือกบริการของคุณ\n1.ฝากเงิน\n2.ถอนเงิน\n3.ตรวจสอบยอด\n4.ออก\nเลือกบริการของคุณ : ")
15     )
16     if menu_list == 1:
17         DepoMoney = int(input("ป้อนยอดฝากของคุณ : "))
18         Deposit(DepoMoney)
19     elif menu_list == 2:
20         WithMoney = int(input("ยอดถอนของคุณ : "))
21         if WithMoney <= count:
22             Withdraw(WithMoney)
23         elif WithMoney > count:
24             print("ยอดเงินของคุณไม่เพียงพอ")
25     elif menu_list == 3:
26         print(f"ยอดเงินของคุณ {count}")
27     elif menu_list == 4:
28         print("ขอบคุณที่ให้บริการ")
29         break
30     else:
31         print("โปรดป้อนให้ถูกต้อง")
32

```

ข้อที่ 7 เขียนโปรแกรมรับค่าตัวเลข แล้วคำนวณหาค่าของแฟกทอเรียลจากตัวเลขที่รับมา

```
Test.py X
Test.py > ...
1 def factorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return n * factorial(n - 1)
6
7
8 n = int(input("ป้อนตัวเลข : "))
9 print(f"{n}! = {factorial(n)}")
10
```