**1. Типы ошибок в коде**

**1.1.Синтаксическая ошибка**

Определение: Нарушение правил языка программирования, из-за которого код не может быть выполнен.

Пример:

if x == 5 # Ошибка: пропущено двоеточие

print("x равно 5")

**1.2Логическая ошибка**

Определение: Код работает, но выдает неверный результат из-за неправильной логики.

Пример:

def sum(a, b):

return a \* b

# Ошибка: вместо суммы возвращается произведение

**1.3.Ошибка в условии (неправильное условие)**

Определение: Условие в if, while или другом операторе работает не так, как задумано.

Пример:

if age >= 18:

print("Ребенок") # Ошибка: должно быть `age < 18`

**1.4.Ошибка обработки граничных случаев**

Определение: Код не учитывает крайние значения или особые ситуации.

Пример:

def divide(a, b):

return a / b # Ошибка: не проверяется деление на ноль (`b == 0`)

**1.5.Отклонение от алгоритма (код делает не то, что описано)**

Определение: Реализация не соответствует заданному алгоритму.

Пример:

# Алгоритм: найти максимум из двух чисел

def max(a, b):

return a # Ошибка: всегда возвращает `a`, даже если `b > a`

**2. Методы поиска ошибок**

1. Визуальный анализ кода

Проверка кода "на глаз" на соответствие синтаксису и логике.

2. Сравнение с описанием алгоритма

Сверка кода с пошаговым описанием, чтобы убедиться, что все шаги выполнены правильно.

3. Ручное тестирование на конкретных примерах

Запуск кода с разными входными данными (включая граничные случаи) и проверка результата.

Пример тестирования:

def max(a, b):

return a # Подозрительный код

# Тест 1: a < b → должен вернуть b

print(max(3, 5)) # Выведет 3, но должен 5 → ошибка!