|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА  Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ**

**Ознакомительная практика**

приказ Университета о направлении на практику от «12» февраля 2025 г. №1427-C

|  |  |  |
| --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студент группы ИКБО-10-24 | «\_\_» июня 2025 | Лесовой К.Р.  (подпись и расшифровка подписи) |
| Отчет утвержден.  Допущен к защите: |  |  |
| Руководитель практики  от кафедры | «\_\_» июня 2025 | Маличенко С.В.  (подпись и расшифровка подписи) |
|  |  |  |

Москва 2025 г.

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА  Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ**

**Ознакомительная практика**

**Студенту 1 курса учебной группы ИКБО-10-24**

**Лесовому Кириллу Романовичу**

**Место и время практики:** РТУ МИРЭА кафедра ИиППО, с 10 февраля 2025 г. по 31 мая 2025 г.

**Должность на практике:**  студент

**1. СОДЕРЖАНИЕ ПРАКТИКИ:**

1.1. Изучить: основные методы отладки программ на языке Python, включая использование встроенных и сторонних инструментов, и подходы к выявлению ошибок в коде.

1.2. Практически выполнить: подготовить примеры программ, которые нужно проанализировать, и отладить их с целью научиться пользоваться инструментами отладки кода.

1.3. Ознакомиться: с библиотеками для анализа работы кода и современными инструментами отладки, такими как интегрированные среды разработки (IDE) с поддержкой дебаггинга, и их функционалом для упрощения процесса.

**2. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ:** подготовить доклад на научно-техническую конференцию студентов и аспирантов РТУ МИРЭА или иную конференцию, подготовить презентационный материал

**3. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ:** В процессе практики рекомендуется использовать периодические издания и отраслевую литературу годом издания не старше 5 лет от даты начала прохождения практики

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Руководитель практики от кафедры  «10» февраля 2025 г. | | | *Подпись* | | (Маличенко С.В.) |
| Задание получил | |  |  | |  |
| «10» февраля 2025 г. | |  | *Подпись* | | (Лесовой К.Р.) |
|  | |  |  | |  |
|  | |  |  | |  |
| **СОГЛАСОВАНО:** | |  |  | |  |
| Заведующий кафедрой: | | | | | |
| «10» февраля 2025 г. | | | *Подпись* | | (Болбаков Р.Г.) |
| **Проведенные инструктажи:** |  | | |  | |
| Охрана труда: |  | | | «10» февраля 2025 г. | |
| Инструктирующий | *Подпись* | | | Болбаков Р.Г., зав. кафедрой ИиППО | |
| Инструктируемый | *Подпись* | | | Лесовой К.Р. | |
|  |  | | |  | |
| Техника безопасности: |  | | | «10» февраля 2025 г. | |
| Инструктирующий | *Подпись* | | | Болбаков Р.Г., зав. кафедрой ИиППО | |
| Инструктируемый | *Подпись* | | | Лесовой К.Р. | |
|  |  | | |  | |
| Пожарная безопасность: |  | | | «10» февраля 2025 г. | |
| Инструктирующий | *Подпись* | | | Болбаков Р.Г., зав. кафедрой ИиППО | |
| Инструктируемый | *Подпись* | | | Лесовой К.Р. | |
|  | | | |  | |
| С правилами внутреннего распорядка ознакомлен: | | | | «10» февраля 2025 г. | |
|  | *Подпись* | | | Лесовой К.Р. | |

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА  Российский технологический университет»**  **РТУ МИРЭА** |

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ  
ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКИ**

студента Жуйковой В.А. 1 курса группы ИКБО-10-24 очной формы обучения, обучающегося по направлению подготовки 09.03.04 Программная инженерия

|  |  |  |  |
| --- | --- | --- | --- |
| **Неделя** | **Сроки**  **выполнения** | **Этап** | **Отметка о выполнении** |
| 1 | 10.02.2025 | Подготовительный этап, включающий в себя организационное собрание (Вводная лекция о порядке организации и прохождения ознакомительной практики, инструктаж по технике безопасности, получение задания на практику) |  |
| 5 | 10.03.2025 | Участие в круглом столе на тему «Проблема достоверности информации в современном мире» |  |
| 6 | 20.03.2025 | Участие в круглом столе на тему «Информационно-коммуникационные технологии для организации информационного процесса» |  |
| 8 | 31.03.2025 | Участие в круглом столе на тему «Информационная и библиографическая культура» |  |
| 8 | 01.04.2025 | Исследовательский этап (Поиск, отбор и анализ материалов для выполнения задания по практике) |  |
| 10 | 18.04.2025 | Согласование с руководителем доклада на научно-техническую конференцию студентов и аспирантов РТУ МИРЭА |  |
| 11 | 21.04.2025 | Представление руководителю структурированного материала: аналитический обзор предметной области |  |
| 11 | 22.04.2025 | Технологический этап (разработка программного продукта) |  |
| 15 | 19.05.2025 | Представление разработанного программного продукта |  |
| 16 | 31.05.2025 | Подготовка окончательной версии отчета и программного продукта (Оформление материалов отчета в полном соответствии с требованиями на оформление ГОСТ 7.32-2017) |  |

Руководитель практики от  
кафедры \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Маличенко С.В. к.т.н., ассистент/

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Лесовой К.Р./

**Согласовано:**

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Болбаков Р.Г., к.т.н., доцент/

**Пункт 1: Изучить основные методы отладки и профилирования программ на языке Python**

Для выполнения этого пункта вам нужно изучить ключевые инструменты и технологии, которые используются для отладки и профилирования кода на Python, а также подходы к выявлению ошибок. Вот конкретный план:

**Инструменты и технологии для изучения:**

1. **Встроенные инструменты для отладки:**
   * **pdb (Python Debugger):** Стандартный отладчик Python. Изучите, как запускать его через командную строку (python -m pdb script.py), устанавливать точки останова (breakpoint()), просматривать значения переменных (p variable) и выполнять код пошагово (n, s, c).
   * **print() и logging:** Простые методы вывода данных. Освойте базовое использование print() для быстрой проверки и модуль logging для структурированного логирования (например, настройка уровней логирования: DEBUG, INFO, ERROR).
2. **Сторонние инструменты для отладки:**
   * **PyCharm:** Интегрированная среда разработки (IDE) с мощным отладчиком. Изучите, как устанавливать точки останова, запускать отладку и анализировать стек вызовов.
   * **Visual Studio Code (VSCode):** Лёгкая IDE с поддержкой отладки через расширение Python. Освойте настройку конфигурации отладки и работу с переменными в процессе выполнения.
   * **IPython и Jupyter Notebook:** Интерактивные среды. Изучите выполнение кода по ячейкам и использование команды %debug для отладки.
3. **Инструменты для профилирования:**
   * **cProfile:** Встроенный модуль для анализа времени выполнения. Изучите запуск (python -m cProfile script.py) и интерпретацию результатов (время выполнения функций).
   * **timeit:** Модуль для замера времени небольших фрагментов кода. Освойте использование через командную строку или внутри скрипта (timeit.timeit()).
   * **memory\_profiler:** Сторонняя библиотека для анализа памяти. Изучите установку (pip install memory\_profiler) и аннотацию функций (@profile).
   * **line\_profiler:** Сторонний инструмент для построчного профилирования. Освойте установку (pip install line\_profiler) и использование (kernprof -l script.py).
4. **Подходы к выявлению ошибок:**
   * **Тестирование:** Изучите основы модуля unittest (встроенного) и библиотеки pytest (сторонней) для написания тестов.
   * **Статический анализ:** Освойте инструменты pylint и flake8 для проверки кода на ошибки и соответствие стандартам PEP 8.
   * **Логирование:** Изучите настройку модуля logging для отслеживания ошибок и событий (например, вывод в файл или консоль).

**Рекомендации:**

* Прочитайте официальную документацию Python по pdb, cProfile, timeit, logging.
* Ознакомьтесь с руководствами по установке и базовому использованию сторонних библиотек (memory\_profiler, line\_profiler, pytest, pylint, flake8).
* Попробуйте запустить простые примеры с каждым инструментом, чтобы понять их работу.

**Пункт 3: Ознакомиться с библиотеками и инструментами для анализа работы кода**

Для этого пункта вам нужно изучить встроенные и сторонние библиотеки, а также современные IDE с поддержкой отладки. Вот описание:

**Встроенные библиотеки:**

1. **pdb:** Стандартный отладчик для пошагового выполнения кода и анализа ошибок.
2. **cProfile и profile:** Инструменты профилирования для замера времени выполнения функций и анализа производительности.
3. **timeit:** Утилита для быстрого измерения времени выполнения небольших участков кода.
4. **logging:** Модуль для записи событий и ошибок в процессе выполнения программы.

**Сторонние библиотеки:**

1. **memory\_profiler:** Анализирует использование памяти, помогает выявить утечки и оптимизировать код.
2. **line\_profiler:** Выполняет построчное профилирование, показывая время выполнения каждой строки.
3. **pytest:** Удобная библиотека для написания и запуска тестов, упрощает проверку кода.
4. **pylint и flake8:** Инструменты статического анализа для поиска ошибок и улучшения качества кода. **black(?) ruff! mypy!**

**Интегрированные среды разработки (IDE):**

1. **PyCharm:** Многофункциональная IDE с поддержкой отладки (точки останова, просмотр переменных), профилирования, автодополнения и интеграции с системами контроля версий (например, Git).
2. **Visual Studio Code (VSCode):** Лёгкая IDE с расширением для Python, предоставляет отладку, линтинг и настройку окружения.
3. **Eclipse с PyDev:** Альтернативная IDE с функциями отладки, рефакторинга и анализа кода.

**Другие инструменты:**

1. **Jupyter Notebook:** Интерактивная среда для выполнения и отладки кода по ячейкам.
2. **IPython:** Расширенная оболочка с поддержкой отладки (%debug) и анализа.

**Рекомендации:**

* Установите PyCharm и VSCode, настройте их для работы с Python.
* Попробуйте Jupyter Notebook для интерактивного анализа кода.
* Ознакомьтесь с примерами использования каждой библиотеки и IDE через документацию или обучающие материалы.

**Пункт 2: Практически выполнить анализ и отладку примеров программ**

Для этого пункта вам нужно подготовить примеры программ, провести их отладку и профилирование с использованием изученных инструментов, а затем описать процесс в отчёте.

**План выполнения:**

1. **Выбор программ:**
   * Подготовьте 4-5 программ разной сложности с типичными ошибками (синтаксическими, логическими, ошибками времени выполнения) и неоптимальными участками.
   * Убедитесь, что программы выполняют полезные функции, чтобы анализ был интересным.
2. **Отладка:**
   * Используйте pdb для пошагового выполнения и выявления ошибок.
   * Примените print() и logging для отслеживания значений переменных.
   * Проведите отладку в IDE (PyCharm или VSCode) с использованием точек останова.
3. **Профилирование:**
   * Примените cProfile для анализа времени выполнения.
   * Используйте memory\_profiler для проверки использования памяти.
   * Выполните построчное профилирование с line\_profiler.
4. **Оптимизация:**
   * На основе результатов профилирования внесите улучшения в код.
   * Повторно проверьте производительность после изменений.
5. **Документирование:**
   * Опишите процесс отладки и профилирования для каждой программы в отчёте: какие ошибки найдены, какие инструменты использованы, как оптимизирован код.

**Краткое описание программ для анализа:**

1. **Простой калькулятор:**
   * Функция: Принимает два числа и операцию (+, -, \*, /), возвращает результат.
   * Ошибки: Деление на ноль, неверный ввод операции.
   * Цель: Отладить исключения и улучшить обработку ввода.
2. **Обработка списка:**
   * Функция: Вычисляет сумму, среднее, максимум и минимум списка чисел.
   * Ошибки: Пустой список, нечисловые элементы.
   * Цель: Исправить ошибки и оптимизировать вычисления.
3. **Работа с файлами:**
   * Функция: Читает текстовый файл и подсчитывает слова, строки, символы.
   * Ошибки: Файл не найден, проблемы с кодировкой.
   * Цель: Обработать исключения и ускорить чтение файла.
4. **Рекурсивная функция:**
   * Функция: Вычисляет факториал или числа Фибоначчи.
   * Проблемы: Переполнение стека, низкая производительность.
   * Цель: Оптимизировать алгоритм (например, добавить мемоизацию).
5. **Многопоточное приложение:**
   * Функция: Выполняет несколько задач параллельно (например, загрузка данных).
   * Ошибки: Гонки данных, взаимоблокировки.
   * Цель: Устранить проблемы синхронизации.

**Итоговый порядок выполнения для отчёта:**

1. **Пункт 1:** Изучите указанные инструменты и технологии, задокументируйте основные выводы (что узнали, как работают инструменты).
2. **Пункт 3:** Ознакомьтесь с библиотеками и IDE, опишите их возможности и преимущества в отчёте.
3. **Пункт 2:** Подготовьте программы, выполните отладку и профилирование, опишите процесс и результаты для каждой программы.

Этот план позволит вам последовательно выполнить все задачи и подготовить полный отчёт по практике. Удачи!

**Ключевые моменты**

* Исследование показывает, что статьи с Habr содержат полезную информацию для вашей практики по отладке и профилированию на Python.
* Наиболее релевантные материалы включают инструменты, такие как cProfile, PDB, и IDE, а также практические примеры.
* Кажется вероятным, что эти статьи помогут дополнить ваш отчет, особенно в теоретической и практической частях.

**Анализ релевантных статей**

Ваша практика фокусируется на методах и инструментах отладки и профилирования программ на Python, включая изучение инструментов, практическое применение и знакомство с библиотеками. Анализ статей с Habr выявил несколько релевантных источников, которые можно использовать для дополнения вашего отчета.

**Релевантные статьи и их использование**

Ниже приведены статьи, которые напрямую связаны с вашей темой, и способы их интеграции в отчет:

* **Профилирование Python-программ и анализ их производительности** ([Профилирование Python-программ и анализ их производительности](https://habr.com/ru/companies/wunderfund/articles/656571/)): Эта статья охватывает инструменты, такие как cProfile, py-spy, line\_profiler, pyheat и memory\_profiler, а также методы оптимизации. Она идеально подходит для изучения методов профилирования (пункт 1) и знакомства с библиотеками (пункт 3).
* **Немного об отладке. Часть 1 и Часть 2** ([Немного об отладке. Часть 1](https://habr.com/ru/articles/867478/), [Немного об отладке. Часть 2](https://habr.com/ru/articles/867482/)): Эти статьи подробно описывают основы отладки, включая механизмы для Python (CPython), что полезно для понимания теоретических аспектов (пункт 1) и практического применения (пункт 2).
* **Серия статей от VK о профилировании и отладке Python**:
  + [Профилирование и отладка Python](https://habr.com/ru/companies/vk/articles/201594/) — вводит в теорию профилирования и отладки.
  + [Профилирование и отладка Python, переходим к практике](https://habr.com/ru/companies/vk/articles/201778/) — предлагает практические примеры, полезные для пункта 2.
  + [Профилирование и отладка Python, инструменты](https://habr.com/ru/companies/vk/articles/202832/) — фокусируется на инструментах, что подходит для пункта 3.
  + [Профилирование и отладка Python, отладка](https://habr.com/ru/companies/vk/articles/205426/) — детально описывает отладочные инструменты, такие как PDB и IPDB, что также релевантно для пункта 3.  
    Эти статьи охватывают как теорию, так и практику, что делает их ценным ресурсом для всех трех пунктов вашей практики.

**Дополнение отчета**

Вы можете использовать эти статьи для:

* **Теоретической части (пункт 1)**: Включите обзор инструментов, таких как cProfile, PDB, и методов профилирования, упомянутых в статьях. Например, статья от Wunderfund подробно описывает статистические и событийные профайлеры.
* **Практической части (пункт 2)**: Используйте практические примеры из статей VK для создания программ для анализа. Например, статьи предлагают примеры оптимизации кода, которые можно отладить с помощью PDB или профилировать с помощью line\_profiler.
* **Знакомство с библиотеками и IDE (пункт 3)**: Статьи упоминают библиотеки, такие как memory\_profiler, и IDE, такие как PyCharm, что можно включить в описание функционала для упрощения процесса.

**Остальные статьи**

Остальные статьи, такие как "Простые лайфхаки для автоматизации работы с помощью Python" ([Простые лайфхаки для автоматизации работы с помощью Python](https://habr.com/ru/companies/netologyru/articles/881796/)) или "Нагрузочное тестирование: простым языком" ([Нагрузочное тестирование: простым языком](https://habr.com/ru/companies/vsk_insurance/articles/866304/)), менее релевантны, но могут быть упомянуты как дополнительный контекст для анализа производительности, если это необходимо. Однако они не фокусируются на конкретных инструментах отладки и профилирования.

**Отчет: Подробный анализ статей с Habr для практики по отладке и профилированию на Python**

Этот раздел представляет собой детальный обзор анализа статей с Habr, предоставленных для дополнения вашего отчета по учебной практике на тему "Методы и инструменты отладки и профилирования программного приложения при использовании языка Python". Анализ проведен с учетом содержания вашей практики, включающего изучение методов, практическое выполнение и знакомство с библиотеками.

**Методология анализа**

Были рассмотрены все предоставленные статьи, и каждая из них оценена на релевантность к вашей теме. Основное внимание уделено статьям, которые напрямую касаются отладки, профилирования и инструментов для Python. Для каждой релевантной статьи предоставлено краткое описание содержания и рекомендации по использованию в отчете.

**Релевантные статьи и их содержание**

**1. Профилирование Python-программ и анализ их производительности**

* **Ссылка**: [Профилирование Python-программ и анализ их производительности](https://habr.com/ru/companies/wunderfund/articles/656571/)
* **Содержание**: Статья фокусируется на профилировании Python-кода, включая инструменты, такие как cProfile, py-spy, line\_profiler, pyheat и memory\_profiler. Описаны методы оптимизации, такие как использование PyPy, Cython и интеграция с C/Fortran. Упомянуты преимущества и недостатки статистических и событийных профайлеров, их пригодность для продакшн.
* **Релевантность**: Высока для пункта 1 (изучение методов профилирования) и пункта 3 (знакомство с библиотеками).
* **Использование в отчете**: Включите обзор инструментов профилирования, таких как cProfile и memory\_profiler, и обсудите их применение для анализа производительности. Это также может быть полезно для практической части, если вы выберете программы для оптимизации.

**2. Немного об отладке. Часть 1 и Часть 2**

* **Ссылки**: [Немного об отладке. Часть 1](https://habr.com/ru/articles/867478/), [Немного об отладке. Часть 2](https://habr.com/ru/articles/867482/)
* **Содержание**:
  + Часть 1: Охватывает основы отладки, включая использование ptrace, INT 3, и форматы отладочной информации, такие как DWARF. Упомянута отладка в различных средах, включая CPython (Python), что включает механизмы отладки интерпретатора.
  + Часть 2: Продолжает тему, обсуждая разработку отладчиков (например, Dumbugger), парсинг DWARF и отладку в управляемых средах, включая Python.
* **Релевантность**: Высока для пункта 1 (теоретическое изучение) и пункта 2 (практическое применение, особенно для понимания механизмов).
* **Использование в отчете**: Используйте эти статьи для описания теоретических основ отладки в Python, включая внутренние механизмы CPython. Это может быть полезно для объяснения, как работают встроенные инструменты, такие как PDB.

**3. Серия статей от VK о профилировании и отладке Python**

Ниже представлена таблица с описанием каждой статьи из серии, их содержания и релевантности:

| **Название статьи** | **Ссылка** | **Содержание** | **Релевантность** | **Использование в отчете** |
| --- | --- | --- | --- | --- |
| Профилирование и отладка Python | [Ссылка](https://habr.com/ru/companies/vk/articles/201594/) | Введение в профилирование и отладку, обсуждение подходов, таких как метод детального анализа. | Пункт 1 (теория), Пункт 3 (инструменты) | Опишите теоретические основы и подходы к профилированию. |
| Профилирование и отладка Python, переходим к практике | [Ссылка](https://habr.com/ru/companies/vk/articles/201778/) | Практические примеры, включая использование timeit, Graphite, StatsD и других инструментов. | Пункт 2 (практика), Пункт 3 (инструменты) | Используйте для создания примеров программ для анализа и отладки. |
| Профилирование и отладка Python, инструменты | [Ссылка](https://habr.com/ru/companies/vk/articles/202832/) | Описание конкретных инструментов профилирования и отладки, таких как StatProf, django-live-profiler. | Пункт 3 (знакомство с библиотеками) | Включите обзор инструментов для анализа производительности. |
| Профилирование и отладка Python, отладка | [Ссылка](https://habr.com/ru/companies/vk/articles/205426/) | Детальное описание отладочных инструментов, таких как PDB, IPDB, PuDB, и IDE, таких как PyCharm. | Пункт 1 (методы отладки), Пункт 3 (IDE) | Опишите инструменты отладки и их функционал для упрощения процесса. |

* **Релевантность**: Эти статьи охватывают все аспекты вашей практики, предоставляя как теоретическую базу, так и практические примеры.
* **Использование в отчете**: Интегрируйте их для создания полного обзора методов, инструментов и практических примеров. Например, статьи о практике могут быть использованы для пункта 2, где вы готовите программы для анализа.

**Менее релевантные статьи**

Следующие статьи имеют меньшую релевантность, но могут быть упомянуты как дополнительный контекст:

* **Простые лайфхаки для автоматизации работы с помощью Python** ([Простые лайфхаки для автоматизации работы с помощью Python](https://habr.com/ru/companies/netologyru/articles/881796/)): Фокусируется на автоматизации задач, включая тестирование и DevOps, но не на конкретных инструментах отладки. Может быть полезна для контекста, если вы анализируете производительность автоматизированных скриптов.
* **Нагрузочное тестирование: простым языком** ([Нагрузочное тестирование: простым языком](https://habr.com/ru/companies/vsk_insurance/articles/866304/)): Обсуждает тестирование производительности, но не фокусируется на Python или инструментах профилирования. Может быть упомянута для общего понимания анализа производительности.
* Другие статьи, такие как "Порядок разрешения методов в Python" ([Порядок разрешения методов в Python](https://habr.com/ru/articles/62203/)) или "Параллелизм может быть только 1" ([Параллелизм может быть только 1](https://habr.com/ru/articles/869856/)), менее релевантны, так как они касаются других аспектов Python, не связанных с отладкой и профилированием.

**Рекомендации по интеграции**

Для дополнения вашего отчета рекомендуется:

* Включить обзор инструментов из статей, таких как cProfile, PDB, memory\_profiler, и IDE (PyCharm, VSCode), упомянутых в релевантных источниках.
* Использовать практические примеры из статей VK для создания программ для анализа в пункте 2. Например, вы можете выбрать программы, связанные с оптимизацией, и применить профилирование с помощью line\_profiler.
* Цитировать эти статьи в отчете для поддержки ваших выводов и предоставления дополнительного чтения.

**Заключение**

Анализ показал, что статьи с Habr, особенно те, что фокусируются на профилировании и отладке Python, значительно обогащают ваш отчет. Они предоставляют как теоретическую базу, так и практические примеры, что соответствует всем трем пунктам вашей практики. Рекомендуется сосредоточиться на указанных релевантных источниках для создания полного и информативного отчета.

**Ключевые цитирования**

* [Профилирование Python-программ и анализ их производительности на Habr](https://habr.com/ru/companies/wunderfund/articles/656571/)
* [Немного об отладке. Часть 1 на Habr](https://habr.com/ru/articles/867478/)
* [Немного об отладке. Часть 2 на Habr](https://habr.com/ru/articles/867482/)
* [Профилирование и отладка Python на Habr от VK](https://habr.com/ru/companies/vk/articles/201594/)
* [Профилирование и отладка Python, переходим к практике на Habr от VK](https://habr.com/ru/companies/vk/articles/201778/)
* [Профилирование и отладка Python, инструменты на Habr от VK](https://habr.com/ru/companies/vk/articles/202832/)
* [Профилирование и отладка Python, отладка на Habr от VK](https://habr.com/ru/companies/vk/articles/205426/)
* [Простые лайфхаки для автоматизации работы с помощью Python на Habr](https://habr.com/ru/companies/netologyru/articles/881796/)
* [Нагрузочное тестирование: простым языком на Habr от VSK Insurance](https://habr.com/ru/companies/vsk_insurance/articles/866304/)
* [Порядок разрешения методов в Python на Habr](https://habr.com/ru/articles/62203/)
* [Параллелизм может быть только 1 на Habr](https://habr.com/ru/articles/869856/)

**Объединенный план для отчета по учебной практике**

Ниже представлен объединенный пошаговый план, который включает изучение основных методов отладки и профилирования, ознакомление с библиотеками и инструментами для анализа работы кода, практическое выполнение анализа и отладки примеров программ, а также интеграцию анализа статей с Habr. Этот план объединяет все данные из предыдущих шагов и предоставляет структурированный подход к выполнению задач практики.

**1. Изучение основных методов отладки и профилирования**

Цель этого раздела — освоить базовые и продвинутые инструменты для отладки и профилирования кода на Python, а также изучить подходы к выявлению ошибок.

* **Встроенные инструменты для отладки**:
  + **pdb (Python Debugger)**:
    - Запуск через командную строку (python -m pdb script.py).
    - Установка точек останова (breakpoint() или команда b).
    - Просмотр значений переменных (p variable).
    - Пошаговое выполнение (n — следующий шаг, s — шаг внутрь, c — продолжить).
  + **print() и logging**:
    - Использование print() для простого вывода данных.
    - Настройка модуля logging для структурированного логирования (уровни: DEBUG, INFO, ERROR; вывод в файл или консоль).
* **Сторонние инструменты для отладки**:
  + **PyCharm**:
    - Установка точек останова (клик на полях редактора).
    - Запуск отладки (кнопка "Debug").
    - Анализ стека вызовов и значений переменных.
  + **Visual Studio Code (VSCode)**:
    - Настройка конфигурации отладки через launch.json.
    - Работа с переменными в панели отладки.
  + **IPython и Jupyter Notebook**:
    - Выполнение кода по ячейкам.
    - Использование команды %debug для пост-отладки исключений.
* **Инструменты для профилирования**:
  + **cProfile**:
    - Запуск через python -m cProfile script.py.
    - Интерпретация результатов (время выполнения функций, количество вызовов).
  + **timeit**:
    - Измерение времени через командную строку (python -m timeit) или в коде (timeit.timeit()).
  + **memory\_profiler**:
    - Установка (pip install memory\_profiler).
    - Аннотация функций с @profile для анализа памяти.
  + **line\_profiler**:
    - Установка (pip install line\_profiler).
    - Использование через kernprof -l script.py для построчного анализа.
* **Подходы к выявлению ошибок**:
  + **Тестирование**:
    - Основы unittest (встроенный модуль) — написание тестовых классов и методов.
    - Основы pytest (сторонняя библиотека) — запуск тестов через pytest test\_file.py.
  + **Статический анализ**:
    - Использование pylint и flake8 для проверки стиля и потенциальных ошибок.
  + **Логирование**:
    - Настройка модуля logging (формат сообщений, ротация логов).

**Интеграция статей с Habr**:

* Использовать статью [Профилирование Python-программ и анализ их производительности](https://habr.com/ru/companies/wunderfund/articles/656571/) для детального описания cProfile и memory\_profiler.
* Включить материалы из [Немного об отладке. Часть 1](https://habr.com/ru/articles/867478/) и [Часть 2](https://habr.com/ru/articles/867482/) для теоретического обоснования отладки с помощью pdb.
* Цитировать серию статей от VK ([Профилирование и отладка Python](https://habr.com/ru/companies/vk/articles/201594/), [инструменты](https://habr.com/ru/companies/vk/articles/202832/)) для описания подходов к профилированию.

**2. Ознакомление с библиотеками и инструментами для анализа работы кода**

Этот раздел посвящен изучению встроенных и сторонних библиотек, а также интегрированных сред разработки (IDE) для анализа и улучшения кода.

* **Встроенные библиотеки**:
  + **pdb**: Стандартный отладчик для пошагового выполнения.
  + **cProfile и profile**: Инструменты для анализа производительности.
  + **timeit**: Утилита для измерения времени выполнения небольших фрагментов кода.
  + **logging**: Модуль для записи событий и ошибок.
* **Сторонние библиотеки**:
  + **memory\_profiler**: Анализ потребления памяти программами.
  + **line\_profiler**: Построчное профилирование кода.
  + **pytest**: Удобный инструмент для написания и запуска тестов.
  + **pylint и flake8**: Средства статического анализа кода.
* **Интегрированные среды разработки (IDE)**:
  + **PyCharm**:
    - Поддержка отладки, профилирования, автодополнения.
    - Интеграция с тестированием и статическим анализом.
  + **Visual Studio Code (VSCode)**:
    - Легкая IDE с расширением Python.
    - Поддержка отладки, линтинга и терминалов.
  + **Eclipse с PyDev**:
    - Альтернативная IDE с функциями отладки и рефакторинга.
* **Другие инструменты**:
  + **Jupyter Notebook**: Интерактивная среда для выполнения кода по ячейкам и отладки.
  + **IPython**: Расширенная оболочка Python с поддержкой %debug.

**Интеграция статей с Habr**:

* Использовать статью [Профилирование и отладка Python, инструменты](https://habr.com/ru/companies/vk/articles/202832/) для описания дополнительных инструментов, таких как StatProf.
* Включить информацию из [Профилирование и отладка Python, отладка](https://habr.com/ru/companies/vk/articles/205426/) для анализа отладочных возможностей PyCharm и IPython.
* Упомянуть [Простые лайфхаки для автоматизации работы с помощью Python](https://habr.com/ru/companies/netologyru/articles/881796/) как контекст для анализа производительности скриптов.

**3. Практическое выполнение анализа и отладки примеров программ**

Этот раздел включает практическую работу с программами: отладку, профилирование и оптимизацию.

* **Выбор программ**:
  + Подготовить 4-5 программ с типичными проблемами:
    1. Простой калькулятор (ошибки: деление на ноль, неверный ввод).
    2. Обработка списка (ошибки: пустой список, нечисловые элементы).
    3. Работа с файлами (ошибки: файл не найден, проблемы с кодировкой).
    4. Рекурсивная функция (проблемы: переполнение стека, низкая производительность).
    5. Многопоточное приложение (ошибки: гонки данных, взаимоблокировки).
* **Отладка**:
  + Использовать pdb для пошагового выполнения и поиска ошибок.
  + Применить print() и logging для вывода промежуточных значений.
  + Провести отладку в IDE (PyCharm или VSCode) с точками останова.
* **Профилирование**:
  + Применить cProfile для анализа времени выполнения.
  + Использовать memory\_profiler для проверки потребления памяти.
  + Выполнить построчное профилирование с line\_profiler.
* **Оптимизация**:
  + Внести улучшения в код на основе результатов профилирования (например, заменить рекурсию на цикл, устранить утечки памяти).
  + Повторно измерить производительность после изменений.
* **Документирование**:
  + Описать процесс для каждой программы:
    1. Какие ошибки найдены.
    2. Какие инструменты использованы.
    3. Как оптимизирован код.

**Интеграция статей с Habr**:

* Использовать примеры из [Профилирование и отладка Python, переходим к практике](https://habr.com/ru/companies/vk/articles/201778/) для создания программ.
* Применить методы оптимизации из [Профилирование Python-программ и анализ их производительности](https://habr.com/ru/companies/wunderfund/articles/656571/).
* Описать процесс отладки, основываясь на [Немного об отладке. Часть 1 и Часть 2](https://habr.com/ru/articles/867478/).

**4. Интеграция анализа статей с Habr в отчет**

Статьи с Habr используются для обогащения отчета теоретическими и практическими данными.

* **Теоретическая часть**:
  + Обзор инструментов (cProfile, pdb) с примерами из статей.
  + Цитирование для поддержки описания методов.
* **Практическая часть**:
  + Примеры программ и их анализ, вдохновленные статьями.
  + Описание применения инструментов на основе практических кейсов из статей.
* **Знакомство с библиотеками и IDE**:
  + Описание возможностей PyCharm, memory\_profiler и других инструментов, упомянутых в статьях.

**Рекомендации**:

* Цитировать статьи в отчете для обоснования выводов.
* Использовать статьи для углубленного понимания и добавления современных подходов.

**Итоговый порядок выполнения для отчета**

1. **Изучение методов (Пункт 1)**: Освоить инструменты и подходы, задокументировать выводы.
2. **Ознакомление с библиотеками и IDE (Пункт 3)**: Описать их возможности и преимущества.
3. **Практическая работа (Пункт 2)**: Подготовить программы, провести отладку и профилирование, описать результаты.
4. **Интеграция статей**: Добавить материалы с Habr для теоретической и практической частей.

Этот план обеспечивает полное выполнение задач практики и создание качественного отчета с учетом теоретических знаний, практических навыков и современных источников.