Урок по Python: Волшебный щит для программы — Try Except

Длительность: 40 минут
Целевая аудитория: школьники 10 лет
Цель: Научить "ловить" ошибки, чтобы программа не ломалась

Часть 1: Введение (5 минут)

Объяснение через аналогию:

"Представьте, что ваша программа — это робот, который выполняет команды. Иногда он может столкнуться с неожиданной ситуацией:

- Попросить поделить яблоки между 0 друзей (деление на ноль)
- Попытаться открыть дверь, которой нет (файл не найден)
- Попросить превратить слово 'кошка' в число

Обычно программа тогда **ломается** и останавливается. Но мы можем дать нашему роботу **волшебный щит** — команды `try` и `except`, которые поймают ошибку и скажут роботу, что делать дальше!"

Показываем пример без защиты:

```
print("Делим яблоки между друзьями")
друзья = 0
яблоки = 10
результат = яблоки / друзей  # Ошибка! На ноль делить нельзя!
print("Каждому достанется:", результат)
```

Часть 2: Базовый try-except (10 минут)

Код с защитой:

```
print("=== Волшебный щтив try-except ===")

try:
    # Пробуем выполнить опасный код
    print("Делим яблоки между друзьями")
    друзья = 0
```

```
    яблоки = 10
    результат = яблоки / друзей
    print("Каждому достанется:", результат)
except:
    # Что делать, если ошибка случилась
    print("Ой! Кажется, друзей нет... Деление на ноль невозможно!")

print("Программа продолжает работать!  ")
```

Объяснение структуры:

```
try:
    # Код, который может сломаться
    [опасные команды]
except:
    # Код, который спасёт ситуацию
    [что делать при ошибке]
```

Интерактивный пример:

```
print("=== Калькулятор с защитой ===")

try:
    a = int(input("Введи первое число: "))
    b = int(input("Введи второе число: "))
    ответ = a / b
    print(f"{a} / {b} = {ответ}")
except:
    print("Что-то пошло не так! Проверь, что ты вводишь числа и не делишь на ноль!")

print("Спасибо за использование калькулятора!")
```

Часть 3: Ловим конкретные ошибки (10 минут)

Разные виды ошибок:

```
print("=== Умный щит, который знает разные ошибки ===")

try:
    # 1. Пробуем преобразовать текст в число
    возраст = int(input("Сколько тебе лет? (можно написать 'десять'): "))

    # 2. Пробуем поделить
    конфеты = 20
    результат = конфеты / (возраст - 10)  # Если ввели 10, будет деление на 0!

    print(f"Тебе достанется {результат} конфет!")
```

```
except ValueError:  # Если не смогли превратить текст в число
    print("Кажется, нужно вводить число (например, 10), а не слово!")

except ZeroDivisionError:  # Если деление на ноль
    print("Ой! Получилось делить на ноль! Так нельзя!")

except:  # Все остальные ошибки
    print("Произошла какая-то другая ошибка!")

print("Мы справились с ошибками!  ")
```

Игра "Угадай ошибку":

```
print("=== Игра: Угадай, какая ошибка произойдет ===")

задачи = [
    "int('привет')",           # ValueError
    "10 / 0",                  # ZeroDivisionError
    "число = 5 + 'яблоко'",    # TypeError
    "print(неизвестная_переменная)", # NameError
]

for задача in задачи:
    print(f"\nПробуем: {задача}")
    try:
        выполнить = eval(задача)
        print(f"Результат: {выполнить}")
    except ValueError as e:
        print(f"Поймали ValueError: {e}")
    except ZeroDivisionError as e:
        print(f"Поймали ZeroDivisionError: {e}")
    except TypeError as e:
        print(f"Поймали TypeError: {e}")
    except NameError as e:
        print(f"Поймали NameError: {e}")
```

Часть 4: Else и Finally (5 минут)

Полная структура:

```
print("=== Полная защита с else и finally ===")

try:
    print("Пробуем открыть сундук с сокровищами...")
    код = int(input("Введи код от сундука (число): "))
    сокровище = 100 / код  # Может быть ошибка деления на 0
    print(f"Отлично! Код подошёл!")
```

```
except (ValueError, ZeroDivisionError):
    print("Неверный код! Сундук не открывается...")

else:  # Выполняется ТОЛЬКО если НЕ было ошибки в try
    print(f"  Ты нашёл {сокровище} золотых монет!")

finally:  # Выполняется ВСЕГДА, даже если была ошибка
    print("🔒 Сундук закрывается...")
    print("Игра продолжается!")

print("\nПереходим к следующему уровню!")
```

Часть 5: Практическое задание (7 минут)

Задание: "Несгораемый калькулятор"

Напиши калькулятор, который:

1. Просит ввести два числа
2. Просит выбрать операцию (+, -, *, /)
3. Выполняет операцию
4. **Никогда не ломается**, даже если:
    - Ввели не числа
    - Выбрали деление на ноль
    - Случилась любая другая ошибка

```
# Каркас для решения
print("=== Несгораемый калькулятор ===")

try:
    # Здесь будет основной код
    pass
except ValueError:
    print("Пожалуйста, вводи только числа!")
except ZeroDivisionError:
    print("На ноль делить нельзя!")
except:
    print("Что-то пошло не так...")
```

Пример решения:

```
print("=== Несгораемый калькулятор ===")

try:
    a = float(input("Первое число: "))
    b = float(input("Второе число: "))
```

```
    операция = input("Операция (+, -, *, /): ")

    if операция == "+":
        результат = a + b
    elif операция == "-":
        результат = a - b
    elif операция == "*":
        результат = a * b
    elif операция == "/":
        результат = a / b
    else:
        print("Неизвестная операция!")
        результат = None

    if результат is not None:
        print(f"Результат: {a} {операция} {b} = {результат}")

except ValueError:
    print("  Ошибка! Нужно вводить числа (можно с точкой, например 5.5)")
except ZeroDivisionError:
    print("  Ошибка! На ноль делить нельзя!")
except Exception as e:
    print(f"  Неизвестная ошибка: {e}")
finally:
    print("\nКалькулятор завершил работу. До встречи!  ")
```

Часть 6: Итог и домашнее задание (3 минуты)

Ключевые моменты:

1. `try` — пробуем выполнить код, который может сломаться
2. `except` — ловим ошибку и говорим, что делать
3. Можно ловить конкретные ошибки: `except ValueError:`
4. `else` — выполняется, если ошибок не было
5. `finally` — выполняется всегда, даже при ошибке

Домашнее задание:

Создай "Умного робота-помощника" с защитой от ошибок:

```
# Робот должен уметь:
# 1. Принимать команды: "привет", "сложи", "пока"
# 2. При команде "сложи" — спрашивать два числа и выводить сумму
# 3. Не ломаться, если ввели не число или неизвестную команду
# 4. Всегда прощаться в конце (использовать finally)

while True:
    команда = input("\nВведи команду: ")
```

```
    if команда == "пока":
        break

    # Твоя задача — добавить try-except для защиты!
```

Бонусная задача:

Создай функцию `безопасное_деление(a, b)`, которая:

- Возвращает результат деления a на b
- Если происходит ошибка, возвращает строку "Ошибка!"
- Использует try-except внутри функции

Визуальные материалы для урока:

Аналогии:

1. **Try** — как попытка пройти по канату
2. **Except** — как страховочная сетка внизу
3. **Else** — как аплодисменты, если прошёл успешно
4. **Finally** — как обязательный поклон артиста в любом случае

Пример ошибок в жизни:

```
ValueError: Попытка засунуть квадратный кубик в круглое отверстие
ZeroDivisionError: Поделить пиццу между 0 друзей
TypeError: Попытка съесть цифру 5
NameError: Позвать друга, которого нет в комнате
```

Чек-лист для учеников:

- [ ] Понял, зачем нужен try-except
- [ ] Умею ловить все ошибки одним except
- [ ] Умею ловить конкретные ошибки
- [ ] Знаю, чем else отличается от finally
- [ ] Создал свою программу с защитой от ошибок

Формат урока: Можно провести как интерактивное занятие, где ученики по очереди вводят "неправильные" данные, а программа демонстрирует, как их обрабатывает. Особенно весело пробовать деление на ноль или ввод текста вместо чисел!