МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Инженерно-технологическая академия**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра Математического обеспечения и применения ЭВМ**

**ОТЧЕТ**

по лабораторной работе № 3

по курсу «Методология программной инженерии и верификация программного обеспечения»

Выполнил:

студенты группы КТмо1-3

Шепель И. О.

Куприянова А. А.

Проверил:

преподаватель каф. МОП ЭВМ

Пирская Л. В.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2016 г.

Таганрог 2016

**Цели работы:**

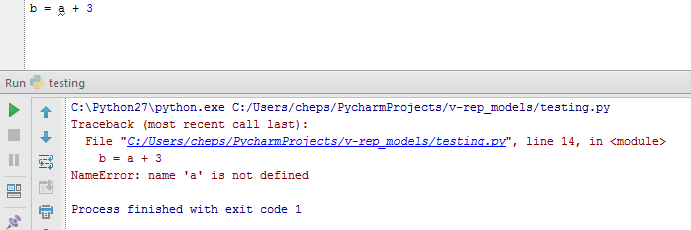
Целью работы является получение навыков применения инструментов статического анализа кода. Для этого была выбрана интегрированная среда разработки PyCharm для языка программирования Python. Необходимо описать эту среду, выбрать программу длиной не менее 1000 строк, пропустить ее через статический анализатор кода, встроенный в среду, и исправить найденные предупреждения.

**Выполнение работы:**

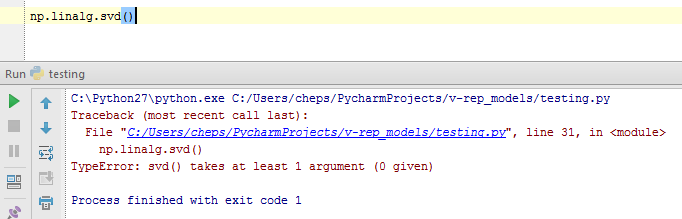
1. Описание статического анализатора кода в IDE PyCharm.

Статический анализатор кода в PyCharm может обнаруживать следующие типы ошибок:

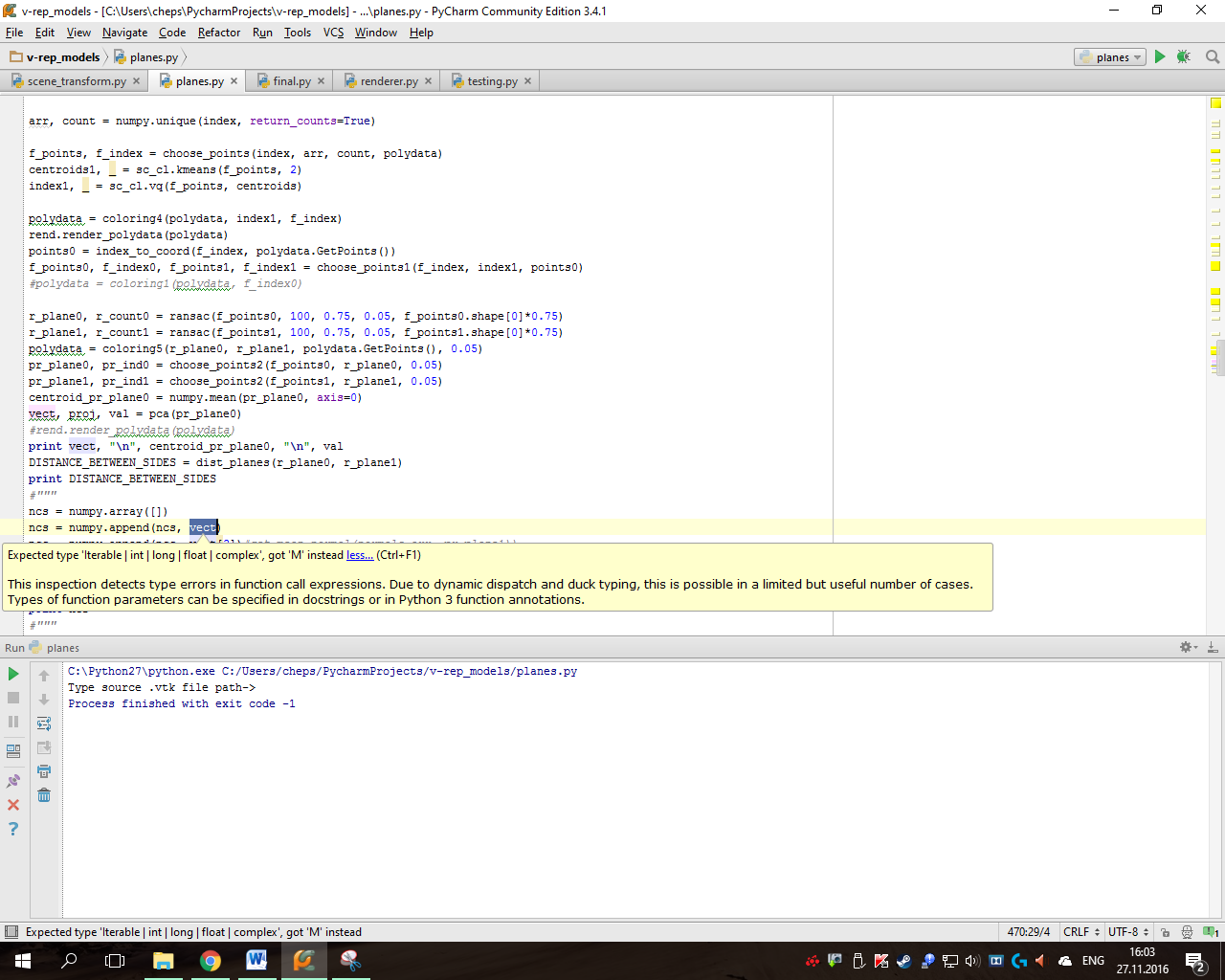
1) Неопределенное поведение (неинициализированные переменные, обращение к пустым указателям).



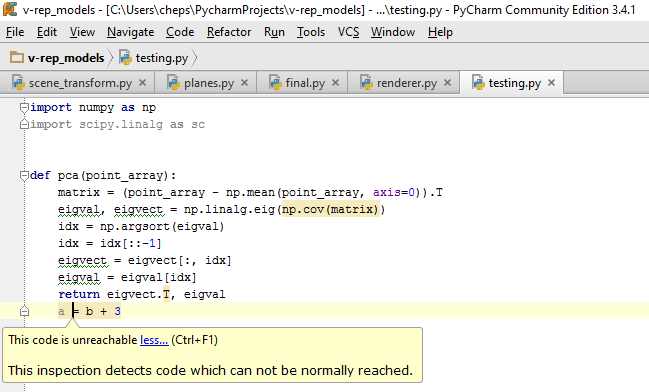
2) Неправильное поведение функций (неправильное количество переменных на входе и на выходе в частности)

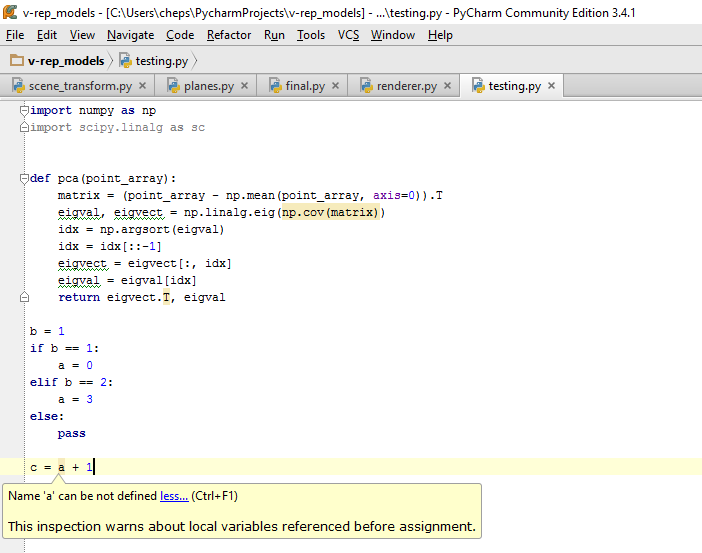


3) Несоответствие типов переменных внутри функций. Проверка кода проводится just in time, то есть «на лету» и пользователю сразу выдается предупреждение о любой ошибке и без запуска скрипта.



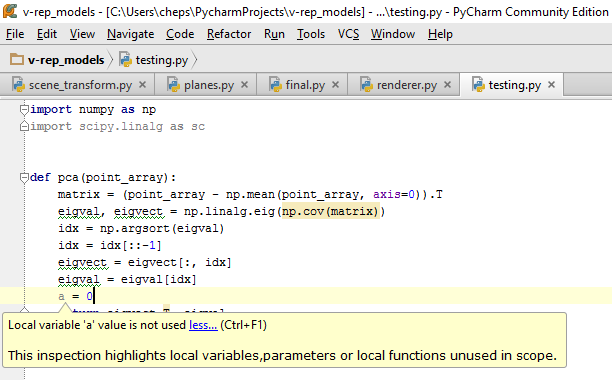
4) Также анализатор сообщает о недостижимых участках кода, и о возможном неправильном поведении программы при прохождении блоков условий.





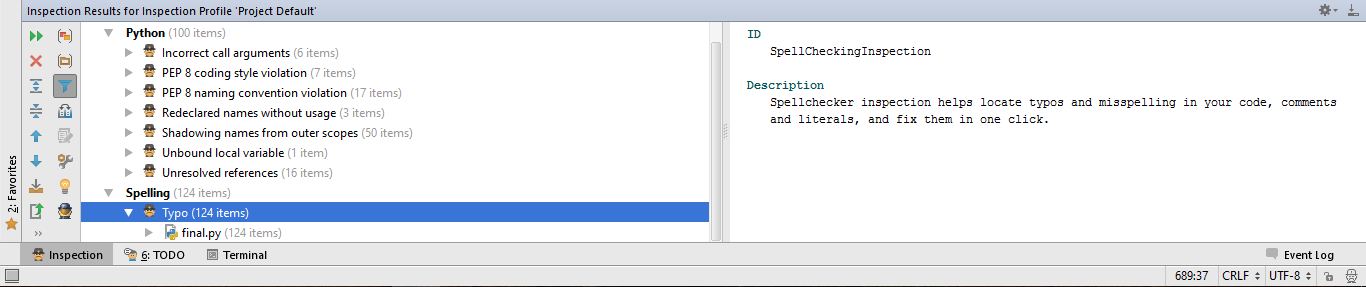
При таких ошибках код можно запустить, и он будет работать корректно, если переменная задана. Анализатор сообщает лишь о *возможных* в таком случае проблемах.

5) Анализатор также сообщает о неиспользуемых кусках кода.



6) Кроме этого анализатор может находить множество других ошибок и кроме того предлагать их автоматическое решение – например, генерацию кода метода при отсутствии такового в скрипте. Подробнее об этом написано в пункте 1 списка использованных источников.

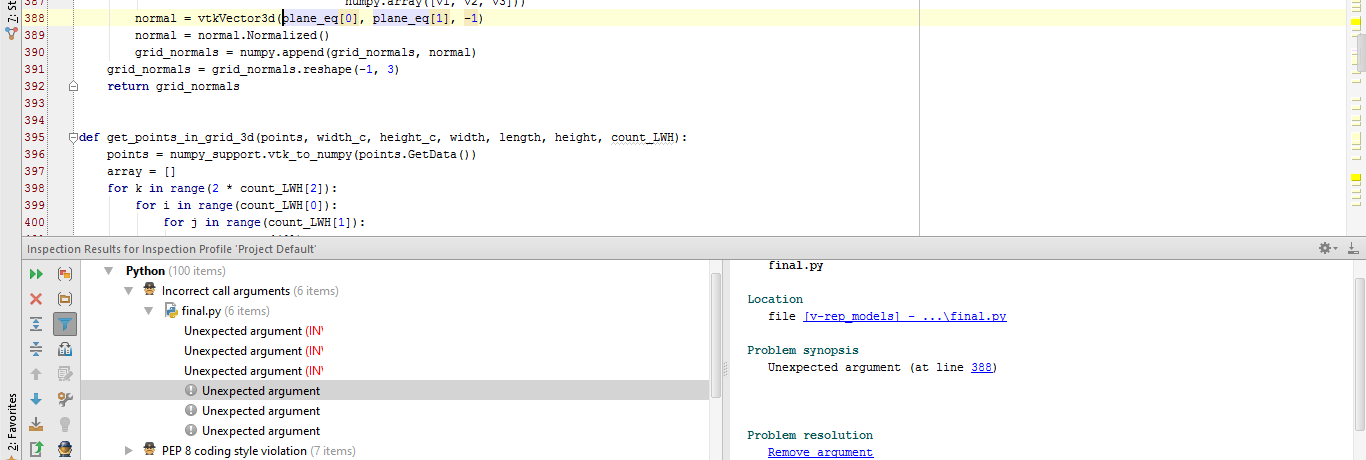
Чтобы запустить инспекцию кода необходимо перейти из строки меню в Code -> Inspect Code.



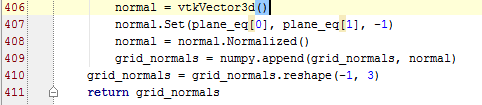
Слева выводятся все типы ошибок, в данном случае ошибки в коде (вкладка Python) и грамматические ошибки в написании слов (Spelling), а справа описание типа ошибки и возможности по ее исправлению или игнорированию.

2. Исправление ошибок.

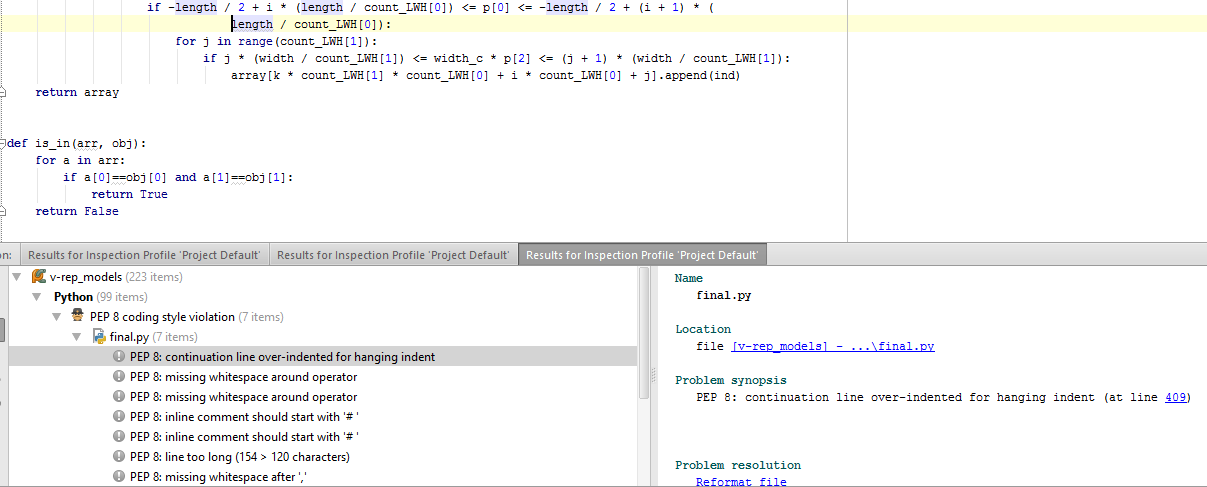
1) Сперва исправим все ошибки *Incorrect call arguments.*



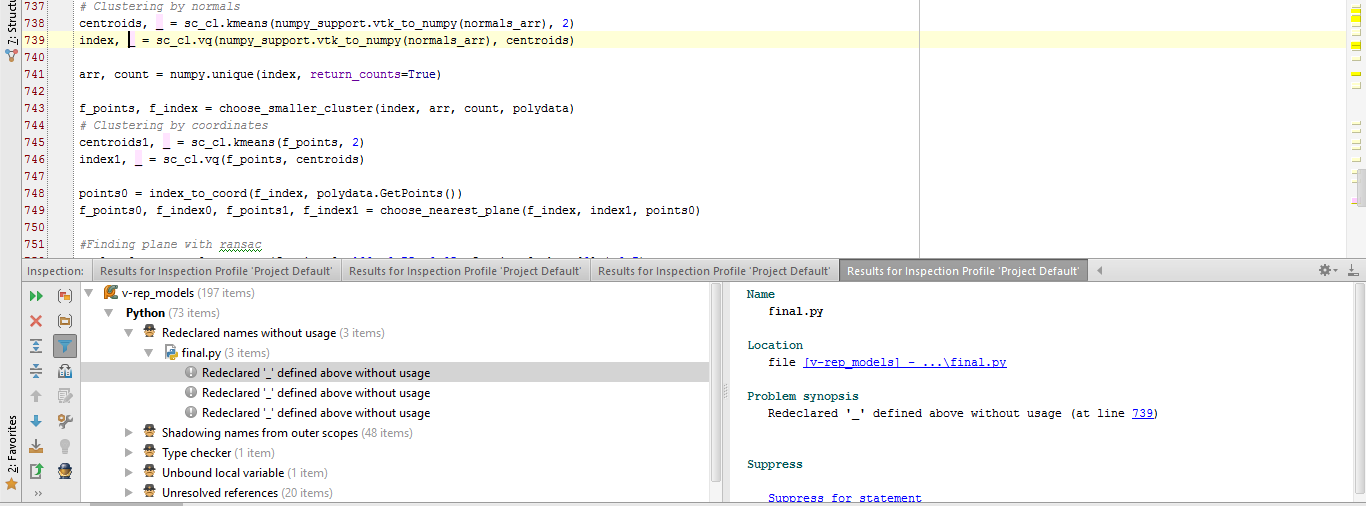
Ошибка в том, что конструктор vtkVector3d по умолчанию не принимает 3 значения на вход, хотя при такой инициализации библиотека vtk отрабатывает корректно и вызывает метод Set. Сделаем это вручную.



2) Далее анализатор нашел ошибки в оформлении кода по стандарту PEP 8: *PEP 8 coding style violation*. Сюда входят ошибки слишком длинных строчек кода, пропущенных пробелов и т. п. Исправим их, пользуясь автоматически предлагаемым решением.

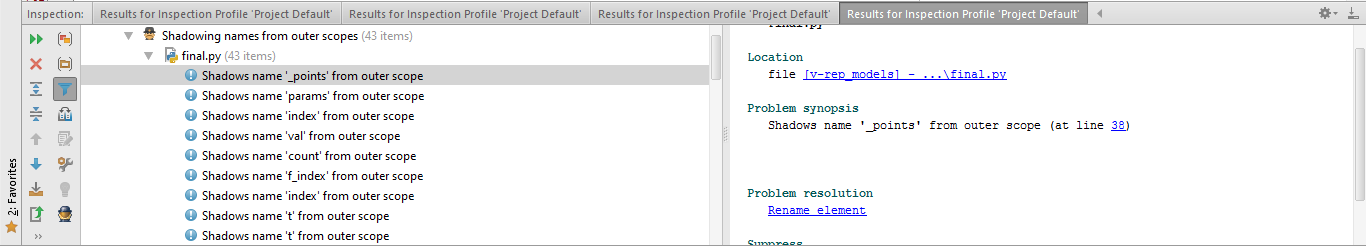


3) Перейдем к ошибке *Redeclared names without usage*.



Она сообщает о переменных, которые не были использованы до их перезадания. В частности в коде в переменную \_ записываются индексы кластеризованных точек, которые в дальнейшем не используются. Тем не менее функция сторонней библиотеки, реализующая кластеризацию, возвращает две переменные, чем и вызвана необходимость использовать переменную заглушку \_. Исправим ошибку, указав анализатору кода игнорировать такое поведение, поскольку оно нормально для этой программы.

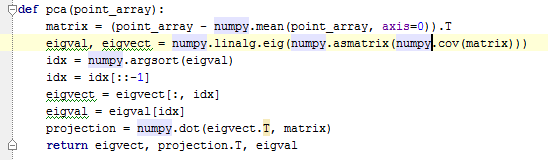
4) Ошибка *Shadowing names from outer scope.*  Она указывает на то, что локальные и глобальные переменные имеют одинаковые имена. При этом конфликта не возникает, так как в локальных функциях используется та переменная, что ближе по иерархии, то есть локальная. Исправим ошибки, переименовав глобальные переменные.



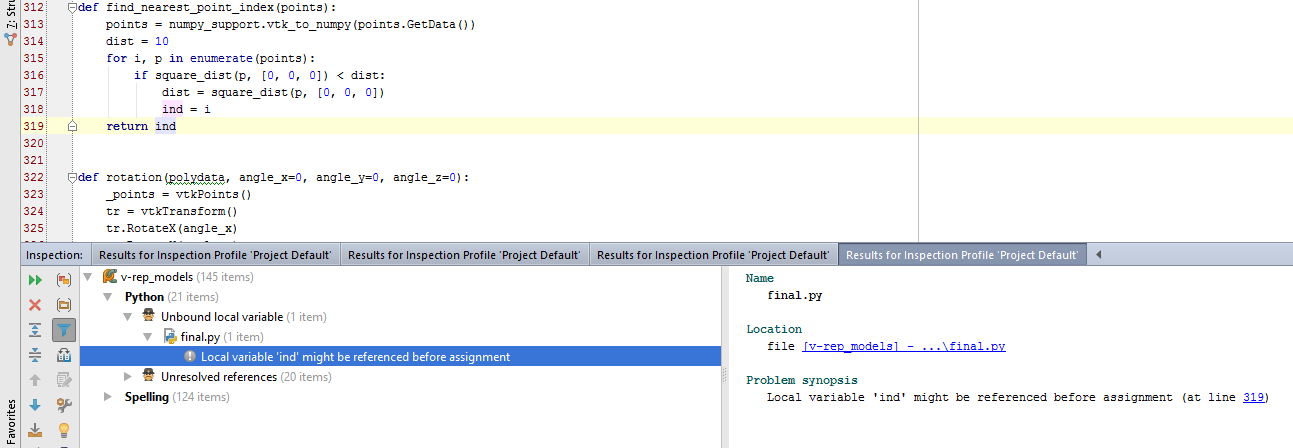
5) Следующая ошибка – *Type checker.* Она возникает, когда типы переменных при присваивании не совпадают.



Здесь библиотекой numpy ожидался тип переменной «М» (матрица) на входе, а был получен тип «ndarray» (n-мерный массив). Эта библиотека способна автоматически приводить один тип к другому, но мы сделаем это вручную.



5) *Unbound local variable.* Переменная может быть вызвана до того, как будет задана.



Исправим эту ошибку, просто задав переменную нулем до цикла.

**Вывод.**

В ходе работы был изучен встроенный в среду разработки PyCharm статический анализатор кода. Также были исправлены ошибки, найденные в коде с его помощью.

**Список использованной литературы.**

1. <https://www.jetbrains.com/help/pycharm/2016.1/code-inspection.html>