

Exceptions အကြောင်း

ခြင်းချက် Error (Exception) တွေကို ပြီးခဲ့တဲ့ကုန်တွေမှာ မြင်တွေ့ခဲ့ပါပြီ။ ကုန်မှာ အမှားအယွင်း တစ်ခုခုပါတဲ့အခါ ဒီလို Exception တွေ ဖြစ်ပေါ်လာပါတယ်။ Exception တစ်ခုဖြစ်ပေါ်လာတာနဲ့ ပရိုဂရမ်က run နေတာကို ချက်ချင်းရပ်လိုက်ပါတယ်။ အောက်မှာနမူနာပြထားတာက 7 ကို 0 နဲ့ စားဖို့ကြိုးစားတဲ့အတွက် ဖြစ်လာတဲ့ ZeroDivisionError exception ဖြစ်ပါတယ်။

```
test_1.py x
1 num1 = 7
2 num2 = 0
3 print(num1/num2)
.

Shell x
Python 3.7.7 (bundled)
>>> %Run test_1.py
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 3, in <module>
    print(num1/num2)
ZeroDivisionError: division by zero
>>>
```

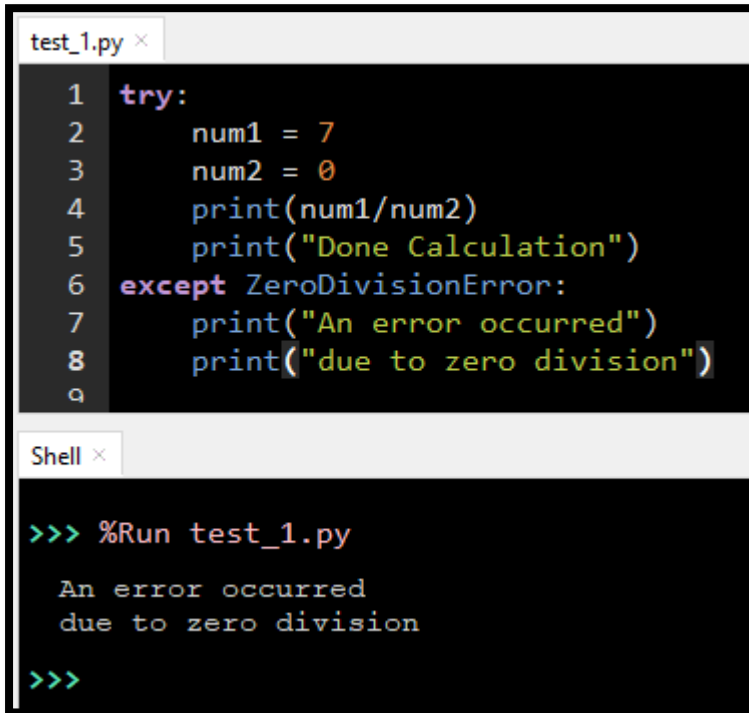
မတူတဲ့အကြောင်းရင်းတွေအတွက် မတူညီတဲ့ Exceptions တွေ ဖြစ်ပေါ်ပါတယ်။ အတွေ့ရများတဲ့ Exceptions တွေကတော့ -

- **ImportError:** import ပြုလုပ်လို့မရသောအခါ
- **IndexError:** list ပြင်ပမှာရှိတဲ့/လက်လှမ်းမမီတဲ့နံပါတ်တစ်ခုနဲ့ list တန်ဖိုးကို index လုပ်ပြီးရယူဖို့ ကြိုးစားတဲ့အခါ
- **NameError:** နာမည်နဲ့တန်ဖိုးသတ်မှတ်ပေးထားခြင်းမရှိတဲ့ variable အမည်တစ်ခု ပါလာသောအခါ
- **SyntaxError:** ကုန်တည်ဆောက်ပုံစနစ်မကျ၍ စနစ်တကျ တဆင့်စီ run လို့ မရသောအခါ
- **TypeError:** သတ်မှတ်ထားတဲ့ဒေတာအမျိုးအစား type မဟုတ်တဲ့ တန်ဖိုး value ကို ထည့်သွင်းပြီး function ကို အသုံးပြုသောအခါ
- **ValueError:** သတ်မှတ်ထားတဲ့ဒေတာအမျိုးအစား type ဟုတ်သော်လည်း မမှန်ကန်မသင့်တော်တဲ့ တန်ဖိုး value ကို function ထဲမှာ ထည့်သွင်းအသုံးပြုသောအခါ

Python မှာ ZeroDivisionError နဲ့ OSError တို့လို built-in exceptions တွေအများအပြားပါပါတယ်။
Third – party library တွေကတော့ ကိုယ်ပိုင် exceptions တွေကို အသုံးပြုပါတယ်။

Exception Handling

Exception တွေကို ဖြေရှင်းဖို့ **try/except** statement ကို အသုံးပြုပါတယ်။ **try** ကုဒ်အစုအဝေး (**try block**) ထဲမှာ **exception** ဖြစ်လာနိုင်ခြေရှိတဲ့ ကုဒ်တွေကို ထည့်သွင်းထားပါတယ်။ **exception** ဖြစ်လာပြီဆိုတာနဲ့ **try** block က အလုပ်မလုပ်တော့ဘဲ ရပ်သွားပြီး **except** block ထဲမှာရှိတဲ့ ကုဒ်တွေကို ဆက်လက် run မှာ ဖြစ်ပါတယ်။ Error မတွေ့လို့ **exception** ဖြစ်မလာဘူးဆိုရင်တော့ **try** block ထဲက ကုဒ်တွေကိုပဲ run သွားမှာဖြစ်ပြီး **except** ထဲက ကုဒ်တွေကို run မှာ မဟုတ်ပါဘူး။ အောက်ပါပုံကို ကြည့်ပါ။



```
test_1.py x
1 try:
2     num1 = 7
3     num2 = 0
4     print(num1/num2)
5     print("Done Calculation")
6 except ZeroDivisionError:
7     print("An error occurred")
8     print("due to zero division")
9

Shell x

>>> %Run test_1.py
An error occurred
due to zero division
>>>
```

(အပေါ်ကပုံထဲမှာ ဖြစ်ပေါ်လာနိုင်တဲ့ **ZeroDivisionError** ကို **except** ထဲမှာ ဖော်ပြထားပါတယ်)

try statement တစ်ခုမှာ မတူညီတဲ့ error တွေကို ဖြေရှင်းဖို့အတွက် **except** block တစ်ခုထက်ပိုပြီး ပါဝင်နိုင်ပါတယ်။ **except** block တစ်ခုတည်းမှာလည်း **exception** တွေအများကြီး ပါနိုင်ပါတယ်။ **exception** တွေကို () ထဲမှာ ထည့်သွင်းရေးသားရပါတယ်။

```
test_1.py x
1 try:
2     variable = 10
3     print(variable + "hello")
4     print(variable / 2)
5 except ZeroDivisionError:
6     print("Divided by zero")
7 except (ValueError, TypeError):
8     print("Error occurred")
9

Shell x
>>> %Run test_1.py
Error occurred
>>>
```

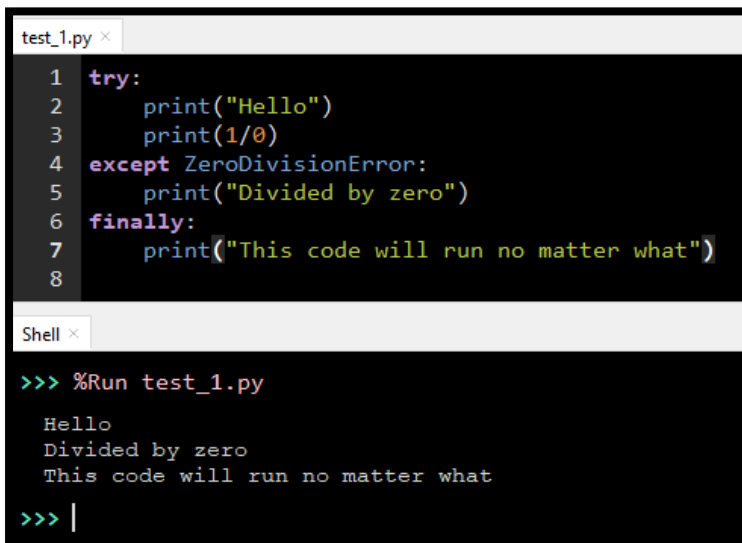
except statement ထဲမှာ **exception** သတ်မှတ်ချက်ကိုမဖော်ပြဘဲ ဒီအတိုင်း **except:** ဆိုပြီး အသုံးပြုရင် Error တွေအကုန်လုံးကို ဖမ်းပေးတော့မှာ ဖြစ်ပါတယ်။ ဒီလို **except:** တစ်မျိုးတည်းရေးတာကို အရံအနေနဲ့ အသုံးပြုသင့်ပါတယ်။ ဘာကြောင့်လဲဆိုတော့ သူတို့က မမျှော်လင့်တဲ့ Error တွေကို ဖမ်းထားပေးနိုင်ပြီး Programming အမှားအယွင်းတွေကို အသေးစိတ်မဖော်ပြတော့ဘဲ ဖုံးကွယ်ထားပေးနိုင်တဲ့အတွက် ဖြစ်ပါတယ်။

```
test_1.py x
1 try:
2     word = "spam"
3     print(word / 0)
4 except:
5     print("An error occurred")
6

Shell x
>>> %Run test_1.py
An error occurred
>>> |
```

finally

ဘယ်လို error ချိုးပဲတက်တက် အချို့ကုဒ်တွေကို ဆက်လက်အလုပ်လုပ်စေဖို့ **finally** statement ကို အသုံးပြုနိုင်ပါတယ်။ **finally** statement ကို **try/except** statement ရဲ့ အောက်ခြေမှာ ရေးသားပါတယ်။ **finally** statement ထဲက ကုဒ်တွေက **try:** , **except:** ထဲက ကုဒ်တွေကို run ပြီးတဲ့နောက်မှာ အမြဲတမ်း run ပေးမှာဖြစ်ပါတယ်။ **try:** ကိုပဲ run ပြီး **except:** ကို မ run တာပဲ ဖြစ်ဖြစ် **try:** ကို မ run ဘဲ **except:** ကို run တာပဲဖြစ်ဖြစ် **finally:** ကတော့ အမြဲတမ်း run နေမှာ ဖြစ်ပါတယ်။



```
test_1.py x
1 try:
2     print("Hello")
3     print(1/0)
4 except ZeroDivisionError:
5     print("Divided by zero")
6 finally:
7     print("This code will run no matter what")
8

Shell x
>>> %Run test_1.py
Hello
Divided by zero
This code will run no matter what
>>> |
```

အကယ်၍ **except:** ကနေ မဖမ်းမိဘဲ Error တစ်ခုခု တက်ခဲ့ရင်လည်း **finally** ထဲက ကုဒ်ကတော့ ဆက်လက် run နေမှာ ဖြစ်ပါတယ်။

```

test_1.py ×
1 try:
2     print(1)
3     print(10/0)
4 except ZeroDivisionError:
5     print(unknown_var)
6 finally:
7     print("This is executed last")

Shell ×

>>> %Run test_1.py
1
This is executed last
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 3, in <module>
    print(10/0)
ZeroDivisionError: division by zero

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 5, in <module>
    print(unknown_var)
NameError: name 'unknown_var' is not defined

>>>

```

Raising Exceptions

`raise` statement ကို အသုံးပြုပြီး `exceptions` တွေကို ဖန်တီးနိုင်ပါတယ်။

```

test_1.py ×
1 print(1)
2 raise ValueError
3

Shell ×

>>> %Run test_1.py
1
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 2, in <module>
    raise ValueError
ValueError

>>> |

```

`raise` ကိုအသုံးပြုတဲ့အခါမှာ မိမိဖြစ်စေချင်တဲ့ `exception type` ကို သတ်မှတ်ပေးဖို့လိုပါတယ်။

`exceptions` တွေကို ဖန်တီးတဲ့အခါမှာ အသေးစိတ်ဖော်ပြချက်တွေကိုထည့်သွင်းပြီးတော့လည်း ဖန်တီးနိုင်ပါတယ်။

```
test_1.py x
1 name = "123"
2 raise NameError("Invalid name!")
3

Shell x
>>> %Run test_1.py
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 2, in <module>
    raise NameError("Invalid name!")
NameError: Invalid name!
>>> |
```

except: ကိုအသုံးပြုပြီး error မှန်သမျှကို ဖမ်းထားပြီးတဲ့အခါမှာ ဘာက error တက်မှန်းမသိတဲ့အတွက် Programming အမှားအယွင်းတွေကို ဖုံးထားသလိုဖြစ်နေတယ်လို့ ပြောခဲ့ပါတယ်။ ဘာ error တက်သွားတယ်ဆိုတာကို သိဖို့အတွက် **raise** ကို သုံးနိုင်ပါတယ်။ အောက်ပါပုံကို ကြည့်ပါ။

```
test_1.py x
1 try:
2     num = 5/0
3 except:
4     print("An error occurred")
5     raise
6

Shell x
>>> %Run test_1.py
An error occurred
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 2, in <module>
    num = 5/0
ZeroDivisionError: division by zero
>>>
```

Assertions

assertion ဆိုတာက ပရိုဂရမ်ကို run တဲ့အခါမှာ ပရိုဂရမ်ထဲကကုဒ်တွေကို စမ်းသပ် run ကြည့်ဖို့ အသုံးပြုပါတယ်။ **assertion** ပါဝင်တဲ့ ကုဒ်စာကြောင်းဖော်ပြချက် (expression) ကို စမ်းသပ်ကြည့်ပြီး ရလဒ်က **False** ဖြစ်နေ (မှားယွင်းနေခဲ့ရင်) **exception** တစ်ခု ဖြစ်ပေါ်လာမှာ ဖြစ်ပါတယ်။

assertion ကို အသုံးပြုဖို့အတွက် **assert** statement နဲ့ ရေးသားရပါတယ်။

```

test_1.py ×
1 print(1)
2 assert 2 + 2 == 4
3
4 print(2)
5 assert 1 + 1 == 3
6
7 print(3)

Shell ×
>>> %Run test_1.py
1
2
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 5, in <module>
    assert 1 + 1 == 3
AssertionError
>>>

```

`assert` statement ကိုအသုံးပြုတဲ့အခါမှာ `AssertionError` ဖြစ်လာခဲ့ရင် အသေးစိတ်ဖော်ပြချက်နဲ့ ဖော်ပြပေးဖို့အတွက် ဒုတိယ `argument` ကို ထည့်သွင်းပေးနိုင်ပါတယ်။

```

test_1.py ×
1 temp = -10
2 assert (temp >= 0), "Colder than absolute zero!"

Shell ×
>>> %Run test_1.py
Traceback (most recent call last):
  File "D:\Job\Python Myanmar Syllabus\test 1.py", line 2, in <module>
    assert (temp >= 0), "Colder than absolute zero!"
AssertionError: Colder than absolute zero!
>>> |

```