



# Возможности среды разработки для анализа, модификации и ассистирования при написании кода на Python

Выпускная квалификационная работа

Кузнецов Илья Александрович  
Группа 24.M71-MM

Научный руководитель:  
профессор кафедры Системного программирования, д.т.н. Д. В.  
Кознов

Консультант:  
ведущий инженер ключевых проектов ООО «Техкомпания Хуавэй» Н. В. Тропин

Рецензент:  
ведущий инженер ключевых проектов ООО «Техкомпания Хуавэй», ?

Санкт-Петербургский государственный университет  
Программная инженерия

- Saint-Petersburg Research Center (SRC)
- SRC IDE – мультязыковая среда разработки, основной компонент семейства продуктов SRC<sup>1</sup>
- SRC IDE поддерживала Java
- Для Python разрабатывалась модель кода, которая уже позволяла приступить к реализации основных возможностей языкового сервера и клиента, но в то же время она требовала апробации, доработки и стабилизации базовых интерфейсов

<sup>1</sup> [Повторно используемая инфраструктура мультязыковой среды разработки для языка Python // Кузнецов И. А., 2023 // СПбГУ](#)

## Цель и задачи

**Цель:** реализация с использованием модели кода основных возможностей SRC IDE для Python: анализов кода, действий для исправления проблем и реструктуризации кода, а также ассистентов для написания кода

### **Задачи выпускной квалификационной работы:**

- провести обзор моделей кода и возможностей в инструментах разработки для языка Python
- реализовать основные возможности SRC IDE для языка Python с использованием модели кода
- провести многостороннее тестирование разработанного решения, создав соответствующую инфраструктуру
- настроить многосторонний мониторинг разработанного решения, создав соответствующую инфраструктуру
- выполнить апробацию разработанного решения с помощью мониторинга и пользователей SRC IDE для Python

- Статические анализаторы – предлагают долгий, но наиболее точный анализ кода на Python
  - MyPy, PyRight
- Среды разработки – предлагают быстрый и точный анализ кода, исправления обнаруженных проблем и действия по реструктуризации кода, а также ассистируют при написании кода на Python
  - SRC IDE, PyCharm, VSCode (Pylance)

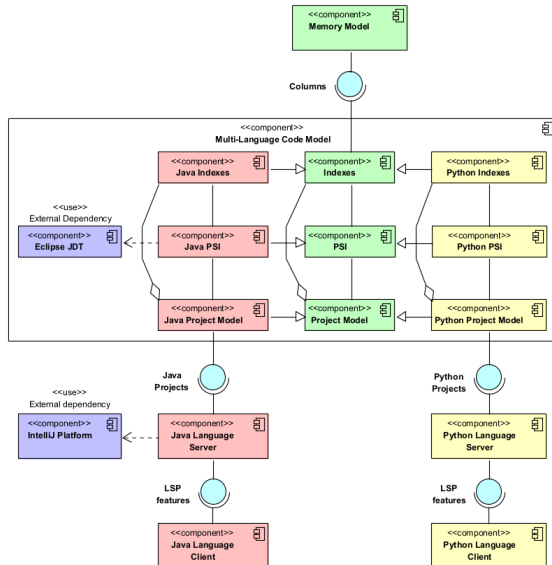
## Обзор предметной области

- **Анализы** (инспекции, диагностики) – обнаруживают и интерпретируют проблемы в коде, определяя исправления
- **Исправления** (фиксы) – это действия по модификации кода для устранения проблем, найденных инспекциями
- **Реструктуризации** (рефакторинги) – это действия по модификации кода для изменения его структуры, абстракции или упрощения
- **Ассистенты** (помощники) – представляют информацию о коде в удобном виде, упрощая его понимание и снижая вероятность ошибок

# Обзор предметной области

## Архитектура мультязыковой SRC IDE<sup>1</sup>:

- Memory Model
  - Indexes
- Python Code Model
  - AST, PSI
  - Resolve, Type Inference, IR
  - Project Model
- Python Language Server
  - Features
- Python Language Client
  - Features
  - UI



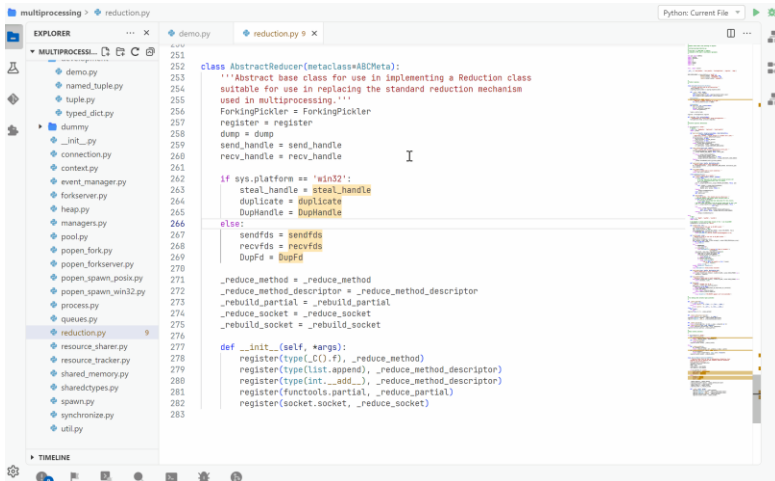
## Обзор предметной области

- Производительность и точность возможностей среды разработки напрямую зависят от модели кода
- Качество и полнота возможностей среды разработки определяется соответствием спецификациям Python и библиотек, покрытием различных сценариев как в правильном, так и в неправильном коде, а также проработкой UX и UI

	<b>Pylance (PyRight)</b>	<b>PyCharm</b>	<b>SRC IDE</b>
Flask (5072 QR)	32.96 %	29.20 %	39.33 %
Jedi (10884 QR)	32.77 %	45.61 %	60.63 %
PyTorch (599914 QR)		54.29 %	61.36 %
Tensorflow (491977 QR)		65.96 %	84.09 %
Django (202578 QR)	28.75 %	59.16 %	76.08 %
Numpy (85004 QR)		61.51 %	87.64 %
ToolBench (2913 QR)	57.56 %	62.92 %	79.71 %
SciPy (123591 QR)		32.37 %	85.63 %
Scikit-learn (90175 QR)		33.78 %	69.52 %

# Ход работы: инспекции

- На текущий момент реализовано более 60-ти различных инспекций
- Некоторые из наиболее важных: Import, Simple и Qualified Resolve, Expected Types, Redeclaration, Shadowing, Unused, Version Compatibility, ...





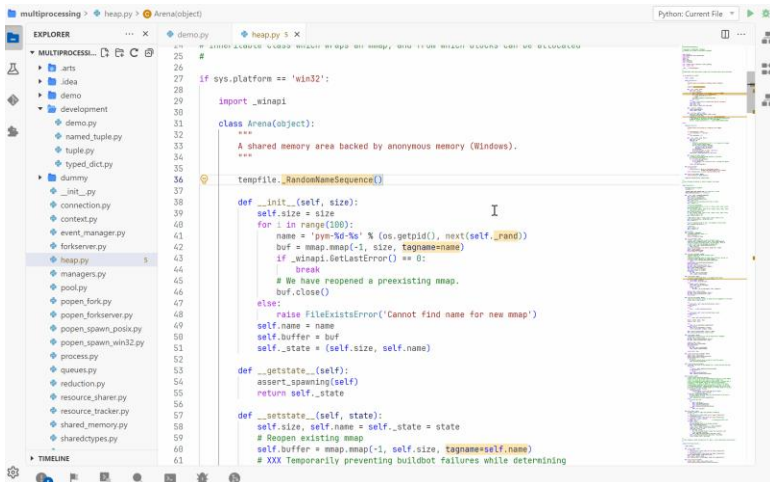
# Ход работы: исправления

- На текущий момент реализовано более 40-ка различных исправлений
- Некоторые из наиболее важных: Install Packages, Import, Create, Remove Class, Function, Field, Variable, ...

```
350 class AuthenticationString(bytes):
351     def __reduce__(self):
352         from .context import get_spawning_popen
353         if get_spawning_popen() is None:
354             raise TypeError(
355                 'Pickling an AuthenticationString object is '
356                 'disallowed for security reasons'
357             )
358         return AuthenticationString, (bytes(self),)
359
360 #
361 # Create object representing the parent process
362 #
363
364 SuperClasses (2)
365 class _ParentProcess(BaseProcess):
366     I
367     Overrides (2)
368     def __init__(self, name, pid, sentinel):
369         self._identity = ()
370         self._name = name
371         self._pid = pid
372         self._parent_pid = None
373         self._popen = None
374         self._closed = False
375         self._sentinel = sentinel
376         self._config = {}
377
378     Overrides (1)
379     def is_alive(self):
380         from multiprocessing.connection import wait
381         return not wait([self._sentinel], timeout=0)
382
383 @property
384 Overrides (1)
385 def ident(self):
386     return self._pid
```

# Ход работы: реструктуризации

- На текущий момент реализовано 10 различных реструктуризаций
- Некоторые из наиболее важных: Rename, Rearrange, Inline, Extract Variable, Method ...

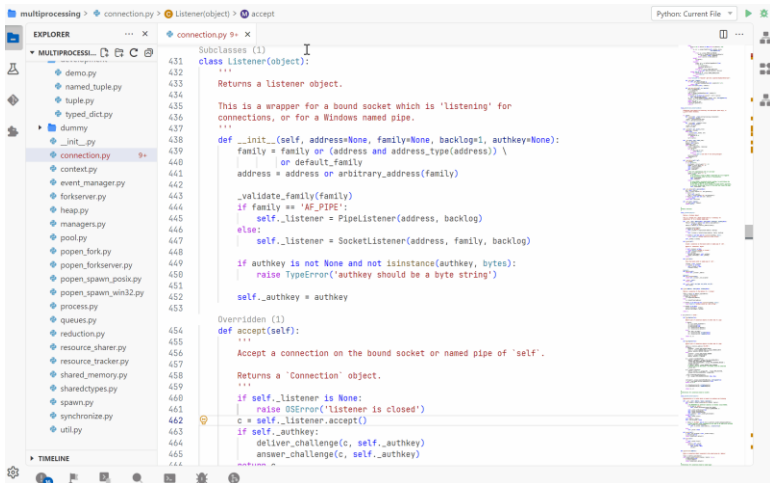


The screenshot shows a Python IDE interface. On the left is a file explorer showing a project structure with folders like 'MULTIPROCESSING...', 'arts', 'idea', 'demo', and 'development'. The 'demo' folder is expanded, showing files like 'demo.py', 'named\_tuple.py', 'tuple.py', 'typed\_dict.py', 'dummy', 'init\_.py', 'connection.py', 'context.py', 'event\_manager.py', 'forkserver.py', 'heap.py' (selected), 'managers.py', 'pool.py', 'popen\_fork.py', 'popen\_forkserver.py', 'popen\_spawn\_posix.py', 'popen\_spawn\_win32.py', 'process.py', 'queues.py', 'reduction.py', 'resource\_sharer.py', 'resource\_tracker.py', 'shared\_memory.py', and 'sharedctypes.py'. The main editor window displays the code for 'demo.py'. The code defines a class 'Arena(object)' which is a shared memory area backed by anonymous memory (Windows). It includes methods for initialization, getting and setting state, and reopening existing mmap. The code is as follows:

```
25 #
26
27 if sys.platform == 'win32':
28
29     import _winapi
30
31     class Arena(object):
32     """
33     A shared memory area backed by anonymous memory (Windows).
34     """
35
36     tempfile._RandomNameSequence()
37
38     def __init__(self, size):
39         self.size = size
40         for i in range(100):
41             name = 'pym-%d-%s' % (os.getpid(), next(self._rand))
42             buf = mmap.mmap(-1, size, tagname=name)
43             if _winapi.GetLastError() == 0:
44                 break
45             # We have reopened a preexisting mmap.
46             buf.close()
47         else:
48             raise FileExistsError('Cannot find name for new mmap')
49         self.name = name
50         self.buffer = buf
51         self._state = (self.size, self.name)
52
53     def __getstate__(self):
54         assert_spawning(self)
55         return self._state
56
57     def __setstate__(self, state):
58         self.size, self.name = self._state = state
59         # Reopen existing mmap
60         self.buffer = mmap.mmap(-1, self.size, tagname=self.name)
61         # XXX Temporarily preventing buildbot failures while determining
```

# Ход работы: ассистенты

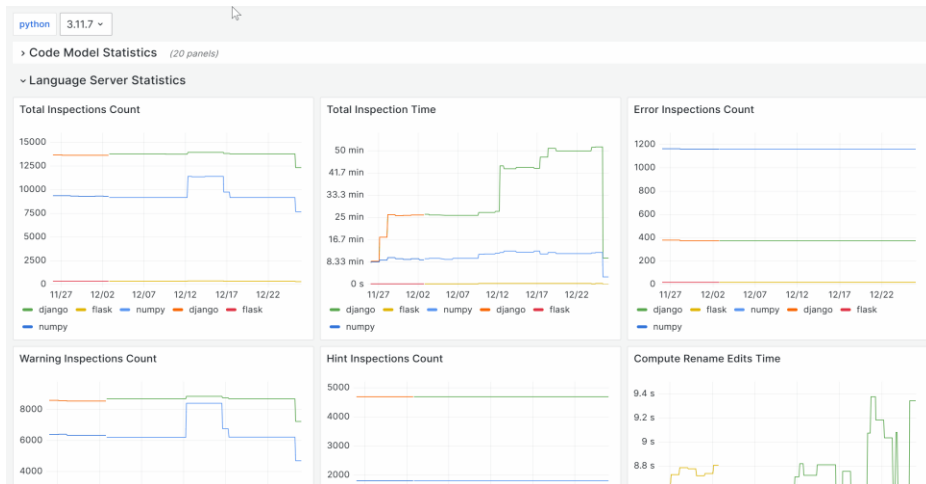
- На текущий момент реализовано 3 ассистента: Hover, Signature Help и Inlay Hints



- Модульное
- Интеграционное
- Регрессионное
- Комплексное

# Ход работы: мониторинг

- Ведётся мониторинг производительности, улучшений и регрессий в состоянии разработанного решения на основе VictoriaMetrics и Grafana



## Заключение

Для достижения **цели выпускной квалификационной работы** были получены следующие **результаты**:

- рассмотрены модели кода и возможности в статических анализаторах – MyPy, PyRight, и средах разработки – SRC IDE, PyCharm, VSCode (Pylance)
- на основе модели кода были реализованы основные возможности SRC IDE для Python, а именно: анализы кода, действия для исправления проблем и реструктуризации кода, а также ассистенты для написания кода
- проведено модульное, интеграционное, регрессионное и комплексное тестирование разработанного решения, создана соответствующая инфраструктура
- настроен мониторинг производительности и точности разработанного решения, создана инфраструктура на основе базы данных VictoriaMetrics и системы визуализации Grafana
- выполнена апробация разработанного решения на основе мониторинга, а также отзывов, сценариев и обнаруженных проблем у пользователей SRC IDE для Python