

Exceptions ในภาษา Python

ในบทนี้ คุณจะได้เรียนรู้เกี่ยวกับการจัดการข้อผิดพลาด (Error) ในภาษา Python ที่เรียกว่า Exception ซึ่งสามารถเกิดขึ้นได้ถึงแม้ว่า syntax ของโปรแกรมถูกต้อง แต่บางคำสั่งในโค้ดโปรแกรมนั้นทำให้เกิดข้อผิดพลาดขึ้น ซึ่งการจัดการข้อผิดพลาดนั้นเป็นสิ่งที่ควรทำในการเขียนโปรแกรม เพราะมันจะทำให้โปรแกรมของคุณไม่แสดงข้อผิดพลาดให้กับผู้ใช้ได้เห็น

Syntax Errors

เหมือนที่คุณได้เรียนรู้โครงสร้างของภาษา Python ในบทก่อนหน้านี้ทั้งหมดมาแล้ว เมื่อคุณเขียนโปรแกรมไม่ถูกต้องตามหลักไวยากรณ์ของภาษาจะทำให้เกิดข้อผิดพลาดขึ้นขณะที่โปรแกรมได้ทำการตรวจสอบข้อโค้ดของคุณ หรือเราเรียกว่า Syntax errors

```
if True  
  
    print('Enter the if block')
```

ในตัวอย่าง เป็นโค้ดของโปรแกรมที่จะทำให้เกิด Syntax errors ขึ้น เพราะว่าในคำสั่ง if จะต้องมีความหมายโคลอน (:) หลังจากเงื่อนไขของมัน เมื่อคุณรันโปรแกรมจึงทำให้เกิดข้อผิดพลาดขึ้น ดังนั้นในกรณีเกิด Syntax errors เราจำเป็นต้องแก้ไขโค้ดของโปรแกรมให้ถูกต้องก่อนจึงจะสามารถรันโปรแกรมได้

File "exception.py", line 1

```
if True  
  
    ^
```

SyntaxError: invalid syntax

นี่เป็นตัวอย่างผลลัพธ์ของข้อผิดพลาดที่เกิดขึ้นเมื่อคุณรันโค้ดดังกล่าว โดย Python จะแสดงชนิดของข้อผิดพลาดที่เกิดขึ้น ซึ่งประกอบไปด้วยชื่อไฟล์ บรรทัดที่เกิดข้อผิดพลาด และประเภทของข้อผิดพลาด ตามด้วยข้อความอธิบายสิ่งที่เกิดขึ้น

Exceptions

Exception นั้นแตกต่างจากข้อผิดพลาดก่อนหน้านี้ มันสามารถเกิดขึ้นได้เมื่อโปรแกรมของคุณทำงานบางอย่าง เช่น การหารด้วยศูนย์ การใช้งานตัวแปรที่ไม่ได้ประกาศไว้ หรือการแปลงข้อมูลคนละประเภทกัน โดยพื้นฐานของภาษาแล้ว ข้อผิดพลาดส่วนมากไม่ได้รับการจัดการอัตโนมัติ ดังนั้น คุณจำเป็นต้องจัดการกับมันเอง (Exception handing) ต่อไปเป็นตัวอย่างของคำสั่งที่สามารถทำให้เกิด Exception ได้

```
print (10 / 0)
```

```
print (5 * money)
```

```
print (1 + '2')
```

ในตัวอย่าง เป็นชุดของคำสั่งที่จะทำให้เกิด Exception ขึ้นและโปรแกรมจะหยุดการทำงานในทันที เราได้ทำการรันโปรแกรมสามครั้ง ในคำสั่งแรกเป็นการหารตัวเลขด้วยศูนย์ คำสั่งต่อมาเป็นการใช้งานตัวแปรที่ไม่ได้ประกาศ money และคำสั่งสุดท้ายเป็นการใช้งาน operand + กับประเภทข้อมูลที่ไม่ถูกต้อง

Traceback (most recent call last):

File "exception.py", line 1, in <module>

print (10 / 0)

ZeroDivisionError: division by zero

Traceback (most recent call last):

File "exception.py", line 3, in <module>

print (5 * money)

NameError: name 'money' is not defined

Traceback (most recent call last):

File "exception.py", line 5, in <module>

```
print (1 + '2')
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

เมื่อเราได้รับโปรแกรม ในคำสั่งแรกการหารตัวเลขด้วยศูนย์จะทำให้เกิด ZeroDivisionError exception ขึ้น บรรทัดถัดไป การใช้ตัวแปรที่ไม่ได้ประกาศไว้จะทำให้เกิด NameError exception และบรรทัดสุดท้าย การใช้ operand ไม่ถูกต้องกับประเภทของข้อมูลทำให้เกิด TypeError ตามลำดับ นี่เป็นข้อผิดพลาดพื้นฐานที่จะเกิดขึ้นเมื่อคุณเขียนโปรแกรมแล้วไม่ได้ตรวจสอบการทำงานให้ดี

Handling Exceptions

อย่างที่เรารู้ได้บอก เมื่อเกิด Exception ขึ้นโปรแกรมจะหยุดการทำงานในทันที ดังนั้นเพื่อให้โปรแกรมของเราสามารถทำงานต่อไปได้ เราจำเป็นต้องจัดการกับ Exception เหล่านั้น ซึ่งในภาษา Python มีรูปแบบในการจัดการกับ Exception ดังนี้

```
try:
```

```
    # do something
```

```
except firstError:
```

```
    # handling exception
```

```
except secondError as e:
```

```
    # handling exception
```

```
except:
```

```
    # handling exception
```

else:

```
# excuted when no exception
```

ในการจัดการกับ Exception จะใช้คำสั่ง try ... except สำหรับตรวจจับข้อผิดพลาดที่จะเกิดขึ้น ในบล็อกของคำสั่ง try จะเป็นการทำงานที่จะทำให้เกิดข้อผิดพลาดขึ้น และเราสามารถมีบล็อกคำสั่ง except ได้หลายอันเพื่อจัดการข้อผิดพลาดประเภทต่างๆ และถ้าหากคุณไม่ได้กำหนดประเภทให้กับ except หมายความว่ามันสามารถจัดการกับข้อผิดพลาดได้ทุกประเภทที่สืบทอดมาจากคลาส Exception นอกจากนี้ คุณยังสามารถใช้ else clause ซึ่งจะทำงานเมื่อไม่เกิดข้อผิดพลาดขึ้นในขณะที่โปรแกรมทำงานในบล็อกคำสั่ง try ต่อไปมาดูตัวอย่างการจัดการข้อผิดพลาดในภาษา Python

try:

```
a = int(input('Enter first number: '))
```

```
b = int(input('Enter second number: '))
```

```
print("%d / %d = %f" % (a, b, a / b))
```

except ValueError as e:

```
print ('You should enter a valid number')
```

except ZeroDivisionError as e:

```
print ('Handling error: ', e)
```

ในตัวอย่าง เป็นโปรแกรมสำหรับรับค่าตัวเลขสองตัวจากทางคีย์บอร์ดและแสดงผลหาร ในบล็อกของคำสั่ง try เป็นการรับค่าตัวเลขและเก็บใส่ตัวแปร a และ b ตามลำดับ เนื่องต้องการแปลงค่าที่รับมาเป็นตัวเลขด้วยฟังก์ชัน int() ดังนั้น เราต้องมีบล็อก except เพื่อจัดการกับข้อผิดพลาด ValueError ที่จะเกิดขึ้นเมื่อค่าที่ใส่เข้ามานั้นไม่ได้เป็นตัวเลข

ต่อมาเป็นการแสดงผลการหารของตัวเลขเหล่านั้น ซึ่งสามารถทำให้เกิด ZeroDivisionError exception ขึ้นได้เช่นกัน เราจึงได้ทำการใช้คำสั่ง except เพื่อจัดการกับข้อผิดพลาดนี้ ในการจัดการข้อผิดพลาดคุณอาจจะบอกวิธีการแก้ไขหรือแสดงรายละเอียดของข้อผิดพลาดนั้นให้ผู้ใช้โปรแกรมได้เห็นก็ได้

```
Enter first number: hello
```

```
You should enter a valid number
```

```
Enter first number: 10
```

```
Enter second number: 0
```

```
Handling error: division by zero
```

```
Enter first number: 5
```

```
Enter second number: 3
```

```
5 / 3 = 1.666667
```

นี่เป็นผลลัพธ์เมื่อเราได้รันโปรแกรมเป็นจำนวนสามครั้ง ครั้งแรกเป็นการใส่ข้อมูลที่ไม่ใช่ตัวเลขเข้ามา ทำให้เกิดข้อผิดพลาดขึ้นและโปรแกรมทำงานในบล็อกคำสั่ง exception ของ ValueError ต่อมาเราได้ใส่ตัวเลขตัวที่สองเป็นศูนย์ ทำให้เกิดข้อผิดพลาด ZeroDivisionError ขึ้น และสุดท้ายโปรแกรมของเราทำงานได้โดยไม่มีข้อผิดพลาด

ต่อไปมาดูตัวอย่างเพิ่มเติมสำหรับการจัดการข้อผิดพลาดในการทำงานกับไฟล์ ซึ่งในการทำงานกับไฟล์นั้นมีข้อผิดพลาดต่างๆ ที่สามารถเกิดขึ้นได้ เช่น การเปิดไฟล์ที่ไม่มีอยู่ หรือไฟล์นั้นไม่พร้อมใช้งาน เป็นต้น

```
import sys
```

```
try:
```

```
    f = open('file.txt')
```

```
    s = f.readline()
```

```
    print(s)
```

```
except OSError as err:

    print("OS error: ", err)

except:

    print("Unexpected error occured")

else:

    print("File closed successfully")

f.close()
```

ในการทำงานกับไฟล์ เราทำการนำเข้าไลบรารีจาก sys มายังโปรแกรม และในบล็อกของคำสั่ง try เป็นการเปิดไฟล์ชื่อ file.txt และอ่านข้อมูลบรรทัดแรกมาแสดงผลทางหน้าจอ ในการเปิดไฟล์ถ้าหากไม่มีไฟล์อยู่จะทำให้เกิดข้อผิดพลาด OSError ขึ้นและเราได้ทำการแสดงข้อความบอกทางหน้าจอ เรายังทำการจัดการกับข้อผิดพลาดต่างๆ ที่อาจจะเกิดขึ้นด้วย

ในบล็อกของคำสั่ง else จะทำงานเมื่อไม่มีข้อผิดพลาดเกิดขึ้น นั่นหมายถึงเราเปิดไฟล์เพื่ออ่านข้อมูลได้สำเร็จ ดังนั้นเราจึงควรจะทำการปิดไฟล์ในบล็อกคำสั่งนี้

```
OS error: [Errno 2] No such file or directory: 'file.txt'
```

```
marcuscode.com
```

```
File closed successfully
```

นี่เป็นผลลัพธ์ของโปรแกรมที่จะแสดงข้อผิดพลาดขึ้นเมื่อเรารันโปรแกรมในทันทีโดยที่ยังไม่มีไฟล์ file.txt อยู่ และถัดมาเป็นผลลัพธ์ของโปรแกรมเมื่อเราสร้างไฟล์ file.txt และภายในไฟล์มีข้อความ "marcuscode.com" อยู่ข้างใน ทำให้โปรแกรมสามารถอ่านไฟล์ได้และนำข้อความมาแสดงผลบนหน้าจอ และหลังจากนั้นโปรแกรมทำงานในบล็อกคำสั่ง else

Raising Exceptions

ในภาษา Python มี built-in exception ที่จะเกิดขึ้นโดยพื้นฐานเมื่อโปรแกรมมีข้อผิดพลาดขึ้น อย่างไรก็ตาม โปรแกรมเมอร์สามารถสั่งให้เกิด Exception ขึ้นเองได้ โดยใช้คำสั่ง raise มาดูตัวอย่างการใช้งาน

```
try:

    name = input('Enter your name: ')

    if name == 'mateo':

        raise Exception('Whoa! Mateo you are not allowed here')

    print('Hi ', name)

except Exception as err:

    print("Exception: ", err)

else:

    print('Bye')
```

ในตัวอย่าง เป็นโปรแกรมสำหรับรับชื่อจากทางคีย์บอร์ดและทักทาย ถ้าหากชื่อที่ใส่เข้ามานั้นเป็น "mateo" เราจะทำให้เกิด exception ขึ้นด้วยคำสั่ง raise โดยสร้างออบเจ็กต์จากคลาส Exception ซึ่งเป็นคลาสในภาษา Python และกำหนดข้อความของเราเอง และถ้าหากชื่อที่ใส่เข้ามาเป็นอย่างอื่นที่ไม่ใช่ "mateo" โปรแกรมจะแสดงการทักทายและจบการทำงาน

```
Enter your name: mateo

Exception:  Whoa! Mateo you are not allowed here

Enter your name: Marcus

Hi  Marcus
```

Bye

และนี่เป็นผลลัพธ์การทำงานของโปรแกรม โดยครั้งแรกจะเกิดข้อผิดพลาดขึ้นเพราะเราได้ใส่ชื่อเข้ามาเป็น "mateo" และครั้งที่สองไม่เกิดข้อผิดพลาดเพราะชื่อที่ใส่เข้ามาเป็น "marcus" และหลังจากนั้นโปรแกรมแสดงข้อความทักทายและคำบอกลาในบล็อกของคำสั่ง else

การสร้าง Exceptions

นอกจากการใช้งาน build-in exception จากภาษา Python แล้ว คุณยังสามารถสร้างคลาส Exception ขึ้นมาเองได้ เพื่อให้สามารถทำงานได้ตามที่ต้องการ ยกตัวอย่างเช่น การเพิ่มแอตทริบิวต์หรือเมธอดต่างๆ ภายในคลาส ต่อไปเราจะมาสร้างคลาสเพื่อจัดการข้อผิดพลาดของเราเอง โดยในการสร้างคลาสนั้นเราต้องทำการสืบทอดมาจากคลาส Exception เสมอ มาดูตัวอย่าง

```
# การสร้างคลาสซึ่งคุณอาจจะสร้างไว้ที่โมดูลอื่นเพื่อเรียกใช้งาน
```

```
class UsernameError(Exception):
```

```
    def __init__(self, message, error):
```

```
        super().__init__(message)
```

```
        self.message = message
```

```
        self.error = error
```

```
    def getMessage(self):
```

```
        return self.message + ' \'' + self.error + '\'
```

```
class PasswordError(Exception):
```

```
    def __init__(self, message, error):
```



```
super().__init__(message)

self.message = message

self.error = error


def getMessage(self):

    return self.message + ' \' + ('*' * len(self.error)) + \'

# โปรแกรมเริ่มการทำงาน

try:

    print('Login')

    username = input('Username: ')

    password = input('Password: ')

    if (username != 'mateo'):

        raise UsernameError('Invalid username', username)

    if (password != '1234'):

        raise PasswordError('Invalid password', password)

    print('Login success')

except UsernameError as e:

    print('Exception: ', e.getMessage())
```

```
except PasswordError as e:
```

```
    print('Exception: ', e.getMessage())
```

ในตัวอย่างของโปรแกรมนั้นจะแบ่งออกเป็นสองส่วน ในส่วนแรกเป็นการสร้างคลาสโดยเราได้สร้างคลาสมาสองคลาสคือ UsernameError เป็นคลาสของ Exception สำหรับจัดการเมื่อ username ไม่ถูกต้อง และคลาส PasswordError เป็นคลาสของ Exception สำหรับจัดการข้อผิดพลาดเมื่อรหัสผ่านไม่ถูกต้อง โดยในคลาสเราได้กำหนดแอตทริบิวต์สองตัวคือ message เป็นความสำหรับอธิบายข้อผิดพลาด และ error เป็นข้อมูลที่เกิดข้อผิดพลาดขึ้น และภายในคลาสทั้งสองมีเมธอด getMessage() สำหรับรับข้อความการแสดงผลข้อผิดพลาดที่แตกต่างกันออกไป

ในส่วนต่อมา เป็นการทดสอบการจัดการข้อผิดพลาดของเรา โดยการจำลองการทำงานของระบบ Login สำหรับให้ผู้ใช้เข้าสู่ระบบโดยการใส่ username และ password โดยเราจะทำการตรวจสอบถ้าหากชื่อผู้ใช้ไม่เป็น "mateo" เราจะทำให้เกิดข้อผิดพลาด UsernameError ขึ้น แต่ถ้าชื่อผู้ใช้ถูกต้องแต่รหัสผ่านยังผิดจะทำให้เกิดข้อผิดพลาด PasswordError ขึ้น นอกจากนี้ หมายความว่า การเข้าสู่ระบบสำเร็จ

Login

Username: guest

Password: 1111

Exception: Invalid username 'guest'

Login

Username: mateo

Password: 5555

Exception: Invalid password '*****'

Login

Username: mateo

Password: 1234

Login success

นี่เป็นผลลัพธ์การทำงานของโปรแกรมจากการรันสามครั้ง ครั้งแรกเราได้ใส่ชื่อผู้ใช้ที่ไม่ถูกต้อง และครั้งที่สองเราได้ใส่ชื่อผู้ใช้ถูกต้องแต่รหัสผ่านผิด คุณจะสังเกตเห็นสิ่งที่แตกต่างคือการแสดงข้อผิดพลาดของรหัสผ่านจะถูกปกปิดไว้ เพราะเราเรียกใช้เมธอด `getMessage()` ซึ่งมีการทำงานที่ไม่เหมือนกันสำหรับแต่ละคลาส และสุดท้ายเป็นการใส่ชื่อผู้ใช้และรหัสผ่านถูกต้อง โปรแกรมจะบอกว่าเข้าสู่ระบบสำเร็จ

การใช้คำสั่ง Finally

อีกคำสั่งหนึ่งที่ใช้สำหรับการจัดการข้อผิดพลาดก็คือคำสั่ง `finally` ที่สามารถใช้ร่วมกับคำสั่ง `try ... except` ได้ โดยการทำงานของมันนั้นจะแตกต่างจาก `else` คือจะทำงานในบล็อกคำสั่งนี้เสมอ ไม่ว่าจะเกิดข้อผิดพลาดหรือไม่ก็ตาม มาดูตัวอย่างการใช้งาน

`try:`

```
items = ['Mac', 'iPhone', 'iPad']
```

```
print('Avilable items: ', items)
```

```
need = input('What do you want to buy?: ')
```

```
if need not in items:
```

```
    raise Exception('Sorry, \' + need + \' out of stock')
```

```
print('You have purchased ' + \' + need + \')
```

`except Exception as e:`

```
print(e)
```

`finally:`

```
print("Thank you for shopping with us")
```

ในตัวอย่าง เป็นโปรแกรมสำหรับสั่งซื้อสินค้าชนิด เราได้ประกาศสินค้าที่มีอยู่ใน Stock ใส่ไว้ในตัวแปร
list items โดยโปรแกรมจะถามว่าต้องการซื้ออะไร หลังจากที่เราใส่ชื่อสินค้าเข้ามาแล้ว เราได้ทำการตรวจสอบว่ามี
ใน list items หรือไม่ ถ้าหากไม่มีจะแสดงข้อผิดพลาดขึ้นว่าสินค้าหมด และถ้าหากมี จะแสดงข้อความว่าได้ซื้อ
สำเร็จแล้ว และในตอนสุดท้ายเราได้แสดงข้อความขอบคุณไม่ว่าลูกค้าได้จะได้อะไรหรือไม่ก็ตาม ดังนั้น เราจึง
กำหนดให้มันทำงานในบล็อกคำสั่ง finally

```
Available items: ['Mac', 'iPhone', 'iPad']
```

```
What do you want to buy?: xbox
```

```
Sorry, 'xbox' out of stock
```

```
Thank you for shopping with us
```

```
Available items: ['Mac', 'iPhone', 'iPad']
```

```
What do you want to buy?: iPad
```

```
You have purchased 'iPad'
```

```
Thank you for shopping with us
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม อันแรกเป็นการใส่ชื่อสินค้าที่ไม่มีอยู่ ดังนั้นโปรแกรมจะแสดงว่าสินค้า
หมดแล้ว และต่อมาเป็นการใส่ชื่อสินค้าที่มีอยู่คือ "iPad" และในตอนท้ายเราแสดงคำขอบคุณเสมอ

ในบทนี้ คุณได้เรียนรู้เกี่ยวกับการจัดการข้อผิดพลาดหรือ Exception ในภาษา Python เราได้ให้ตัวอย่างแบบ
ต่างๆ ที่คุณสามารถนำไปประยุกต์เขียนโปรแกรมในขั้นสูงได้ต่อไป นอกจากนี้เรายังพูดถึงเกี่ยวกับการสร้างคลาส
Exception ขึ้นมาเองพร้อมวิธีการเรียกใช้งาน ซึ่งจะช่วยให้คุณสามารถขยายความสามารถในการจัดการ
ข้อผิดพลาดตามที่คุณต้องการได้