

Regular Expression (RegEx)

Regular Expression หรือที่เรียกกันโดยย่อว่า RegEx คือการสร้างรูปแบบสำหรับตรวจจับข้อความ เพื่อประโยชน์ในการค้นหาสิ่งที่ตรงกับรูปแบบที่กำหนดไว้

ตารางสัญลักษณ์ Meta Character ที่ใช้งานบ่อย

สัญลักษณ์	ความหมาย	ตัวอย่างการใช้
Anchors (หมวดการยึดตำแหน่งของอักขระ หรือกลุ่มอักขระ)		
. (any)	แทนอักขระใดก็ได้ 1 ตัว	.ncle เช่น uncle, ancle, 0ncle ฯลฯ
^ (start)	จุดเริ่มต้น (ใช้กับหลายบรรทัด)	^py เช่น python, pyautogui, pypercilp ฯลฯ
\$ (end)	จุดสิ้นสุด (ใช้กับหลายบรรทัด)	thon\$ เช่น python, marathon
Groups/Ranges (หมวดการจัดกลุ่ม และการระบุขนาดของอักขระ)		
[] (lists)	เลือกอักขระ 1 ตัวที่อยู่ในลิสต์	[bcr]at, [A-Z], [0-9]
() (passive group)	เลือกกลุ่มอักขระตามที่ระบุไว้ จะมีเครื่องหมาย “ ” คั่นหรือไม่ก็ได้	(.com), (bat cat rat)
{ } (range)	ระบุจำนวนอักขระที่ต้องการ ทั้งที่ระบุตายตัว หรือเป็นช่วง (อย่างน้อยที่สุด, อย่างมากที่สุด)	{3} คือ ต้องมีอักขระ 5 ตัว {1,3} คือ มีอักขระอย่างน้อย 1 ตัว แต่ไม่เกิน 3 ตัว {2, } คือ มีอักขระอย่างน้อย 2 ตัวขึ้นไป
* (0 - x)	ไม่มีอักขระที่ซ้ำกันเลย หรือ มีอักขระซ้ำกันอย่างน้อย 1 ตัว	01* => 0, 01, 011, 0111, ...
+ (1 - x)	มีอักขระซ้ำกันอย่างน้อย 1 ตัว	01+ => 01, 011, 0111, ...
? (0 - 1)	ไม่มีอักขระที่ซ้ำกันเลย หรือ มีอักขระซ้ำกันเพียง 1 ตัว	01? => 0, 01
Character Classes (หมวดอักขระ จะมีเครื่องหมาย \ นำหน้า)		
\w (word)	พยัญชนะ/ตัวอักษร/ตัวเลข	\wat เช่น bat, Cat, 0at ฯลฯ
\W (not word)	ไม่ใช่พยัญชนะ/ตัวอักษร/ตัวเลข	\Wat เช่น _at, @at, #at ฯลฯ
\d (digit)	ตัวเลข	\d99 เช่น 099, 599, 999 ฯลฯ
\D (not digit)	ไม่ใช่ตัวเลข	\D99 เช่น A99, z99, ก99 ฯลฯ
\s (space)	ช่องว่าง	[A-Z][a-z]+\s[A-Z][a-z]+ เช่น Uncle Engineer
\S (not space)	ไม่ใช่ช่องว่าง	[A-Z][a-z]+\S[A-Z][a-z]+ เช่น UncleEngineer

RegEx ในภาษา Python

การใช้งาน RegEx ในภาษา Python จะใช้ไลบรารีที่ชื่อว่า re สามารถเรียกใช้ได้ทันที ไม่ต้องติดตั้ง

```
import re
```

เมธอดใน re (RegEx) ที่ใช้งานบ่อย

เมธอด	ความหมาย
re.compile()	สร้างออบเจกต์สำหรับ RegEx
re.findall()	ผลการค้นหาทั้งหมดจะคืนค่าเป็นลิสต์
re.finditer()	ผลการค้นหาทั้งหมดจะคืนค่าเป็นลิสต์ และบอกตำแหน่งที่พบ
re.match()	ค้นหาในตำแหน่งที่เป็นจุดเริ่มต้นของสตริง (index 0) ถ้าพบแล้วจะไม่ค้นหาต่อในส่วนที่เหลือ
re.search()	ค้นหาในตำแหน่งใดก็ได้ภายในสตริง ถ้าพบแล้วจะไม่ค้นหาต่อในส่วนที่เหลือ
re.split()	ตัดอักขระที่ต้องการ และแยกออกจากกัน
re.sub()	แทนที่อักขระตามที่ระบุ ในรูปแบบ RegEx

ตัวอย่างการใช้งานเมธอดใน re (RegEx)

กำหนดให้ตัวแปร my_string เก็บค่าสตริง

```
my_string = 'Uncle Engineer Loong Wissawakorn'
```

- re.findall()

ค้นหาตำแหน่งที่มี 'ng' ทุกตำแหน่ง

```
find1 = re.findall('ng', my_string)
print(find1)

# ['ng', 'ng']
```

ค้นหาอักขระ 1 ตัว อะไรก็ได้ แล้วตามด้วยตัว n

```
find2 = re.findall(r'\wn', my_string)
print(find2)

# ['Un', 'En', 'in', 'on', 'rn']
```

- re.finditer()

ค้นหาอักขระ 1 ตัว อะไรก็ได้ แล้วตามด้วยตัว n พร้อมกับบอกตำแหน่งที่ค้นหาเจอ ซึ่งต้องใช้การวนลูปเข้ามาด้วย

```
for f in re.finditer(r'\wn', my_string):
    print(f)

# <re.Match object; span=(0, 2), match='Un'>
# <re.Match object; span=(6, 8), match='En'>
# <re.Match object; span=(9, 11), match='in'>
# <re.Match object; span=(18, 20), match='on'>
# <re.Match object; span=(33, 35), match='rn'>
```

- re.match()

ค้นหาคำว่า Uncle ถ้าอยู่ที่จุดเริ่มต้นของสตริง (index 0) จะคืนค่าเป็น True แต่ถ้าไม่ใช่จุดเริ่มต้น จะคืนค่าเป็น None และจะค้นหาเฉพาะตำแหน่งแรกที่เจอเท่านั้น

```
match1 = re.match('Uncle', my_string)
print(match1)

# <re.Match object; span=(0, 5), match='Uncle'>
```

ค้นหาอักขระ 1 ตัว อะไรก็ได้ แล้วตามด้วยตัว n ถ้าอยู่ที่จุดเริ่มต้นของสตริง (index 0) จะคืนค่าเป็น True แต่ถ้าไม่ใช่จุดเริ่มต้น จะคืนค่าเป็น None และจะค้นหาเฉพาะตำแหน่งแรกที่เจอเท่านั้น

```
match2 = re.match(r'\wn', my_string)
print(match2)

# <re.Match object; span=(0, 2), match='Un'>
```

- re.search()

ค้นหาคำว่า Engineer ภายในตัวแปร my_string ถ้าหาเจอ จะคืนค่าเป็น True แต่ถ้าหาไม่เจอ จะคืนค่าเป็น None และจะค้นหาเฉพาะตำแหน่งแรกที่เจอเท่านั้น

```
search1 = re.search('Engineer', my_string)
print(search1)

# <re.Match object; span=(6, 14), match='Engineer'>
```

ค้นหาอักขระ 1 ตัว อะไรก็ได้ แล้วตามด้วยตัว n ภายในตัวแปร my_string ถ้าหาเจอ จะคืนค่าเป็น True แต่ถ้าหาไม่เจอ จะคืนค่าเป็น None และจะค้นหาเฉพาะตำแหน่งแรกที่เจอเท่านั้น

```
search2 = re.search(r'\wn', my_string)
print(search2)

# <re.Match object; span=(0, 2), match='Un'>
```

- re.split()

แยกสตริงภายในตัวแปร text ด้วย space 1 เคาะ (' ')

```
split_text = re.split(' ', my_string)
print(split_text)

# ['Uncle', 'Engineer', '', 'Loong', '', '', 'Wissawakorn']
```

- re.sub()

แทนที่ space ที่เกิน 1 เคาะ ('+') ด้วย space 1 เคาะ (' ')

```
replace_text = re.sub(' +', ' ', my_string)
print(replace_text)

# Uncle Engineer Loong Wissawakorn
```

เมธอดในออบเจกต์ Match

```
m = re.match('uncle', 'uncle engineer')
```

เมธอด & แอตทริบิวต์	ความหมาย	ตัวอย่างการใช้
group([group1, ...])	คืนค่า “อักขระที่พบ”	m.group() -> 'uncle'
start([group])	คืนค่า “ตำแหน่งแรกที่พบ”	m.start() -> 0
end([group1])	คืนค่า “ตำแหน่งสุดท้ายที่พบ”	m.end() -> 5
span([group1])	คืนค่า “tuple” (start, end)	m.span() -> (0, 5)

ตัวอย่างการเลือกข้อมูลด้วย RegEx ในภาษา Python

กำหนดให้ ตัวแปร text เก็บค่าสตริงขนาดยาว

```
text = '''
    My name is Uncle Engineer. I have 1,000.25 Baht on 19 Aug 2023.
    Please contact me at 099-555-3210 or uncle.engineer@email.com
    '''
```

ใช้คำสั่ง re.compile() เพื่อสร้างออบเจกต์สำหรับ RegEx และใช้คำสั่ง re.search() เพื่อค้นหาสิ่งที่อยู่ในตัวแปร text ที่เราสนใจจากรูปแบบ RegEx

1. ชื่อ-นามสกุล : Uncle Engineer

```
name_pattern = r'[A-Z][a-z]+(\s[A-Z][a-z]*(.)?)?\s[A-Z][a-z]+'
name_regex = re.compile(name_pattern)
name_result = name_regex.search(text)
print(name_result.group())
```

[A-Z] แทน การเลือกตัวอักษรพิมพ์ใหญ่ A-Z มา 1 ตัว (ชื่อ และ นามสกุล)

[a-z]+ แทน การเลือกตัวอักษรพิมพ์เล็ก a-z อย่างน้อย 1 ตัวขึ้นไป

(\s[A-Z][a-z]*(.)?)? แทน กลุ่มของอักขระ เริ่มจากการเว้นวรรค 1 ครั้ง ต่อด้วยการเลือกตัวอักษรพิมพ์ใหญ่ A-Z มา 1 ตัว และเลือกตัวอักษรพิมพ์เล็ก a-z หรือไม่ก็ได้ จากนั้นเลือกกลุ่มของอักขระย่อย ที่มีเครื่องหมายจุดอยู่ ถ้าค้นหาเจอ จะแสดงเครื่องหมายจุด แต่ถ้าค้นหาไม่เจอ ก็จะไม่แสดงเครื่องหมายจุด ซึ่งเครื่องหมาย ? ที่อยู่ท้ายสุด หมายถึงอักขระทั้งหมดในกลุ่มนี้ จะมีหรือไม่ก็ได้ (ชื่อกลาง)

2. จำนวน : 1,000.25 หรือจำนวนตั้งแต่ 4 หลักขึ้นไป มี comma (,) คั่นทุกๆ 3 หลัก และจุดทศนิยม 2 ตำแหน่ง

```
money_pattern = r'(-)?\d{1,3}(:|\d{3})*(\.\d{0,2})?'
money_regex = re.compile(money_pattern)
money_result = money_regex.search(text)
print(money_result.group())
```

(-)? แทน กลุ่มของอักขระ ที่มีเครื่องหมายลบ ถ้าค้นหาเจอ จะแสดงเครื่องหมายลบ แต่ถ้าค้นหาไม่เจอ ก็จะไม่แสดงเครื่องหมายลบ

\d{1, 3} แทน ตัวเลขจำนวนเต็มที่สามารถมีได้ 1, 2 หรือ 3 หลัก

(?:\d{3})* แทน การจัดกลุ่มตัวเลข ใส่ comma (,) ทุกๆ 3 หลัก และจะมีก็ได้ หรือไม่เลยก็ได้

(\.\d{0, 2})? แทน กลุ่มของอักขระ เริ่มจากเครื่องหมายจุด ต่อด้วยทศนิยมที่สามารถมีได้ 1 หรือ 2 ตำแหน่ง หรือไม่เลยก็ได้ ปิดท้ายด้วยเครื่องหมาย ? หมายถึงอักขระทั้งหมดในกลุ่มนี้ จะมีหรือไม่ก็ได้

3. วันที่ : 19 AUG 2023 / 19 Aug 2023 / 19 August 2023

```
date_pattern = r'\d{1,2}\s[ADFJMNOS]\w+\s\d{4}'
date_regex = re.compile(date_pattern)
date_result = date_regex.search(text)
print(date_result.group())
```

\d{1, 2} แทน การเลือกตัวเลขอะไรก็ได้มา 1 หรือ 2 ตัว

\s แทน การเว้นวรรค 1 เคาะ

[ADFJMNOS] แทน การเลือกตัวอักษรตัวแรกของชื่อเดือนในภาษาอังกฤษ ซึ่งเป็นตัวพิมพ์ใหญ่มา 1 ตัว

\w+ แทน การเลือกตัวอักษร หรือพยัญชนะมาอย่างน้อย 1 ตัวขึ้นไป

\d{4} แทน การเลือกตัวเลขอะไรก็ได้มา 4 ตัว

4. หมายเลขโทรศัพท์ : 0995553210 / 099-5553210 / 099-555-3210 / +6699-555-3210

```
tel_pattern = r'(\+66|0)\d{1,2}(-)?\d{3}(-)?\d{4}'
tel_regex = re.compile(tel_pattern)
tel_result = tel_regex.search(text)
print(tel_result.group())
```

(\+66|0) แทน การเลือกกลุ่มอักขระ ระหว่าง +66 หรือ 0

\d{1,2} แทน การเลือกตัวเลขอะไรก็ได้มา 1 หรือ 2 ตัว

(-)? แทน กลุ่มของเครื่องหมายขีด ถ้าค้นหาเจอ จะแสดงเครื่องหมายขีด แต่ถ้าค้นหาไม่เจอ ก็จะไม่แสดงเครื่องหมายขีด

[d]{3} แทน การเลือกตัวเลขอะไรก็ได้มา 3 ตัว

[d]{4} แทน การเลือกตัวเลขอะไรก็ได้มา 4 ตัว

5. อีเมล : uncle-engineer@email.com / uncle.engineer@email.com / uncle_engineer@email.com

```
email_pattern = r'[\w\-\._]+@\w+\.\w{2,3}(\.\w{0,2})?'
email_regex = re.compile(email_pattern)
email_result = email_regex.search(text)
print(email_result.group())
```

`[\w\-\._]+` แทน การเลือกตัวอักษร หรือพยัญชนะ รวมถึงเครื่องหมาย (-), (.) และ (_) จากภายในลิสต์ มาอย่างน้อย 1 ตัวขึ้นไป

`@` แทน เครื่องหมาย At the rate sign (@)

`\w+` แทน การเลือกตัวอักษร หรือพยัญชนะ มาอย่างน้อย 1 ตัวขึ้นไป

`\.` แทน เครื่องหมายจุด (.)

`\w{2,3}` แทน การเลือกตัวอักษร หรือพยัญชนะ 2 หรือ 3 ตัว

`(.)?` แทน กลุ่มของเครื่องหมายจุด ถ้าค้นหาเจอ จะแสดงเครื่องหมายจุด แต่ถ้าค้นหาไม่เจอ ก็จะไม่แสดงเครื่องหมายจุด

`(\w{0, 2})?` แทน แทน กลุ่มของอักขระ เริ่มจากการเลือกตัวอักษร หรือพยัญชนะ ที่สามารถมีได้ 1 หรือ 2 ตัว หรือไม่มีเลยก็ได้ ปิดท้ายด้วยเครื่องหมาย ? หมายถึงอักขระทั้งหมดในกลุ่มนี้ จะมีหรือไม่ก็ได้

Test RegEx online : <https://regex101.com/>

อ้างอิง :

- <https://docs.python.org/3/library/re.html>

- <https://cheatography.com/mutanclan/cheat-sheets/python-regular-expression-regex/pdf/>

- <https://drive.google.com/file/d/1XPcNtR8z9MvP7Mkcwyubd3T6Rc0fmNj6/view>