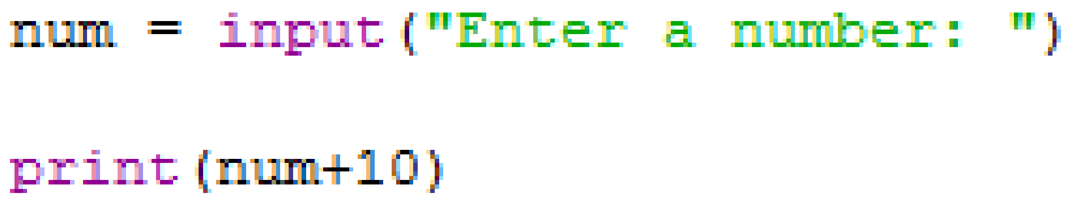
Ch10: Error and Exceptions

305172 Computer Programming Laboratory
Jiraporn Pooksook
Naresuan University

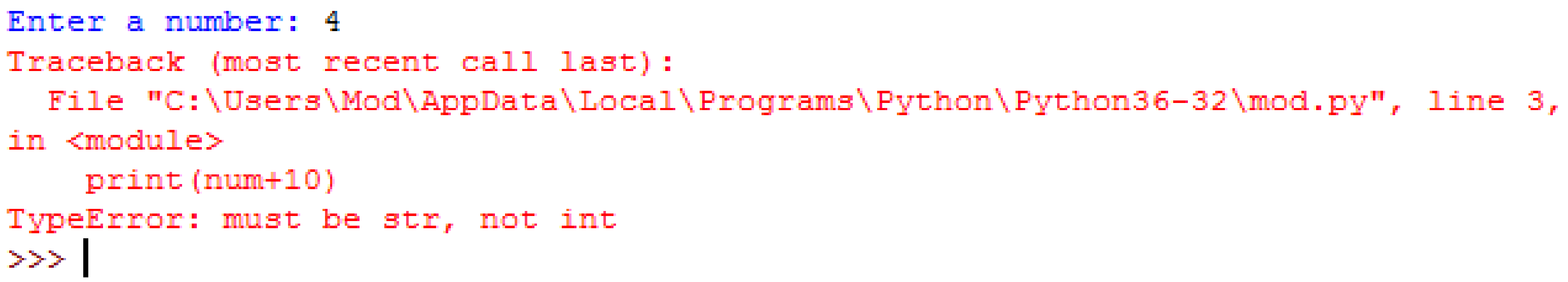
Syntax Errors

```
>>> while True print('Hello world')
  File "<stdin>", line 1
    while True print('Hello world')
                   ^
SyntaxError: invalid syntax
```

Exceptions



```
num = input("Enter a number: ")

print(num+10)
```



```
Enter a number: 4
Traceback (most recent call last):
  File "C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py", line 3,
in <module>
    print(num+10)
TypeError: must be str, not int
>>> |
```

Handling Exceptions

```
num = input("Enter a number: ")

try:
    print(num+10)
    print("after addition")
except TypeError:
    print("Must be type number. !!!")
```

Execute first

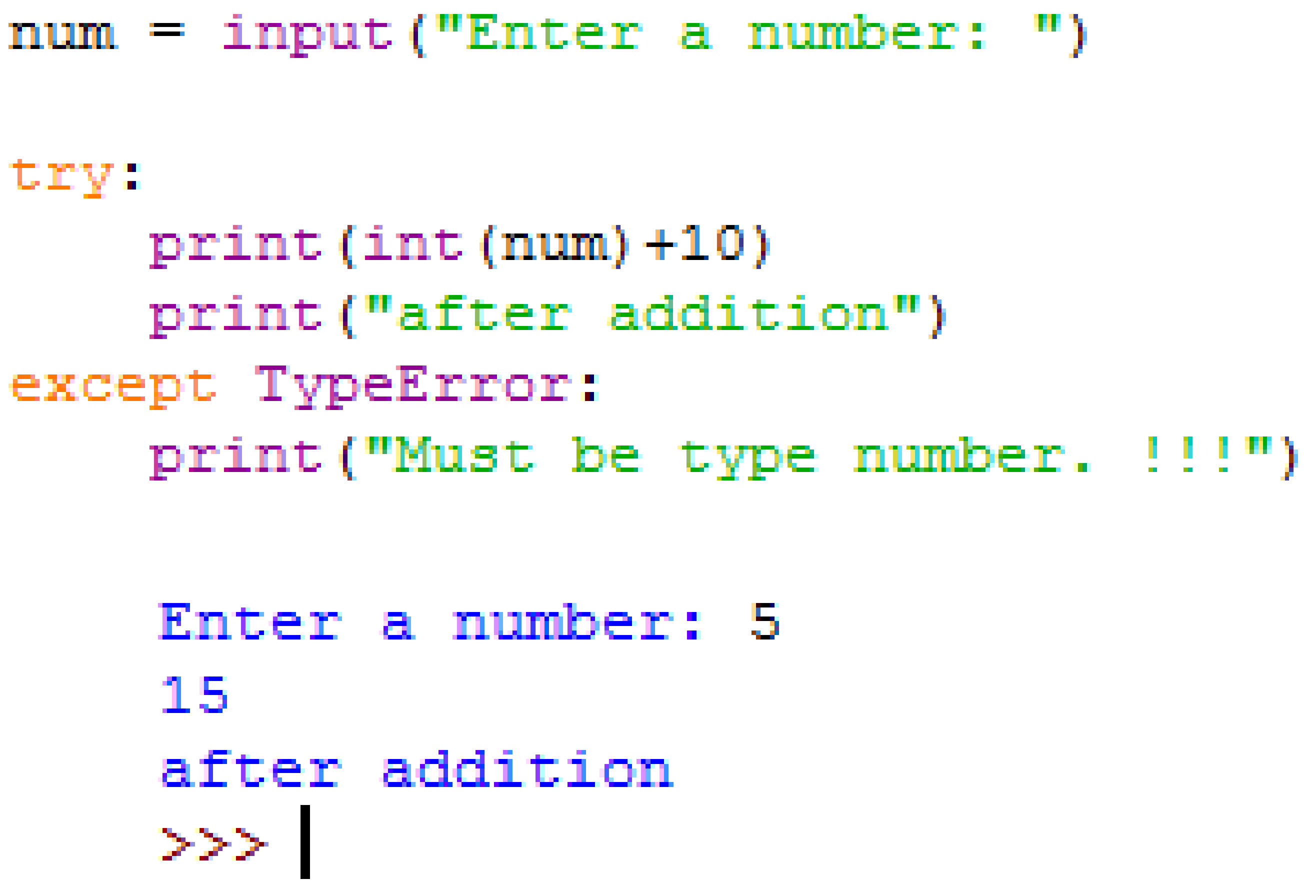
Execute when error

```
Enter a number: 5
Must be type number. !!!
>>>
```

Handling Exceptions

Execute first

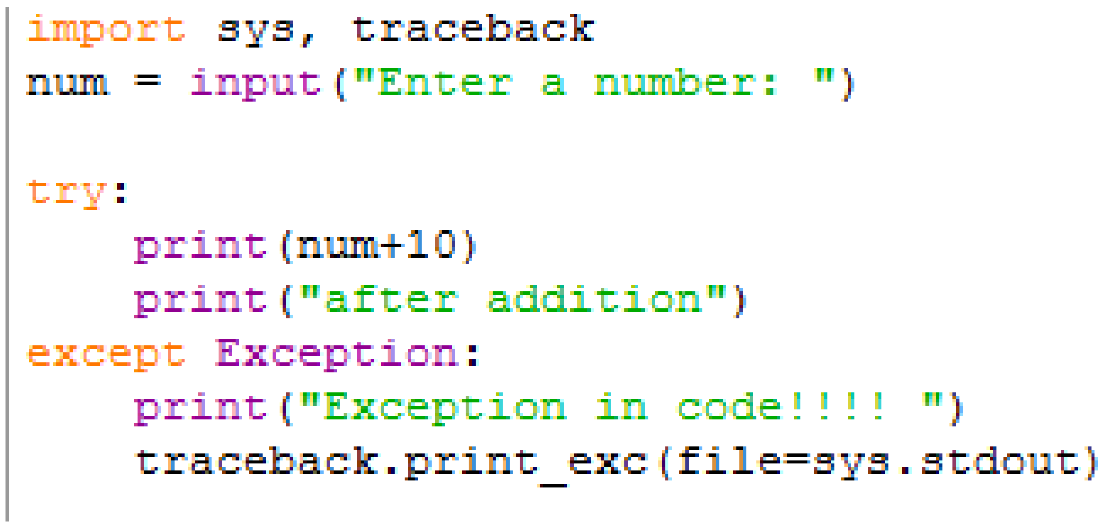
Execute when error



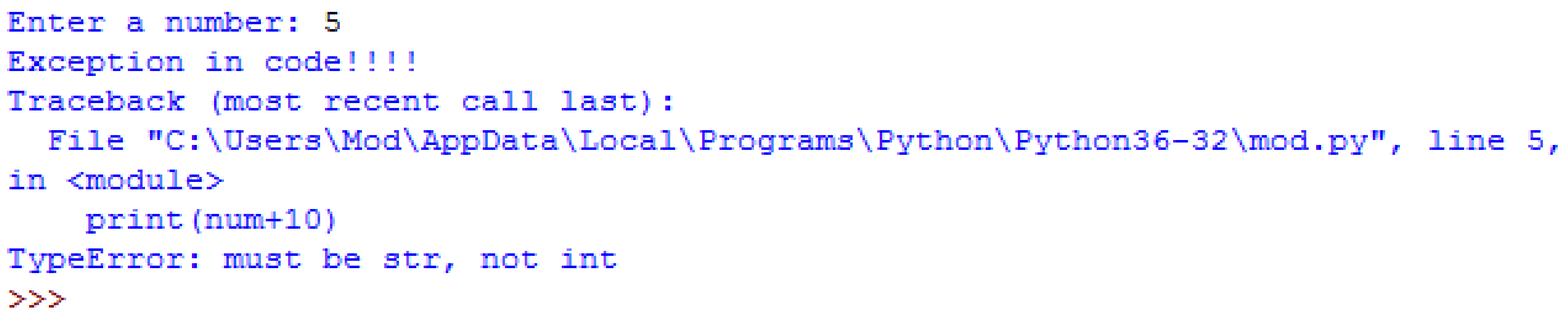
```
num = input("Enter a number: ")

try:
    print(int(num)+10)
    print("after addition")
except TypeError:
    print("Must be type number. !!!")
```

```
Enter a number: 5
15
after addition
>>> |
```

Handling Exceptions



```python
import sys, traceback
num = input("Enter a number: ")

try:
    print(num+10)
    print("after addition")
except Exception:
    print("Exception in code!!!! ")
    traceback.print_exc(file=sys.stdout)
```



```
Enter a number: 5
Exception in code!!!!
Traceback (most recent call last):
  File "C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py", line 5,
in <module>
    print(num+10)
TypeError: must be str, not int
>>>
```

Error Types

- Reference: https://docs.python.org/3/library/exceptions.html
- https://www.tutorialsteacher.com/python/error-types-in-python

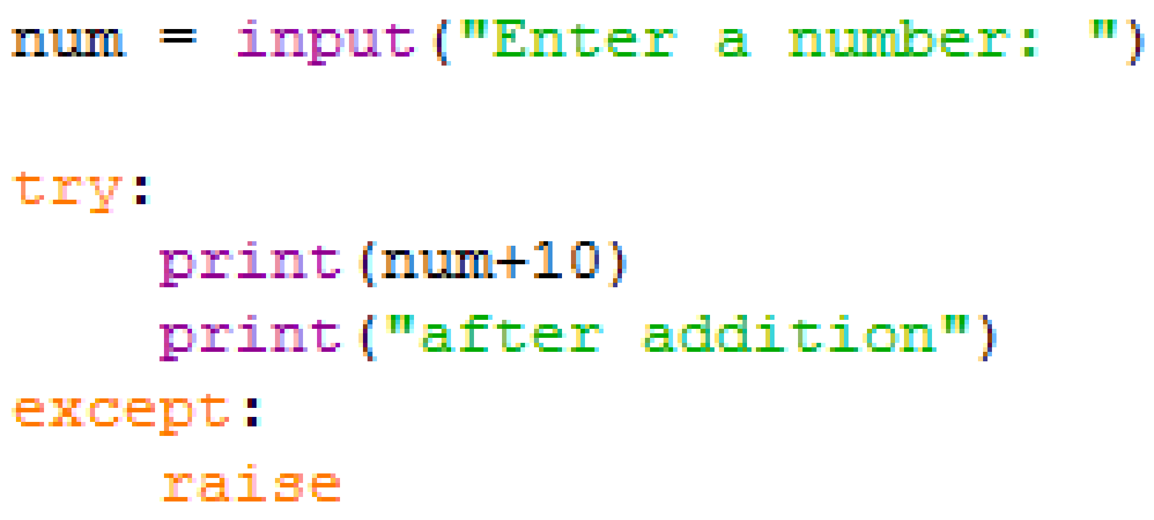
Exception	Description
ImportError	Raised when the imported module is not found.
NameError	Raised when a variable is not found in the local or global scope.
TypeError	Raised when a function or operation is applied to an object of an incorrect type.
ValueError	Raised when a function gets an argument of correct type but improper value.

Handling Exceptions

```
import sys, traceback
num = input("Enter a number: ")

try:
    print(num+10)
    print("after addition")
except ValueError:
    print("Cannot convert data to integer")
except:
    print("Unexpected error: ",sys.exc_info()[0])
    print(traceback.print_exc(file=sys.stdout))
```

```
Enter a number: 5
Unexpected error:  <class 'TypeError'>
Traceback (most recent call last):
  File "C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py", line 5,
in <module>
    print(num+10)
TypeError: must be str, not int
None
>>>
```

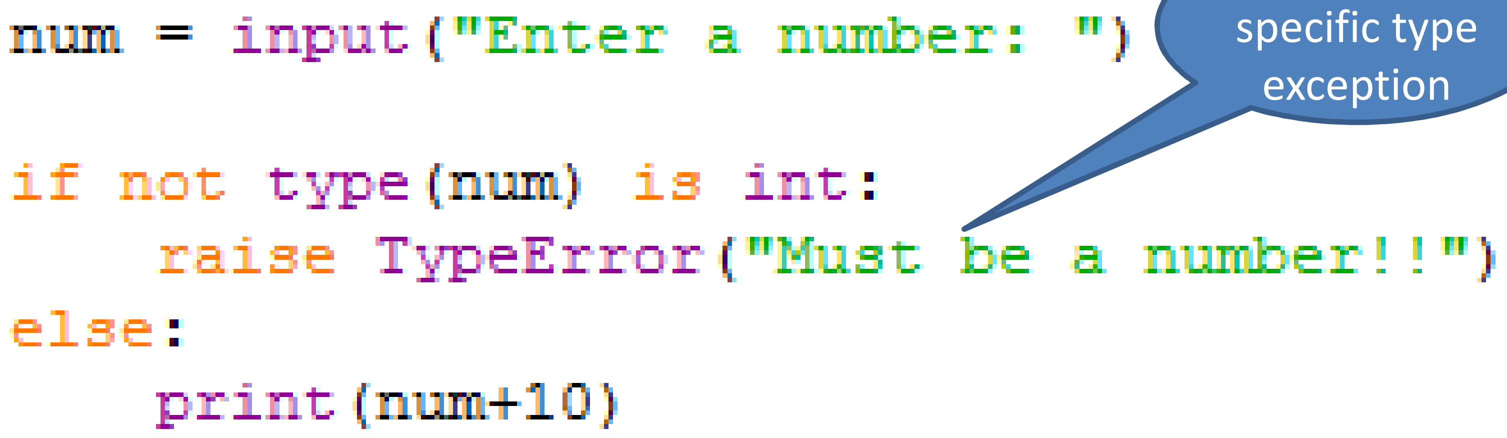
Raise Exception



```
num = input("Enter a number: ")

try:
    print(num+10)
    print("after addition")
except:
    raise
```

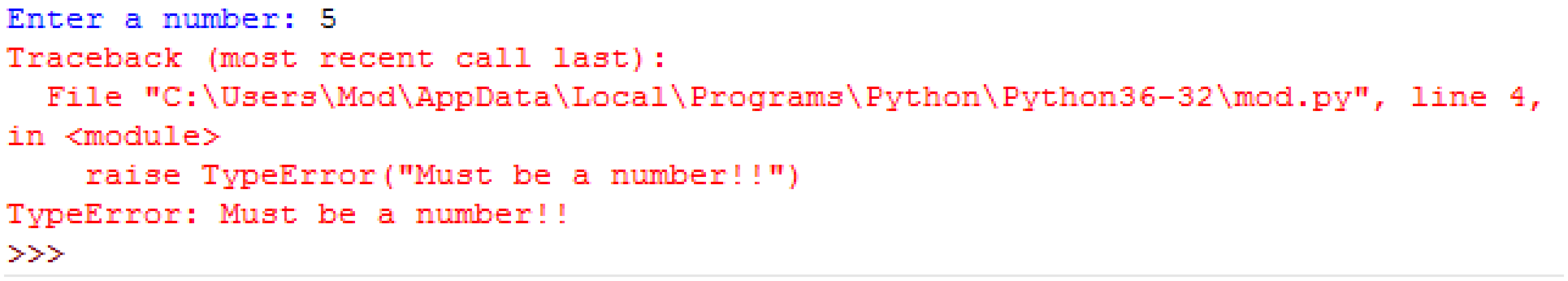
Raise exception without handler

Raise Exception



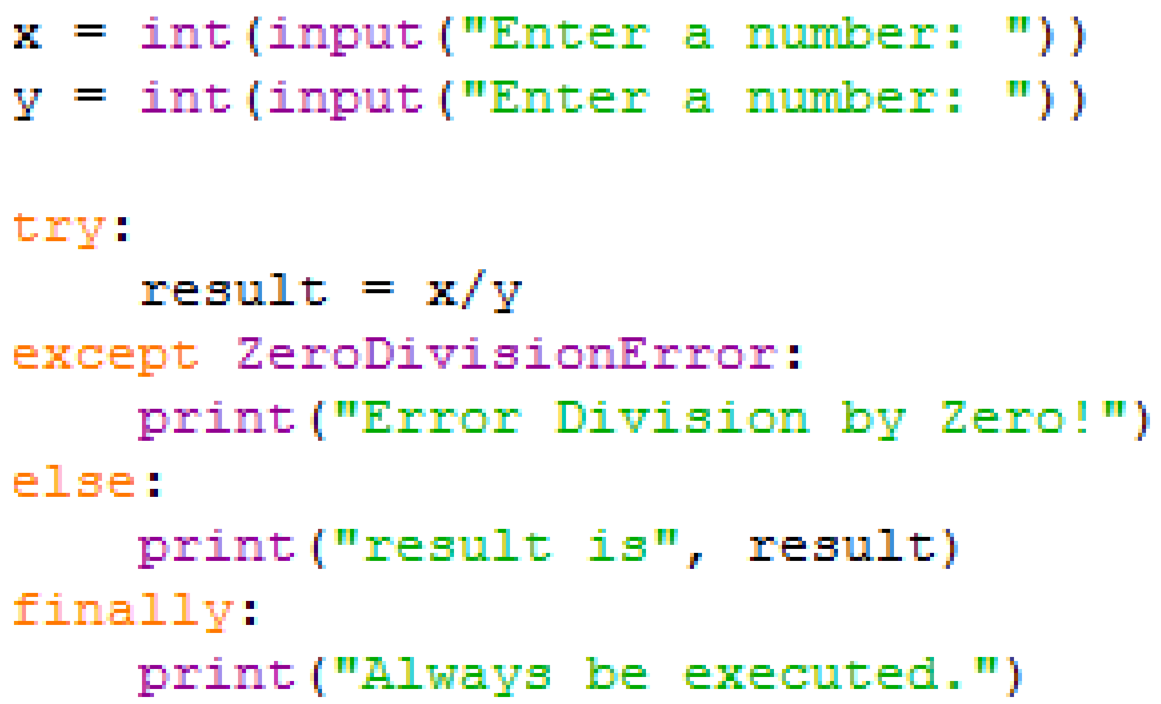
```
num = input("Enter a number: ")

if not type(num) is int:
    raise TypeError("Must be a number!!")
else:
    print(num+10)
```

Raise specific type exception



```
Enter a number: 5
Traceback (most recent call last):
  File "C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py", line 4,
in <module>
    raise TypeError("Must be a number!!")
TypeError: Must be a number!!
>>>
```

Exception Handling



```
x = int(input("Enter a number: "))
y = int(input("Enter a number: "))

try:
    result = x/y
except ZeroDivisionError:
    print("Error Division by Zero!")
else:
    print("result is", result)
finally:
    print("Always be executed.")
```