

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with faint, lighter blue diagonal stripes.

Basic Python (Updated 2022)

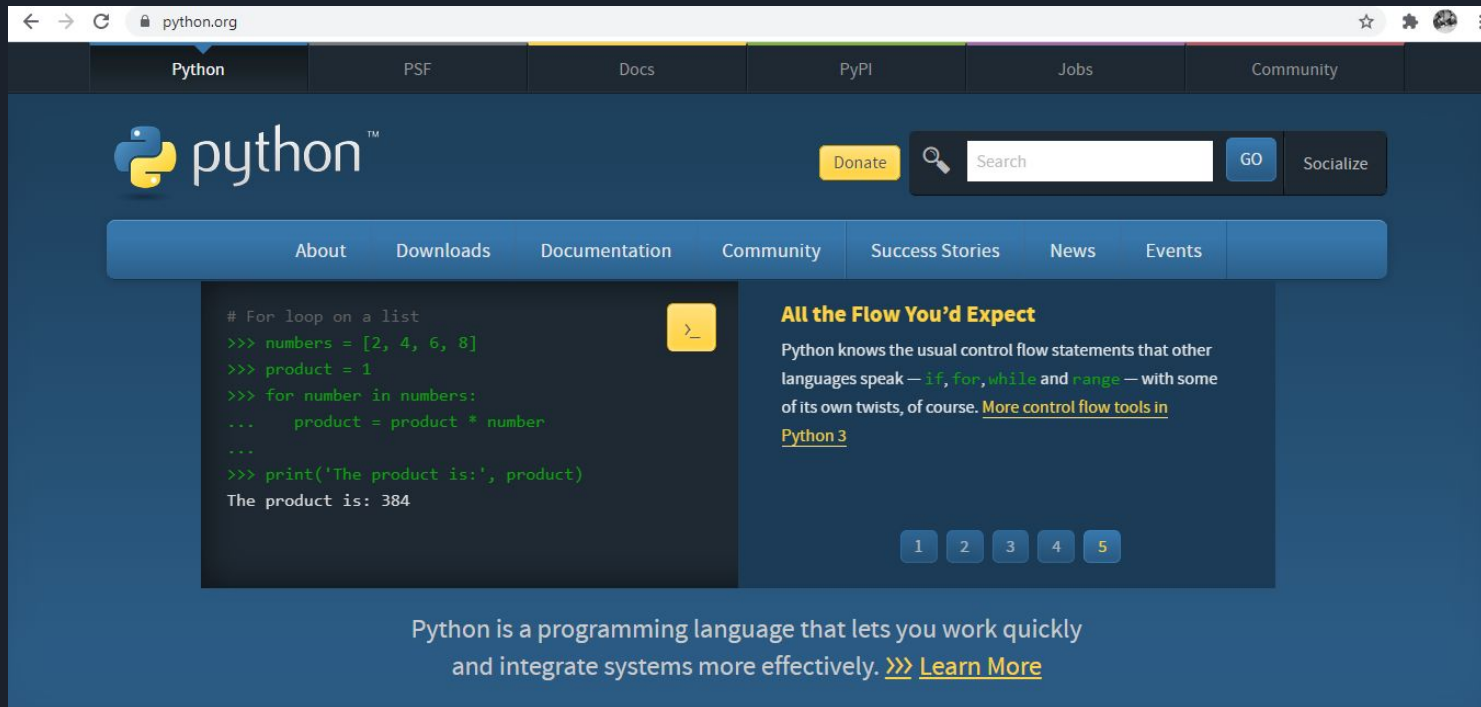


ความรู้เบื้องต้นเกี่ยวกับการเขียนโปรแกรม

- การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน
- การติดตั้งโปรแกรมสำหรับเขียน code เพื่อใช้งาน
- ความรู้เบื้องต้นเกี่ยวกับการเขียนโปรแกรม
- แนะนำภาษา Python

การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่1: เข้าเว็บ python.org



The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links: Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar with a 'GO' button. A secondary navigation bar contains links: About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a code snippet on the left and an article titled 'All the Flow You'd Expect' on the right. The code snippet demonstrates a for loop that calculates the product of numbers in a list. The article text explains that Python supports standard control flow statements like if, for, while, and range, along with some unique features. At the bottom, a footer states: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

All the Flow You'd Expect

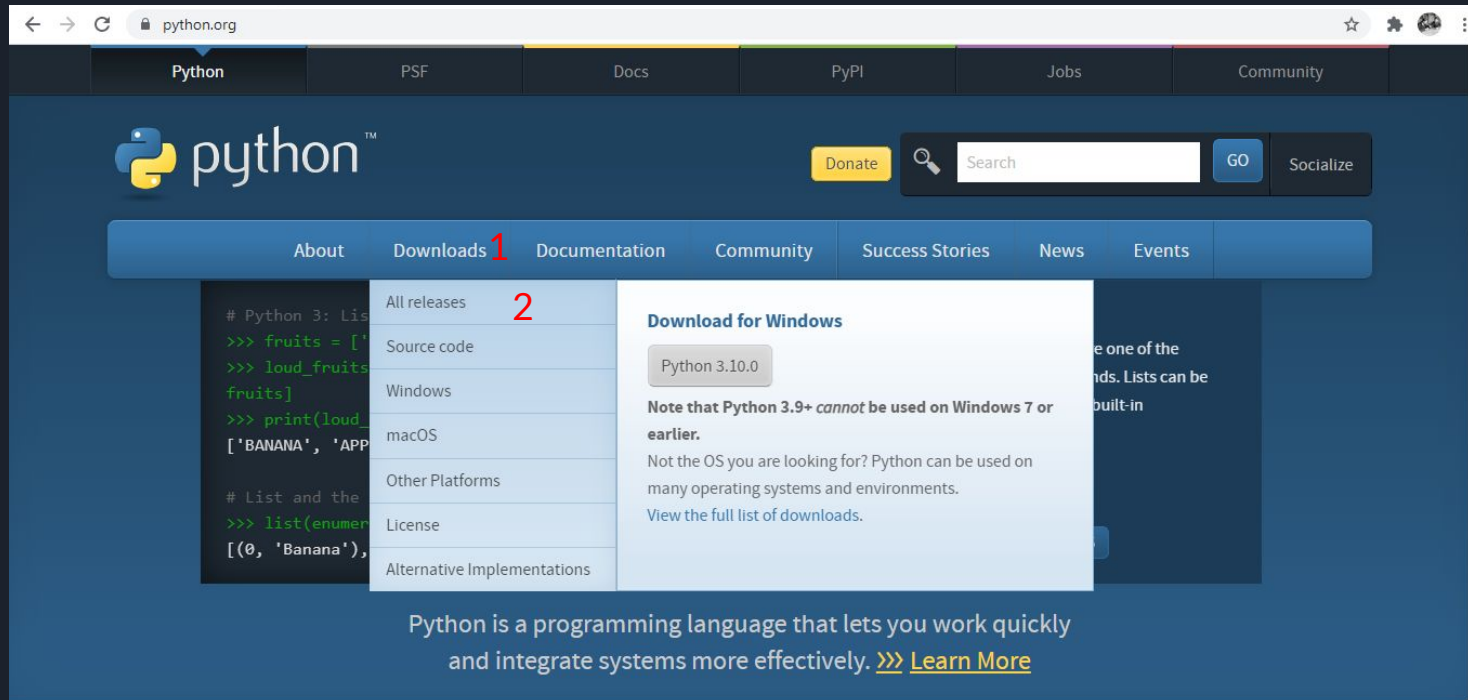
Python knows the usual control flow statements that other languages speak — `if`, `for`, `while` and `range` — with some of its own twists, of course. [More control flow tools in Python 3](#)

1 2 3 4 5

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

การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

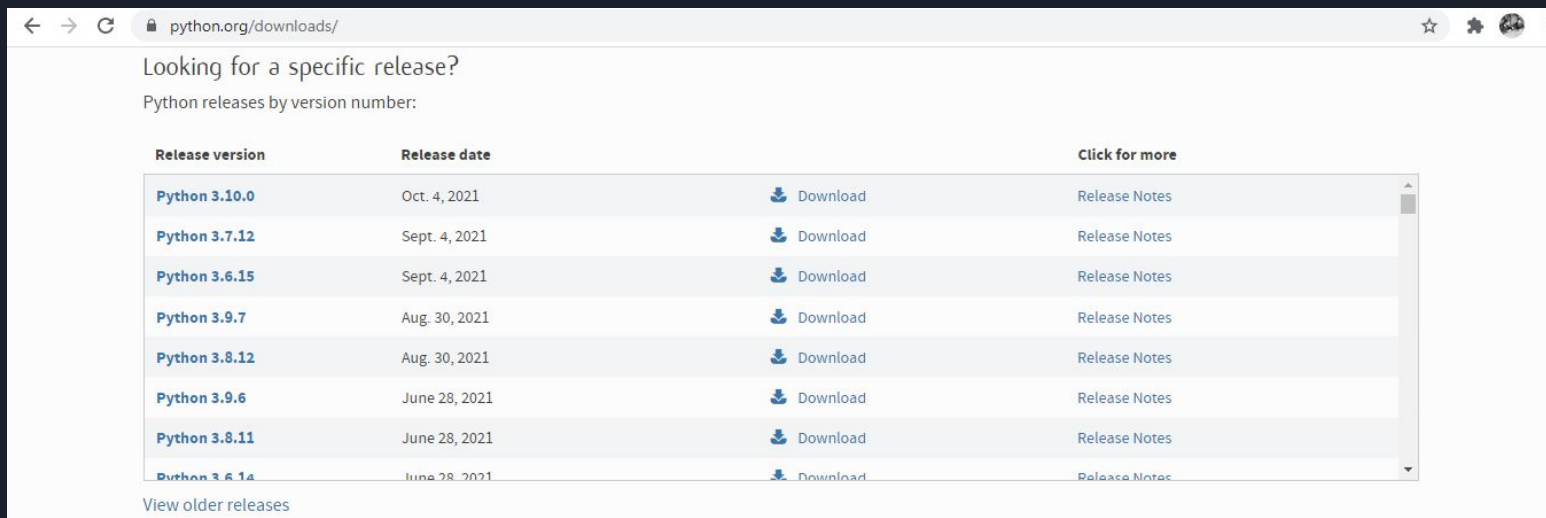
ขั้นตอนที่2: คลิก Downloads แล้วเลือก All releases



The screenshot shows the Python.org website. The navigation bar at the top includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a secondary navigation bar with links for About, Downloads (marked with a red '1'), Documentation, Community, Success Stories, News, and Events. The Downloads dropdown menu is open, showing options: All releases (marked with a red '2'), Source code, Windows, macOS, Other Platforms, License, and Alternative Implementations. The 'All releases' option is highlighted. To the right of the dropdown, there is a section titled 'Download for Windows' which includes a button for 'Python 3.10.0' and text stating that Python 3.9+ cannot be used on Windows 7 or earlier. At the bottom of the page, there is a footer text: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่3: คลิก Python 3.10.0



Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.10.0	Oct. 4, 2021	Download	Release Notes
Python 3.7.12	Sept. 4, 2021	Download	Release Notes
Python 3.6.15	Sept. 4, 2021	Download	Release Notes
Python 3.9.7	Aug. 30, 2021	Download	Release Notes
Python 3.8.12	Aug. 30, 2021	Download	Release Notes
Python 3.9.6	June 28, 2021	Download	Release Notes
Python 3.8.11	June 28, 2021	Download	Release Notes
Python 3.6.14	June 28, 2021	Download	Release Notes

[View older releases](#)

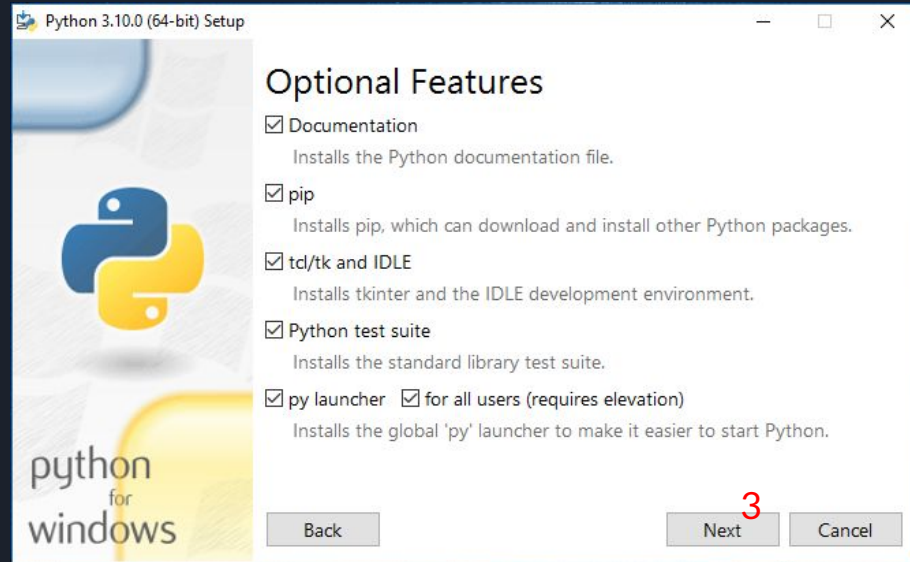
การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

ขั้นตอนที่4: เลือกตัวติดตั้งตามระบบปฏิบัติการ

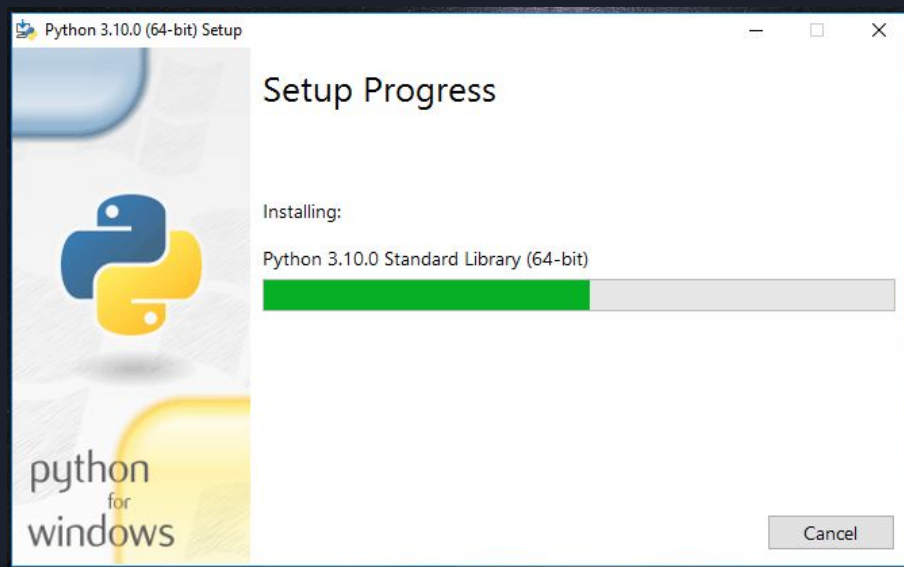
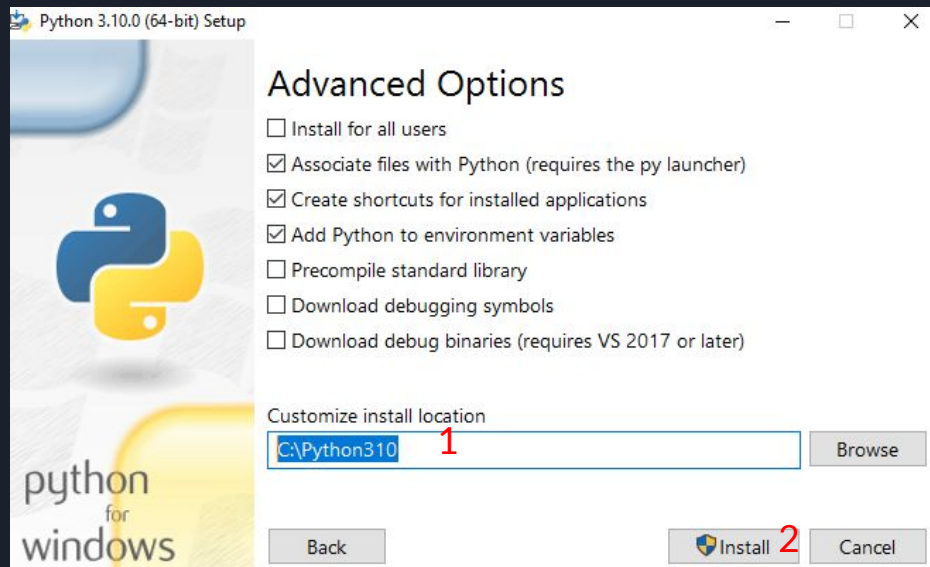
Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		729e36388ae9a832b01cf9138921b383	25007016	SIG
XZ compressed source tarball	Source release		3e7035d272680f80e3ce4e8eb492d580	18726176	SIG
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	d70b83e8af2260244e7723eb0b0f79ff	39741684	SIG
Windows embeddable package (32-bit)	Windows		dc9d1abc644dd78f5e48edae38c7bc6b	7521592	SIG
Windows embeddable package (64-bit)	Windows		340408540eff359d5eaf93139ab90fd	8474319	SIG
Windows help file	Windows		9d7b80c1c23cfb2cecd63ac4fac9766e	9559706	SIG
Windows installer (32-bit)	Windows		133aa48145032e341ad2a000cd3bff50	27194856	SIG
Windows installer (64-bit)	Windows	Recommended	c3917c08a7fe85db7203da6dcaa99a70	28315928	SIG

การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน

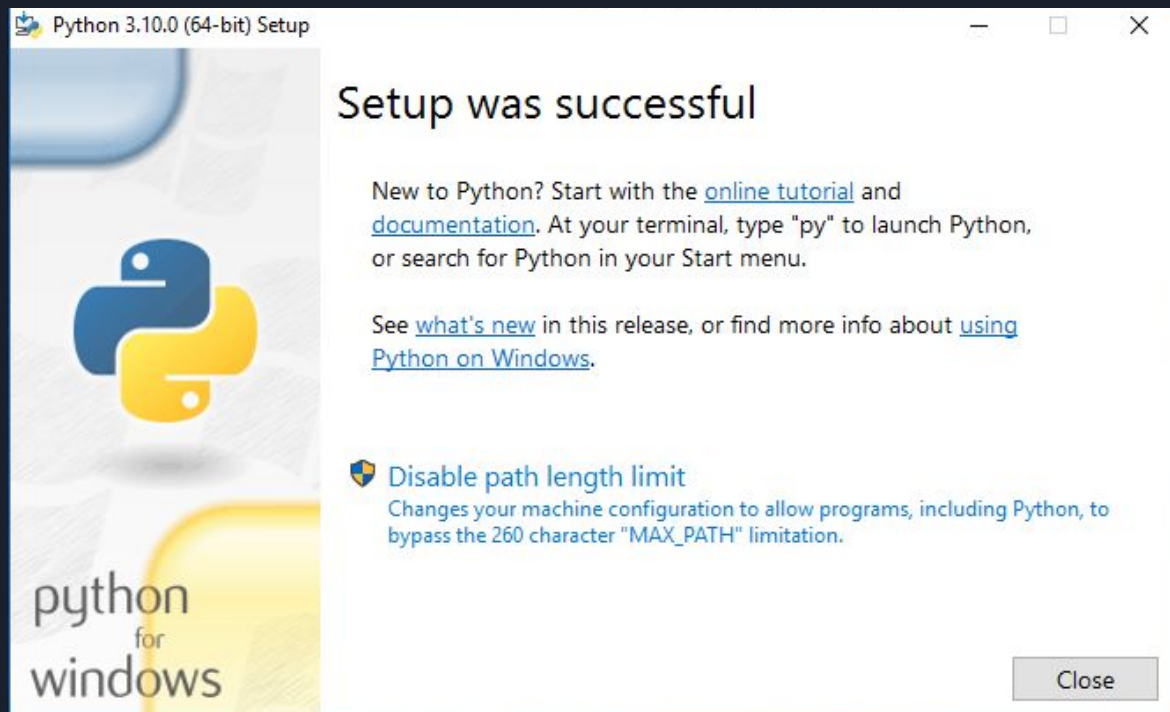
ขั้นตอนที่ 5: เลือก Add Python 3.10 to PATH แล้ว Customize installation



การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน ขั้นตอนที่6: แก้ไขที่อยู่โปรแกรม



การติดตั้งภาษาโปรแกรม Python เพื่อใช้งาน ขั้นตอนที่7: เรียบร้อย Close ได้เลย



Python คืออะไร?

ภาษาคอมพิวเตอร์ที่ใช้ในการเขียนโปรแกรม (Programming Language) คิดค้นโดย Guido van Rossum เริ่มเผยแพร่ให้ใช้ในปี ค.ศ. 1990

Python เป็นภาษาที่เปิดให้ใช้ฟรี (Open Source) สามารถใช้ได้ทั้งใน Windows, Mac , Linux



คำว่า Python มาจากไหน?

มาจากคณะตลกที่ชื่อ “Monty Python's Flying Circus”

ชาวบ้านรู้จักกันในนาม “Pythons”

Guido van Rossum ผู้คิดค้น เป็น
แฟนคลับคณะตลกวงนี้ ในขณะที่กำลัง
คิดค้นภาษาไพทอนอยู่นั้น คิดชื่อไม่ออก
เลยเอาชื่อนี้มาตั้งซะเลย “Python”

555





Python ทำอะไรได้บ้าง?

- เขียนโปรแกรมทั่วไป
- งาน Data Science
- คำนวณเชิงคณิตศาสตร์ สถิติ
- คำนวณเชิงวิศวกรรม
- เขียนเว็บไซต์
- Internet of Things
- 3D Modeling
- เขียนเกม
- งาน Network Engineering
- งานคำนวณเชิงฟิสิกส์
- Big Data, Data Analysis
- Machine Learning, AI
- Image Processing
- ฝึกเด็กๆเขียนโปรแกรม
- เขียนโปรแกรม Desktop , แอปมือถือ
- อื่นๆ อีกมากมาย



แนะนำภาษาไพธอน

ภาษาไพธอนเป็นภาษาระดับสูง (high level) โดยคำว่าระดับสูงหมายถึงเป็นภาษาที่เขียนได้ง่ายต่อความเข้าใจของมนุษย์ แล้วจึงแปลงเป็นภาษาระดับต่ำ (low level) หรือภาษาเครื่อง ให้คอมพิวเตอร์เข้าใจการทำงาน

โดยภาษาไพธอน ถูกสร้างขึ้นเมื่อปี พ.ศ. 2533 โดย Guido van Rossum โดย Guido ได้ให้ชื่อ ไพธอน ตามคณะตลกที่เขาชื่นชอบ

แนะนำภาษาไพธอน

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



WIKIPEDIA



facebook

amazon



หลักภาษาและไวยากรณ์ทั่วของภาษา Python

- Operators (Arithmetic Operators & Orders)
- Variables (Naming, Numeric (Integers & floats))
- Data types
- Expressions (Boolean)
- Statements (True 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 ควรมีการใส่ลำดับด้วยการวงเล็บให้กับ
สมการเสมอ 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



Variables (Naming)

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

การแสดงข้อความ

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

การประกาศตัวแปร (Variable)

```
print(msg)
```

การแสดงค่าตัวแปร



Numeric (Integers & floats)

Integers -> **int()**


จำนวนเต็ม

Ex. -11, 0, 800

Floats -> **float()**

จำนวนทศนิยม

Ex. -8.25, 0.7542, 78.0



Data types

`str()`

`msg = 'This is Basic'`


`msg2 = "This is Python"`

`print(type(msg))`

`print(type(msg2))`

ชนิดข้อมูลแบบ string คือ ข้อมูลที่เป็นตัวอักษร หรือข้อความ

โดย จะมีการใช้ เครื่องหมาย " หรือ " " เปิดไว้ก่อนข้อความ และปิดไว้ท้ายข้อความ เช่นตัวอย่างที่เป็นข้อความสี่เหลี่ยม



Data types

int()

```
point = 10
```

```
print(type(point))
```

ชนิดข้อมูลแบบ integer คือ

ข้อมูลที่เป็นเลขจำนวนเต็ม


float()

```
point = 2.5
```

```
print(type(point))
```

ชนิดข้อมูลแบบ float คือ

ข้อมูลที่เป็นเลขจำนวนทศนิยม



Data types

list()

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


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

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

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

ชนิดข้อมูลแบบ list คือ การรวบรวมข้อมูลไว้ในตัวแปรเดียว

โดยมีตำแหน่งของข้อมูลแต่ละตัว



Data types

list()

การเรียกข้อมูลใน ข้อมูลแบบ list สามารถเรียก

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

โดยการอ้างตำแหน่ง (index)


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

print(room[2])

-> 'bird'

print(number[-1])

-> 1.5



Data types


dict()

ชนิดข้อมูลแบบ dictionary คือ การรวบรวมข้อมูลไว้ในตัวแปรเดียว

โดยมีการอ้างถึง keys ในการเข้าถึง values ต่าง ๆ

```
box1 = {'color': 'green', 'size': 5}
```

```
print(type(box1))
```



Data types

`dict()`

การเรียกข้อมูลใน ข้อมูลแบบ dict สามารถเรียก
โดยการอ้างจาก keys

```
box = {'color': 'red', 'size': 5}
```

```
print(box['color'])
```

-> 'red'

```
print(box['size'])
```

-> 5

เครื่องหมาย ใช้ในการเทียบ

==	เท่ากัน
!=	ไม่เท่ากัน
>	มากกว่า
>=	มากกว่าหรือเท่ากัน
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากัน

ค่าจริง - เท็จ boolean

True	มีค่าเป็นจริง แทนด้วยตัวเลขเป็น 1
False	มีค่าเป็นเท็จ แทนด้วยตัวเลขเป็น 0



3. การสร้างและใช้งานตัวแปรข้อความ (String)

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



String Slice

```
fullname = 'Somchai Rukchard'
name = fullname[:7]
lastname = fullname[-8:]

print(name)
print(lastname)
```

การเรียกตำแหน่งของข้อมูลแบบ string

ใช้การใส่ index ได้ตามรูปแบบนี้

[_ตำแหน่งเริ่ม_:_ตำแหน่งที่จะหยุดแสดงค่า_]



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')
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 ไปเช็คว่า ตรงกับเงื่อนไขใด ก็จะทำ

คำสั่งหลัง case นั้น แต่ถ้าไม่ตรงกับเงื่อนไขใดเลย ก็จะทำ

คำสั่งหลัง case _:



5. การเขียนคำสั่งวนลูป

- 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. โครงสร้างข้อมูลในภาษาโปรแกรม Python

- List
- Tuple
- Dictionary
- Set



List

```
number1 = list(range(100))  
number2 = list(range(1,51))
```

การสร้าง list แบบช่วง อย่างง่าย

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

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



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)



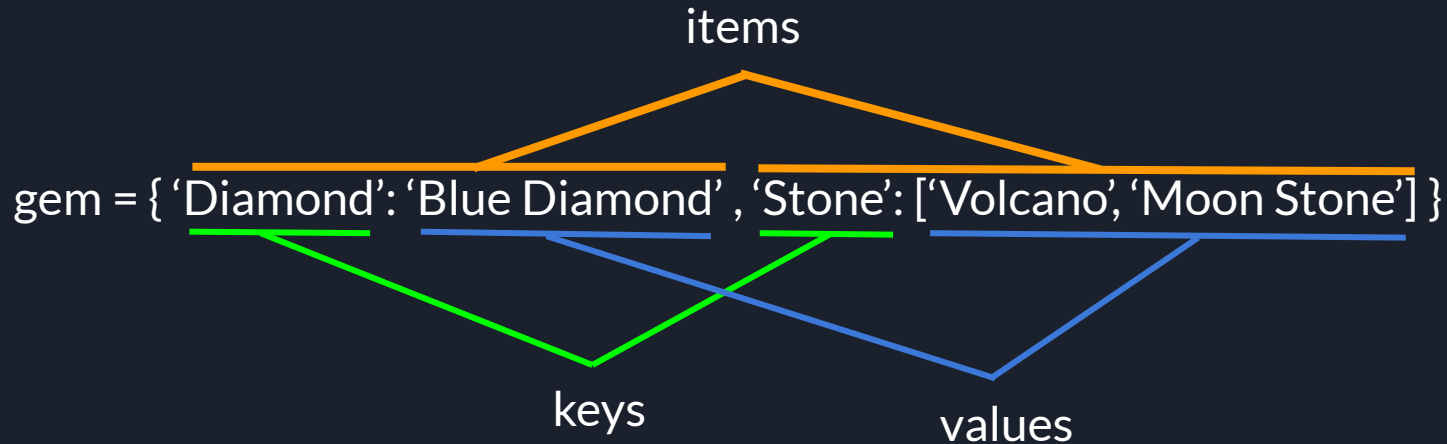
Tuple

location = (1500,750)

ค่าที่เก็บค่ามากกว่า 1 ค่า ไว้ในค่าเดียว
โดยค่าแต่ละค่าไม่สามารถแก้ไขได้

Dictionary

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





Dictionary

การเรียกค่าใน dict()

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

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



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. การสร้างและใช้งานโมดูล (Modules)

- 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. การดักจับและตรวจสอบข้อผิดพลาดต่างๆ

- Error Types
- Exceptions
- Bug
- Debugging

10. การดักจับและตรวจสอบข้อผิดพลาดต่างๆ

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. การดักจับและตรวจสอบข้อผิดพลาดต่างๆ

Exceptions (Error Type : ZeroDivisionError)

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

```
try:
    result = 10 / int(number)
except ZeroDivisionError:
    pass
else:
    print(result)
```

pass คือการข้ามการทำงานไปยังบล็อกต่อไป

10. การดักจับและตรวจสอบข้อผิดพลาดต่างๆ

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
(ไม่แนะนำให้ใช้ finally)



11. การเขียน Unit Testing เบื้องต้น

- Unit Testing Fundamental
- unittest



11. การเขียน Unit Testing 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 Testing 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 Testing 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()
```