Московский авиационный институт

(Национальный исследовательский университет)

**Лабораторные работы 1 - 10**

**Архитектура программной среды**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Шарудин Д.В./

(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_/Семенов АС./

(Подпись)

Дата : \_\_ 2024

**Лабораторная 1.**  Интегрированные среды разработки и их базовые отличия

1. Интегрированная среда разработки (Integrated Development Environment, IDE) представляет собой программное обеспечение, которое объединяет в себе различные инструменты, необходимые для разработки программного обеспечения. Обычно IDE включает в себя текстовый редактор, компилятор или интерпретатор, отладчик, инструменты для автоматизации сборки и управления проектами, а также другие инструменты, упрощающие процесс разработки.
2. Интегрированная среда разработки (IDE) обычно включает в себя следующие компоненты:

* **Текстовый редактор:** Позволяет разработчику создавать, редактировать и просматривать исходный код программы.
* **Компилятор/Интерпретатор:** Компилятор используется для преобразования исходного кода в машинный код, который может быть выполнен компьютером. Интерпретатор выполняет исходный код построчно без предварительной компиляции.
* **Отладчик:** Инструмент, который помогает разработчику идентифицировать и исправлять ошибки в программе.
* **Управление проектом:** Включает инструменты для создания, управления и организации файлов и ресурсов проекта.
* **Автоматизация сборки:** Позволяет автоматизировать процесс компиляции и сборки программы, что упрощает развертывание и обновление приложения.

1. Интегрированная среда разработки (IDE) и текстовый редактор кода имеют схожие цели - обеспечить удобную среду для написания программного кода. Однако существуют существенные различия между ними:

* **Функциональность:**
  1. **IDE:** Предоставляет полноценную интегрированную среду для разработки, включая компиляторы, отладчики, системы управления проектами, автоматизацию сборки и другие инструменты.
  2. **Редактор кода:** Обычно ограничивается функциями редактирования текста и может иметь некоторые расширения или плагины, но не обладает полной функциональностью IDE.
* **Интеграция инструментов:**
  1. **IDE:** Интегрирует в себя все необходимые инструменты для разработки, что упрощает рабочий процесс разработчика.
  2. **Редактор кода:** Обычно ориентирован на базовые функции редактирования текста и может требовать установки дополнительных инструментов для полноценной разработки.
* **Уровень абстракции:**
  1. **IDE:** Предоставляет высокий уровень абстракции и обычно ориентирован на конкретные языки программирования.
  2. **Редактор кода:** Обычно имеет более низкий уровень абстракции и ориентирован на работу с текстовыми файлами.
* **Сложность использования:**
  1. **IDE:** Имеет более сложный интерфейс и может потребовать времени для освоения.
  2. **Редактор кода:** Обычно более прост в использовании и может быть предпочтителен для начинающих разработчиков или для выполнения простых задач.
* **Ресурсы:**
  1. **IDE:** Возможно, потребуется больше ресурсов компьютера для работы из-за интеграции множества инструментов.
  2. **Редактор кода:** Обычно имеет более низкие требования к ресурсам компьютера.

1. Плагины представляют собой расширения, которые позволяют настраивать функционал интегрированных сред разработки (IDE) под конкретные проекты, задачи или предпочтения разработчика. Вот несколько способов, как плагины могут быть полезны:

* **Дополнительные инструменты:** Плагины могут добавлять новые инструменты и функции, которых нет в стандартной поставке IDE. Например, плагин для анализа кода или интеграции с внешними сервисами.
* **Поддержка новых языков:** Если IDE не поддерживает определенный язык программирования по умолчанию, можно установить плагин, добавляющий поддержку этого языка.
* **Интеграция с внешними инструментами:** Плагины могут обеспечивать интеграцию с системами контроля версий, средствами непрерывной интеграции, системами управления задачами и другими внешними инструментами, упрощая рабочий процесс разработчика.
* **Настройка интерфейса:** Плагины могут изменять внешний вид и поведение IDE в соответствии с предпочтениями разработчика, добавляя новые темы оформления, расширенные возможности настройки интерфейса и т. д.
* **Улучшение производительности:** Некоторые плагины направлены на повышение производительности разработки, например, путем автоматизации рутинных задач или улучшения инструментов отладки и анализа кода.

1. Основная цель использования интегрированной среды разработки (IDE) заключается в обеспечении удобной и эффективной среды для написания, отладки и управления программным кодом. Вот основные цели использования IDE:

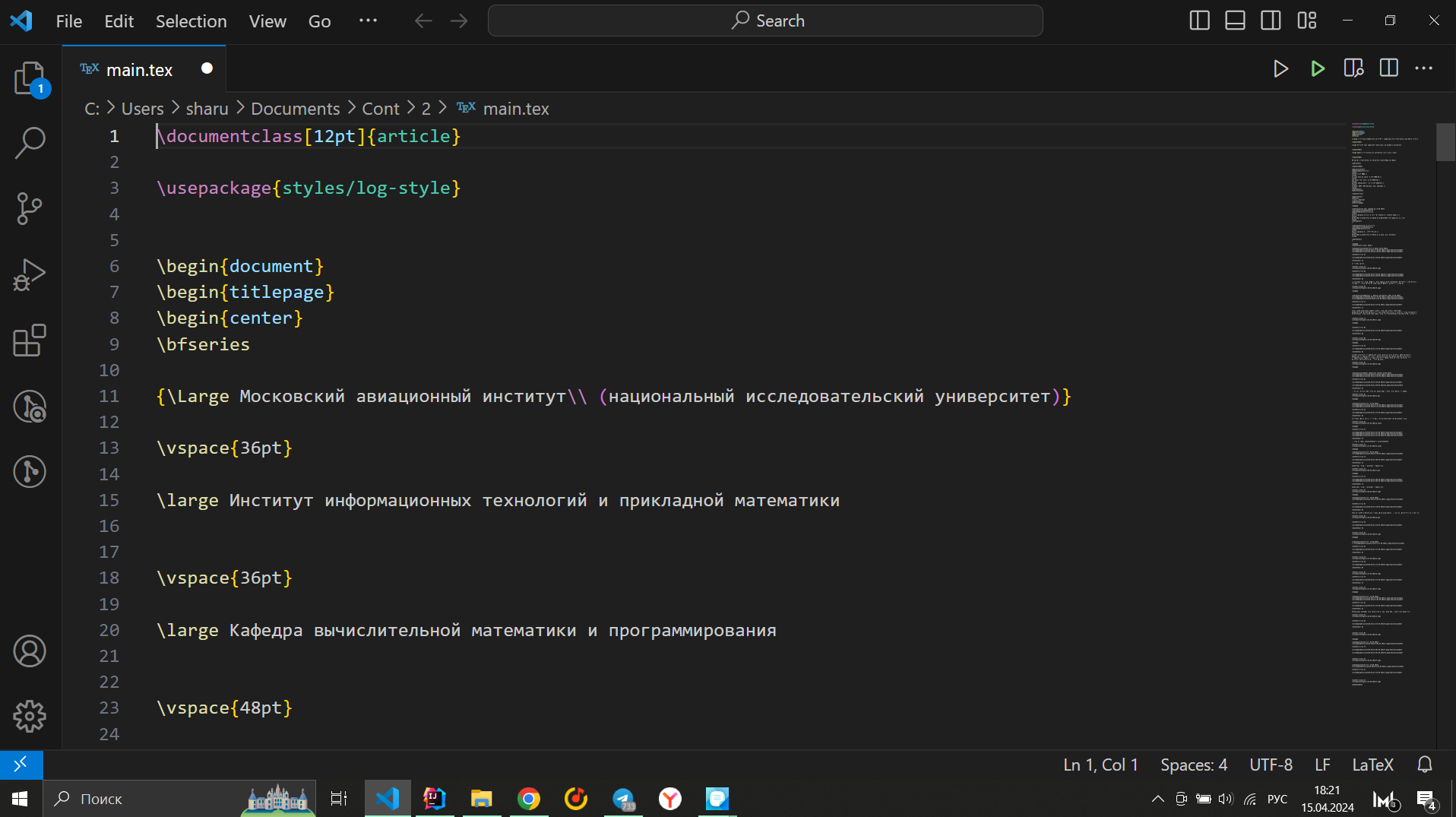
* **Увеличение производительности:** IDE предоставляет разработчику все необходимые инструменты и функции в одной среде, что упрощает рабочий процесс и повышает производительность. Вместо того чтобы использовать отдельные инструменты, разработчик может работать непосредственно в IDE, минимизируя время, затраченное на переключение между приложениями.
* **Упрощение разработки:** IDE предоставляет интегрированные инструменты для автоматизации рутинных задач, таких как сборка и отладка программы, управление зависимостями и версионирование кода. Это упрощает процесс разработки и позволяет сосредоточиться на создании функциональности приложения.
* **Улучшение качества кода:** IDE обеспечивает доступ к инструментам анализа кода, отладке и проверке синтаксиса, что помогает выявлять и исправлять ошибки на ранних стадиях разработки. Это способствует улучшению качества программного продукта и уменьшению количества ошибок.

**Задача 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Языки программирования | Свойства среды (автоматизация программирования) | Поддержка методологии разработки | Неучтенные факторы |
| VSCode | C, C#, Java, JavaScript, Ruby, PHP и др. | Интегрированный терминал, автозаполнение кода, отладчик, Git интеграция, IntelliSense, рефакторинг | Водопадная модель  Гибкая методология  DevOps  Lean | VSCode, или Visual Studio Code, это бесплатный и открытый исходный код редактор кода, разработанный компанией Microsoft. Он доступен для различных операционных систем, включая Windows, macOS и Linux. |
| PyCharm | Python | Интегрированный терминал, автозаполнение кода, отладчик, Git интеграция, IntelliSense, рефакторинг | V-модель  Водопадная модель | PyCharm - это интегрированная среда разработки (IDE) для языка программирования Python, разработанная компанией JetBrains. Она предоставляет множество инструментов и функций, облегчающих разработку, отладку и тестирование приложений на Python. |
| Intellij Idea | Java, Kotlin | Интегрированный терминал, автозаполнение кода, отладчик, Git интеграция, IntelliSense, рефакторинг | DevOps  Водопадная модель | IntelliJ IDEA - это интегрированная среда разработки (IDE) для различных языков программирования, таких как Java, Kotlin, Scala, Groovy и других. Она создана компанией JetBrains и широко используется разработчиками для создания профессиональных приложений. |
| Anaconda | Python, R | Интегрированный терминал, Git интеграция | Lean | Anaconda - это платформа и среда разработки для анализа данных и научных вычислений на языке программирования Python. Она включает в себя инструменты для работы с большими объемами данных, различные библиотеки и фреймворки, а также среду разработки Jupyter Notebook. |
| Visual Studio | C#, C++,PHP, Visual Basic .NET, Python и др. | Интегрированный терминал, автозаполнение кода, отладчик, Git интеграция, IntelliSense, рефакторинг | Водопадная модель  Гибкая методология | Visual Studio – это интегрированная среда разработки (IDE) разработанная компанией Microsoft. Она предоставляет разработчикам широкий спектр инструментов для создания приложений для различных платформ, включая Windows, macOS, Android и iOS. |

**Задача 2.**

Visual Studio Code (VS Code) - это бесплатный и открытый исходный код текстовый редактор, разработанный компанией Microsoft.



Вот несколько ключевых особенностей и характеристик Visual Studio Code:

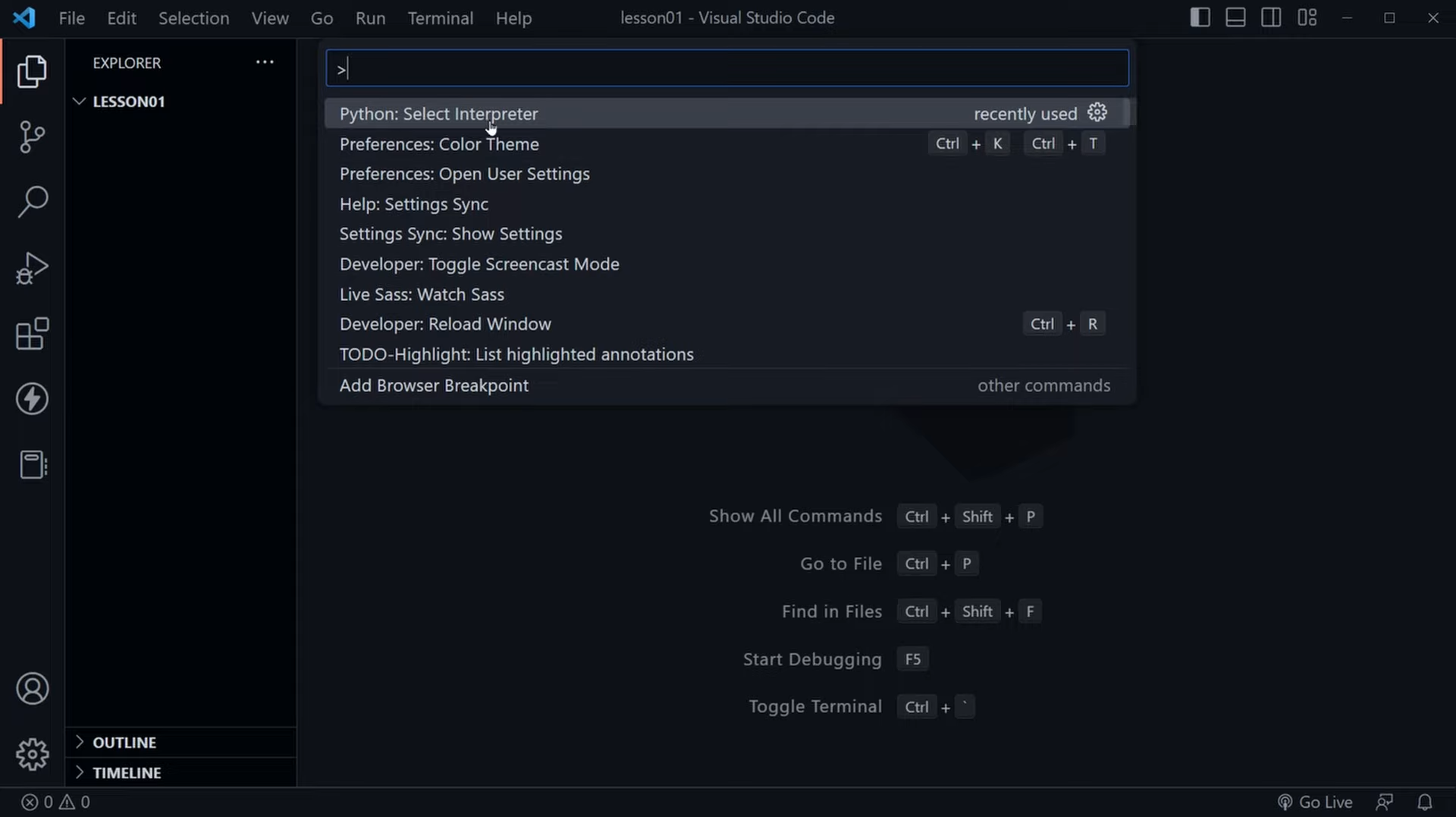
1. **Кроссплатформенность:** Visual Studio Code доступен для операционных систем Windows, macOS и Linux, что делает его доступным для широкого круга разработчиков.
2. **Легковесность и производительность:** В отличие от полноценной IDE, Visual Studio Code является легковесным текстовым редактором с возможностью расширения функционала с помощью плагинов. Это позволяет сохранить высокую производительность и меньшие требования к ресурсам компьютера.
3. **Интеграция с Git:** Visual Studio Code обладает встроенной поддержкой системы контроля версий Git, что позволяет разработчикам управлять своими проектами и делать коммиты прямо из редактора.
4. **Расширяемость:** С помощью расширений и плагинов Visual Studio Code можно расширить его функциональность для работы с различными языками программирования, интеграции с различными сервисами и инструментами, а также настроить его под собственные потребности.
5. **Интеллектуальные функции редактора:** Visual Studio Code предоставляет различные интеллектуальные функции, такие как автозавершение кода, подсветка синтаксиса, быстрая навигация по коду и многие другие, что улучшает процесс написания и редактирования кода.
6. **Интегрированная отладка:** VS Code предоставляет возможности для отладки кода прямо из редактора, что позволяет быстро находить и исправлять ошибки в программном коде.
7. **Широкое сообщество и поддержка:** Visual Studio Code имеет активное сообщество пользователей и разработчиков, что обеспечивает постоянное развитие и поддержку редактора, а также доступ к множеству полезных расширений и ресурсов.

Ниже представлен пример кода на языке программирования Python с учётом предустановленных расширений (Python).

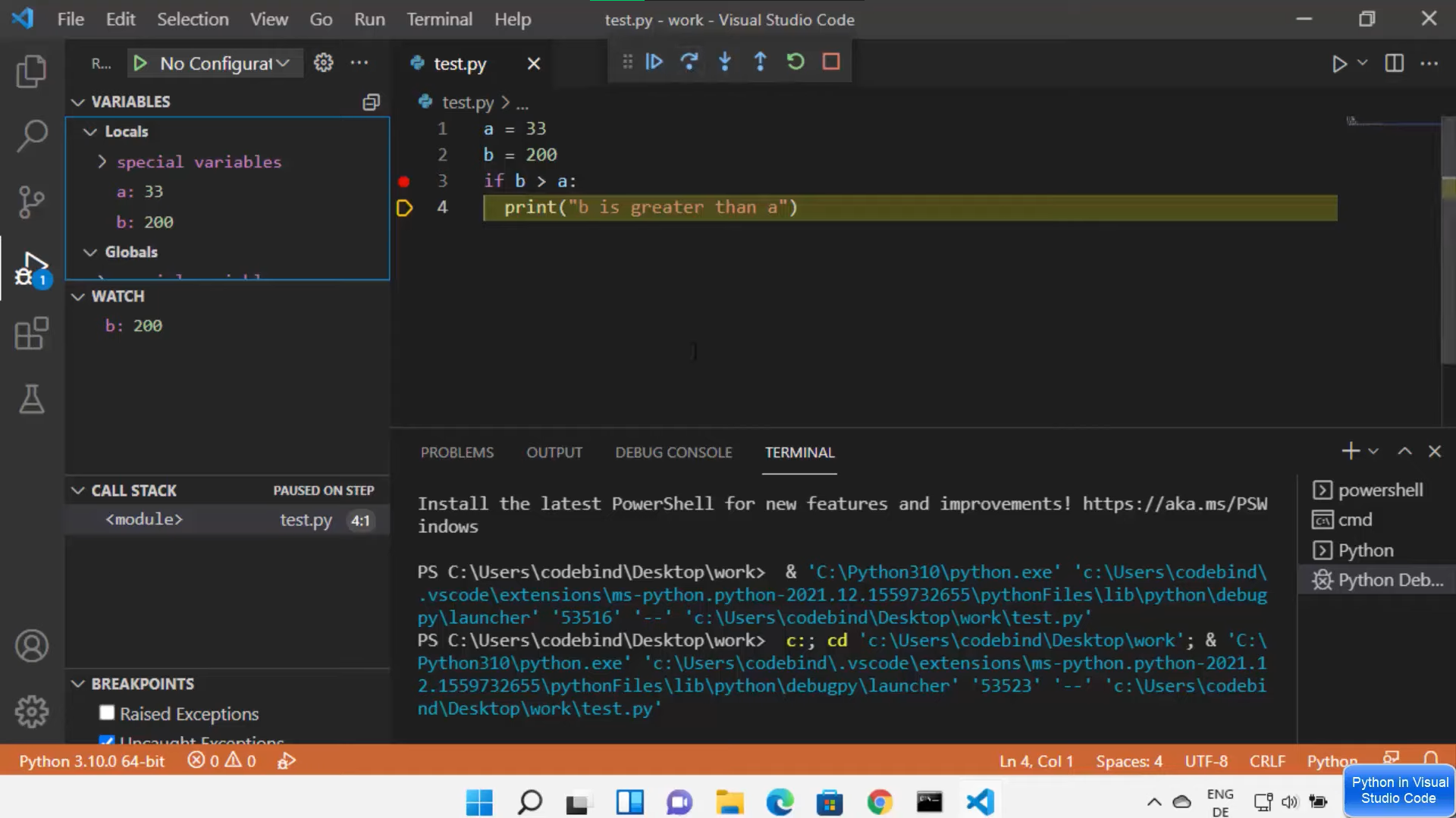
**Устанавливаем необходимые расширения:**



Расширение Visual Studio Code с широкой поддержкой языка Python (для всех активно поддерживаемых версий языка: >=3.7), включая такие функции, как IntelliSense (Pylance), анализ, отладку (Python Debugger), навигацию по коду, форматирование кода, рефакторинг, обозреватель переменных, обозреватель тестов и многое другое**.**

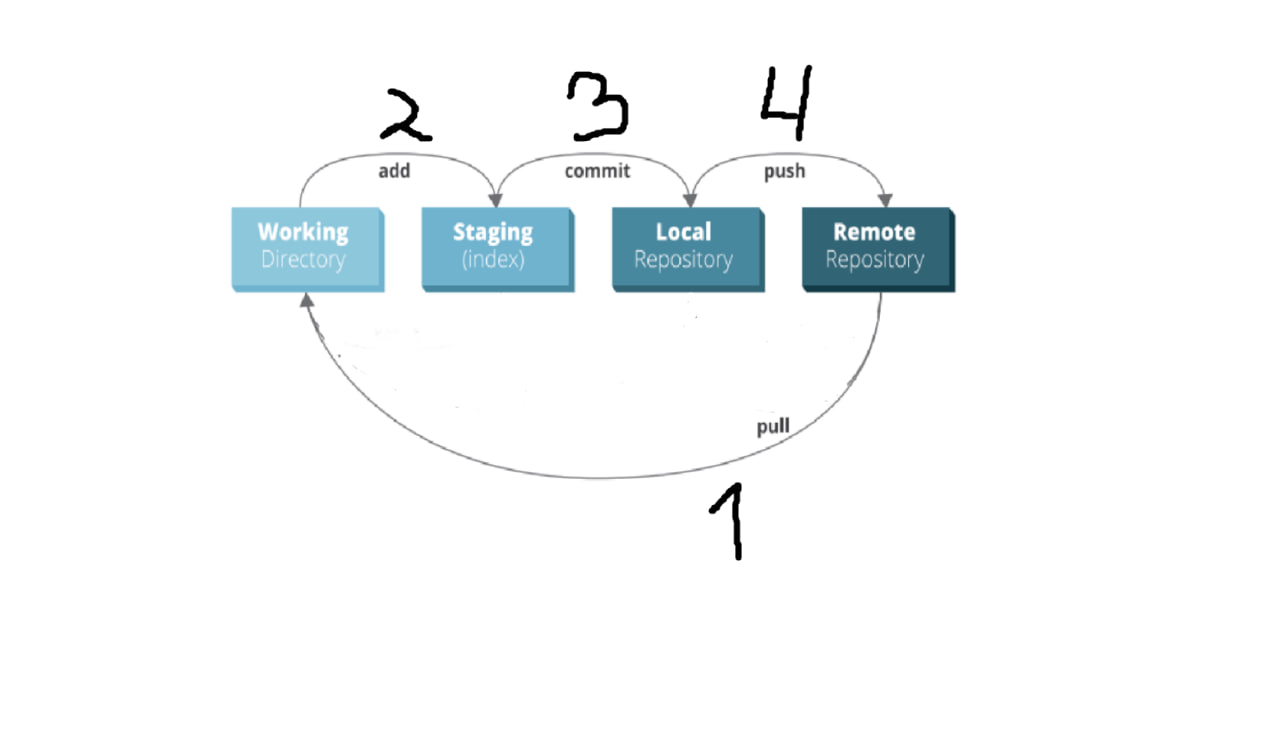
Выбираем интерпретатор:

**Пишем простейшую программу на Python и запускаем её в режиме дебага с помощью средств VSCode:**



**Лабораторная 2.** Потоковая диаграмма GitHub и Архитектура среды.

Ниже представлена общая потоковая диаграмма при работе над проектом как в локальном, так и в удаленном репозитории.

1. 

Потоковая диаграмма GitHub



Архитектура среды

1. Использование веток способствует коллективной разработке, предотвращает внезапные ошибки в коде и делает процесс разработки более прозрачным.



1. Основные принципы, лежащие в основе архитектуры GitHub, включают:

* **Репозитории (Repositories):** GitHub основан на концепции репозиториев, которые являются центральным элементом для хранения и управления исходным кодом и другими файлами проекта.
* **Ветвление (Branching):** GitHub поощряет использование ветвления в разработке, что позволяет разработчикам работать над отдельными фичами или исправлениями ошибок параллельно без влияния на основную ветку кода.
* **Слияние (Merging):** Слияние веток представляет собой важный аспект архитектуры GitHub, который позволяет объединять изменения из различных веток, обеспечивая интеграцию нового кода в основную разработку.
* **Pull-запросы (Pull Requests):** GitHub использует механизм pull-запросов, который позволяет разработчикам предложить изменения из своей ветки кода в основную ветку проекта для обсуждения и слияния.
* **Отслеживание задач (Issue Tracking):** GitHub предоставляет функциональность для отслеживания задач и обсуждения проблем в проекте с помощью системы issue tracking, что способствует управлению проектом и коммуникации в команде.
* **Коллаборация и обсуждение (Collaboration and Discussions):** GitHub облегчает сотрудничество и обсуждение проектов с помощью комментариев к коду, обсуждений в issue и pull-запросах, а также возможности работы в команде над проектом.

4.Локальный репозиторий создается на основе существующего с помощью команд:

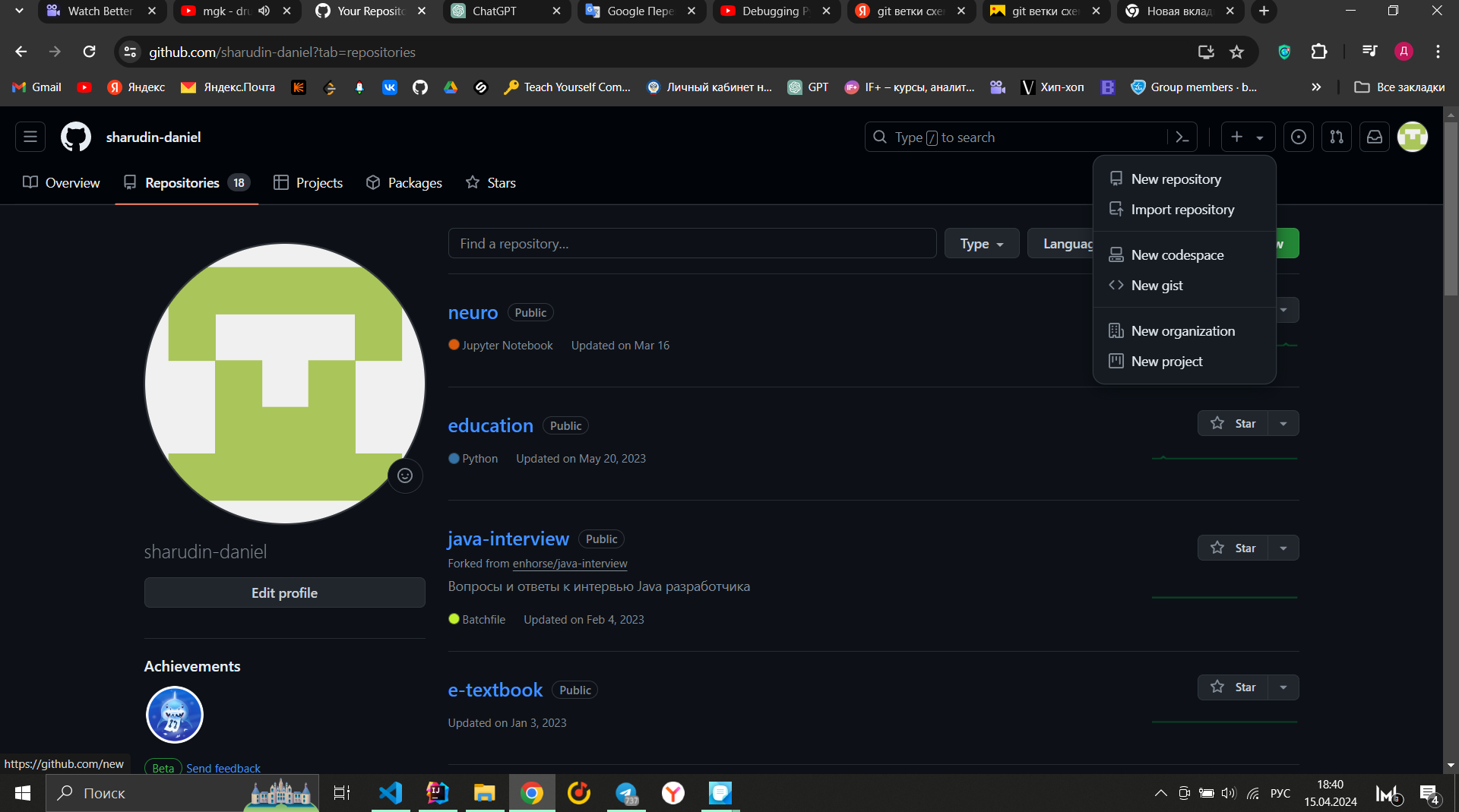
* **Клонирование (Clone):** Эта команда используется для создания локальной копии существующего удаленного репозитория. Она загружает все файлы и историю версий с удаленного репозитория на локальную машину.

git clone <URL\_удаленного\_репозитория>

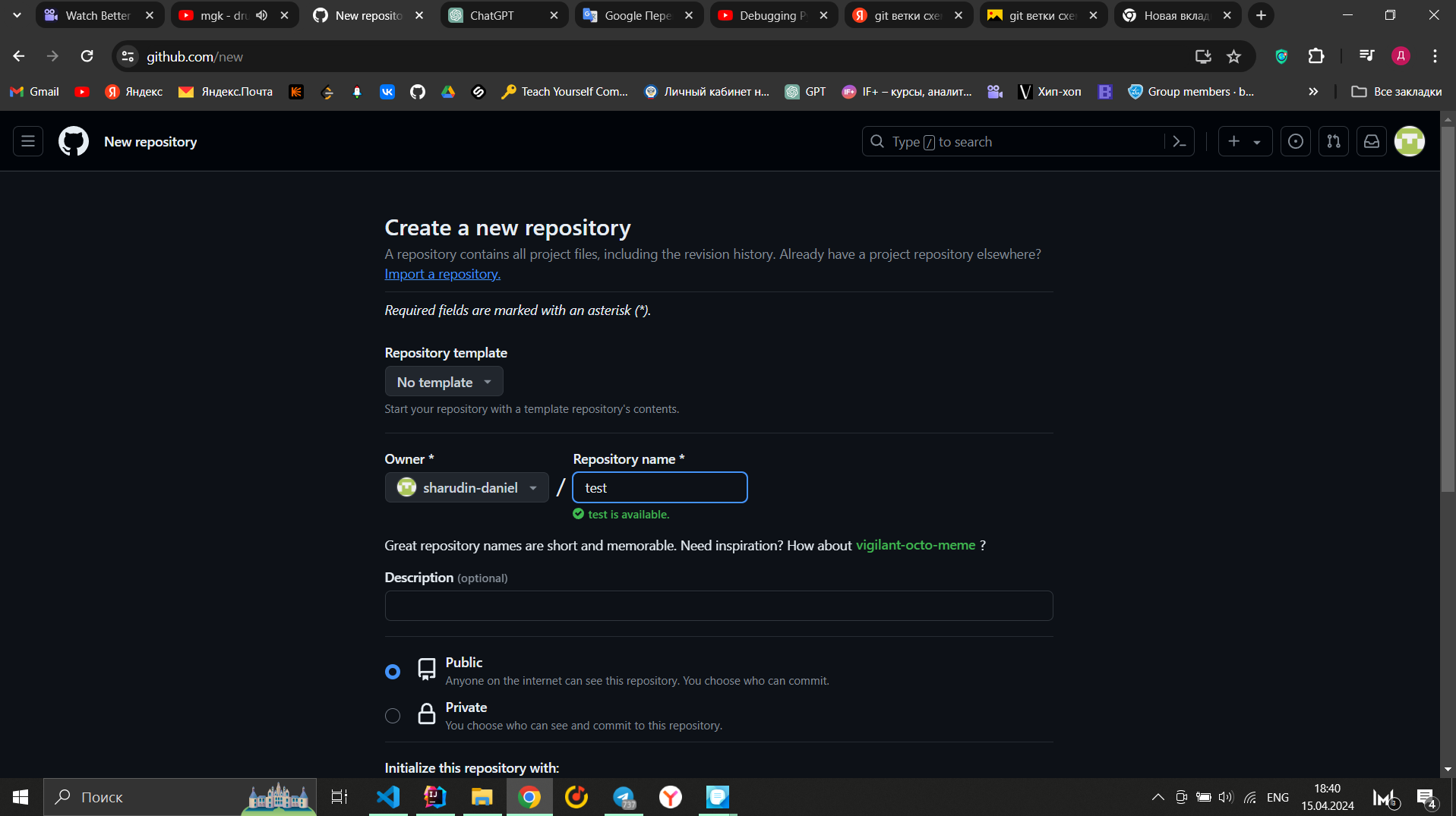
* **Инициализация (Init) и Добавление удаленного репозитория (Remote):** Эти команды используются в случае, если у вас уже есть локальный проект, и вы хотите привязать его к существующему удаленному репозиторию.

git init

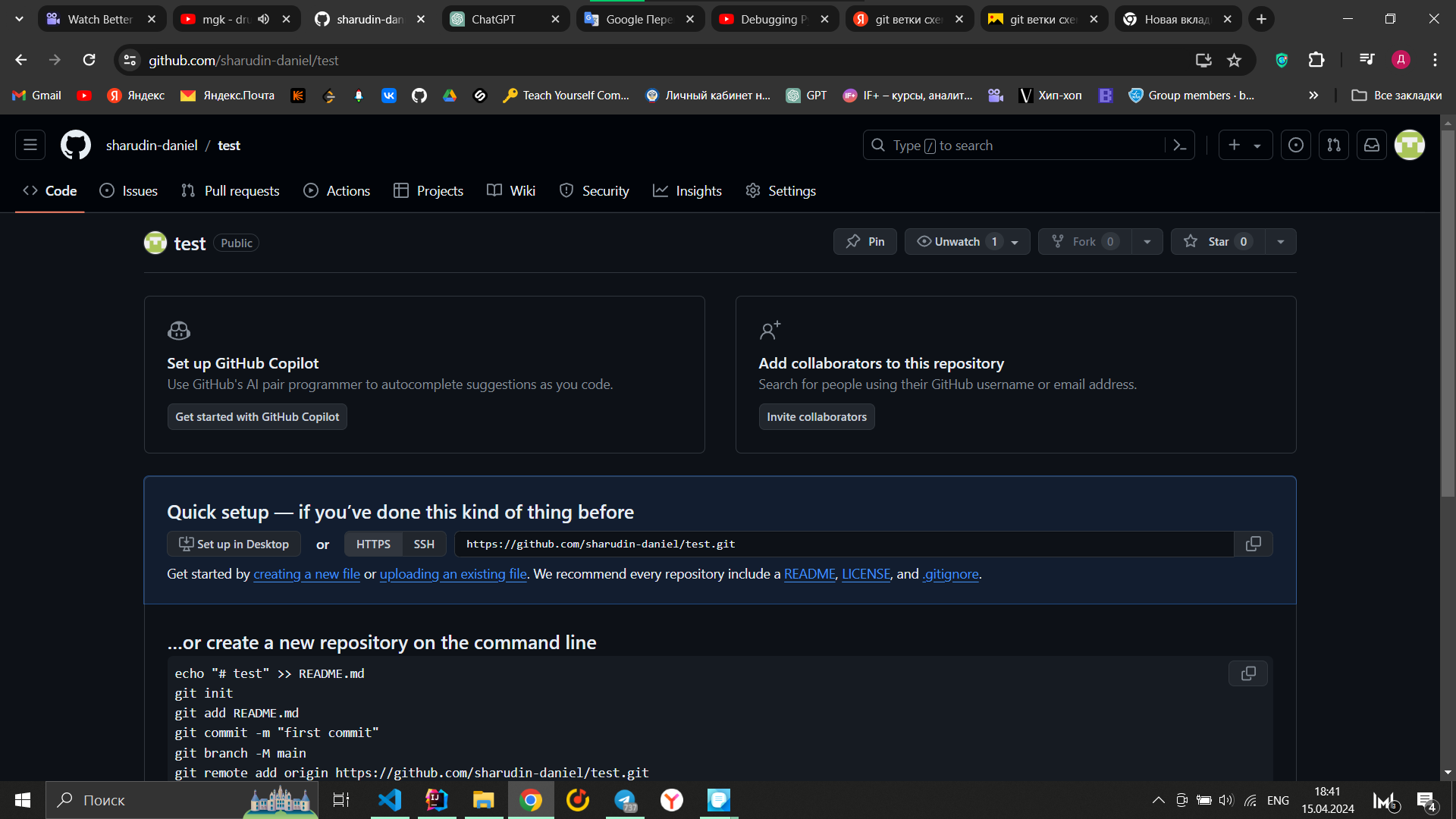
5. На главной странице нажать на кнопку создания репозитория.

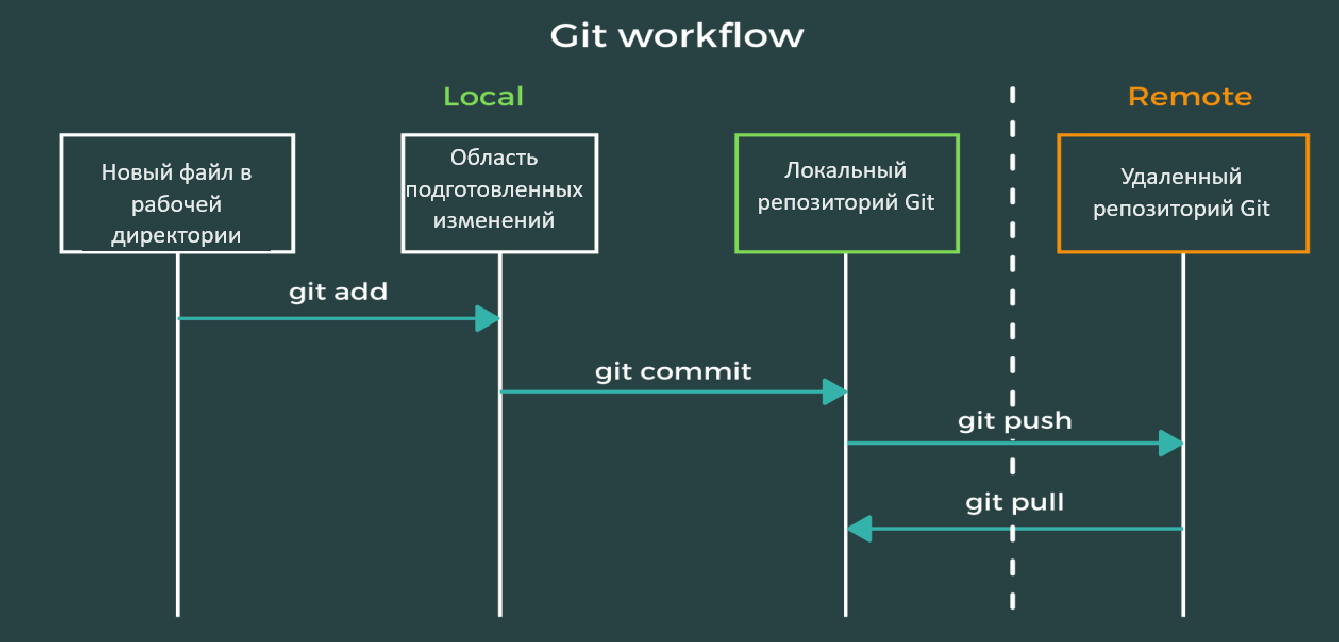


Задать нужные параметры



Репозиторий создан



**Задание 1**

**Задание 2**

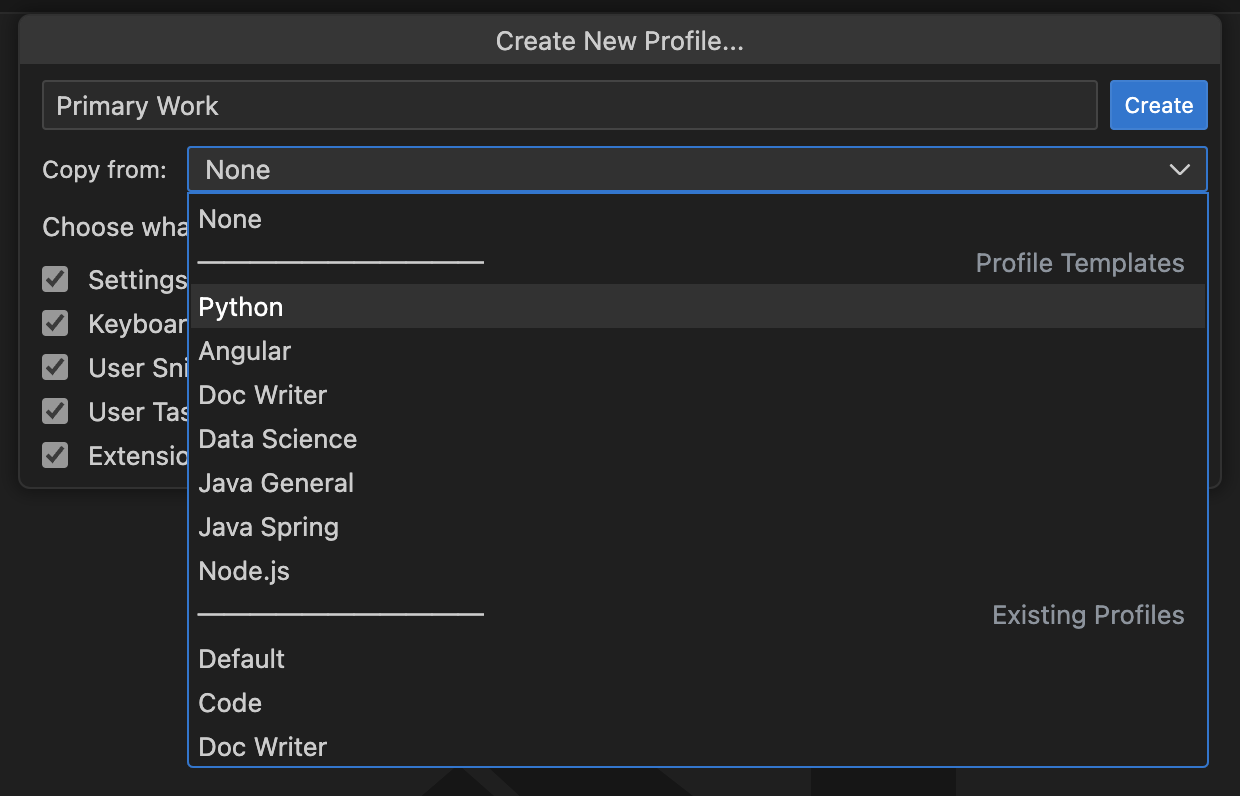
Ниже будет показан пример прохождения вышеуказанных этапов. Сначала был склонирован удаленный репозиторий. Создали текущий документ и внесли в него изменения.

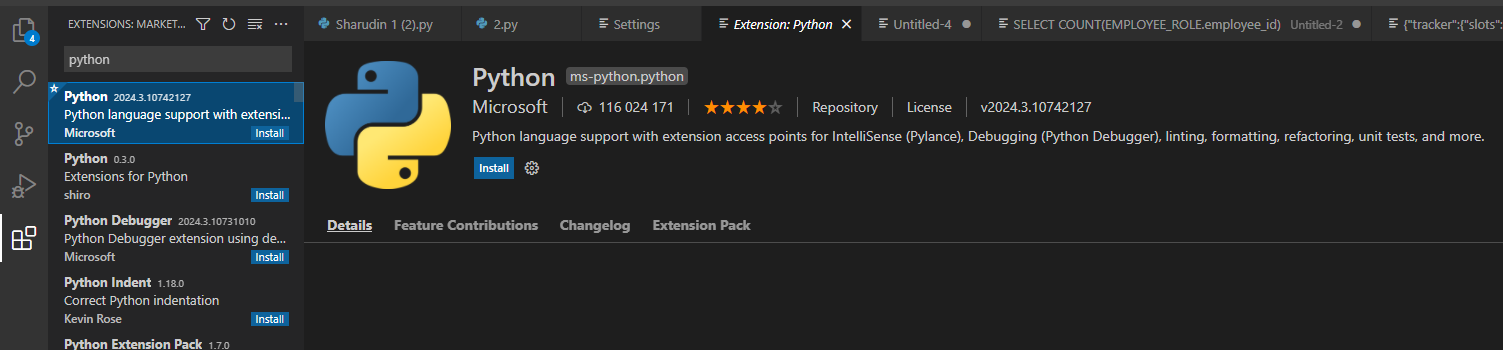
**Лабораторная 3.** Разработка потоковых диаграмм их реализация в среде VS Code.

**Лабораторная 4.** Создание профайла VS code для работы в среде Python. Виды профайлов. Их использование при распределенной разработки. Структура профайла. GitHub copilot. UML (Unified Model Language) - диаграмма взаимодействий.

Профиль отвечает за практически все настройки редактора - регулирует набор расширений, горячих клавиш, сниппетов, задач по отладке и непосредственно настроек не только редактора, но и его расширений. Профиль можно применить на несколько проектов, и только от вас зависит набор профилей и настроек в них.

Создание профиля через **File** > **Preferences** > **Profiles** > **Create Profile**



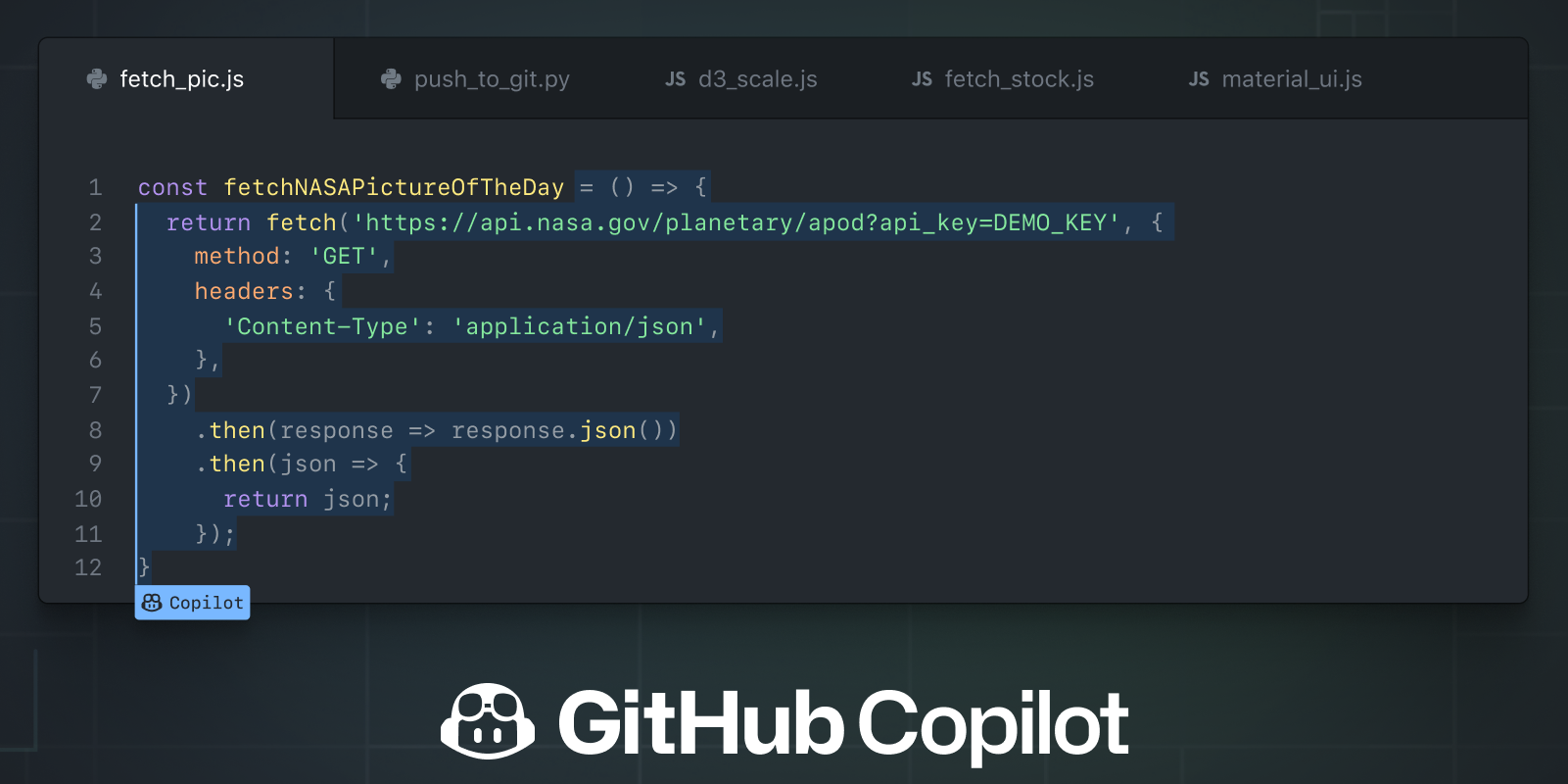


Установка расширения для работы с Python из магазина VSCode

GitHub Copilot — это **инструмент от компании GitHub, который в реальном времени предоставляет разработчикам подсказки по написанию кода и функции автодополнения кода**. Этот инструмент будет полезен как новичкам, которые только погружаются в язык или библиотеку, так и опытным разработчикам, которые могут не тратить время на создания стандартных функций и конструкций.



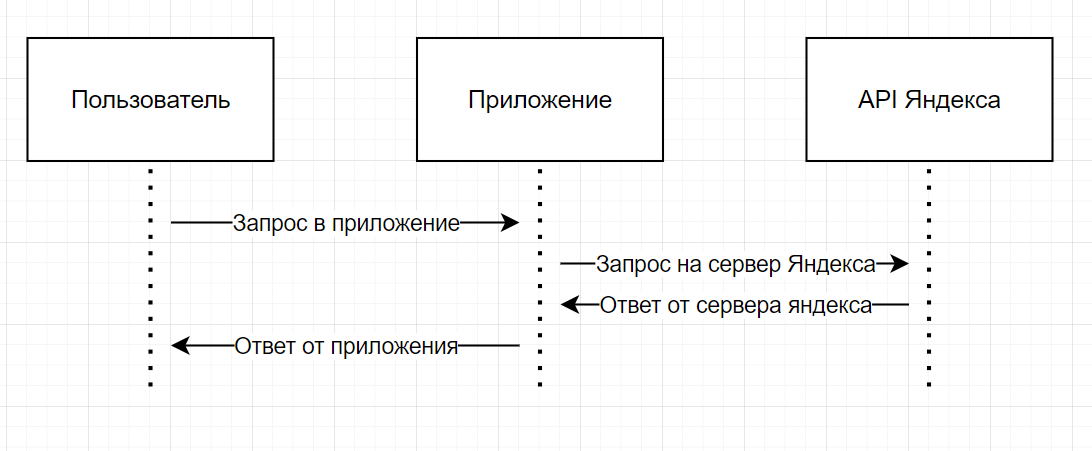
Установка расширения GitHub Copilot из магазина VSCode



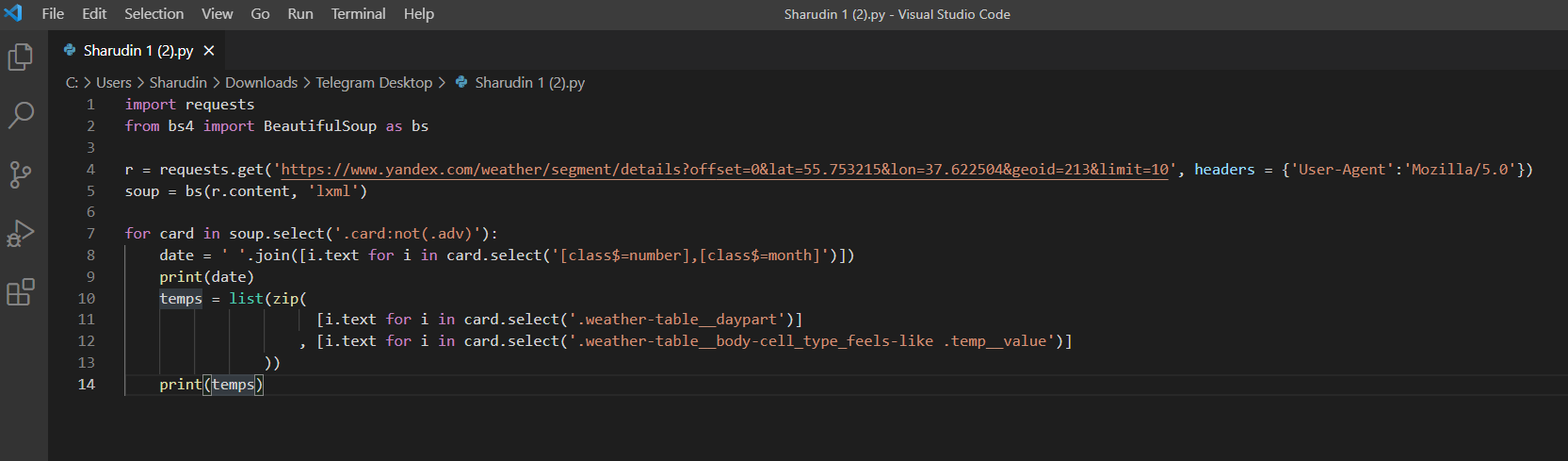
**Пример работы GitHub Copilot**

**UML**, или **Unified Modeling Language**, — это унифицированный язык моделирования. Его используют, чтобы создавать диаграммы и схемы для визуализации процессов и явлений.

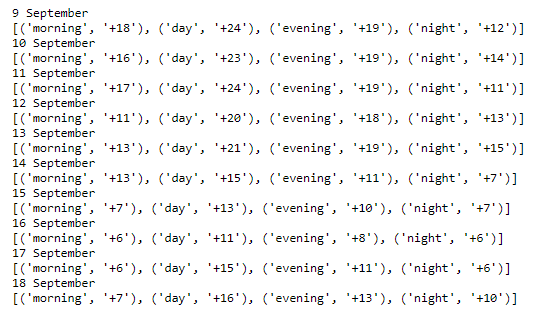
Слово «унифицированный» означает, что схемы на UML будут понятны всем, кто знаком с ним. То есть у языка есть определённые правила, по которым применяются все стрелочки, кружки и квадраты.



UML диаграмма программы



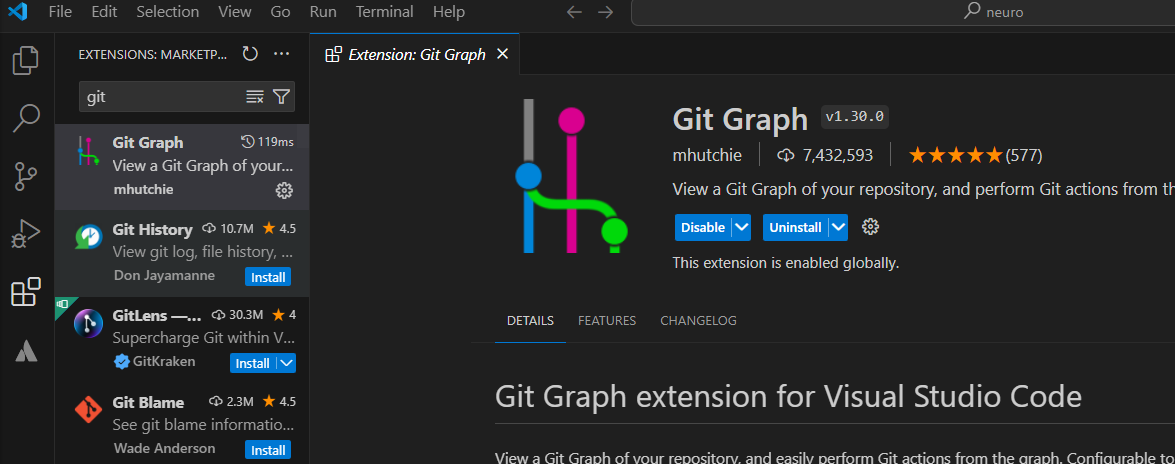
Исходная программа



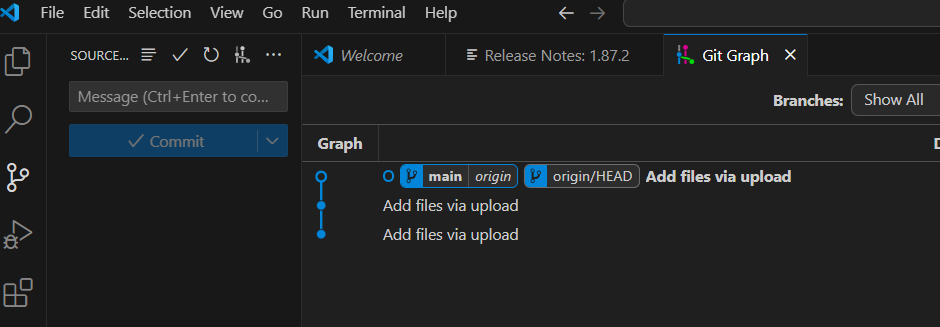
Результат работы

**Лабораторная 5**. Создание ветвей в GitHub для распределенной разработки и их слияние из среды VS Code.

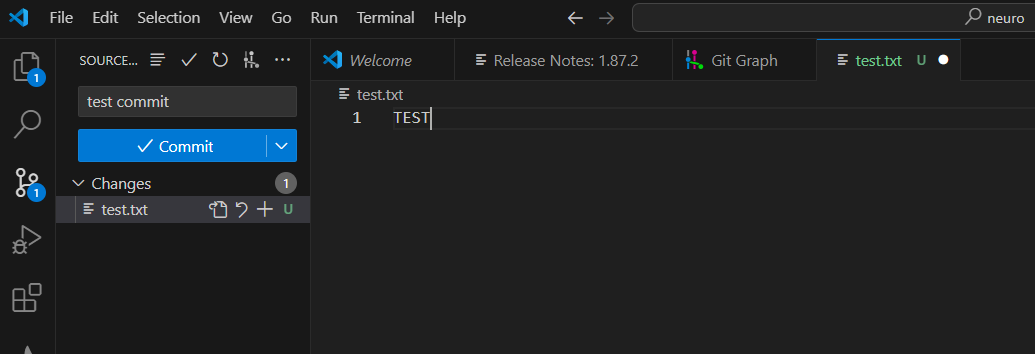
Установим расширение Git Graph для комфортной работы с ветками Git:



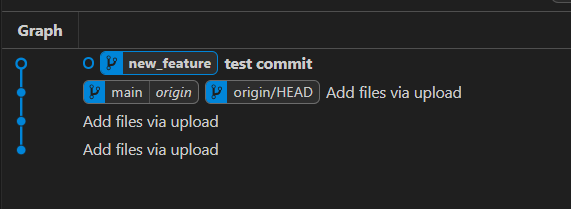
Изначально наш граф выглядит следующим образом:



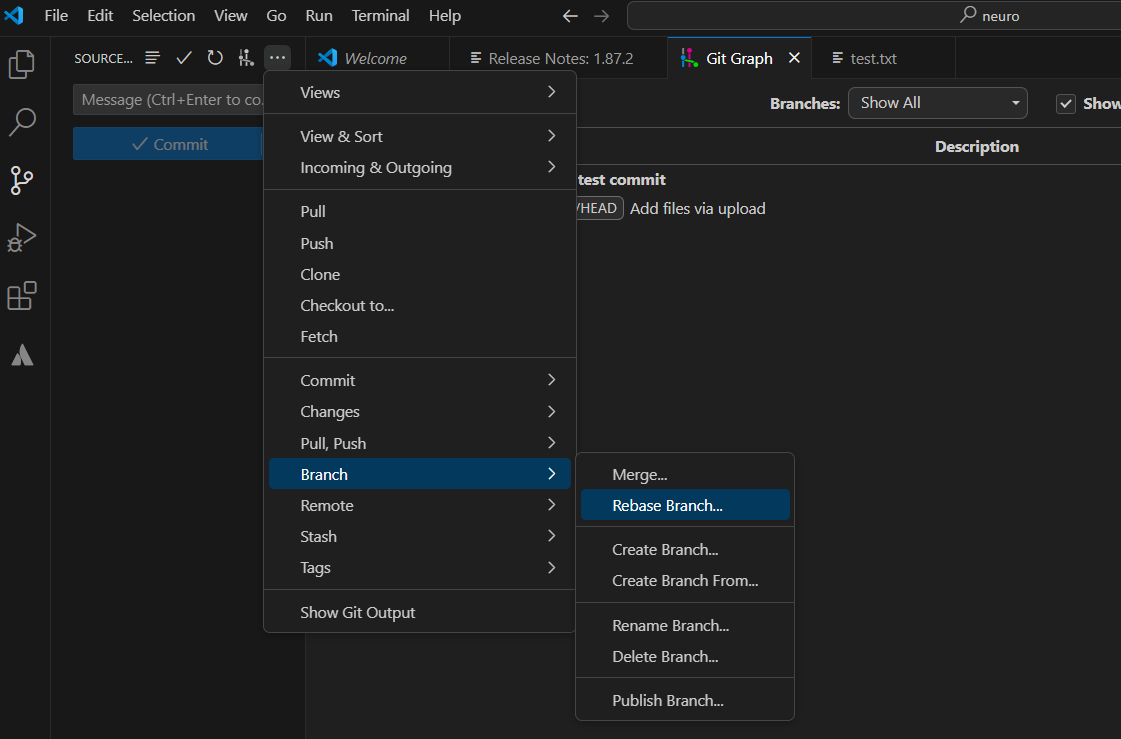
Создадим новую ветку, затем создадим новый файл и закомитим его:



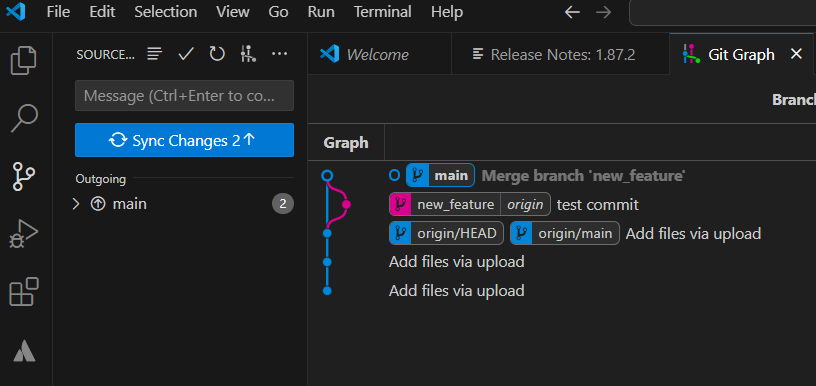
Наша новая ветка выглядит так:



Сделаем рибейз новой ветки на master (на случай, если ктото успел внести свои изменения):



Вольем новую ветку в ветку master:



В результате, наши новые изменения, сделанные в новой ветке, попали в ветку master.