

Basic Python

(ทฤษฎีเบื้องต้น)



Uncle Engineer

ลุงวิศวกร สอนคำนวณ

แนะนำภาษา Python



ภาษาไพธอน (Python) เป็นภาษาระดับสูง (high level) โดยคำว่าระดับสูง หมายถึง เป็นภาษาที่เขียนได้ง่ายต่อความเข้าใจของมนุษย์ แล้วจึงแปลงเป็นภาษาระดับต่ำ (low level) หรือภาษาเครื่อง ให้คอมพิวเตอร์เข้าใจการทำงาน

โดยภาษาไพธอน ถูกสร้างขึ้นมาเมื่อปี พ.ศ. 2533 (ค.ศ. 1990) โดย Guido van Rossum โปรแกรมเมอร์ชาวเนเธอร์แลนด์

ภาษา Python มีที่มาจากไหน ?



ภาษา Python มีที่มาจากเจ้าตัวนี้

ใช่ หรือ ไม่ ?

ภาษา Python มีที่มาจากไหน ?



ภาษา Python มีที่มาจากชื่อ “Monty Python’s Flying Circus” รายการทีวีแนวตลกทางช่อง BBC ของอังกฤษ ซึ่ง Guido van Rossum เป็นแฟนตัวยงของรายการนี้ จึงได้นำคำว่า “Python” มาใช้ตั้งชื่อภาษาโปรแกรมที่เขาสร้างขึ้น

ใครบ้างที่ใช้ Python ?



WIKIPEDIA

Google



amazon



facebook

Python เป็นภาษาที่มี Syntax ที่ดูง่าย และสะดวกต่อผู้เริ่มเขียนโปรแกรม และสามารถนำไปใช้งาน เพื่อเขียนโปรแกรมให้กับ Hardware ได้ง่าย ดังนั้นจึงมีหลายบริษัท และหลายองค์กร ที่พัฒนางานด้วย Python

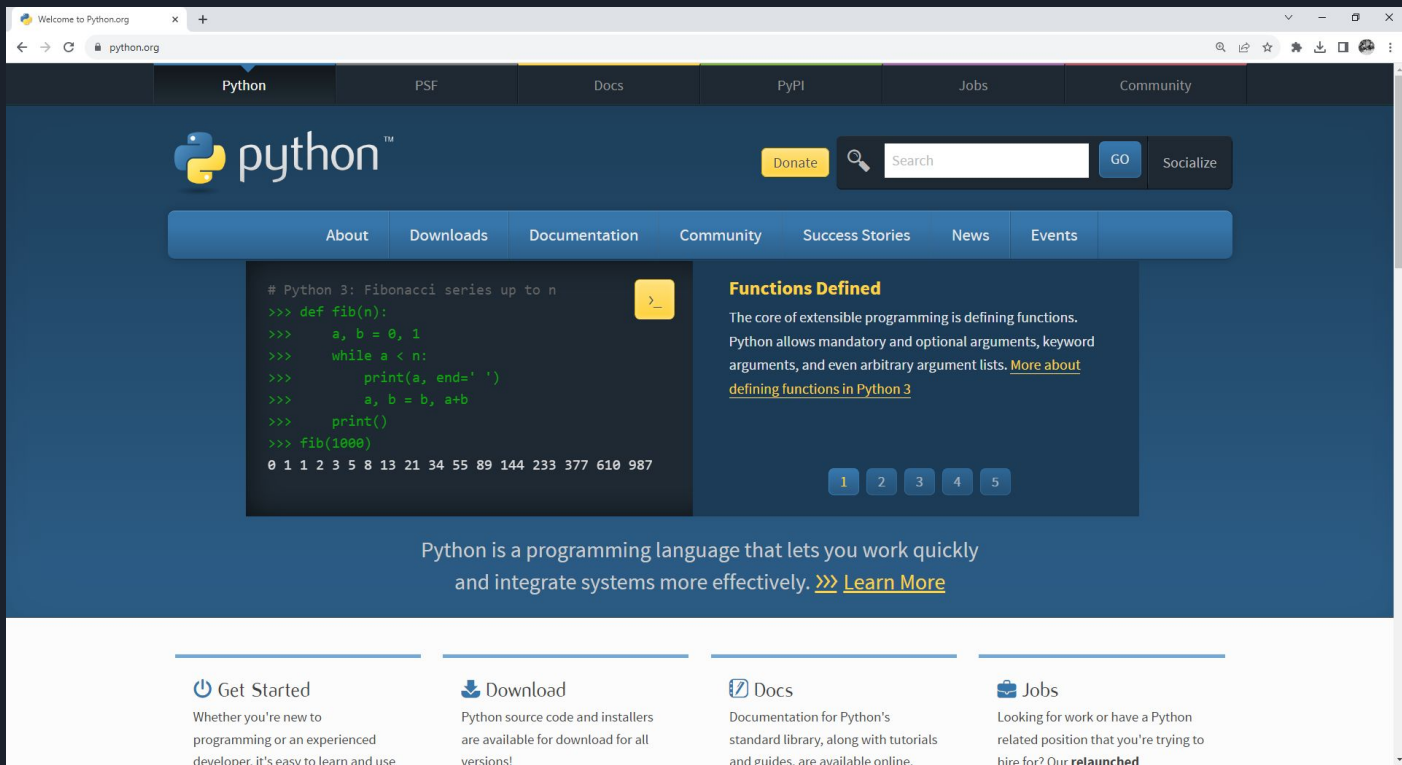


Python ทำอะไรได้บ้าง ?

- เขียนโปรแกรมสำหรับเด็ก
- เขียนโปรแกรม GUI บน Desktop
- เขียนเกม
- เขียนเว็บไซต์
- งาน Network Engineering
- คำนวณเชิงคณิตศาสตร์, สถิติ
- คำนวณเชิงฟิสิกส์
- คำนวณเชิงวิศวกรรม
- Image Processing
- AI, Machine Learning
- งาน Data Science
- Big Data, Data Analysis
- 3D Modeling
- Internet of Things
- Robotic Process Automation
- อื่นๆ อีกมากมาย

การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่1: เข้าเว็บ python.org



The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a large blue banner with the Python logo, a search bar, and a 'Donate' button. A secondary navigation bar contains links to About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a code snippet for a Fibonacci function, a section titled 'Functions Defined' explaining the core of extensible programming, and a list of five numbered links. At the bottom, there's a white section with four columns: 'Get Started', 'Download', 'Docs', and 'Jobs', each with a brief description.

Welcome to Python.org

python.org

Python PSF Docs PyPI Jobs Community

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>>     fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [>>> Learn More](#)

Get Started
Whether you're new to programming or an experienced developer, it's easy to learn and use.

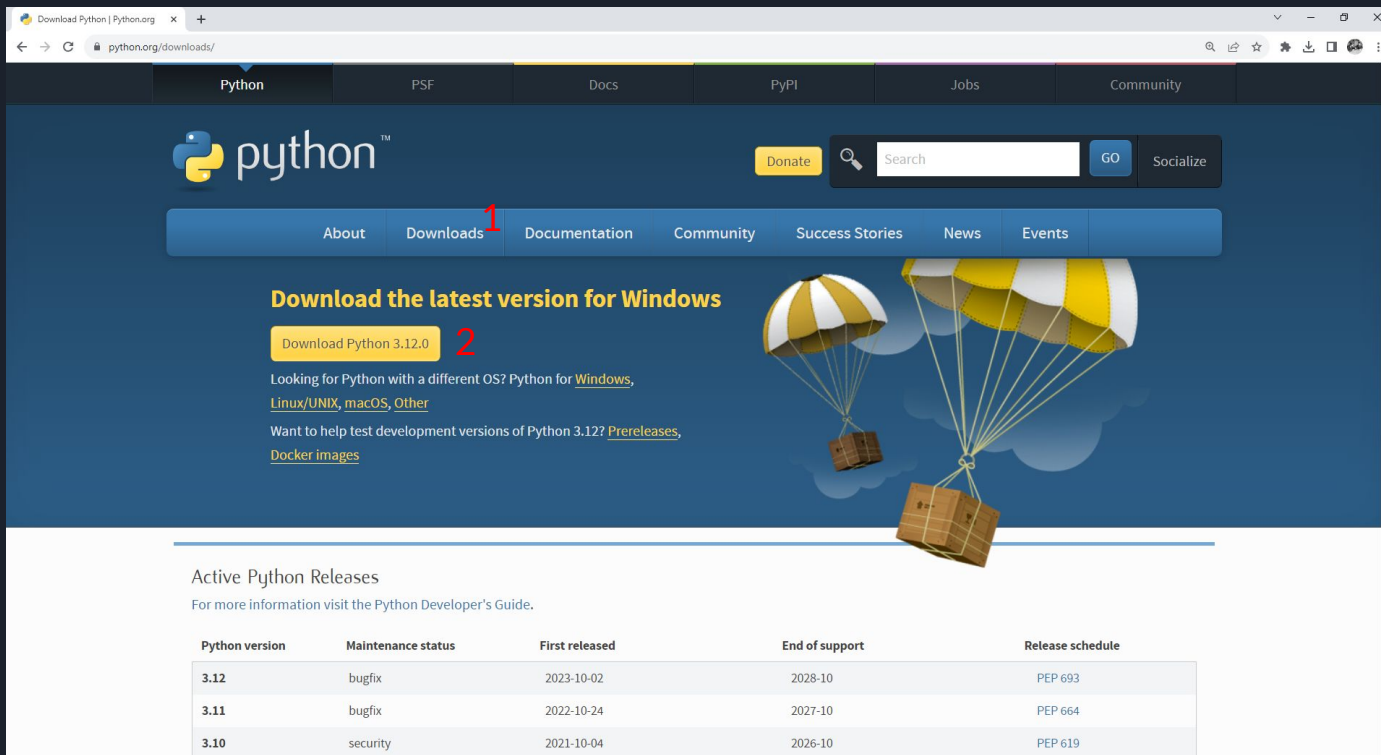
Download
Python source code and installers are available for download for all versions!

Docs
Documentation for Python's standard library, along with tutorials and guides, are available online.

Jobs
Looking for work or have a Python related position that you're trying to hire for? [Our relaunched](#)

การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่2: ซึ่ปุ่ม Downloads แล้วเลือก Download Python 3.xx.x



The screenshot shows the Python.org website. The navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this, a secondary navigation bar has links for About, Downloads (marked with a red '1'), Documentation, Community, Success Stories, News, and Events. The main content area features the Python logo, a search bar, and a 'Donate' button. Below the navigation bar, the 'Downloads' section is highlighted, showing the text 'Download the latest version for Windows' and a yellow button labeled 'Download Python 3.12.0' (marked with a red '2'). Below this button, there are links for 'Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other' and 'Want to help test development versions of Python 3.12? Prereleases, Docker images'.

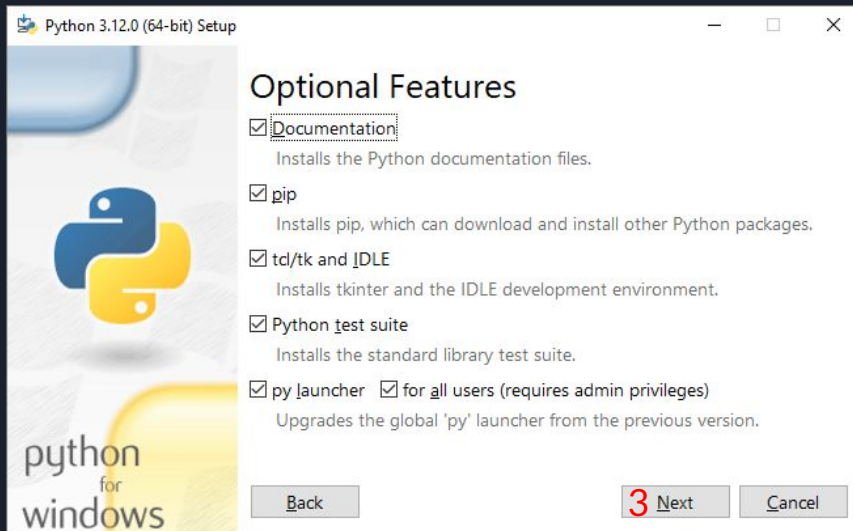
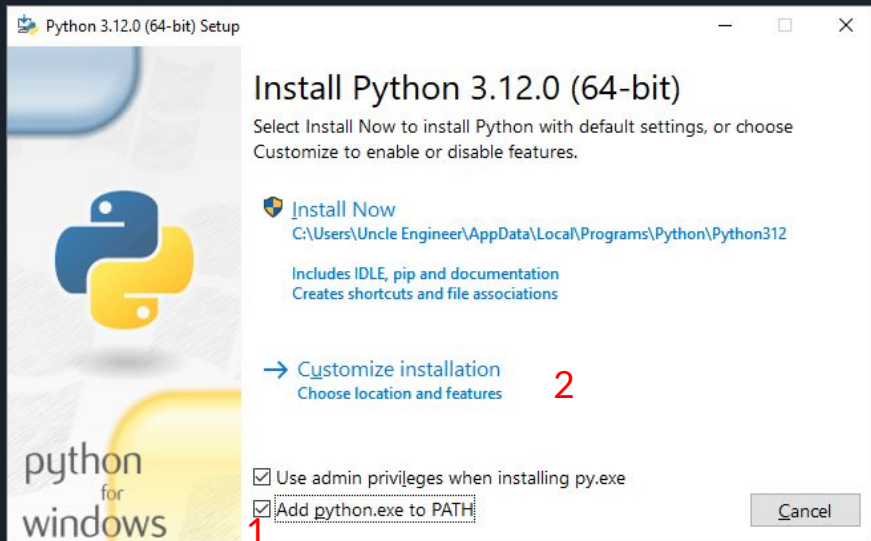
Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619

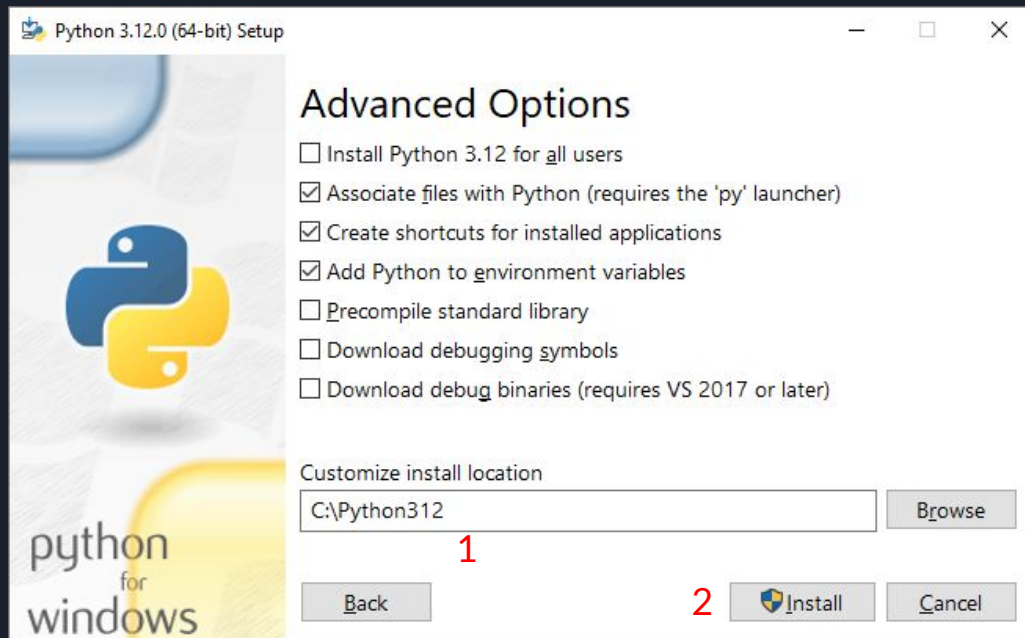
การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่3: เลือก Add Python.exe to PATH แล้ว Customize installation



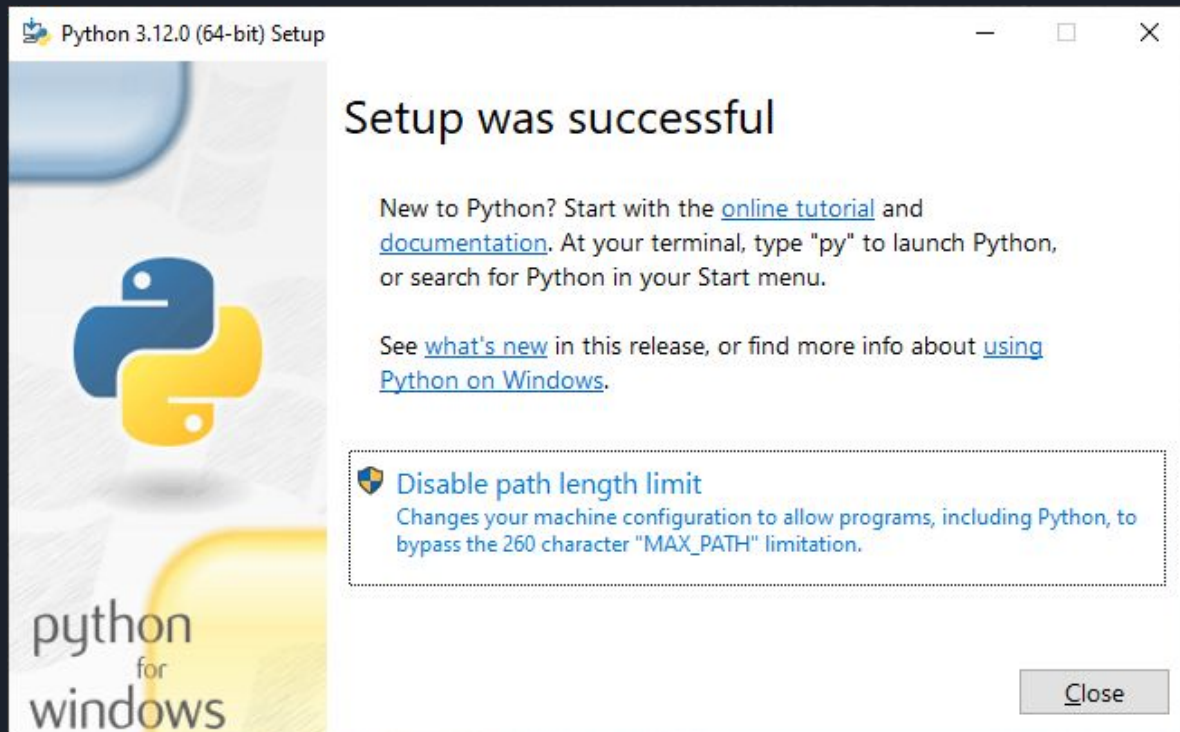
การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่4: แก้ไขที่อยู่โปรแกรมเป็น C:\Python312



การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่ 5 : เรียบร้อย Close ได้เลย



การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

โปรแกรมที่นิยมใช้ คือ Visual Studio Code หรือ PyCharm





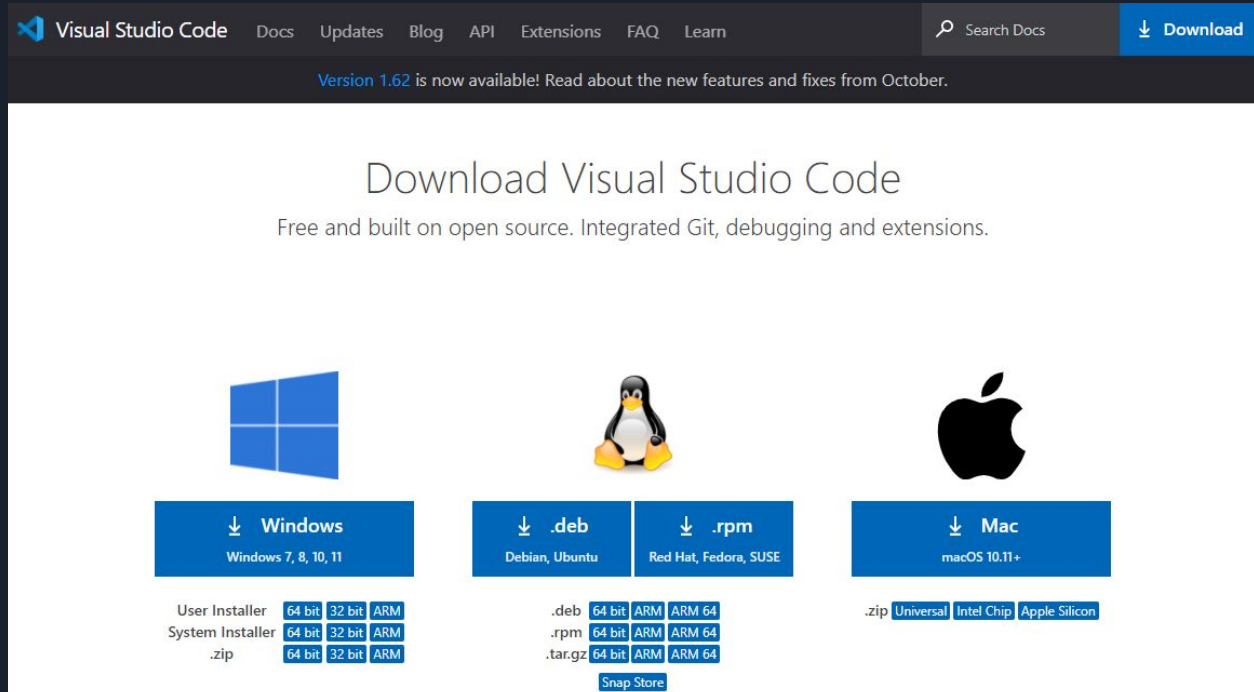
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

Visual Studio Code คือ IDE (Integrated Development Environment)
พัฒนาโดยบริษัท Microsoft



การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

1. <https://code.visualstudio.com/Download> เลือกดาวน์โหลดไฟล์ติดตั้งตาม OS ที่ใช้งาน (สำหรับ Windows ให้เลือก System Installer 64 Bit)




The screenshot shows the Visual Studio Code download page. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, and Learn. A search bar for 'Search Docs' and a 'Download' button are also present. Below the navigation bar, a message states 'Version 1.62 is now available! Read about the new features and fixes from October.' The main heading is 'Download Visual Studio Code', followed by the tagline 'Free and built on open source. Integrated Git, debugging and extensions.' The page is divided into three main sections for different operating systems: Windows, Linux, and Mac. Each section has a logo (Windows logo, Tux penguin, and Apple logo respectively) and a blue button with a download icon and the OS name. Below these buttons, specific download options are listed with their respective bit architectures.

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Search Docs Download

Version 1.62 is now available! Read about the new features and fixes from October.


Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows
Windows 7, 8, 10, 11

User Installer	64 bit	32 bit	ARM
System Installer	64 bit	32 bit	ARM
.zip	64 bit	32 bit	ARM




↓ .deb
Debian, Ubuntu

↓ .rpm
Red Hat, Fedora, SUSE

.deb	64 bit	ARM	ARM 64
.rpm	64 bit	ARM	ARM 64
.tar.gz	64 bit	ARM	ARM 64

Snap Store

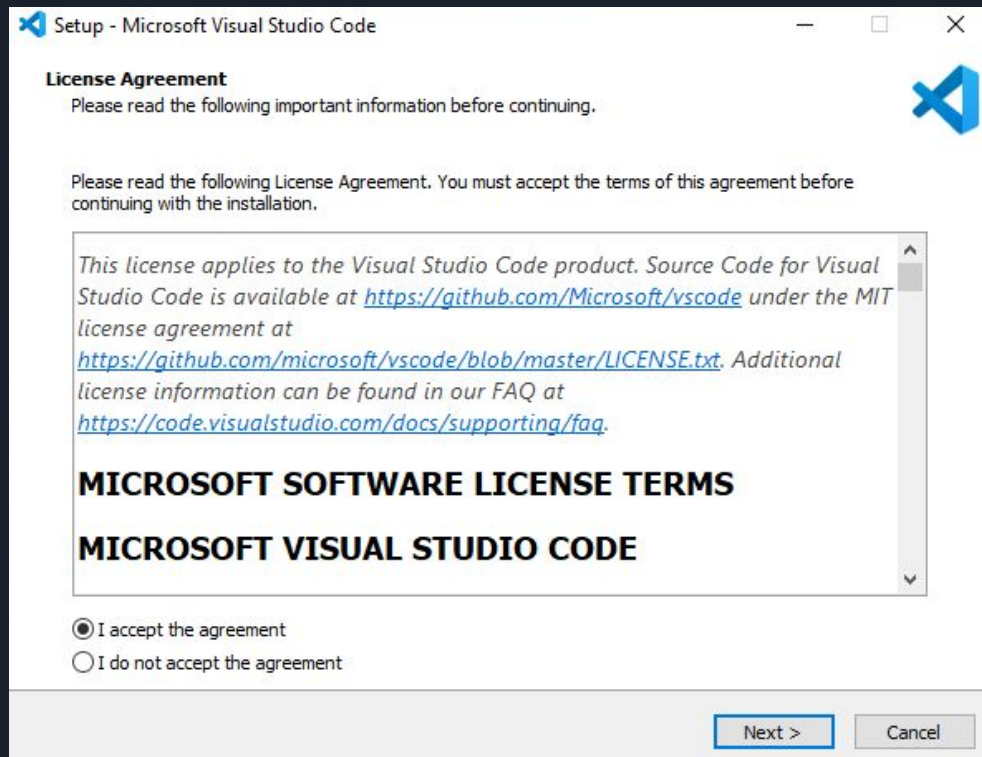


↓ Mac
macOS 10.11+

.zip Universal Intel Chip Apple Silicon

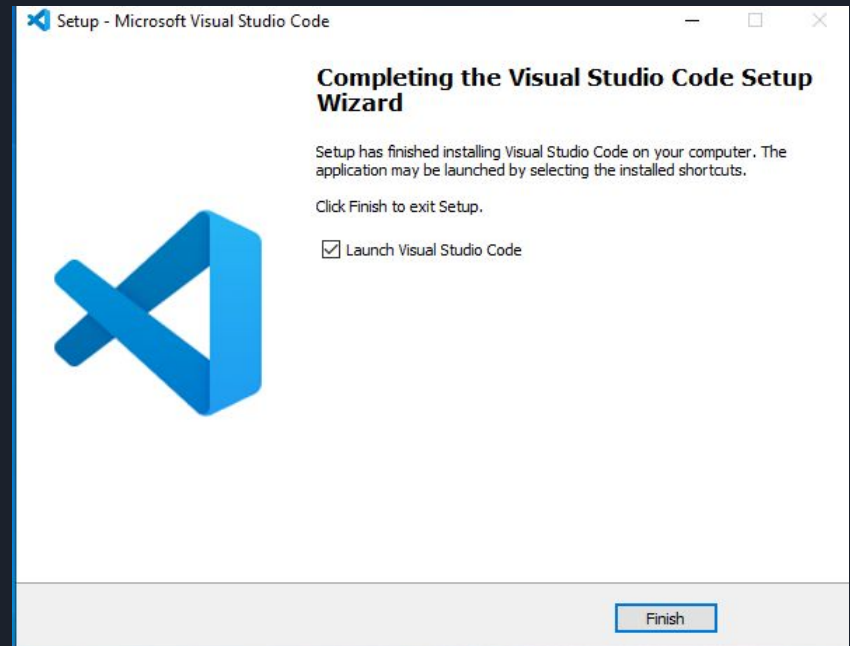
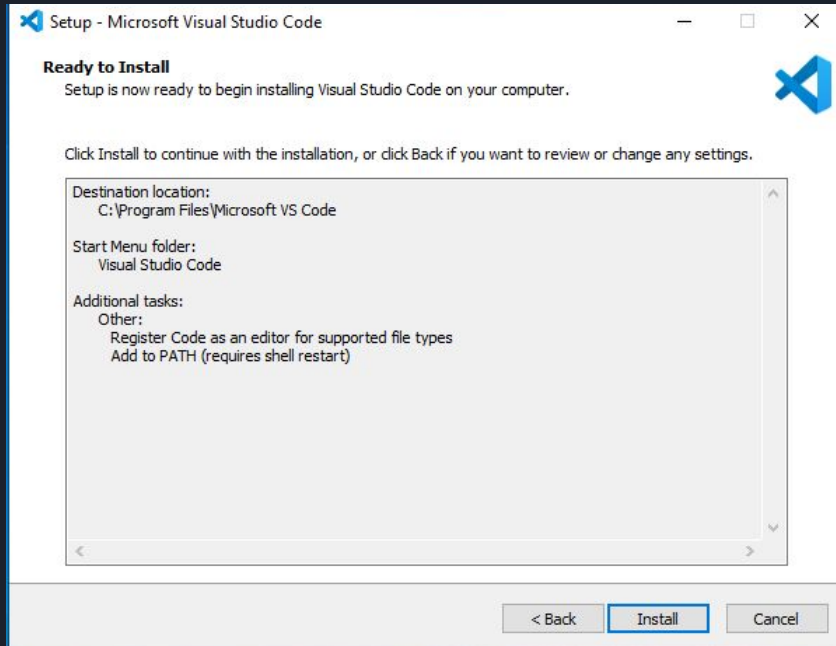
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

2. คลิกเลือก I accept the agreement กด Next >



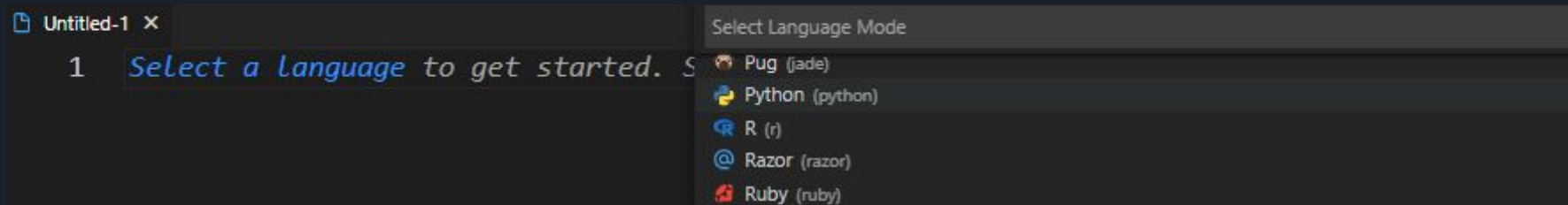
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

3. กดปุ่ม Next > อีก 3 ครั้ง แล้วกด Install เมื่อติดตั้งเสร็จกดปุ่ม Finish



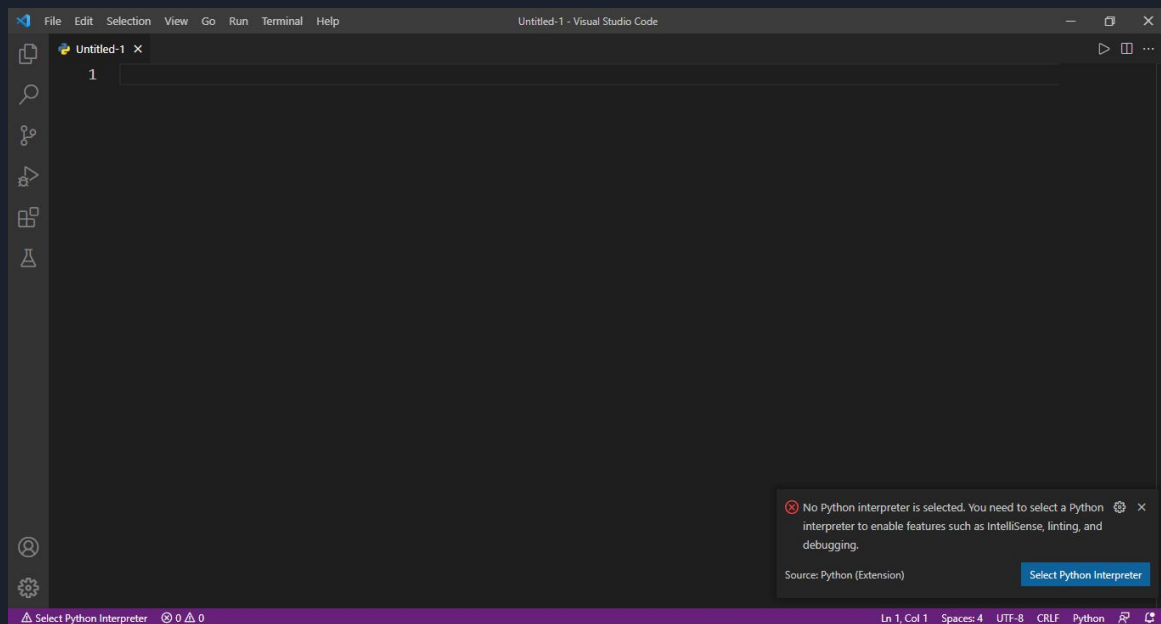
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

4. เปิดโปรแกรม VS Code สร้างไฟล์ใหม่ คลิกที่คำว่า Select a language
เลือก Python



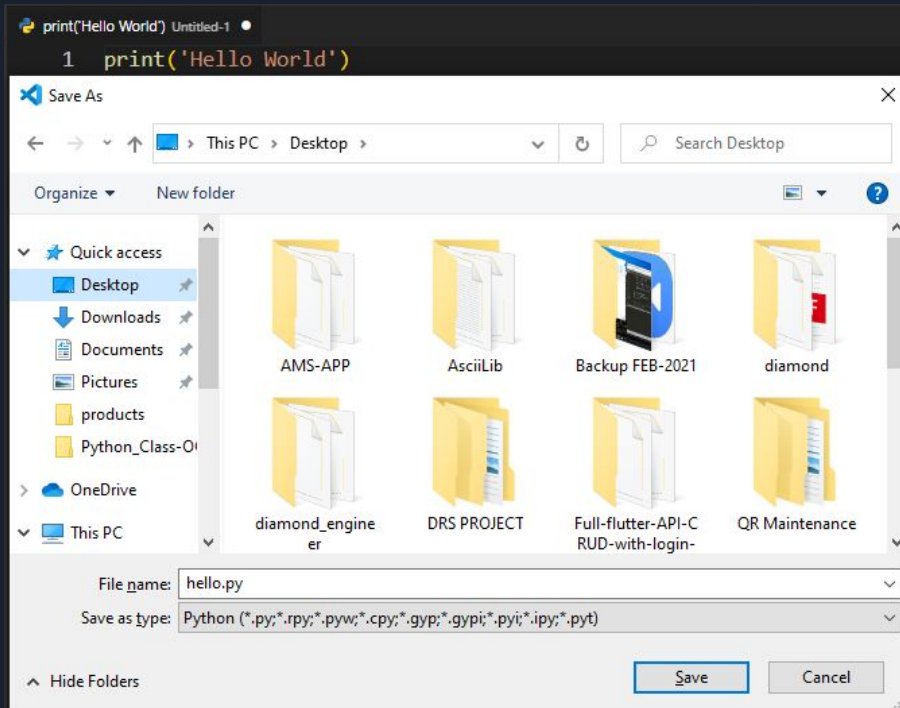
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

5. การติดตั้งในครั้งแรก จะต้องเลือก Interpreter หรือตัวแปลภาษา Python (อาจมีการติดตั้งมากกว่า 1 เวอร์ชัน) ให้คลิกปุ่ม Select Python Interpreter



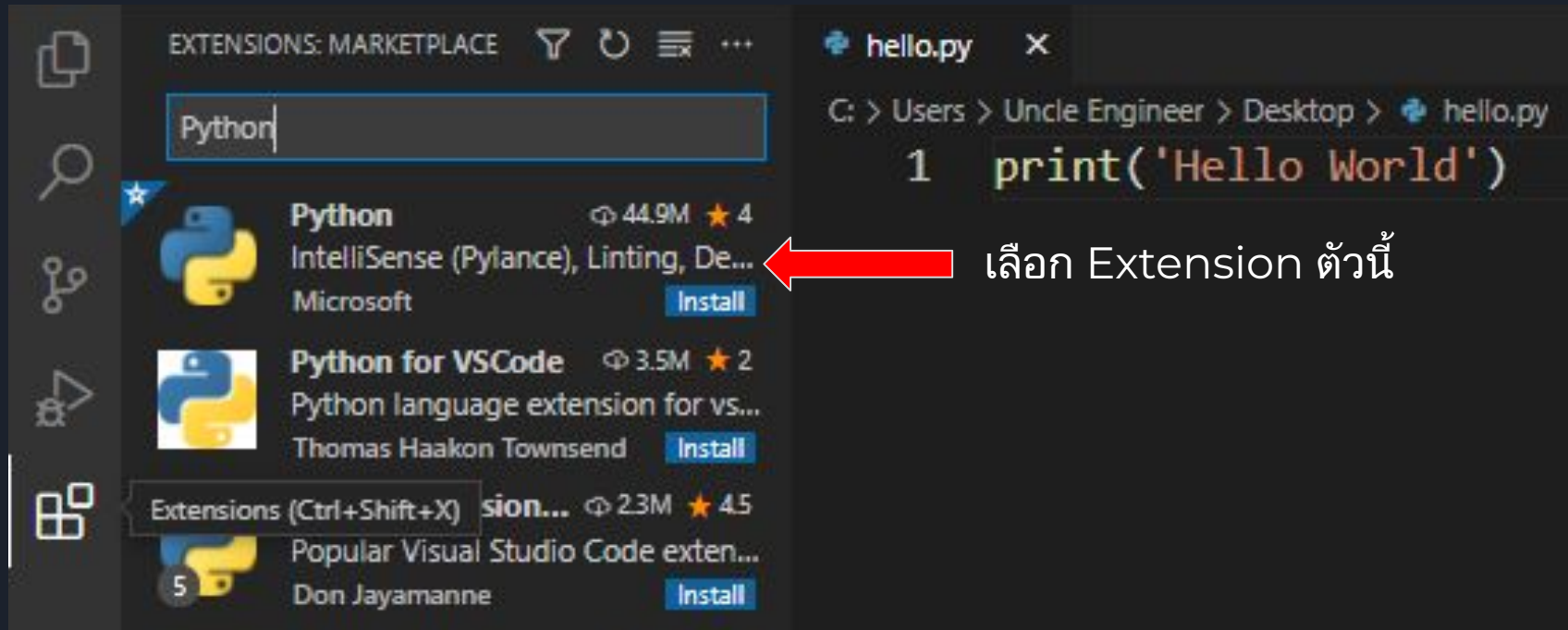
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

6. ทดลองเขียนโค้ด Python แล้วเซฟไฟล์



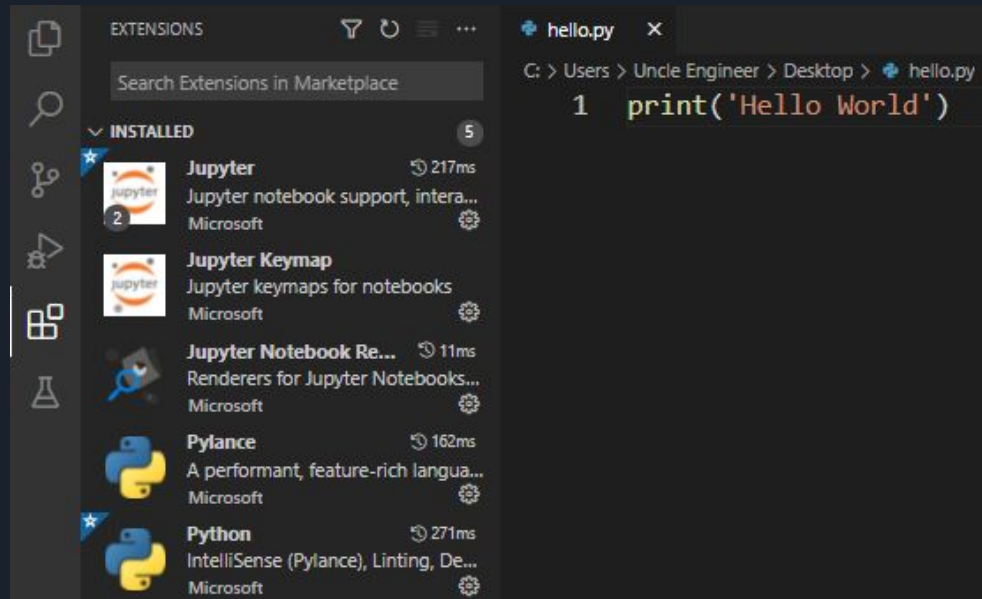
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

7. คลิกที่เมนู Extensions พิมพ์ในช่องค้นหาว่า Python แล้วกดปุ่ม Install ที่ Extension ตัวบนสุด



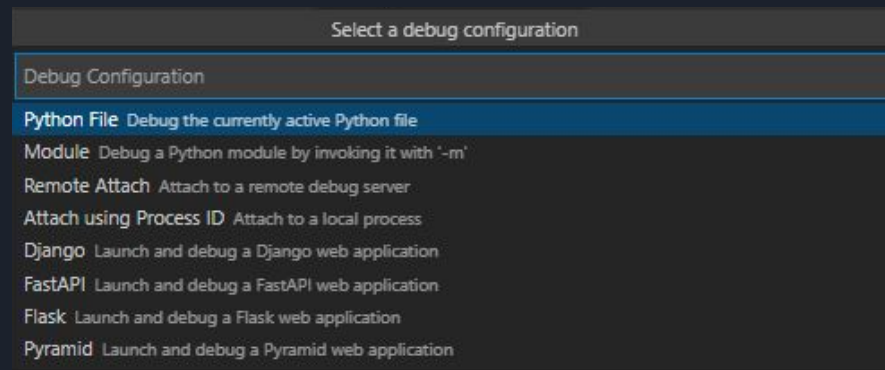
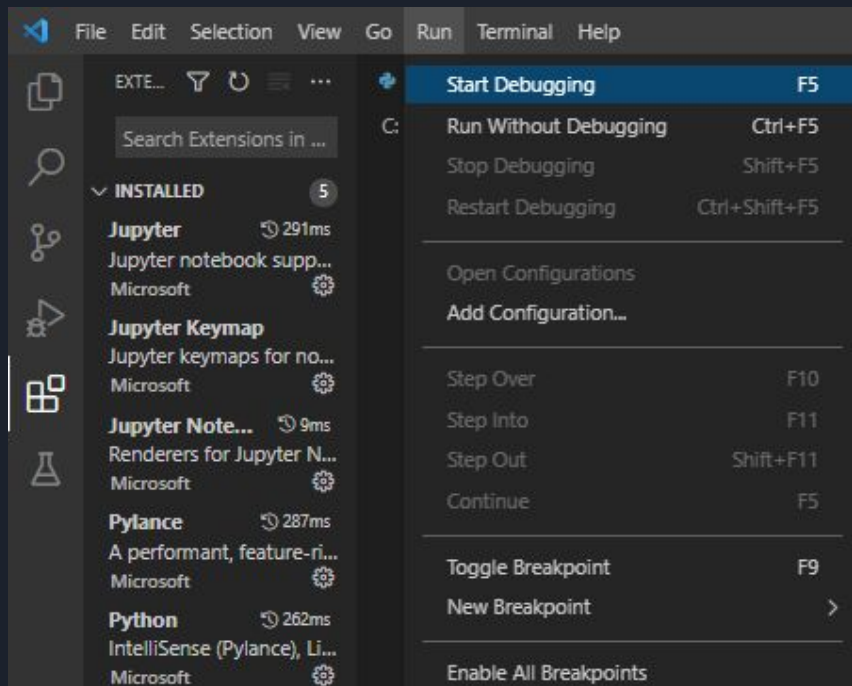
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

8. ในระหว่างที่ติดตั้ง Python extension ก็จะติดตั้ง extension เพิ่มอีก 4 ตัวให้โดยอัตโนมัติ คือ Jupyter, Jupyter Keymap, Jupyter Notebook Renderers และ Pylance



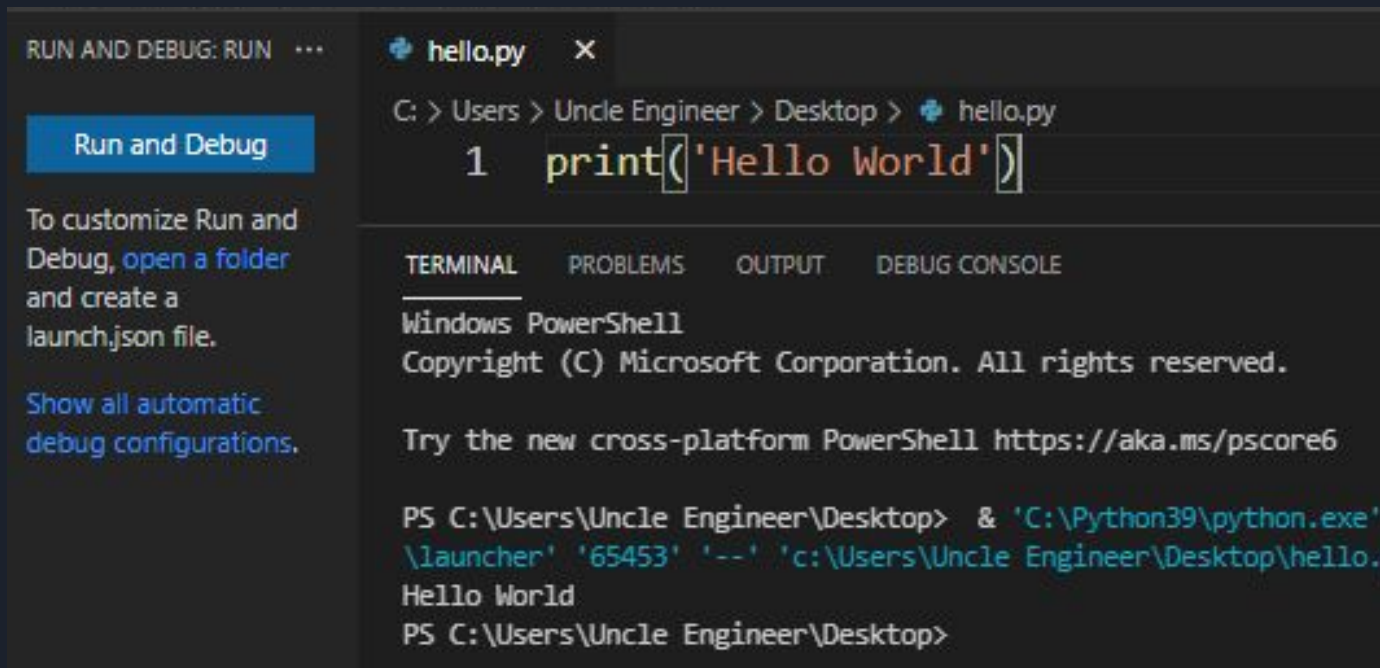
การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

9. รันโค้ด โดยคลิกเมนู Run > Start Debugging แล้วเลือก Python File



การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

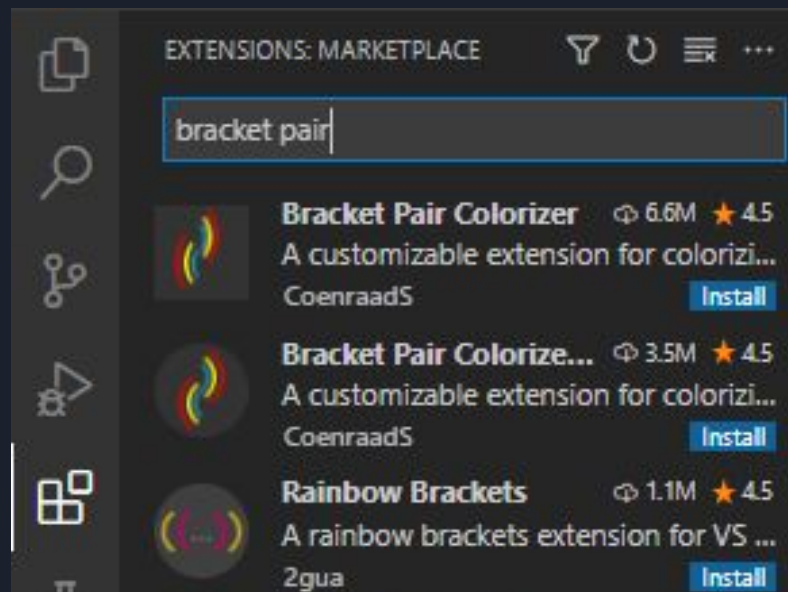
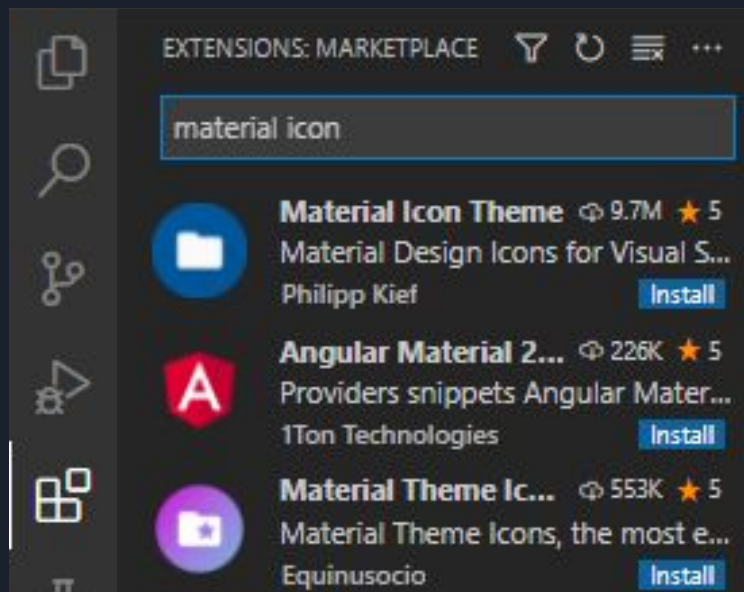
10. จะ print คำว่า Hello World ออกมา และในการรันครั้งต่อไป สามารถคลิกปุ่ม Run and Debug ได้เช่นกัน



การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน

11. ให้ติดตั้ง Extensions เพิ่มอีก 2 ตัว

- Material Icon Theme แสดงไอคอนที่สวยงาม
- Bracket Pair Colorizer จับคู่สีวงเล็บ เพื่อให้สังเกตได้ง่าย





ความรู้เบื้องต้นเกี่ยวกับการเขียนโปรแกรม

- print("Hello World!"), Variables (function)
- Data types, Operators
- String functions & method
- การเขียนคำสั่งเกี่ยวกับทางเลือก
- การเขียนคำสั่งทำซ้ำ (Loop)
- โครงสร้างข้อมูลแบบชุด (Collection)
- การสร้างและใช้งานฟังก์ชัน
- โมดูล/แพ็คเกจ (Modules/Packages)
- การเขียนและอ่านไฟล์
- การดักจับและตรวจสอบข้อผิดพลาด
- การเขียน Unit Test เบื้องต้น
- การจัดการข้อมูล



1. print("Hello World!"), Variables

```
print("Hello world!")
```

การแสดงความข้อความ

```
msg = "Hello world!"
```

การประกาศตัวแปร (Variable)

```
print(msg)
```

การแสดงค่าตัวแปร



2. Data types, Operators

- String

`str()`

ชนิดข้อมูลแบบ string คือ ข้อมูลที่เป็นตัวอักษร หรือข้อความ

`msg = 'This is Basic'`

`msg2 = "This is Python"`

โดย จะมีการใช้ เครื่องหมาย " หรือ " " เปิดไว้ก่อนข้อความ และปิดไว้ท้ายข้อความ เช่นตัวอย่างที่เป็นข้อความสี่เหลี่ยม

`print(type(msg))`

`print(type(msg2))`



- Numeric (Integers & floats)

Integers -> **int()**

จำนวนเต็ม **Ex.**

-11, 0, 800

Floats -> **float()**

จำนวนทศนิยม **Ex.**

-8.25, 0.7542, 78.0

int()

point = 10

print(type(point))

ชนิดข้อมูลแบบ integer คือ

ข้อมูลที่เป็นเลขจำนวนเต็ม

float()

point = 2.5

print(type(point))

ชนิดข้อมูลแบบ float คือ

ข้อมูลที่เป็นเลขจำนวนทศนิยม



- Boolean

```
status = True
```

```
isEmpty = False
```

```
print(type(status))
```

```
print(type(isEmpty))
```

ชนิดข้อมูลแบบ boolean คือ ข้อมูลที่บอกค่าความจริง มีเพียง 2 ค่าเท่านั้นคือ จริง (True) และเท็จ (False)



- Operators (Arithmetic Operators & Orders)

>>> 10+2

การบวก

12

>>> 10-5

การลบ

5

>>> 5*2

การคูณ

10



- Operators (Arithmetic Operators & Orders)

>>> 10/3

การหาร(หาค่าจำนวนจริง)

3.3333333333333335

>>> 10//3

การหาร(หาค่าจำนวนเต็ม/ไม่เอาเศษ)

3

>>> 10%3

การหาร(หาค่าเศษ)

1



- Operators (Arithmetic Operators & Orders)

```
>>> 10**2
```

การยกกำลัง

100

โดยการเขียนโปรแกรม หรือการคำนวณเชิงวิศวกรรม ควรมีการใส่ลำดับด้วยการวงเล็บให้กับสมการเสมอ เช่น

$((20/2)**2)+(5+(8/2))) * 2$ จะเป็น 218

- Operators (Arithmetic Operators & Orders)

ตารางลำดับการคำนวณ

ลำดับแรก	การคูณ(*) และการหาร(/) จากซ้ายไปขวา
ลำดับสอง	การบวก(+) และการลบ(-) จากซ้ายไปขวา

เช่น จากโจทย์นี้

$2**3*3-6/2*1+1-2*3$ ผลลัพธ์ จะเป็น 16.0

- Operators (Assignment)

เครื่องหมาย	ความหมาย	การใช้
=	กำหนดค่า	x = 5
+=	x = x + y	x += 5
-=	x = x - y	x -= 5
*=	x = x * y	x *= 5
/=	x = x / y	x /= 5
:=	x = 3 print(x)	print(x:=3)



- Operators (Comparision)

เครื่องหมาย	ความหมาย	การใช้
==	เท่ากัน	$x == y$
!=	ไม่เท่ากัน	$x != y$
>	มากกว่า	$x > y$
>=	มากกว่าหรือเท่ากัน	$x >= y$
<	น้อยกว่า	$x < y$
<=	น้อยกว่าหรือเท่ากัน	$x <= y$



- Operators (Logical)

เครื่องหมาย	ความหมาย	การใช้
and	และ	$x > 5$ and $x < 10$
or	หรือ	$x > 5$ or $x > 10$
not	นิเสธ (ให้ค่าตรงกันข้าม)	$\text{not}(x > 5 \text{ and } x < 10)$



- Operators (Membership)

เครื่องหมาย	ความหมาย	การใช้	ผลลัพธ์
in	อยู่ใน	1 in [1,2,3]	True
not in	ไม่อยู่ใน	4 not in [2,3,4]	False

- Operators (Identity)

เครื่องหมาย	ความหมาย	การใช้	ผลลัพธ์
is	เหมือนกัน	4 is 4.00	False
is not	ไม่เหมือนกัน	4 is not '4'	True



3. String functions & method

- String Slice
- len string
- .format()
- .lower()
- .upper()



String Slice

```
fullname = 'Somchai Rukchard'
```

การเรียกตำแหน่งของข้อมูลแบบ string

```
name = fullname[:7]
```

ใช้การใส่ index ได้ตามรูปแบบนี้

```
lastname = fullname[-8:]
```

[_ตำแหน่งเริ่ม_:_ตำแหน่งที่จะหยุดแสดงค่า_]

```
print(name)
```

```
print(lastname)
```



Len String

```
msg = 'This is my message'
```

```
print(len(msg))
```

len เป็นคำสั่ง ที่ใช้ในการบอกจำนวนอักขระ
ในข้อความนั้น ๆ



.format()

name = 'Somchai'

age = '65'

```
print('My name is {}, I am {} yrs old'.format(name,age))  
print('My name is {0}, I am {1} yrs old'.format(name,age))  
print('My name is {n}, I am {a} yrs old'.format(n=name,a=age))  
print(f'My name is {name}, I am {age} yrs old')
```



.lower()

msg = 'HELLO MY NAME IS LINCOLN'

print(msg.lower())



.upper()

msg = 'this is my letter'

print(msg.upper())



4. การเขียนคำสั่งเกี่ยวกับทางเลือก

- if ... else และ if ... elif ... else
- match-case (เริ่มใช้ใน Python 3.10)



if ... else

```
name = 'Somsak'  
if name == 'Somsak':  
    print('Hello',name)  
else:  
    print('You are not Somsak')
```

การสร้างทางเลือกด้วยเงื่อนไข โดยตามโปรแกรมนี้ คือ การเทียบ

ข้อมูลในตัวแปร name หากเข้าตามเงื่อนไข จะมีการทำงานตาม

ขั้นตอนต่อไป หากไม่ตรง จะทำตาม else



if ... elif ... else

```
name = input('Please Enter Your Name: ')
if name == 'Prayut':
    print('Sawatdee Krub Lungtu ,Love you')
elif name == 'Prawit':
    print('Sawatdee Krub Lungpom ')
else:
    print('Who are you?')
```

การสร้างทางเลือกด้วยเงื่อนไข โดยตามโปรแกรมนี้ คือ
การเทียบข้อมูลในตัวแปร name หากเข้าตามเงื่อนไข จะมี
การทำงานตามขั้นตอนต่อไป หรือหากตัวแปรไม่ตรงตาม
if
จะมีการเช็คเงื่อนไข elif และทำตามเงื่อนไขต่อไป
หากไม่ตรง จะทำตาม else



match-case (เริ่มใช้ใน Python 3.10)

```
name = input('Please Enter Your Name: ')
match name:
    case 'Prayut':
        print('Sawatdee Krub Lungtu')
    case 'Prawit':
        print('Sawatdee Krub Lungpom')
    case _:
        print('Who are you?')
```

match-case ฟีเจอร์ใหม่ใน Python เริ่มตั้งแต่เวอร์ชัน 3.10

เป็นการสร้างทางเลือกด้วยเงื่อนไข เช่นเดียวกับ if-else

match-case ใน Python เทียบได้กับ switch-case ใน

ภาษา C, Java

นำตัวแปร name ไปเช็คค่า ตรงกับเงื่อนไขใด ก็จะทำ

คำสั่งหลัง case นั้น แต่ถ้าไม่ตรงกับเงื่อนไขใดเลย ก็จะทำ

คำสั่งหลัง case _:



5. การเขียนคำสั่งทำซ้ำ (Loop)

- For loops
- While loops
- Enumerating iterators
- Continue break and else



For loops

```
for number in range(10):
```

```
    print(number)
```

การทำลูป for แบบแสดงค่าในช่วง range 10 ค่า

```
for number2 in range(1,11):
```

```
    print(number2)
```

การทำลูป for แบบแสดงค่าในช่วง range 10 ค่า

โดยเริ่มจากค่า 1 - 10



While loops

```
while True:  
    print('this is WHILE')
```

```
name = 'Mr.A'  
while name == 'Mr.A':  
    print('Hello',name)
```

การทำ while loop คือการทำซ้ำตามเงื่อนไขที่มีค่าเป็นจริง (True) แล้วเมื่อไรที่เงื่อนไขเป็นจริงจะทำงานในลูปไปเรื่อยไม่รู้จบ



Enumerating iterators

```
names = ['Blue', 'Red', 'Pink']  
for number, name in enumerate(names):  
    print(f'{number} is {name}')  
for number, name in enumerate(names, 1):  
    print(f'{number} is {name}')
```

การลำดับค่า จะมีฟังก์ชัน `enumerate()`
ในการเพิ่มค่าการลำดับมาช่วย

โดยสามารถกำหนดค่าเริ่มต้นได้เป็น
พารามิเตอร์ ตัวที่ 2 หลังจากตัวแปร `names`



Continue break and else

```
while True:
    name = input('Enter Your name: ')
    if name == 'exit':
        break
    elif name == '':
        continue
    else:
        print('name')
```

การใช้คำสั่ง break เพื่อออกจากลูป while

Continue ใช้ย้อนการทำงานลูป while



6. โครงสร้างข้อมูลแบบชุด (Collection)

- List
- Dictionary
- Tuple
- Set



List

```
room = ['dog', 'cat', 'bird']
```

```
number = [-5,0,3,1.5]
```

```
print(type(room))
```

```
print(type(number))
```

```
number1 = list(range(100))
```

```
number2 = list(range(1,50))
```

ชนิดข้อมูลแบบ list คือ การรวบรวมข้อมูลไว้ในตัวแปรเดียว

โดยมีตำแหน่งของข้อมูลแต่ละตัว

การสร้าง list แบบช่วง อย่างง่าย

```
number1 = [0,1,2,...,99]
```

```
number2 = [1,2,3,...,50]
```



List

```
room = ['dog', 'cat', 'bird']
```

```
number = [-5, 0, 3, 1.5]
```

```
print(room[2])
```

```
print(number[-1])
```

การเรียกข้อมูลใน ข้อมูลแบบ list สามารถเรียก
โดยการอ้างตำแหน่ง (index)

-> 'bird'

-> 1.5



List

ฟังก์ชัน ที่ใช้กับ list

```
Box = ['A','B']  
Box.append('C')
```

การเพิ่มค่าใน list
โดยเพิ่มไปที่ตำแหน่งสุดท้าย (index -1)



List

ฟังก์ชัน ที่ใช้กับ list

```
Box = ['A','B','C']  
Box.insert(1,'D')
```

การแทรกค่าใน list
โดยการอ้างอิงถึง index ที่จะเอาค่าใหม่ไปแทน
แล้วตามด้วยค่าใหม่



List

ฟังก์ชัน ที่ใช้กับ list

```
Box = ['A','D','B','C']  
Box.remove('D')
```

การลบค่าใน list
โดยการใส่ค่าที่จะลบ



List

ฟังก์ชัน ที่ใช้กับ list

```
Box = ['A','B','C']
```

```
Box.pop(0)
```

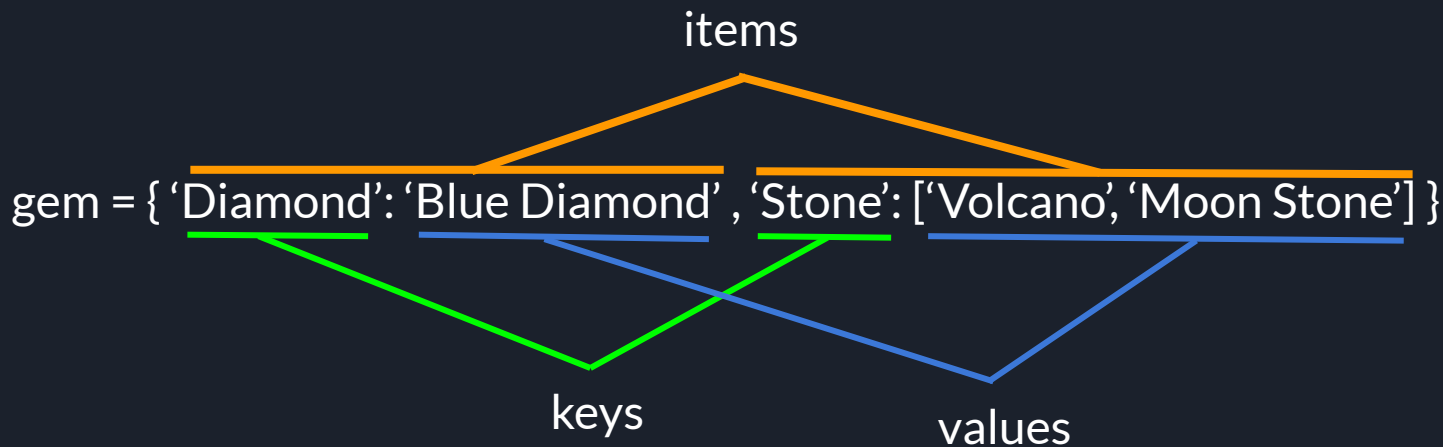
```
Box.pop()
```

การลบค่าใน list

โดยการใส่ค่า index ของค่าที่จะเอาออก
(ถ้าไม่ใส่ จะเอาค่าท้ายสุดออก)

Dictionary

เป็นตัวแปรที่มีรูปแบบในการจัดการข้อมูล โดยมี ข้อมูลที่ถูกเก็บไว้แบ่งเป็น 2 แบบหลัก เรียกว่า 'key' และ 'value' ตามตัวอย่างต่อไปนี้





Dictionary

การเรียกค่าใน dict() สามารถเรียกโดยการอ้างจากชื่อ keys

```
gem = { 'Diamond': 'Blue Diamond', 'Stone': ['Volcano', 'Moon Stone'] }
```

```
print(gem['Diamond'])
```

-> 'Blue Diamond'

```
print(gem['Stone'])
```

-> ['Volcano', 'Moon Stone']



Dictionary

การให้ค่าใหม่ และแก้ไขค่าใน dict()

```
gem = { 'Diamond': 'Blue Diamond', 'Stone': ['Volcano', 'Moon Stone'] }
```

```
gem['Ruby'] = ['Pastel', 'Royal']
```

```
print(gem)
```

```
gem['Diamond'] = 'Pink'
```

```
print(gem['Diamond'])
```



Tuple

location = (1500,750)

ค่าที่เก็บค่ามากกว่า 1 ค่า ไว้ในค่าเดียว
โดยค่าแต่ละค่าไม่สามารถแก้ไขได้



Set

เป็นค่าที่คล้ายกับ dict() แต่ มีเพียง key หรือค่าเพียงอย่างเดียว สร้างได้ดังนี้

```
animal = {'cat', 'dog', 'bird', 'pig'}
```

แต่ค่าของ set() จะไม่มีการลำดับ

```
print(animal)
```




Set

ฟังก์ชันที่ใช้กับ set

```
animal.add('fish')  
print(animal)
```

การเพิ่มค่าใน set
โดยการใส่ค่าที่จะเพิ่ม



Set

ฟังก์ชันที่ใช้กับ set

```
animal.update(['tiger', 'Owl'])  
print(animal)
```

การเพิ่มค่าใน set หลายค่า
โดยการใส่ค่าที่จะเพิ่ม



Set

ฟังก์ชันที่ใช้กับ set

```
animal.remove('tiger')  
print(animal)  
animal.discard('tiger')  
print(animal)
```

การลบค่าใน set
โดยการใส่ค่าที่จะลบ
discard จะไม่มี error หากค่าลบไปแล้ว



7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def first_function():  
    """Display a simple greeting."""  
    print("Hello! My name is Somchai")  
  
first_function()
```

ตัวอย่างการประกาศฟังก์ชัน
พื้นฐาน (แบบไม่มีพารามิเตอร์)



7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def second_function(name):  
    """Display a simple greeting."""  
    print("Hello! My name is " + name)
```

```
second_function("Somchai")  
second_function("Somsak")
```

ตัวอย่างการประกาศฟังก์ชัน
พื้นฐาน (แบบมีพารามิเตอร์ 1
ตัว)



7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def third_function(name, age):  
    """Display information."""  
    print("Hello! My name is " + name)  
    print("I am " + str(age) + " years old")  
  
third_function("Somchai", 80)
```

ตัวอย่างการประกาศฟังก์ชันพื้นฐาน
(แบบมีพารามิเตอร์มากกว่า 1 ตัว)



7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def fourth_function(name, age):  
    """Display information."""  
    print("Hello! My name is " + name)  
    print("I am " + str(age) + " years old")
```

```
fourth_function(name="Somchai", age=80)  
fourth_function(age=100, name="Somsak")
```

ตัวอย่างการประกาศฟังก์ชันพื้นฐาน
สามารถสลับตำแหน่ง และกำหนดค่า
ในพารามิเตอร์ เวลาเรียกใช้งานได้

7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def fifth_function(name, age=80):  
    """Display information."""  
    print("Hello! My name is " + name)  
    print("I am " + str(age) + " years old")
```

```
fifth_function("Somchai")  
fifth_function("Somsak", 100)
```

ตัวอย่างการประกาศฟังก์ชันพื้นฐาน
Optional Parameter สามารถ
กำหนดค่า Default ภายในพารามิเตอร์ได้

7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def sixth_function(name, age=None):  
    """Display information."""  
    print("Hello! My name is " + name)  
    if age:  
        print("I am " + str(age) + " years old")
```

```
sixth_function("Somchai")  
sixth_function("Somsak", 100)
```

ตัวอย่างการประกาศฟังก์ชันพื้นฐาน
None Parameter ถ้าพารามิเตอร์ตัวใด
ถูกกำหนดค่าเป็น None และไม่มีการ
เรียกใช้พารามิเตอร์ตัวนั้น จะไม่ print
ค่าออกมา



7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def first_return():  
    """Display a simple greeting."""  
    return "Hello! My name is Somchai"
```

```
hello = first_return()  
print(hello)
```

ตัวอย่างการประกาศฟังก์ชันพื้นฐาน
ไม่มีพารามิเตอร์ และมีการคืนค่า
กลับไป



7. การสร้างและใช้งานฟังก์ชัน (Function)

```
def second_return(name):  
    """Display a simple greeting."""  
    return "Hello! My name is " + name
```

```
hello = second_return("Somchai")  
print(hello)
```

ตัวอย่างการประกาศฟังก์ชันพื้นฐาน
มีพารามิเตอร์ และมีการคืนค่ากลับ
ไป



8. การใช้งานโมดูล/แพ็คเกจ (Modules/Packages)

- Python Standard Library
- Python Package
- Module



8. การสร้างและใช้งานโมดูล (Modules)

Python Standard Library

เป็นไลบรารีมาตรฐานที่ Python กำหนดมาให้
สามารถ import ไปใช้งานได้ทันที ไม่ต้อง install

<https://docs.python.org/3/library/>

8. การสร้างและใช้งานโมดูล (Modules)

• Python Standard Library

```
import random
import time
import webbrowser as web
```

```
for i in range(1, 7):
    number = random.randint(0, 9)
    time.sleep(1)
    print(number)
```

```
time.sleep(5)
url = 'http://www.glo.or.th/'
web.open(url)
```

ตัวอย่างการใช้งานไลบรารี
มาตรฐาน
random, time และ webbrowser



8. การสร้างและใช้งานโมดูล (Modules)

Python Package

เป็นแหล่งรวบรวมไลบรารีสำหรับ Python การใช้งานต้อง
ใช้คำสั่ง `pip install` สามารถดูได้ที่ Python Package Index

<https://pypi.org/>



8. การสร้างและใช้งานโมดูล (Modules)

- Python Package

`pip install pyautogui`

เปิด cmd / terminal
ติดตั้ง package ชื่อ pyautogui

8. การสร้างและใช้งานโมดูล (Modules)

• Python Package

```
import webbrowser as web  
import time
```

```
import pyautogui as pg
```

```
url = 'https://www.google.com/'  
web.open(url)  
time.sleep(2)  
pg.write('thailand', interval=0.25)  
pg.press('enter')  
time.sleep(2)  
pg.screenshot('thailand.png')
```

ตัวอย่างการใช้งานไลบรารี
pyautogui



8. การสร้างและใช้งานโมดูล (Modules)

Module

คือกลุ่มของตัวแปร ฟังก์ชัน หรือคลาส ที่อยู่ในไฟล์เดียวกัน

1. Standard module
2. Custom module



8. การสร้างและใช้งานโมดูล (Modules)

• Module

```
import math  
from random import randint
```

```
radius = randint(1, 9)  
area = math.pi * radius ** 2  
print(radius)  
print(area)
```

ตัวอย่างการใช้งานโมดูลมาตรฐาน



8. การสร้างและใช้งานโมดูล (Modules)

• Modules : fullname.py

```
def get_fullname(first, last):  
    """Display a simple greeting."""  
    full_name = f"{first} {last}"  
    return full_name.title()
```

ตัวอย่างการใช้งานโมดูล
ที่กำหนดเอง (ไฟล์ที่ 1)

.title() คือสั่งให้ขึ้นต้นด้วยตัว
อักษรพิมพ์ใหญ่



8. การสร้างและใช้งานโมดูล (Modules)

• Module : import_fullname.py

```
from fullname import get_fullname
```

```
person = get_fullname("uncle", "engineer")  
print(person)
```

ตัวอย่างการใช้งานโมดูล
ที่กำหนดเอง (ไฟล์ที่ 2)



9. การเขียนและอ่านไฟล์

- Text
- CSV (Comma-Separated Values)
- JSON (JavaScript Object Notation)
- XML
- Word
- Excel



9. การเขียนและอ่านไฟล์

- Text (Write File)

```
with open("testtext.txt", "w") as f:  
    f.write("Hello World")
```

ตัวอย่างการเขียนลงบนไฟล์ txt
เปล่า

9. การเขียนและอ่านไฟล์

- Text (Write File)

```
file_name = "testtext.txt"
```

```
with open("testtext.txt", "a") as f:  
    f.write("\n")  
    f.write("My Name is Uncle Engineer.\n")  
    f.write("I love Python!")
```

ตัวอย่างการเขียนไฟล์ csv เพิ่ม
จากของเดิม



9. การเขียนและอ่านไฟล์

- Text (Read File)

```
with open("testtext.txt") as f:  
    contents = f.read()
```

```
print(contents)
```

ตัวอย่างการอ่านข้อมูลในไฟล์ txt



9. การเขียนและอ่านไฟล์

- CSV (Write File)

```
import csv
```

```
with open("testtext.csv", "w", newline="") as f:  
    data = csv.writer(f)  
    data.writerow("Uncle", "Engineer", 50)  
    data.writerow("Somchai", "Sailom", 75)  
    data.writerow("Robert", "Tingnongnoy", 100)
```

ตัวอย่างการเขียนลงบนไฟล์ csv
เปล่า



9. การเขียนและอ่านไฟล์

- CSV (Write File)

```
import csv
```

```
with open("testtext.csv", "a", newline="") as f:  
    data = csv.writer(f)  
    data.writerow("Somsak", "Somsri", 30)
```

ตัวอย่างการเขียนไฟล์ csv เพิ่ม
จากของเดิม



9. การเขียนและอ่านไฟล์

- CSV (Read File)

```
import csv

with open("testtext.csv") as f:
    read_csv = csv.reader(f, delimiter=",")
    for row in read_csv:
        print(row)
        # print(row[0], row[1], row[2])
```

ตัวอย่างการอ่านข้อมูลในไฟล์
csv



9. การเขียนและอ่านไฟล์

· JSON (Write File)

```
import json
```

```
dict_profile = {  
    'name': 'Uncle Engineer',  
    'phone': '0987654321'  
}
```

```
with open("testnumbers.json", "w") as f:  
    json.dump(dict_profile, f)
```

ตัวอย่างการเขียนลงบนไฟล์ json
เปล่า



9. การเขียนและอ่านไฟล์

- JSON (Read File)

```
import json
```

```
with open("testnumbers.json") as f:  
    data = json.load(f)
```

```
print(data)
```

ตัวอย่างการอ่านข้อมูลในไฟล์
json



9. การเขียนและอ่านไฟล์

· XML (Write File)

```
from lxml import etree
```

```
root = etree.Element("root")  
a = etree.Element("a")  
a.text = "1"  
root.append(a)  
tree = etree.ElementTree(root)  
tree.write("testxml.xml")
```

ตัวอย่างการเขียนไฟล์ xml



9. การเขียนและอ่านไฟล์

- XML (Read File)

```
from lxml import etree
```

```
tree = etree.parse("testxml.xml")  
print(etree.tostring(tree))
```

ตัวอย่างการอ่านข้อมูลในไฟล์
xml



9. การเขียนและอ่านไฟล์

- Word (Install package)

`pip install python-docx`

เปิด cmd / terminal
ติดตั้ง package ชื่อ python-docx



9. การเขียนและอ่านไฟล์

· Word (Write File)

```
from docx import Document
```

```
document = Document()  
document.add_heading('สวัสดี :)', 0)  
p = document.add_paragraph('Test Word .docx in  
Python')  
paragraph_format = p.paragraph_format  
p.style = 'Heading 2'  
document.add_paragraph('by Uncle Engineer')  
document.add_page_break()  
document.save('testword.docx')
```

ตัวอย่างการเขียนข้อมูลลงในไฟล์
docx



9. การเขียนและอ่านไฟล์

• Word (Read File)

```
import docx
```

```
document = docx.Document('testword.docx')
```

```
contents = [p.text for p in document.paragraphs]  
print(contents)
```

ตัวอย่างการอ่านข้อมูลลงในไฟล์
docx



9. การเขียนและอ่านไฟล์

- Excel (Install Package)

`pip install openpyxl`

เปิด cmd / terminal
ติดตั้ง package ชื่อ opexpyxl



9. การเขียนและอ่านไฟล์

· Excel (Write File)

```
from openpyxl import Workbook  
import datetime
```

```
work_book = Workbook()
```

```
work_sheet = work_book.active  
work_sheet.title = "Hello"  
work_sheet['A1'] = "UncleEngineer"  
work_sheet['B2'] = datetime.datetime.now()
```

```
work_book.save("testexcel.xlsx")
```

ตัวอย่างการเขียนข้อมูลลงในไฟล์
docx



9. การเขียนและอ่านไฟล์

• Excel (Read File)

```
from openpyxl import load_workbook
```

```
work_book = load_workbook(filename='testexcel.xlsx')
```

```
sheet_ranges = work_book["Hello"]
```

```
print(sheet_ranges['A1'].value)
```

ตัวอย่างการอ่านข้อมูลลงในไฟล์
xlsx



10. การดักจับและตรวจสอบข้อผิดพลาด

- Error Types
- Exceptions
- Bug
- Debugging

10. การดักจับและตรวจสอบข้อผิดพลาด

Exceptions (Error Type : FileNotFoundError)

```
file_name = "testnumber.json"
```

```
try:
```

```
    with open(file_name) as f:  
        lines = f.readlines()
```

```
except FileNotFoundError:
```

```
    msg = f"Cannot find file {file_name}"  
    print(msg)
```

try รันคำสั่งตามปกติ
except จะทำงานถ้ามี error

10. การดักจับและตรวจสอบข้อผิดพลาด

Exceptions (Error Type : ZeroDivisionError)

```
number = input("Divide by : ")

try:
    result = 10 / int(number)
except ZeroDivisionError:
    print("You can't divide by zero!")
else:
    print(result)
finally:
    print("This is the divide by number")
```

try รันคำสั่งตามปกติ
except จะทำงานถ้า error
else จะทำงานถ้าไม่ error
finally ทำงานอย่างแน่อน
ไม่ว่าจะมี error หรือไม่ก็ตาม



10. การดักจับและตรวจสอบข้อผิดพลาด

Exceptions (Error Type : ZeroDivisionError)

```
number = input("Divide by : ")
```

```
try:
```

```
    result = 10 / int(number)
```

```
except ZeroDivisionError:
```

```
    pass
```

```
else:
```

```
    print(result)
```

```
finally:
```

```
    print("This is the divide by number")
```

pass คือการข้ามการทำงานไปยังบล็อกต่อไป



11. การเขียน Unit Test เบื้องต้น

- Unit Testing Fundamental
- unittest



11. การเขียน Unit Test เบื้องต้น

fullname.py

```
def get_fullname(first, last):  
    """Display a simple greeting."""  
    full_name = f"{first} {last}"  
    return full_name.title()
```

ไฟล์ที่ 1 สร้างฟังก์ชัน



11. การเขียน Unit Test เบื้องต้น

import_fullname.py

```
from fullname import get_fullname
```

```
person = get_fullname("uncle", "engineer")  
print(person)
```

ไฟล์ที่ 2 เรียกชื่อไฟล์ที่ 1
และ import โมดูลในไฟล์ที่ 1



11. การเขียน Unit Test เบื้องต้น

test_fullname.py

```
import unittest
from fullname import get_fullname
```

ไฟล์ที่ 3 ใช้ unittest

```
class NamesTestCase(unittest.TestCase):
    def test_first_last(self):
        person = get_fullname("uncle", "engineer")
        self.assertEqual(person, "Uncle Engineer")
```

```
unittest.main()
```



12. การจัดการข้อมูล

- NumPy
- Matplotlib
- Pandas



12. การจัดการข้อมูล

- NumPy

`pip install numpy`

เปิด cmd / terminal
ติดตั้ง package ชื่อ numpy



12. การจัดการข้อมูล

• NumPy

```
import numpy as np
```

```
my_list = np.random.randint(1, 10, 10)
my_array = np.array(my_list)
print(my_array)
print(np.sum(my_array))
print(np.mean(my_array))
print(np.median(my_array))
print(np.max(my_array))
print(np.min(my_array))
print(np.std(my_array))
print(np.var(my_array))
```

ตัวอย่างการใช้งาน numpy
ด้านสถิติ (Statistics)



12. การจัดการข้อมูล

- Matplotlib

`pip install matplotlib`

เปิด cmd / terminal
ติดตั้ง package ชื่อ matplotlib

12. การจัดการข้อมูล

· Matplotlib

```
import matplotlib.pyplot as plt
```

```
x_values = [0, 1, 2, 3, 4, 5, 6]
```

```
y_squares = [0, 1, 4, 9, 16, 25, 36]
```

```
plt.title("Exponential Function", fontsize=20)
```

```
plt.xlabel('Values')
```

```
plt.ylabel('Square Values')
```

```
plt.axis([0, 10, 0, 50])
```

```
plt.plot(x_values, y_squares)
```

```
plt.scatter(x_values, y_squares, s=10)
```

```
plt.show()
```

ตัวอย่างการใช้งาน matplotlib
สำหรับพล็อตกราฟ



12. การจัดการข้อมูล

- Pandas

`pip install pandas`

เปิด cmd / terminal
ติดตั้ง package ชื่อ pandas



12. การจัดการข้อมูล

• Pandas

```
import pandas as pd
```

ตัวอย่างการใช้งาน pandas

```
dict_car = {  
    'Brand': ['Toyota', 'Honda', 'Suzuki'],  
    'Price': [250000, 500000, 1000000]  
}  
df = pd.DataFrame(dict_car)  
print(df)  
print(df[1:])  
print(df.iloc[[0],[0]])  
print(df.loc[[2], ['Price']])  
df.to_csv('car_dataframe.csv')
```