

ПРОГРАММА

ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

**“Python-разработка для БПЛА: паттерны проектирования, API-интерфейсы и библиотеки для оптимизации решений”**

**Задание 4.**

**Использование инструментов для отладки и обнаружения ошибок**

Преподаватель курса

**Максимов Егор Васильевич**

Выполнил

**Зубков Евгений Юрьевич**

**№ потока БПЛА\_256-1, группа 1**

**Москва 2024 г.**

**Задание 4.** **Использование инструментов для отладки и обнаружения ошибок**

**Инструкция:**

Шаг 1. Написать код, который может вызвать ошибку (например, деление на ноль)

Шаг 2. Импортировать модуль pdb и добавить вызов pdb.set\_trace() перед строкой, где предполагается ошибка.

Шаг 3. Запустить код - при достижении точки останова pdb откроется отладчик. Исследовать текущее состояние переменных и шаг за шагом выполнить код до ошибки.

Шаг 4. Добавить блок try/except вокруг опасного кода и перехватывать конкретный тип ошибки.

Шаг 5. В блоке except написать код для обработки исключения - например, вывод сообщения и продолжение работы программы.

Шаг 6. Проверить работу обработки ошибки, запустив код ещё раз. Убедиться, что исключение перехватывается корректно.

Шаг 7. Проверить работу обработки ошибки, запустив код ещё раз. Убедиться, что исключение перехватывается корректно.

Шаг 8. При необходимости добавить блок finally для кода, который должен выполниться в любом случае.

**Решение:**

import pdb  
  
def error\_code():  
 numerator = 10  
 denominator = 0  
 pdb.set\_trace() # Устанавливаем точку останова перед потенциально опасным кодом  
  
 try:  
 result = numerator / denominator  
 except ZeroDivisionError:  
 print("Ошибка: деление на ноль!")  
 else:  
 print(f"Результат: {result}")  
 finally:  
 print("Этот код выполнится в любом случае.")  
  
error\_code()  
print("Программа продолжает работу.")