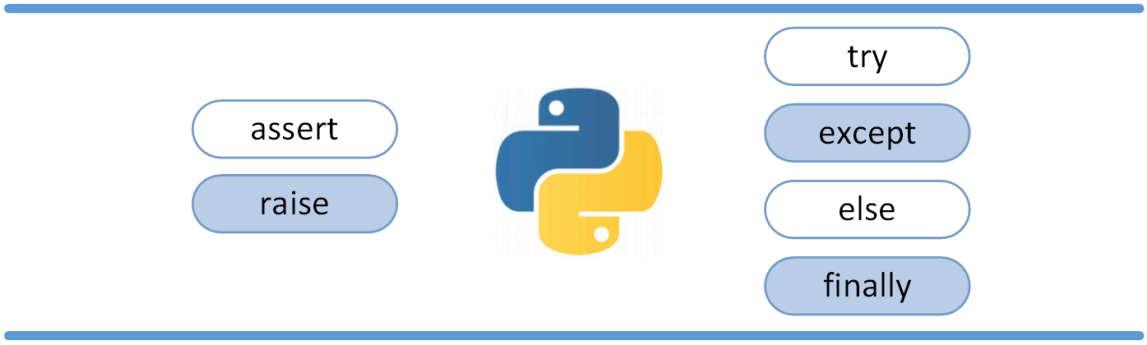


Exception Handling(การจัดการข้อผิดพลาด)

Exception คืออะไร

Exception คือ เหตุการณ์ผิดปกติที่เกิดขึ้นระหว่างการประมวลผลโปรแกรม ส่งผลให้โปรแกรมทำงานไม่เป็นไปตามขั้นตอนที่ได้กำหนดไว้

ตัวอย่างของ exception เช่น

- โค้ดของโปรแกรมมีการหารตัวเลขหนึ่งๆ ด้วยศูนย์
- โปรแกรมพยายามเปิดไฟล์ที่ไม่มีอยู่จริง
- โปรแกรมต้องการรับอินพุตเป็นเลขจำนวนเต็ม แต่ผู้ใช้โปรแกรมป้อนข้อมูลเข้ามาเป็นสตริง

ซึ่งข้อผิดพลาดที่เกิดขึ้นในการเขียนโปรแกรมแบ่งออกได้เป็น 3 ประเภทใหญ่ ๆ ได้แก่

1. Syntax Error คือ การเขียนคำสั่งโปรแกรมผิดหลักไยากรณ์ที่กำนดไว้
2. Runtime Error คือ การเขียนคำสั่งโปรแกรมที่ไม่ถูกต้องโดยผู้เขียนโปรแกรมเอง
3. Logic Error คือ การทำงานของโปรแกรมผิดพลาดที่เกิดมาจากตัวผู้เขียนโปรแกรม เอง

In [1]:
```
# Syntax Error
# การแจ้งเตือนข้อผิดพลาดเมื่อผู้เขียนโปรแกรมไม่ใส่เครื่องหมาย Colon (:) ปิดท้ายคำสั่ง for

for i in range(1,10)
    print(f'รอบที่ {i}')
```

```
  Cell In[1], line 4
    for i in range(1,10)
                       ^
SyntaxError: expected ':'
```

In [5]:
```
# Runtime Error
# ลักษณะการแจ้งเตือนข้อผิดพลาดเมื่อกรอกข้อมูลไม่ถูกต้อง

n = int(input('กรุณาป้อนตัวเลข 1-5: '))
for i in range(1,n):
    print(f'รอบที่ {i}', end = ' ')
```

Loading [MathJax]/extensions/Safe.js 2 รอบที่ 3 รอบที่ 4 รอบที่ 5

In [2]:
```
# Runtime Error
# ลักษณะการแจ้งเตือนข้อผิดพลาดเมื่อกรอกข้อมูลไม่ถูกต้อง

n = int(input('กรุณาป้อนตัวเลข 1-5: '))
for i in range(1,n):
    print(f'รอบที่ {i}', end = ' ')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[2], line 4
      1 # Runtime Error
      2 # ลักษณะการแจ้งเตือนข้อผิดพลาดเมื่อกรอกข้อมูลไม่ถูกต้อง
----> 4 n = int(input('กรุณาป้อนตัวเลข 1-5: '))
      5 for i in range(1,n):
      6     print(f'รอบที่ {i}', end = ' ')

ValueError: invalid literal for int() with base 10: 'o'
```

In [7]:
```
# Logic Error
# ลักษณะการทำงานของโปรแกรมที่ผิดพลาดจาก Logic

b = float(input('กรุณาป้อนความยาวฐาน = '))
h = float(input('กรุณาป้อนความสูง = '))
area = 0.5 * b + h
print(area)
```

```
50.0
```

In [3]:
```
# Logic Error
# ลักษณะการทำงานของโปรแกรมที่ผิดพลาดจาก Logic

b = float(input('กรุณาป้อนความยาวฐาน = '))
h = float(input('กรุณาป้อนความสูง = '))
area = 0.5 * b + h
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[3], line 5
      1 # Logic Error
      2 # ลักษณะการทำงานของโปรแกรมที่ผิดพลาดจาก Logic
      4 b = float(input('กรุณาป้อนความยาวฐาน = '))
----> 5 h = float(input('กรุณาป้อนความสูง = '))
      6 area = 0.5 * b + h

ValueError: could not convert string to float: 'x'
```

In [4]:
```
print(5+0)
print(5-0)
print(5/0)
print(5*0)
```

```
5
5
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[4], line 3
      1 print(5+0)
      2 print(5-0)
----> 3 print(5/0)
      4 print(5*0)

ZeroDivisionError: division by zero
```

จากตัวอย่างด้านบน จะพบว่ามี exception หลายรูปแบบ ทั้งนี้เราสามารถตรวจจับ exception ที่เกิดขึ้นและจัดการกับมันได้ ซึ่งก็จะช่วยให้โปรแกรมยังคงสามารถทำงานต่อไปได้ ไม่จบการทำงานลงกลางคัน

จะจัดการอย่างไรกับ Exception ที่เกิดขึ้น?

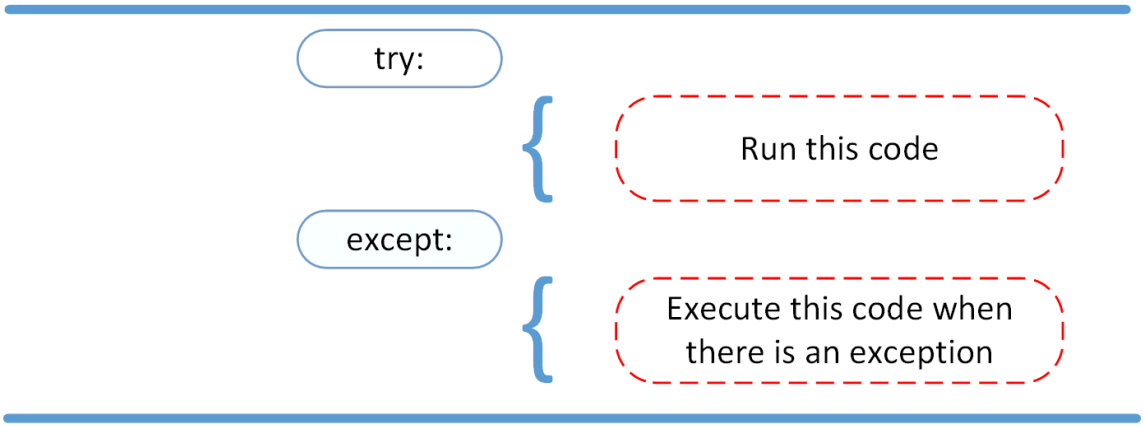
Exception สามารถเกิดขึ้นในโปรแกรมได้เสมอ ดังนั้นจึงควรเพิ่มส่วนของโค้ดที่ใช้สำหรับดักจับและจัดการกับ exception ที่อาจเกิดขึ้นไว้ในโปรแกรมด้วย โดยให้ระบุโค้ดโปรแกรมส่วนนี้ไว้ตรงตำแหน่งที่คาดว่าจะมีโอกาสเกิด exception ขึ้นได้ โดยมีวิธีการต่างๆ ดังนี้

จัดการกับ Exception ด้วย `try` - `except`

`try:` เป็นคำสั่งที่ต้องการให้ตรวจจับความผิดพลาด หรืออาจกล่าวได้ว่า เป็น save zone ที่ให้เราใส่ code ปกติทั่วไป ที่อาจมี error เกิดขึ้นได้

`except:` เป็นคำสั่งที่ต้องการให้ทำงานเมื่อมีข้อผิดพลาด

โดยมีโครงสร้างการใช้คำสั่งดังนี้



In [8]:
```
try:
    print(5+0)
    print(5-0)
    print(5/0)
    print(5*0)
```

```
except:
    print('ไม่สามารถหารด้วย 0 ได้')
```

```
5
5
ไม่สามารถหารด้วย 0 ได้
```

In [9]:
```
try:
    print(5+0)
    print(5-0)
    print(5/0)
    print(5*0)
except ZeroDivisionError:
    print('ไม่สามารถหารด้วย 0 ได้')
```

```
5
5
ไม่สามารถหารด้วย 0 ได้
```

In [10]:
```
try:
    print(5+0)
    print(5-0)
    print(5/0)
    print(5*0)
except ZeroDivisionError as zde:
    print(zde)
```

```
5
5
division by zero
```

++++++++++++++++++++ตัวอย่าง Exception ประเภทอื่นๆ++++++++++++++++++++

In [11]:
```
my_list = [-6, 7, -8, 10, 15, 18]
print(my_list[3])
```

```
10
```

In [12]:
```
my_list = [-6, 7, -8, 10, 15, 18]
print(my_list[6])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[12], line 2
      1 my_list = [-6, 7, -8, 10, 15, 18]
----> 2 print(my_list[6])

IndexError: list index out of range
```

In [13]:
```
# Fix this script

my_list = [-6, 7, -8, 10, 15, 18]
try:
    print(my_list[6])
except IndexError:
    print('ตำแหน่งที่ระบุไม่ถูกต้อง')
```

ตำแหน่งที่ระบุไม่ถูกต้อง

In [14]:
```
my_dict = {1:'Football', 2:'Tennis', 3:'Volleyball'}
print(my_dict[2])
```

```
Tennis
```

In [15]:
```
my_dict = {1:'Football', 2:'Tennis', 3:'Volleyball'}
print(my_dict[4])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[15], line 2
      1 my_dict = {1:'Football', 2:'Tennis', 3:'Volleyball'}
----> 2 print(my_dict[4])

KeyError: 4
```

In [16]:
```
# Fix this script

my_dict = {1:'Football', 2:'Tennis', 3:'Volleyball'}
try:
    print(my_dict[2])
except KeyError:
    print('คียที่ระบุไม่ถูกต้อง')
```

```
Tennis
```

In [17]:
```
my_list = [-6, 7, -8, 10, 15, 18]
my_dict = {1:'Football', 2:'Tennis', 3:'Volleyball'}
try:
    print(my_list[6])
    print(my_dict[4])
except KeyError:
    print('คียที่ระบุไม่ถูกต้อง')
except IndexError:
    print('ตำแหน่งที่ระบุไม่ถูกต้อง')
```

```
ตำแหน่งที่ระบุไม่ถูกต้อง
```

++++++++++++++++++++ตัวอย่าง Exception ประเภทอื่นๆ++++++++++++++++++++

In [18]:
```
with open('file.log') as file:
        read_data = file.read()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[18], line 1
----> 1 with open('file.log') as file:
      2         read_data = file.read()

File ~/.pyenv/versions/3.11.5/Library/Frameworks/Python.framework/Versions/
3.11/lib/python3.11/site-packages/IPython/core/interactiveshell.py:286, in _
modified_open(file, *args, **kwargs)
    279 if file in {0, 1, 2}:
    280     raise ValueError(
    281         f"IPython won't let you open fd={file} by default "
    282         "as it is likely to crash IPython. If you know what you are
doing, "
    283         "you can use builtins' open."
    284     )
--> 286 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'file.log'
```

In [19]:
```
try:
    with open('file.log') as file:
        read_data = file.read()
except:
    print('Could not open file.log')
```

```
Could not open file.log
```

In [20]:
```
try:
    with open('file.log') as file:
        read_data = file.read()
except FileNotFoundError as fnf_error:
    print(fnf_error)
```

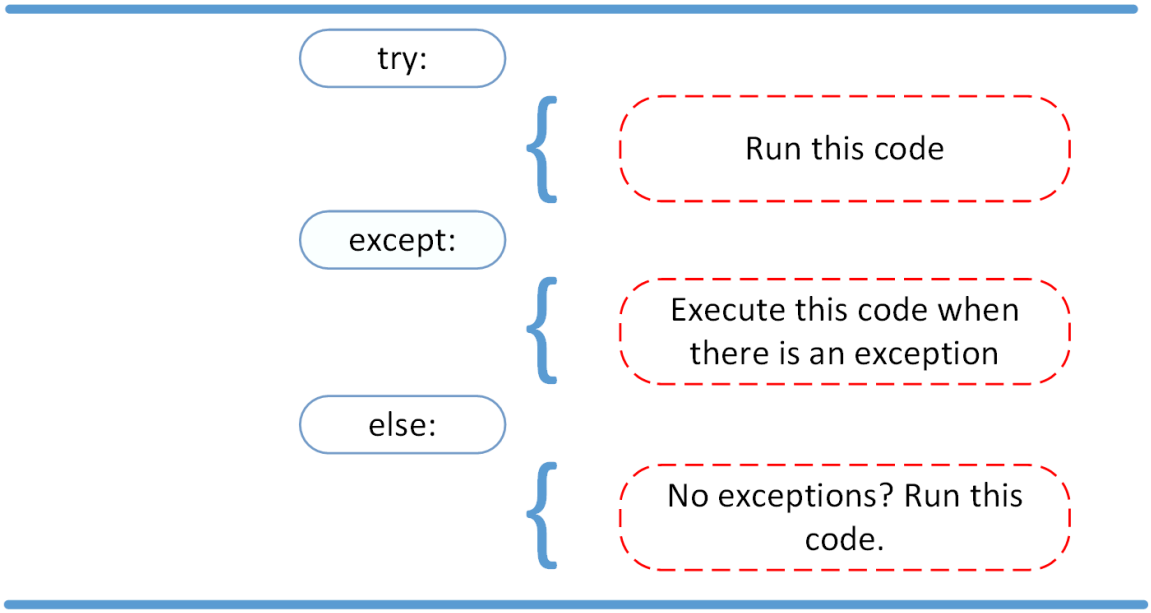
```
[Errno 2] No such file or directory: 'file.log'
```

จัดการกับ Exception ด้วย `try` - `except` - `else`

เราสามารถนำคีย์เวิร์ด `else` มาใช้ร่วมกับการดักจับ Exception ได้ โดยถ้าตรวจสอบไม่พบ Exception ใดๆ โปรแกรมจะทำงานตามคำสั่งที่ระบุไว้ในส่วนของ `else`

`else:` จะทำงานเมื่อไม่มี exception (error) เท่านั้น

โดยมีโครงสร้างการใช้คำสั่งดังนี้



In [21]:
```
x = [5, 12, 6, 9, 13]
n = int(input('กรุณาป้อนตัวเลข: '))
i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
z = x[i]/n
print(f'ผลลัพธ์ที่ได้คือ {z}')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[21], line 4
      2 n = int(input('กรุณาป้อนตัวเลข: '))
      3 i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
----> 4 z = x[i]/n
      5 print(f'ผลลัพธ์ที่ได้คือ {z}')

IndexError: list index out of range
```

In [22]:
```
# Fix this script

x = [5, 12, 6, 9, 13]
try:
    n = int(input('กรุณาป้อนตัวเลข: '))
    i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
    z = x[i]/n
except IndexError:
    print('ไม่พบตำแหน่งที่คุณระบุในlist')
except ZeroDivisionError:
    print('ตัวหารเป็นเลข 0 ไม่ได้')
except ValueError:
    print('ข้อมูลที่คุณป้อนไม่ใช่ตัวเลขจำนวนเต็ม')
else:
    print('ไม่พบข้อผิดพลาดของ Exception')
    print(f'ผลลัพธ์ที่ได้คือ {z}')
```

ไม่พบตำแหน่งที่คุณระบุในlist

In [24]:
```
# Fix this script

x = [5, 12, 6, 9, 13]
try:
    n = int(input('กรุณาป้อนตัวเลข: '))
    i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
    z = x[i]/n
except IndexError:
    print('ไม่พบตำแหน่งที่คุณระบุในlist')
except ZeroDivisionError:
    print('ตัวหารเป็นเลข 0 ไม่ได้')
except ValueError:
    print('ข้อมูลที่คุณป้อนไม่ใช่ตัวเลขจำนวนเต็ม')
else:
    print('ไม่พบข้อผิดพลาดของ Exception')
    print(f'ผลลัพธ์ที่ได้คือ {z}')
```

ตัวหารเป็นเลข 0 ไม่ได้

In [25]:
```
# Fix this script

x = [5, 12, 6, 9, 13]
try:
    n = int(input('กรุณาป้อนตัวเลข: '))
    i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
    z = x[i]/n
except IndexError:
    print('ไม่พบตำแหน่งที่คุณระบุในlist')
except ZeroDivisionError:
    print('ตัวหารเป็นเลข 0 ไม่ได้')
except ValueError:
    print('ข้อมูลที่คุณป้อนไม่ใช่ตัวเลขจำนวนเต็ม')
else:
    print('ไม่พบข้อผิดพลาดของ Exception')
    print(f'ผลลัพธ์ที่ได้คือ {z}')
```

ข้อมูลที่คุณป้อนไม่ใช่ตัวเลขจำนวนเต็ม

In [26]:
```
# Fix this script

x = [5, 12, 6, 9, 13]
try:
    n = int(input('กรุณาป้อนตัวเลข: '))
    i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
    z = x[i]/n
except IndexError:
    print('ไม่พบตำแหน่งที่คุณระบุในlist')
except ZeroDivisionError:
    print('ตัวหารเป็นเลข 0 ไม่ได้')
except ValueError:
    print('ข้อมูลที่คุณป้อนไม่ใช่ตัวเลขจำนวนเต็ม')
else:
    print('ไม่พบข้อผิดพลาดของ Exception')
    print(f'ผลลัพธ์ที่ได้คือ {z}')
```

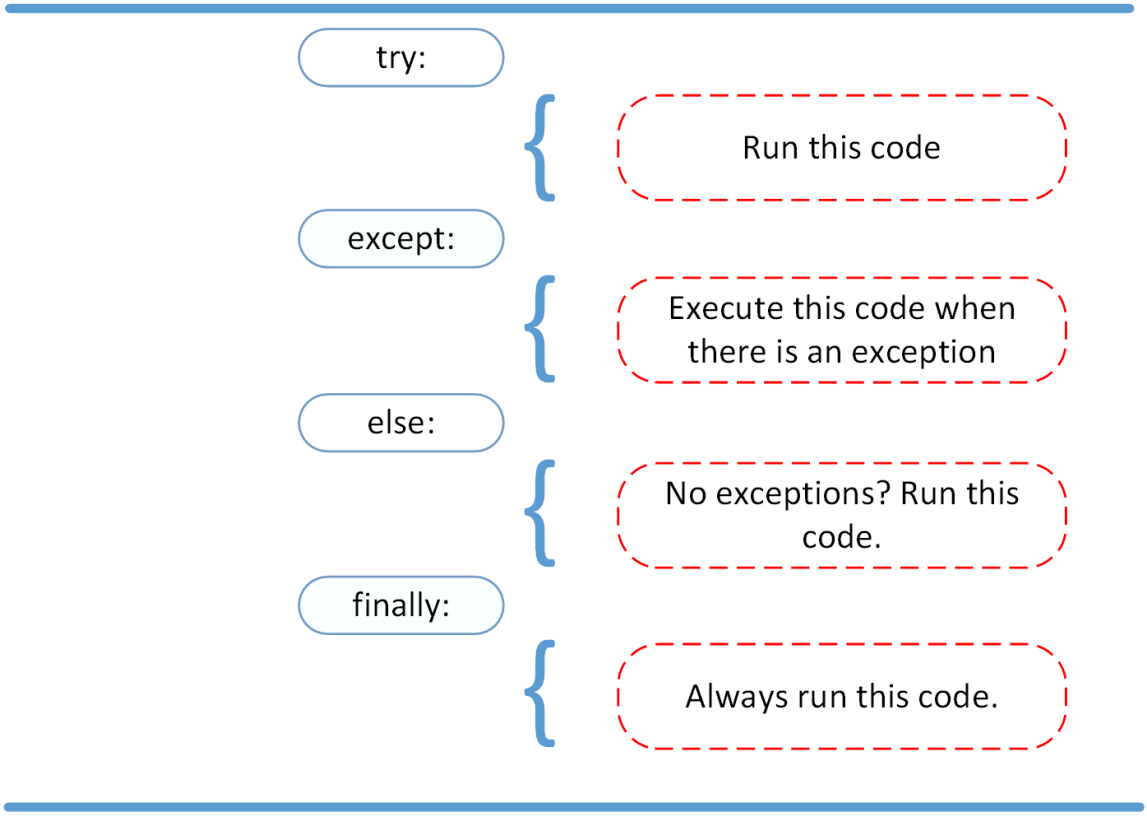
```
ไม่พบข้อผิดพลาดของ Exception
ผลลัพธ์ที่ได้คือ 2.0
```

จัดการกับ Exception ด้วย `try` - `except` - `finally`

เราสามารถนำคีย์เวิร์ด `finally` มาใช้ร่วมกับการดักจับ Exception ได้ โดยโปรแกรมจะทำงานตามคำสั่งที่ระบุไว้ในส่วนของ `finally` เสมอไม่ว่าจะตรวจสอบพบ Exception หรือไม่ก็ตาม

`finally:` เป็น block ที่จะทำงานตลอดเลย ไม่ว่าจะมี exception หรือไม่ก็ตาม

โดยมีโครงสร้างการใช้คำสั่งดังนี้

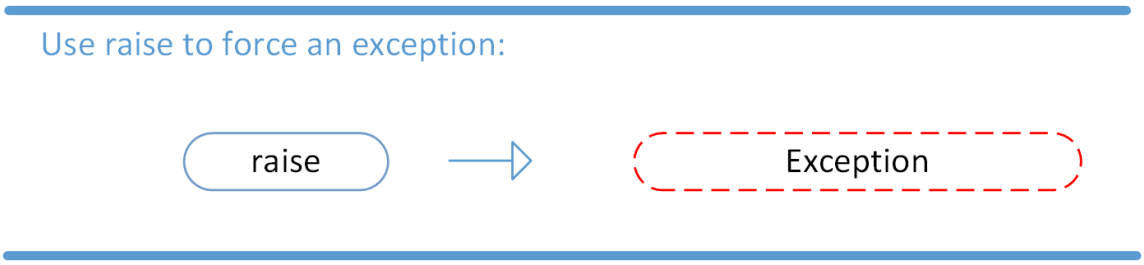


In [27]:
```python
x = [5, 12, 6, 9, 13]
try:
    n = int(input('กรุณาป้อนตัวเลข: '))
    i = int(input('กรุณาป้อนตำแหน่งข้อมูลในlist: '))
    z = x[i]/n
except IndexError:
    print('ไม่พบตำแหน่งที่คุณระบุในlist')
except ZeroDivisionError:
    print('ตัวหารเป็นเลข 0 ไม่ได้')
except ValueError:
    print('ข้อมูลที่คุณป้อนไม่ใช่ตัวเลขจำนวนเต็ม')
else:
    print('ไม่พบข้อผิดพลาดของ Exception')
    print(f'ผลลัพธ์ที่ได้คือ {z}')
finally:
    print('Goog Job')
```

```
ไม่พบข้อผิดพลาดของ Exception
ผลลัพธ์ที่ได้คือ 6.5
Goog Job
```

การสร้างข้อความแจ้งเตือนข้อผิดพลาดด้วยคำสั่ง `raise`

นอกจากเราใช้คำสั่ง `try` - `except` ตรวจจับคำสั่งโปรแกรมที่คาดว่าจะเกิดความผิดพลาด ขึ้น และแจ้งเตือนตามประเภทของ Exception แล้วยังมีคำสั่ง `raise` ให้เราใช้งานสำหรับ สร้างข้อความการแจ้งเตือนข้อผิดพลาดขึ้นมาใช้งานเอง เพื่อให้ทราบถึงสาเหตุที่เกิดขึ้นของข้อ ผิดพลาด ตัวอย่างเช่น การตรวจสอบเงื่อนไขว่าโปรแกรมทำงานถูกต้องตามที่กำนดไว้หรือไม่ โดยคำสั่ง `raise` มีรูปแบบการใช้งานดังนี้



In [29]:
```
x = int(input('Enter numbers 1-5 : '))
if x > 5:
    raise TypeError(f'x should not exceed 5. The value of x was: {x}')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 3
      1 x = int(input('Enter numbers 1-5 : '))
      2 if x > 5:
----> 3     raise TypeError(f'x should not exceed 5. The value of x was:
{x}')

TypeError: x should not exceed 5. The value of x was: 9
```

In [30]:
```
x = int(input('Enter numbers 1-5 : '))
if x > 5:
    raise Exception(f'x should not exceed 5. The value of x was: {x}')
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[30], line 3
      1 x = int(input('Enter numbers 1-5 : '))
      2 if x > 5:
----> 3     raise Exception(f'x should not exceed 5. The value of x was:
{x}')

Exception: x should not exceed 5. The value of x was: 8
```

In [31]:
```
day_dict = {1: 'อาทิตย์',
            2: 'จันทร์',
            3: 'อังคาร',
            4: 'พุธ',
```

```
            5: 'พฤหัสบดี',
            6: 'ศุกร์',
            7: 'เสาร์'}
x = int(input('กรุณากรอกข้อมูลวันในหนึ่งสัปดาห์ (1-7): '))
if x < 1 or x > 7:
    raise Exception(f'ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ {x}')
else:
    print(f'วันที่ {x} ในสัปดาห์ คือวัน {day_dict[x]}')
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[31], line 10
      8 x = int(input('กรุณากรอกข้อมูลวันในหนึ่งสัปดาห์ (1-7): '))
      9 if x < 1 or x > 7:
---> 10     raise Exception(f'ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ
{x}')
     11 else:
     12     print(f'วันที่ {x} ในสัปดาห์ คือวัน {day_dict[x]}')

Exception: ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ 0
```

In [32]:
```
day_dict = {1: 'อาทิตย์',
            2: 'จันทร์',
            3: 'อังคาร',
            4: 'พุธ',
            5: 'พฤหัสบดี',
            6: 'ศุกร์',
            7: 'เสาร์'}
x = int(input('กรุณากรอกข้อมูลวันในหนึ่งสัปดาห์ (1-7): '))
if x < 1 or x > 7:
    raise Exception(f'ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ {x}')
else:
    print(f'วันที่ {x} ในสัปดาห์ คือวัน {day_dict[x]}')
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[32], line 10
      8 x = int(input('กรุณากรอกข้อมูลวันในหนึ่งสัปดาห์ (1-7): '))
      9 if x < 1 or x > 7:
---> 10     raise Exception(f'ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ
{x}')
     11 else:
     12     print(f'วันที่ {x} ในสัปดาห์ คือวัน {day_dict[x]}')

Exception: ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ 9
```

In [33]:
```
day_dict = {1: 'อาทิตย์',
            2: 'จันทร์',
            3: 'อังคาร',
            4: 'พุธ',
            5: 'พฤหัสบดี',
            6: 'ศุกร์',
            7: 'เสาร์'}
x = int(input('กรุณากรอกข้อมูลวันในหนึ่งสัปดาห์ (1-7): '))
if x < 1 or x > 7:
    raise Exception(f'ข้อมูลวันในหนึ่งสัปดาห์ต้องมีค่า 1-7 เท่านั้น ค่าที่คุณป้อนคือ {x}')
```

```
else:
    print(f'วันที่ {x} ในสัปดาห์ คือวัน {day_dict[x]}')
```

วันที่ 5 ในสัปดาห์ คือวัน พฤหัสบดี

In [36]:
```
def func():
    x = input("ระบุตัวเลข :")
    if not x.isdigit():
        raise Exception("โปรดระบุตัวเลข")

    y = int(x)
    if y not in range(1,100):
        raise Exception("ค่าที่ระบุมากเกินไป")
    else:
        return y

try:
    y = func()
except Exception as error:
    print(error)
else:
    print('y =',y)
```

โปรดระบุตัวเลข

In [37]:
```
def func():
    x = input("ระบุตัวเลข :")
    if not x.isdigit():
        raise Exception("โปรดระบุตัวเลข")

    y = int(x)
    if y not in range(1,100):
        raise Exception("ค่าที่ระบุมากเกินไป")
    else:
        return y

try:
    y = func()
except Exception as error:
    print(error)
else:
    print('y =',y)
```

ค่าที่ระบุมากเกินไป

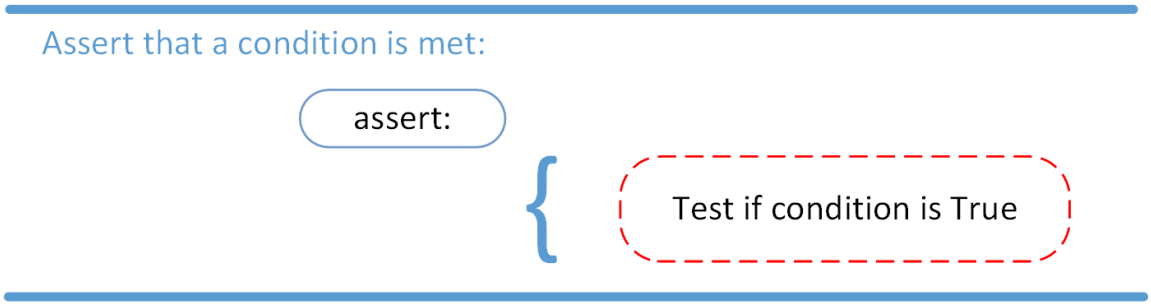
In [38]:
```
def func():
    x = input("ระบุตัวเลข :")
    if not x.isdigit():
        raise Exception("โปรดระบุตัวเลข")

    y = int(x)
    if y not in range(1,100):
        raise Exception("ค่าที่ระบุมากเกินไป")
    else:
        return y
```

```
    y = func()
except Exception as error:
    print(error)
else:
    print('y =',y)
```

```
y = 10
```

การยืนยันความถูกต้องด้วยคำสั่ง `assert`

นอกจากเราใช้คำสั่ง `try` - `except` ตรวจจับคำสั่งโปรแกรมที่คาดว่าจะเกิดความผิดพลาด ขึ้น และแจ้งเตือนตามประเภทของ Exception แล้วยังมีคำสั่ง `assert` เพื่อตรวจจับข้อผิดพลาดที่เกิดขึ้นก็เป็นอีกหนึ่งวิธีที่มีความสะดวก ซึ่งเป็นการยืนยันและเพื่อให้แน่ใจว่าคำสั่งโปรแกรมนั้นจะไม่มีโอกาสเกิดข้อผิดพลาดขึ้นอย่างแน่นอน ถ้าคำสั่ง `assert` ตรวจพบข้อผิดพลาดจะแสดงประเภท Exception ของ `AssertionError` ออกมา โดยมีรูปแบบการใช้งานดังนี้




```
assert เงื่อนไขที่จะไม่เกิดข้อผิดพลาด, ("ข้อความที่จะแสดงถ้าไม่ตรงตามเงื่อนไข")
```

In [39]:
```
x = 10
y = 20
assert x == y, 'ค่าตัวแปร x และ y ไม่เท่ากัน'
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[39], line 3
      1 x = 10
      2 y = 20
----> 3 assert x == y, 'ค่าตัวแปร x และ y ไม่เท่ากัน'

AssertionError: ค่าตัวแปร x และ y ไม่เท่ากัน
```

In [40]:
```
x = 10
y = 20
assert x != y, 'ค่าตัวแปร x และ y ไม่เท่ากัน'
print('x ไม่เท่ากับ y')
```

```
x ไม่เท่ากับ y
```

In [41]:
```
try:
    n = int(input('กรุณากรอกคะแนนไม่เกิน 30 คะแนน : '))
    assert n <= 30, 'bbb'
    f'คะแนนที่คุณกรอกคือ {n}')
```

```
except ValueError:
    print('คุณกรอกข้อมูลตัวเลขไม่ถูกต้อง')
except AssertionError:
    print('คุณกรอกค่าตัวเลขเกินจำนวน')
```

คุณกรอกค่าตัวเลขเกินจำนวน

In [42]:
```
try:
    n = int(input('กรุณากรอกคะแนนไม่เกิน 30 คะแนน : '))
    assert n <= 30, 'bbb'
    print(f'คะแนนที่คุณกรอกคือ {n}')
except ValueError:
    print('คุณกรอกข้อมูลตัวเลขไม่ถูกต้อง')
except AssertionError:
    print('คุณกรอกค่าตัวเลขเกินจำนวน')
```

คุณกรอกข้อมูลตัวเลขไม่ถูกต้อง

In [43]:
```
try:
    n = int(input('กรุณากรอกคะแนนไม่เกิน 30 คะแนน : '))
    assert n <= 30, 'bbb'
    print(f'คะแนนที่คุณกรอกคือ {n}')
except ValueError:
    print('คุณกรอกข้อมูลตัวเลขไม่ถูกต้อง')
except AssertionError:
    print('คุณกรอกค่าตัวเลขเกินจำนวน')
```

คะแนนที่คุณกรอกคือ 29

In [44]:
```
def times_ten(number):
    return number * 100

result = times_ten(20)
assert result == 2000, 'Expected times_ten(20) to return 200, instead got '
```

In [46]:
```
def times_ten(number):
    return number * 100

result = times_ten(10)
assert result == 2000, 'Expected times_ten(20) to return 200, instead got '
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[46], line 5
      2     return number * 100
      4 result = times_ten(10)
----> 5 assert result == 2000, 'Expected times_ten(20) to return 200, instea
d got ' + str(result)

AssertionError: Expected times_ten(20) to return 200, instead got 1000
```

Loading [MathJax]/extensions/Safe.js

In [47]:
```
def times_ten(number):
    return number * 100

result = times_ten(20)
if  result != 2000:
    raise Exception('Expected times_ten(20) to return 200, instead got ' + s
```

In [48]:
```
def times_ten(number):
    return number * 100

result = times_ten(30)
if  result != 2000:
    raise Exception('Expected times_ten(20) to return 200, instead got ' + s
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Cell In[48], line 6
      4 result = times_ten(30)
      5 if  result != 2000:
----> 6     raise Exception('Expected times_ten(20) to return 200, instead g
ot ' + str(result))

Exception: Expected times_ten(20) to return 200, instead got 3000
```

Summing Up

After seeing the difference between syntax errors and exceptions, you learned about various ways to raise, catch, and handle exceptions in Python. In this article, you saw the following options:

- `raise` allows you to throw an exception at any time.
- `assert` enables you to verify if a certain condition is met and throw an exception if it isn't.
- In the `try` clause, all statements are executed until an exception is encountered.
- `except` is used to catch and handle the exception(s) that are encountered in the try clause.
- `else` lets you code sections that should run only when no exceptions are encountered in the try clause.
- `finally` enables you to execute sections of code that should always run, with or without any previously encountered exceptions.

Your turn!

To check your understanding, try to complete the exercise!

Exercise

1

Write a function `division()` that accepts two arguments `x` and `y` and return the value of `x/y`.

The function must be able to catch an exception such as `ZeroDivisionError`, `ValueError`, or any unknown error you might come across when you are doing the division operation.

In [2]:
```
def division(x, y):
    pass
```

In [6]:
```
# This is some test cases. Run this cell to check!

for x in range(-2, 3):
    for y in range(-2, 3):
        z = division(x, y)
        print(f'{x}/{y} = {z}')
print('Test done.')
```

```
-2/-2 = None
-2/-1 = None
-2/0 = None
-2/1 = None
-2/2 = None
-1/-2 = None
-1/-1 = None
-1/0 = None
-1/1 = None
-1/2 = None
0/-2 = None
0/-1 = None
0/0 = None
0/1 = None
0/2 = None
1/-2 = None
1/-1 = None
1/0 = None
1/1 = None
1/2 = None
2/-2 = None
2/-1 = None
2/0 = None
2/1 = None
2/2 = None
Test done.
```

In [5]:
```
# This is another test cases. Run this cell to check!

for x in [11, 22.5, -44, 'Hello', -11.25]:
    for y in range(-5, 5):
        z = division(x, y)
        print(f'{x}/{y} = {z}')
print('Test done.')
```

```
11/-5 = None
11/-4 = None
11/-3 = None
11/-2 = None
11/-1 = None
11/0 = None
11/1 = None
11/2 = None
11/3 = None
11/4 = None
22.5/-5 = None
22.5/-4 = None
22.5/-3 = None
22.5/-2 = None
22.5/-1 = None
22.5/0 = None
22.5/1 = None
22.5/2 = None
22.5/3 = None
22.5/4 = None
-44/-5 = None
-44/-4 = None
-44/-3 = None
-44/-2 = None
-44/-1 = None
-44/0 = None
-44/1 = None
-44/2 = None
-44/3 = None
-44/4 = None
Hello/-5 = None
Hello/-4 = None
Hello/-3 = None
Hello/-2 = None
Hello/-1 = None
Hello/0 = None
Hello/1 = None
Hello/2 = None
Hello/3 = None
Hello/4 = None
-11.25/-5 = None
-11.25/-4 = None
-11.25/-3 = None
-11.25/-2 = None
-11.25/-1 = None
-11.25/0 = None
-11.25/1 = None
-11.25/2 = None
-11.25/3 = None
-11.25/4 = None
Test done.
```

2

Write the guessing number game! Here is the rule:

- The player have to guess the correct number which is randomly chosen by the computer.
- At the beginning, computer choose an integer number randomly between intergers `a` and `b`. This is the number that the player has to guess in this game.
- The computer lets the player guesses until getting the correct number.

In this game, you have to handle some stupid inputs like guessing too low (less than `a`) or too high (greater than `b`) numbers with raising an error then let the player plays continue to play game

In [ ]:
```python
import random

min_number = 0
max_number = 100

the_correct_number = random.randint(min_number, max_number)

game_run = True
# print(the_correct_number) # uncomment this for cheating the game

print('Welcome to the gussing game')
while game_run:
    x = int(input('Enter your guessing: '))
    if x == the_correct_number:
        print('Congratulations! Your guessing is correct')
        game_run = False
    else:
        print(f'Try again, it is not {x}')

print('Thanks for playing')
```

3

Write a program that prompts the user for a number, and then calculates the square of that number. If the user enters a value that is not a number (such as a letter or symbol), the program should display an error message and prompt the user to enter a valid number again.

In [ ]:
```python
# Fix this script

num = float(input("Enter a number: "))
result = num ** 2
print(f'The square of {num} is {result}')
```

4

Create a program that reads a file and prints the contents of the file to the screen. If the file does not exist or if the file cannot be opened for reading, the program should display an appropriate error message and exit.

In [ ]:
```python
# Fix this script

with open("example.txt", "r") as file:
    for line in file:
        print(line)
```
