

Error Handling

Syntax, Index, Key, Type, Value, Name Errors



SoftUni Team
Technical Trainers




SoftUni



Software University

<https://softuni.bg>

Table of Contents

- 
1. Errors and Exceptions
 2. Common Error Types
 3. Catching Exceptions
 4. Custom Exceptions

sli.do

#python-advanced



Errors and Exceptions

Definitions and Examples

What is an Error?

- Every programmer encounters **errors**
- Encountering errors and exceptions can be very frustrating at times
- Once you know why you get certain types of errors, they become much easier to fix
- There are (at least) two distinguishable kinds of errors
 - **Syntax errors**
 - **Exceptions**



- **Syntax errors**, also known as **parsing errors**, are perhaps the most common kind of complaint you get while you are still learning Python

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

The parser displays an 'arrow' pointing at the earliest point where an error was detected

What is an Exception?

- Even if a statement or expression is **syntactically correct**, it may cause an **error** when an attempt is made to **execute** it
- Errors detected during execution are called **exceptions**
- When an exception is not handled it results in **error messages**



Example: Exception

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```




Common Error Types

Definitions and Examples

- A certain statement is not in accordance with the prescribed usage
- It is the most common reason of an error in a Python program

```
>>> print "hello"  
SyntaxError: Missing parentheses in call to 'print'.  
Did you mean print("hello")?
```

- **IndexError** is thrown when trying to access an item at an invalid index

```
>>> L1=[1,2,3]
>>> L1[3]
Traceback (most recent call last):
File "<pyshell#18>", line 1, in <module>
L1[3]
IndexError: list index out of range
```

- **KeyError** is thrown when a key is not found

```
>>> D1={'1':"aa", '2':"bb", '3':"cc"}
>>> D1['4']
Traceback (most recent call last):
File "<pyshell#15>", line 1, in <module>
D1['4']
KeyError: '4'
```

- **TypeError** is thrown when an operation or function is applied to an object of an inappropriate type

```
>>> '2'+2
```

```
Traceback (most recent call last):
```

```
File "<pyshell#23>", line 1, in <module>
```

```
'2'+2
```

```
TypeError: must be str, not int
```

- **ValueError** is thrown when a function's argument is of an inappropriate type

```
>>> int('xyz')
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    int('xyz')
ValueError: invalid literal for int() with base 10: 'xyz'
```

- **NameError** is thrown when an object could not be found

```
>>> age
Traceback (most recent call last):
File "<pyshell#6>", line 1, in <module>
age
NameError: name 'age' is not defined
```

Problem: So Many Exceptions

- You will be provided with a code that raises many exceptions
- Fix the code, so it works properly

```
numbers_list = int(input()).split(", ")
result = 1
for i in range(numbers_list):
    number = numbers_list[i+1]
    if number <= 5
        result *= number
    elif 5 < number <= 10:
        result /= number
print(total)
```

2, 5, 10



1
2 * 5 / 10 = 1

Solution: So Many Exceptions

```
numbers_list = [int(x) for x in input().split(", ")]  
result = 1  
  
for i in range(len(numbers_list)):  
    number = numbers_list[i]  
    if number <= 5:  
        result *= number  
    elif 5 < number <= 10:  
        result /= number  
  
print(result)
```



Catching Exceptions

Try-Except-Finally

- It is possible to write programs that **handle** selected exceptions

```
while True:
    try:
        x = int(input("Please enter a number: "))
        break
    except ValueError:
        print("Oops! That was no valid number. Try again...")
```

- We handle only **ValueError**, so if other error occurs, the error message will show up anyway

The Try Statement

- The Try statement **works as follows**
 - The try **clause is executed**
 - If no exception occurs, the **except clause is skipped**
 - If the type of the exception matches, the **except clause is executed**
 - If the exception does not match, the **exception is unhandled**, and **execution stops** with a message



- An Except clause may name **multiple exceptions** as a parenthesized tuple, for example

```
except (RuntimeError, TypeError, NameError):  
    pass
```

- If some of these exceptions occur, the body of the except statement will be executed

The Finally Statement

- If a **finally** clause is present, the finally clause will execute as the last task before the **try** statement completes
- The **finally** clause runs whether or not the try statement produces an **exception**

```
try:  
    x = int("Peter")  
except ValueError:  
    print("Cannot convert str to int")  
finally:  
    print("Finally block")
```



```
Cannot convert str to int  
Finally block
```

Catching the Exception Object

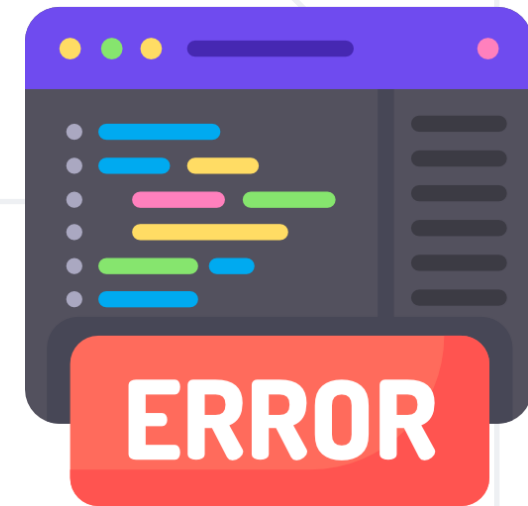
- If you wanted to examine the exception, you can do it using the following syntax

```
try:  
    x = int(input())  
except ValueError as error:  
    print(error)
```



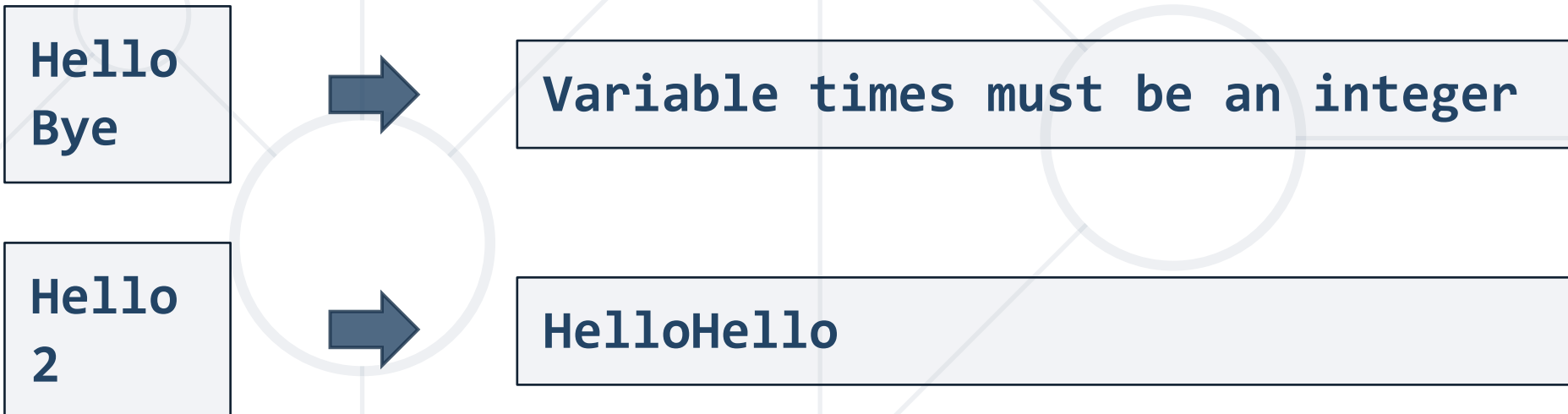
- Sometimes, you want to catch all errors that could possibly be generated, but usually you don't
- In most cases, you want to be as specific as possible

```
try:  
    # some code  
except ValueError:  
    # handle the error  
except TypeError:  
    # handle the error
```



Problem: Repeat Text

- Write a program that receives **text** on the first line and **times** (to repeat the text) that must be an **integer**
- Invalid times** should be handled with exception that prints a message "Variable times must be an integer"



Solution: Repeat Text

```
try:
    text = input()
    times = int(input())
    print(text * times)

except ValueError:
    print("Variable times must be an integer")
```





Custom Exceptions

Exceptions Serving Certain Purpose

- Sometimes you may need to create custom exceptions that serves your purpose
- In Python, users can define such exceptions by creating a new class

Derived from the
Exception class

Raising the
Exception

```
class CustomError(Exception):  
    pass  
  
raise CustomError
```

- Here we illustrate how **user-defined** exceptions can be used in a program to raise errors

```
# define Python user-defined exceptions
class Error(Exception):
    """Base class for other exceptions"""
    pass

class ValueTooSmallError(Error):
    """Raised when the input value is too
    small"""

num = int(input())
if num < 10:
    raise ValueTooSmallError
```



Problem: Value Cannot Be Negative

- Create your own exception called **ValueCannotBeNegative**
- Write a program that reads **five numbers** from the console (on separate lines)
- If a **negative** number occurs, raise the exception

```
1  
4  
-5  
3  
10
```



```
Traceback (most recent call last):  
  File ".\value_cannot_be_negative.py", line 8, in  
<module>  
    raise ValueCannotBeNegative  
__main__.ValueCannotBeNegative
```

Solution: Value Cannot Be Negative

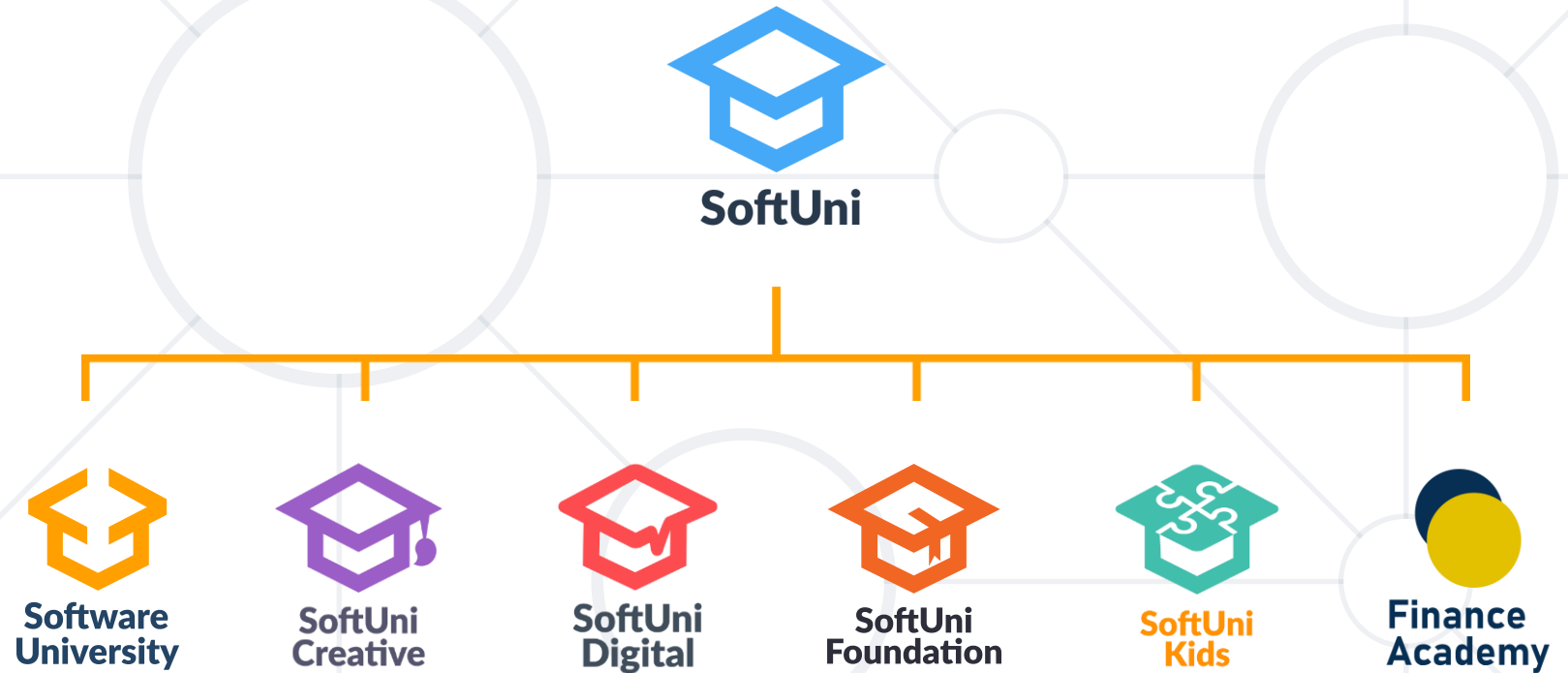
```
class ValueCannotBeNegative(Exception):  
    """Number is below zero"""  
    pass  
  
for i in range(5):  
    number = int(input())  
    if number < 0:  
        raise ValueCannotBeNegative
```



- Errors are the result of bad code
- Errors detected during execution are called exceptions
- Syntax, Index, Key, Type, Value, Name errors
- We can build custom exceptions that serves our purpose
- Handling exceptions with Try-Except block



Questions?



SoftUni Diamond Partners

**SUPER
HOSTING
.BG**



**Coca-Cola HBC
Bulgaria**



POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers



AMBITIONED

 **DRAFT
KINGS**



**SOFTWARE
GROUP**

createX



Postbank

Решения за твоето утре

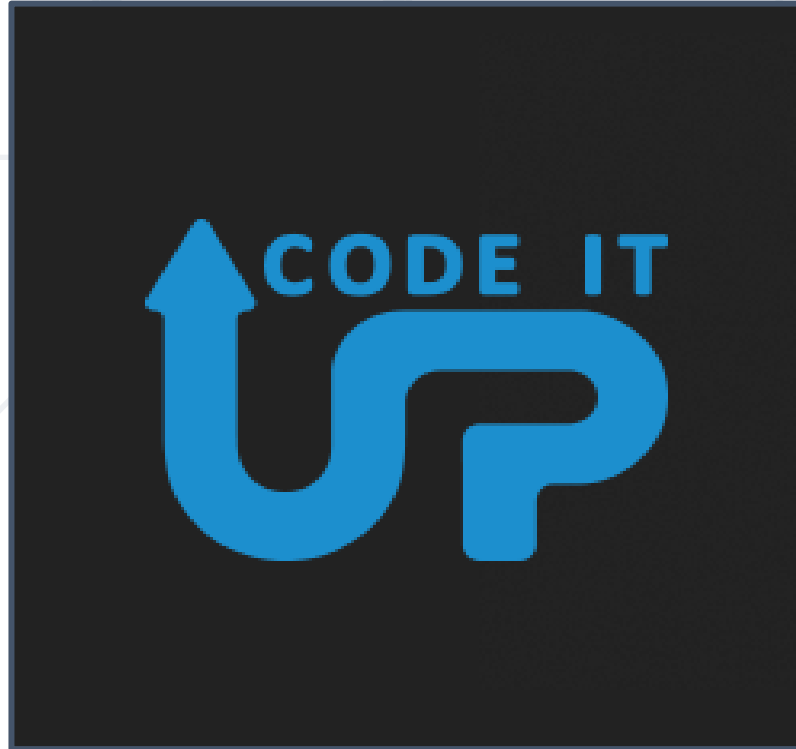


BOSCH

DXC
TECHNOLOGY



SmartIT



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction, or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

