Цифриум

Технологическая карта

Умный код на Python. Базовый уровень

Модуль 4 | Тема 1 | Урок 5

Знакомство и работа с системой контроля версий

Уважаемый коллега, обратите внимание!

Учёт присутствия ребёнка на уроке ведётся только по цифровому следу. Каждый ученик из группы на уроке должен передать цифровой след на нашей платформе.

Вы отвечаете за передачу следа учеником на нашей платформе во время урока.

Передавать цифровой след нужно в личном кабинете ученика.

Цифровой след засчитывается по итогу выполнения учеником трёх задач на платформе в ходе урока.

Обращаем внимание — в задачах с отправкой ответа в виде файла наша платформа поддерживает следующие форматы: .pdf, .doc, .png, .jpeg, .xlsx, .mp3, .py, .ipynb, .txt, .csv, .json, .xml, .sb3, .ino, .hex.

Контролируйте время урока, чтобы не пропустить момент, когда ученики должны залогиниться в личном кабинете и перейти к решению задач на платформе.

Проследите, чтобы каждый ученик залогинился в личном кабинете и выполнил 3 задачи на платформе. В противном случае, Вам не будет засчитано, что вы провели урок, занятие не будет оплачено, а ученику не будет выставлена отметка о присутствии на уроке.

Просим вас в ходе 45 минут урока проконтролировать, чтобы:

1. Каждый из учеников залогинился в личном кабинете.
2. Каждый ученик в разделе **«Решить задания»** отправил решения по 3 задачам — «Практическое задание 1», «Практическое задание 2» и «Практическое задание 3».

О занятии

Краткое содержание занятия:

1. Узнаете, что такое Git — система управления версиями кода.
2. Установите Git и настроите рабочую среду.
3. Создадите свой первый локальный репозиторий.
4. Научитесь выполнять первый коммит и сохранять изменения.
5. Узнаете, как просматривать историю изменений.
6. Освоите базовые команды: init, add, commit, status, log.

Цель:

Познакомить учеников с Git как системой управления версиями.

Задачи:

1. Установить Git и настроить рабочую среду.
2. Создать локальный репозиторий и выполнить первый коммит.
3. Изучить просмотр истории изменений и откат к предыдущим версиям.
4. Освоить базовый рабочий цикл: add, commit, status, log, clone, push, pull.

Термины

1. **Git**
   Система контроля версий для отслеживания изменений в файлах и координации работы над проектом.
2. **Репозиторий (repository)**
   Хранилище проекта с полной историей всех изменений.
3. **Коммит (commit)**
   Снимок состояния проекта в определённый момент времени с описанием изменений.
4. **Индекс / Staging area**
   Область подготовки файлов перед сохранением в коммит.

О занятии

5. **История (history)**
   Последовательность всех коммитов в хронологическом порядке.

6. **git init**
   Команда для создания нового локального репозитория.

7. **git add**
   Команда для добавления файлов в индекс (подготовка к коммиту).

8. **git commit**
   Команда для сохранения подготовленных изменений в репозиторий.

9. **git status**
   Команда для проверки текущего состояния репозитория.

10. **git log**
    Команда для просмотра истории коммитов.

Необходимые материалы:

1. Доступ к платформе для выполнения практических заданий

Темы и время

ЭТАП	ВРЕМЯ
Приветствие	5 мин.
Теория	25 мин.
Итоги занятия	5 мин.
Получение фидбека. Рефлексия. Проверка знаний. Домашнее задание	5 мин.
Вопрос-ответ	5 мин.
Итого	45 мин.

Ход занятия

Номер слайда	Пояснение к слайду
	Приветствие: Здравствуйте! Начинаем новое занятие, посвящённое системе контроля версий Git. Сегодня вы познакомитесь с одним из важнейших инструментов разработчика, который используют миллионы программистов по всему миру. Дополнительно: Git — это не просто программа, это стандарт индустрии. Знание Git обязательно для работы в команде, участия в открытых проектах и профессиональной разработки. Даже если вы работаете в одиночку, Git поможет не потерять код и легко отменять ошибки.
	Титульный слайд с темой урока: «Знакомство и работа с системой контроля версий».
1.	Краткое повторение: • Научились отправлять корректные JSON-запросы к GPT API через OpenRouter • Поняли структуру запроса: model, messages, temperature, max_tokens • Узнали о токенах: как они считаются и почему влияют на стоимость • Практиковались в подборе параметров под разные задачи • Дополнительно: Вы освоили работу с внешними API. Сегодня мы переходим к управлению собственным кодом. Эти навыки дополняют друг друга: вы будете писать код, который взаимодействует с ИИ, и хранить его в надёжном месте.
2.	Сегодня на уроке вы: • Узнаете, что такое Git и зачем он нужен разработчикам • Установите Git и настроите рабочую среду на своём компьютере • Создадите свой первый локальный репозиторий • Освоите базовые команды: init, add, commit, status, log • Выполните практическое задание по шагам • Дополнительно: В конце урока у вас будет работающий репозиторий с историей изменений.

Ход занятия

	Вы сможете применять эти знания в любых проектах: от школьных заданий до личных приложений.
3.	Цель: Познакомить учеников с Git как системой управления версиями и научить выполнять базовые операции. Задачи: Установить Git и проверить работоспособность в терминале Создать локальный репозиторий для учебного проекта Выполнить первый коммит: добавить файл, зафиксировать изменения Освоить команды для отслеживания состояния и просмотра истории Понять принципы именования коммитов и организации работы Дополнительно: После урока вы сможете самостоятельно начинать новые проекты с инициализации Git, фиксировать прогресс и возвращаться к предыдущим версиям при необходимости.
4.	Что такое Git?Git — это распределённая система контроля версий, разработанная Линусом Торвальдсом в 2005 году для управления исходным кодом ядра Linux. Как работает Git: Сохраняет все изменения в проекте как последовательность снимков (коммитов) Позволяет вернуться к любой предыдущей версии за секунды Показывает, кто, когда и какие изменения внёс Позволяет нескольким людям работать над одним проектом без конфликтов Аналогия: «Машина времени для кода» Каждый коммит — точка сохранения, как в игре Вы можете «загрузиться» с любого момента Видите полную историю: что менялось и почему Дополнительно: В отличие от облачных хранилищ (Google Drive, Dropbox), Git хранит не просто файлы, а историю их изменений с возможностью ветвления, слияния и детального сравнения версий.
5.	Репозиторий (repository, repo) — папка проекта, в которой Git отслеживает изменения. Содержит файлы проекта и скрытую папку .git с историей и настройками.

Ход занятия

	Коммит (commit) — снимок состояния всех файлов в определённый момент. Каждый коммит имеет: Уникальный хеш (ID), например: a1b2c3d Автора (имя и email) Дату и время Сообщение с описанием изменений Индекс (staging area) — промежуточная зона, куда вы добавляете файлы перед коммитом. Позволяет фиксировать изменения выборочно, а не все сразу. История (history) — линейная последовательность коммитов в хронологическом порядке. Позволяет проследить эволюцию проекта. Дополнительно: Индекс — уникальная особенность Git. Он даёт гибкость: можно подготовить к коммиту только часть изменённых файлов, оставив остальные на потом.
6.	Базовые команды Git git init — инициализирует новый репозиторий в текущей папке. Создаёт скрытую папку .git. git status — показывает состояние рабочей папки: какие файлы изменены, какие добавлены, какие готовы к коммиту. git add <файл> — добавляет файл в индекс (подготовка к коммиту). git add . — добавляет все изменённые файлы в текущей папке. git commit -m "сообщение" — фиксирует изменения из индекса в истории репозитория. Сообщение должно кратко описывать суть изменений. git log — выводит историю коммитов: хеш, автор, дата, сообщение. git log --oneline — компактный формат: одна строка на коммит. Дополнительно: Все команды выполняются в терминале (командной строке). Важно находиться в папке репозитория при их выполнении. Используйте стрелки вверх/вниз для повторения предыдущих команд.
7.	Практика: создаём первый репозиторий. Проведите практическое занятие по шагам: Шаг 1: Создайте папку проекта

Ход занятия

	mkdir my-first-git Шаг 2: Перейдите в папку cd my-first-git Шаг 3: Инициализируйте репозиторий git init (Появится сообщение: Initialized empty Git repository) Шаг 4: Создайте файл с кодом или текстом echo "Привет, Git!" > hello.txt (Или создайте файл вручную в текстовом редакторе) Шаг 5: Проверьте статус git status (Файл будет показан как «не отслеживается») Шаг 6: Добавьте файл в индекс git add hello.txt Шаг 7: Сделайте коммит git commit -m "Первый коммит: добавлен hello.txt" Дополнительно: После коммита выполните git status снова — вы увидите сообщение «nothing to commit, working tree clean». Это значит, все изменения зафиксированы.
8.	Просмотр истории. Объясните, что команда git log показывает: ● commit: уникальный хеш (идентификатор версии) ● Author: кто сделал изменения ● Date: когда именно ● Сообщение: описание изменений Удобные варианты: ● git log --oneline — компактно: a1b2c3d Первый коммит ● git log -3 — показать только последние 3 коммита ● git log --stat — показать, какие файлы менялись Дополнительно: Хеш коммита можно использовать для возврата к версии: git checkout a1b2c3d. Это основа для отмены ошибок и экспериментов.
9.	Вопросы для обсуждения. Предложите ученикам обсудить: Вопрос 1: Что делает команда git init? Ответ: Создаёт новый репозиторий в текущей папке, инициализирует структуру .git для отслеживания изменений. Вопрос 2: Зачем нужно git add перед коммитом?

Ход занятия

	Ответ: Чтобы выборочно подготовить файлы к фиксации. Позволяет коммитить изменения по логическим группам, а не всё подряд. Вопрос 3: Что такое коммит и почему важно писать понятные сообщения? Ответ: Коммит — точка сохранения. Понятное сообщение помогает вам и команде быстро понять, что менялось, без просмотра кода. Вопрос 4: Как посмотреть историю изменений? Ответ: Команда git log. Для удобства используйте --oneline или ограничьте количество коммитов флагом -n. Вопрос 5: Что будет, если сделать изменения в файле, но не сделать git add? Ответ: Изменения останутся в рабочей папке, но не попадут в коммит. При следующем коммите они не сохранятся в истории. Дополнительно: Обсудите типичные ошибки новичков: коммит с сообщением "fix", забытый git add, коммит конфиденциальных данных. Как их избежать?

Практика

Вопросы для самопроверки

Что делает команда git init?

Зачем нужно git add?

Что такое commit?

Как посмотреть историю изменений?

Задание 1

Откройте терминал и создайте папку с проектом (если не готова):

```
mkdir my-first-git
cd my-first-git
```

Инициализируйте репозиторий:

```
git init
```

Создайте файл и добавьте текст:

```
echo "Привет, Git!" > hello.txt
```

Добавьте файл в индекс и сохрани изменения:

```
git add hello.txt
git commit -m "Первый коммит"
```

Посмотрите историю коммитов:

```
git log
```

Дополнительный вопрос

Почему важно писать понятные сообщения к коммитам? Как это помогает в команде?

Примеры выполнения заданий

Вопросы для самопроверки

git init — создаёт новый репозиторий Git в текущей папке (инициализирует систему контроля версий).

git add — добавляет изменения из рабочей директории в «индекс» (staging area), чтобы включить их в следующий коммит.

Commit — это снимок состояния проекта на определённый момент времени, снабжённый сообщением и уникальным хешем; фиксирует изменения навсегда в истории.

Историю изменений можно посмотреть командой git log (показывает список коммитов с авторами, датами и сообщениями).

Задание 1.

Создастся папка my-first-git и вы перейдёте в неё.

Инициализируется локальный Git-репозиторий — появится скрытая папка .git.

Создастся файл hello.txt с текстом Привет, Git!.

Файл будет добавлен в индекс (git add) и зафиксирован в истории (git commit), создан первый коммит с сообщением "Первый коммит".

Команда git log покажет историю — один коммит с хешем, автором, датой и сообщением.

В результате вы получите рабочий Git-репозиторий с одной версией файла.

Дополнительный вопрос

Объясняют «почему», а не только «что» — это помогает понять причину изменений без изучения кода.

Упрощают поиск багов — по истории легко отследить, когда и зачем была внесена проблемная правка.

Облегчают ревью кода — ревьюер сразу видит цель коммита.

Помогают новым участникам быстрее разобраться в истории проекта.

Делают откат изменений безопаснее — вы точно знаете, что потеряете или вернёте.

Сегодня на занятии мы:

- Узнаете, что такое Git — система управления версиями кода.
- Установите Git и настроите рабочую среду.
- Создадите свой первый локальный репозиторий.
- Освоите базовые команды Git: init, add, commit, status, logI.

На следующем занятии вы:

- Научитесь управлять качеством и стилем вывода через параметры генерации.
- Разберетесь, как temperature и top_p влияют на креативность и повторяемость.
- Поймете природу галлюцинаций: почему модель «выдумывает» информацию.
- Изучите связь между длиной запроса и стоимостью/временем отклика.
- Научитесь выполнять практические замеры и сравнивать разные настройки.

Получение фидбека. Рефлексия. Проверка знаний

Попросите ребят решить интерактивные задачи по теме урока.

Ответы к интерактивным задачам

Практическое задание 1

Какая команда Git используется для создания нового репозитория?

- **git init**
- git start
- git create
- git new

Практическое задание 2

Выберите команды Git, которые входят в базовый рабочий цикл:

- **git add**
- **git commit**
- **git status**
- git install
- git remove

Практическое задание 3

Как называется снимок изменений, который мы сохраняем в Git? (одно слово на английском)
commit

Вопрос-ответ

Если во время подготовки или проведения урока возникают вопросы, их необходимо адресовать в методический чат.