

Multi-dimensional View of Python

Exception

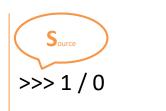
ZHANG Dazhuang

Department of Computer Science and Technology

Department of University Basic Computer Teaching

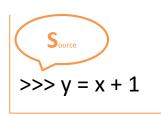
Data Processing Using Python

EXCEPTION



Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>
1/0

ZeroDivisionError: division by zero



Traceback (most recent call last):

File "<pyshell#12>", line 1, in

<module>
y = x + 1

NameError: name 'x' is not defined

exception objects stand for different exception situations

Show the exception class>> dir(_builtins__)

Name	Description
BaseException	The base class for all built-
	in exceptions
Exception	The base class for common
	exceptions
AttributeError	Raised when an attribute reference
	or assignment fails
IndexError	Raised when a sequence subscript
	is out of range.
I0Error	Raised when Input or output
	fails.
KeyboardInterrupt	Raised when the user hits the
	interrupt key (normally
	Control-C or Delete)

Show the exception class>> dir(_builtins__)

Name	Description
KeyError	Raised when a mapping (dictionary)
	key is not found in the set of
	existing keys.
NameError	Raised when a local or global name
	is not found
SyntaxError	Raised when the parser encounters
	a syntax error
TypeError	Raised when an operation or
	function is applied to an object of
	inappropriate type.

Show the exception class>> dir(_builtins__)

Name	Description
ValueError	Raised when a built-in operation or
	function receives an argument that
	has the right type but an
	inappropriate value
ZeroDivisionError	Raised when the second argument
	of a division or modulo operation is
	zero

Handling Exceptions

```
if y != 0:
    print(x / y)
else:
    print('division by zero')
```

try-except

```
# Filename: exception1.py
num1 = int(input('Enter the first number: '))
num2 = int(input('Enter the second number: '))
print(num1 / num2)

Enter the first number: a
```

Traceback (most recent call last):

File "C:\Python\programs\exception1.py", line 1, in <module>

num1 = int(input('Enter the first number: '))

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

try-except statement

```
# Filename: exception2.py
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
except ValueError:
    print('Please input a digit!')
```

```
try:
    raise
except Exception as err:
    print(err)
```

```
# Filename: exception3.py
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
except ZeroDivisionError as err:
    print('The second number cannot be zero!')
    print(err)
```

```
File
```

```
# Filename: exception4.py
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
except ValueError:
    print('Please input a digit!')
except ZeroDivisionError:
    print('The second number cannot be zero!')
```



```
# Filename: exception5.py
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
except (ValueError, ZeroDivisionError):
    print('Invalid input!')
```

Empty except Statement & as



```
# Filename: exception6.py
try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
except:
    print('Something went wrong!')
```

```
# Filename: exception7.py

try:

num1 = int(input('Enter the first number: '))

num2 = int(input('Enter the second number: '))

print(num1 / num2)
```

Handling all kinds of exception: except:

except Exception as err:

print(err)

print('Something went wrong!')

Else Statement

```
# Filename: exception8.py
try:
  num1 = int(input('Enter the first number: '))
                                                      Enter the first number: 3
  num2 = int(input('Enter the second number: '))
                                                      Enter the second number: 5
  print(num1 / num2)
                                                     0.6
except(ValueError, ZeroDivisionError):
                                                     Aha, everything is OK.
  print('Invalid input!')
else:
  print('Aha, everything is OK.')
```

Loop

```
# Filename: exception9.py
while True:
  try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
    break
  except ValueError:
    print('Please input a digit!')
  except ZeroDivisionError:
    print('The second number cannot be zero!')
```

```
Enter the first number: a
Please input a digit!
Enter the first number: 3
Enter the second number: 0
The second number cannot be zero!
Enter the first number: 3
Enter the second number: 5
0.6
```

Position of *break*

```
# Filename: exception10.py
while True:
  try:
    num1 = int(input('Enter the first number: '))
    num2 = int(input('Enter the second number: '))
    print(num1 / num2)
  except Exception as err:
    print(err)
  else:
    break
```

```
File
```

```
# Filename: exception11.py
aList = [1, 2, 3, 4, 5]
i = 0
while True:
  try:
     print(aList[i])
  except IndexError:
     print('index error')
     break
  else:
     i += 1
```

Finally Statement

```
Enter the first number: 3
# Filename: exception12.py
                                                     Enter the second number: 5
def finallyTest():
                                                     0.6
  try:
                                                     It is a finally clause.
    x = int(input('Enter the first number: '))
    y = int(input('Enter the second number: '))
     print(x / y)
    return 1
                                                      Enter the first number: 3
  except Exception as err:
                                                      Enter the second number: 0
     print(err)
    return 0
                                                     division by zero
  finally:
                                                      It is a finally clause.
     print('It is a finally clause.')
result = finallyTest()
print(result)
```

Context Manager & with Statement

```
# Filename: exception13.py
try:
  f = open('data.txt')
  for line in f:
     print(line, end = ")
except IOError:
  print('Cannot open the file!')
finally:
  f.close()
```

```
# Filename: exception14.py
with open('data.txt') as f:
for line in f:
    print(line, end=")
```

Define and control the preparation before execution and final actions after execution of code block