

## Обработка исключений

# Обработка ошибок без использования исключений

`src/13. Exceptions/example_01/errors_processing.py`

# Обработка ошибок с использованием исключений

`src/13. Exceptions/example_02/exception_processing.py`

## Не перехваченное исключение

`src/13. Exceptions/example_03/uncaught_exception.py`

```
if __name__ == "__main__":  
    try:  
        print("До возбуждения исключения.")  
        raise ValueError("Сообщение для исключения.")  
        print("Эта строка никогда не будет выполняться.")  
    except ValueError:  
        print("Возникло исключение ValueError.")  
  
    print("После обработки исключения.")
```

---

До возбуждения исключения.  
Возникло исключение ValueError.  
После обработки исключения.

---

```
from random import randrange
```

```
def raise_error():  
    if randrange(2) == 0:  
        raise ZeroDivisionError("Делить на ноль нельзя.")  
    raise ValueError("Неправильное значение.")
```

```
if __name__ == "__main__":  
    try:  
        raise_error()  
    except ZeroDivisionError as err:  
        print("ZeroDivisionError.", err)  
    except ValueError as err:  
        print("ValueError.", err)  
  
    print("После обработки исключения.")
```

```
from random import randrange

def raise_error():
    if randrange(2) == 0:
        raise ZeroDivisionError("Делить на ноль нельзя.")
        raise ValueError("Неправильное значение.")

if __name__ == "__main__":
    try:
        raise_error()
    except (ZeroDivisionError, ValueError) as err:
        print("Что-то пошло не так.", err)

    print("После обработки исключения.")
```

```
def raise_error():  
    try:  
        raise ValueError("Неправильное значение.")  
    except ValueError:  
        print("Обработка исключения внутри raise_error().")  
        raise  
  
if __name__ == "__main__":  
    try:  
        raise_error()  
    except ValueError as err:  
        print("Обработка исключения в основном скрипте.")  
        print(err)  
  
    print("После обработки исключения.")
```

---

Обработка исключения внутри raise\_error().  
Обработка исключения в основном скрипте.  
Неправильное значение.  
После обработки исключения.

---



# **Иерархия исключений. Пользовательские исключения**

# Иерархия исключений

```
BaseException
├── BaseExceptionGroup
├── GeneratorExit
├── KeyboardInterrupt
├── SystemExit
├── Exception
│   ├── ArithmeticError
│   │   ├── FloatingPointError
│   │   ├── OverflowError
│   │   └── ZeroDivisionError
│   ├── AssertionError
│   ├── AttributeError
│   ├── BufferError
│   ├── EOFError
│   ├── ExceptionGroup [BaseExceptionGroup]
│   ├── ImportError
│   │   └── ModuleNotFoundError
│   ├── LookupError
│   │   ├── IndexError
│   │   └── KeyError
│   ├── MemoryError
│   ├── NameError
│   │   └── UnboundLocalError
│   ├── OSError
│   │   ├── BlockingIOError
│   │   ├── ChildProcessError
│   │   ├── ConnectionError
│   │   │   ├── BrokenPipeError
│   │   │   ├── ConnectionAbortedError
│   │   │   ├── ConnectionRefusedError
│   │   │   └── ConnectionResetError
│   │   ├── FileExistsError
│   │   ├── FileNotFoundError
│   │   ├── InterruptedError
│   │   └── IsADirectoryError
```

# Пользовательские исключения

```
class MyException(Exception):  
    ...  
  
def raise_error():  
    raise MyException("Это MyException.")  
  
if __name__ == "__main__":  
    try:  
        raise_error()  
    except MyException as err:  
        print("Что-то пошло не так.", err)
```

```
class MyException(Exception):  
    ...  
  
def raise_error(filename, value):  
    raise MyException("Это MyException.", filename, value)  
  
if __name__ == "__main__":  
    try:  
        raise_error("invalid.txt", 42)  
    except MyException as err:  
        print("Что-то пошло не так.")  
        print(f"Сообщение: {err.args[0]}")  
        print(f"Имя файла: {err.args[1]}")  
        print(f"Значение: {err.args[2]}")  
        print(err)
```

---

```
Что-то пошло не так.  
Сообщение: Это MyException.  
Имя файла: invalid.txt  
Значение: 42  
( 'Это MyException.', 'invalid.txt', 42)
```

---

```
class MyException(Exception):
    def __init__(self, message:str, filename:str, value:int):
        super().__init__(message, filename, value)
        self.message = message
        self.filename = filename
        self.value = value

def raise_error(filename, value):
    raise MyException("Это MyException.", filename, value)

if __name__ == "__main__":
    try:
        raise_error("invalid.txt", 42)
    except MyException as err:
        print("Что-то пошло не так.")
        print(f"Сообщение: {err.message}")
        print(f"Имя файла: {err.filename}")
        print(f"Значение: {err.value}")
        print(err)
```

# Обработка исключений Exception и BaseException

`src/13. Exceptions/example_11/base_exceptions.py`

```
from math import sqrt

class EquationError(Exception): ...

def equation(a, b, c) -> tuple[float, float]:
    D = b ** 2 - 4 * a * c
    if D < 0:
        raise EquationError("Дискриминант меньше нуля.")
    x1 = (-b + sqrt(D)) / (2 * a)
    x2 = (-b - sqrt(D)) / (2 * a)
    return (x1, x2)

if __name__ == "__main__":
    print("Решение уравнения вида  $ax^2 + bx + c = 0$ ")
    try:
        input_str = input("Введите a, b и c через пробел: ")
        values = [float(val) for val in input_str.split(" ")]
        result = equation(values[0], values[1], values[2])
        print("Результат вычисления:", result)
    except EquationError as err:
        print("Ошибка решения уравнения.", err)
    except Exception as err:
        print("Ошибка вычисления.", err, type(err))
    except:
        print("\nЧто-то пошло не так.")
```

```
from math import sqrt
```

```
class EquationError(Exception): ...
```

```
def equation(a, b, c) -> tuple[float, float]:
```

```
    D = b ** 2 - 4 * a * c
```

```
    if D < 0:
```

```
        raise EquationError("Дискриминант меньше нуля.")
```

```
    x1 = (-b + sqrt(D)) / (2 * a)
```

```
    x2 = (-b - sqrt(D)) / (2 * a)
```

```
    return (x1, x2)
```

```
if __name__ == "__main__":
```

```
    print("Решение уравнения вида  $ax^2 + bx + c = 0$ ")
```

```
    result = None
```

```
    try:
```

```
        input_str = input("Введите a, b и c через пробел: ")
```

```
        values = [float(val) for val in input_str.split(" ")]
```

```
        result = equation(values[0], values[1], values[2])
```

```
    except Exception as err:
```

```
        print("Ошибка вычисления.", err, type(err))
```

```
    else:
```

```
        print("Результат вычисления:", result)
```



```
try:
    print("Вошли в блок try.")
    print("Выходим из блока try.")
except ValueError:
    print("Вошли в блок except.")
else:
    print("Вошли в блок else.")
finally:
    print("Вошли в блок finally.")

print("Вышли из блока try / except / else / finally.")
```

---

```
Вошли в блок try.
Выходим из блока try.
Вошли в блок else.
Вошли в блок finally.
Вышли из блока try / except / else / finally.
```

---

```
try:
    print("Вошли в блок try.")
    raise ValueError()
    print("Выходим из блока try.")
except ValueError:
    print("Вошли в блок except.")
else:
    print("Вошли в блок else.")
finally:
    print("Вошли в блок finally.")

print("Вышли из блока try / except / else / finally.")
```

---

```
Вошли в блок try.
Вошли в блок except.
Вошли в блок finally.
Вышли из блока try / except / else / finally.
```

---

```
try:
    print("Вошли в блок try.")
    raise OSError()
    print("Выходим из блока try.")
except ValueError:
    print("Вошли в блок except.")
else:
    print("Вошли в блок else.")
finally:
    print("Вошли в блок finally.")

print("Вышли из блока try / except / else / finally.")
```

---

```
Вошли в блок try.
Вошли в блок finally.
Traceback (most recent call last):
  File ".../try_finally", line 3, in <module>
    raise OSError()
OSError
```

```
def func():  
    try:  
        print("Вошли в блок try.")  
        return  
        print("Выходим из блока try.")  
    except ValueError:  
        print("Вошли в блок except.")  
    else:  
        print("Вошли в блок else.")  
    finally:  
        print("Вошли в блок finally.")  
  
if __name__ == "__main__":  
    print("Вызов функции func()")  
    func()  
    print("Вышли из функции func()")
```

---

Вызов функции func()  
Вошли в блок try.  
Вошли в блок finally.  
Вышли из функции func()

---