



# Введение

---

ЛЕКЦИЯ 0



# Содержание

---

- Что такое Python
- Основные компоненты
- Версии Python
- Способы запуска
- Использование PyCharm в качестве среды разработки
- Система распределенного контроля версий Git



# Язык Python

---



Python – это высокоуровневый язык программирования общего назначения.

Данный язык был создан Гвидо ван Россумом в 1991 году. Название языка происходит от творческого коллектива Monty Python из Великобритании. Однако в настоящее время Python чаще ассоциируется со змеей чем с комик-группой.

В настоящее время данный язык используется и продвигается такими компаниями как Google, NASA, Los Alamos, FermiLab и многие другие.



# Основные компоненты

---

В процессе установки Python на компьютер создается ряд программных компонентов.

В их число входят (как минимум):

- Интерпретатор
- Библиотека поддержки

**Интерпретатор** представляет собой программу анализирующую и тут же выполняющую код построчно.

В случае Python используется интерпретатор **командного типа**, т.е. это система одновременно включающая как интерпретатор (описанный выше) так и компилятор – переводящий исходный код программы в промежуточное представление (байт-код) исполняемый в дальнейшем виртуальной машиной PVM (Python Virtual Machine).



# Версии Python

---



В настоящее время существуют и развиваются одновременно две версии Python: это 2.\* и 3.\*. Программы написанные на Python в большинстве случаев не будут работать с версией Python другого поколения.

В рамках данного курса используется Python версии 3.\*.

В качестве среды разработки используется **PyCharm**, однако на свое усмотрение можно использовать любую другую среду разработки



# Способы запуска Python

---

## Интерактивный режим работы:

- Данный режим подразумевает что пользователь вводит инструкции непосредственно в командной строке. Данный режим схож с аналогичным режимом из Matlab

## Сценарии:

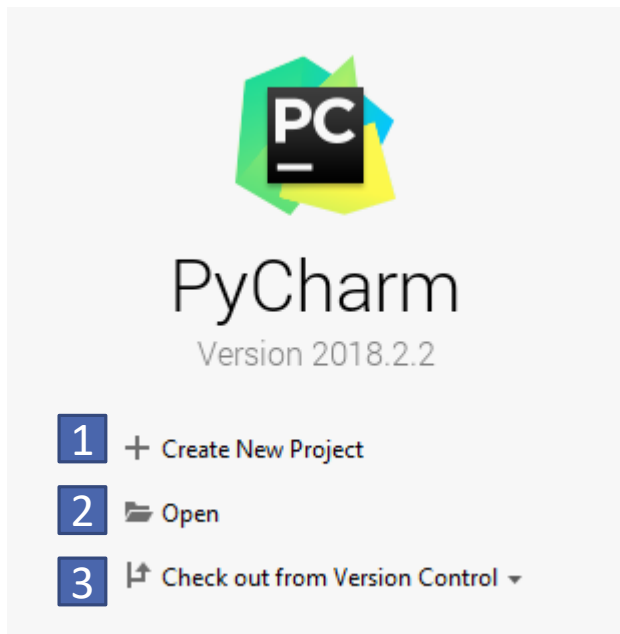
- В данном случае пользователь заранее создает скрипты (имеют расширение \*.py) инструкции из которых транслируются интерпретатором в байт-код , после чего исполняются виртуальной машиной

*Примечание: Стоит отметить что иногда процесс трансляции в байт-код может занимать продолжительное время. По этой причине Python может создавать уже скомпилированные файлы (\*.pyc). В таком случае при запуске программы этап трансляции игнорируется*



# Работа с PyCharm

---

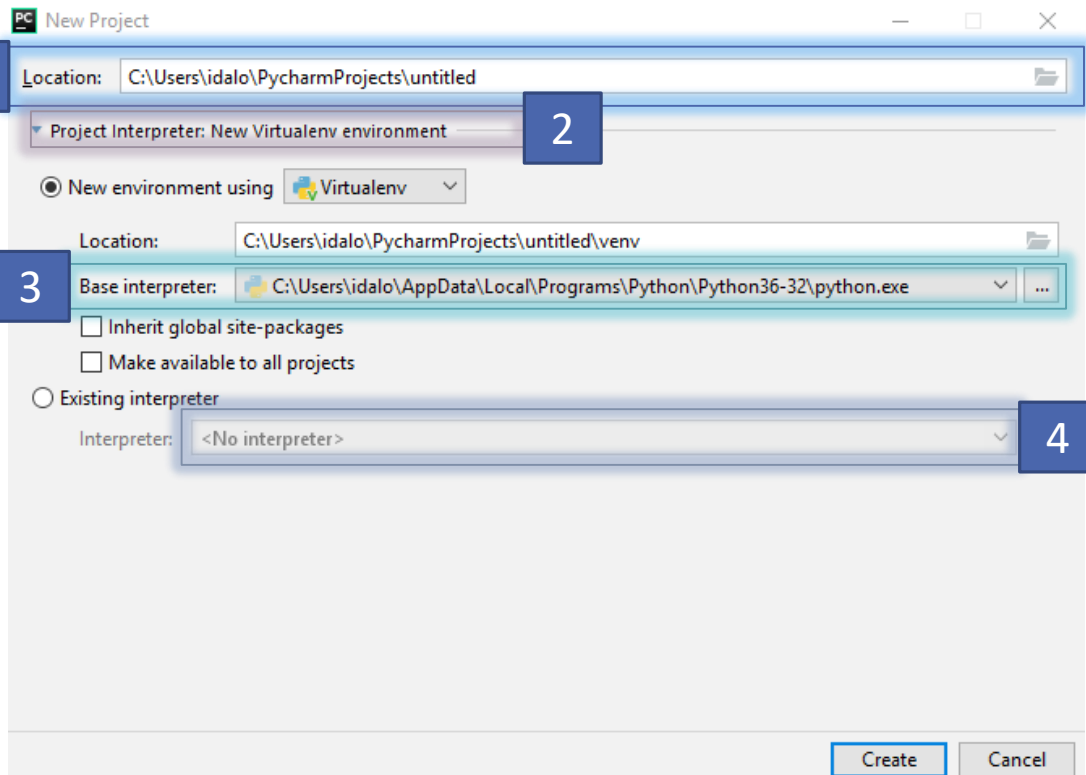


Главное окно **PyCharm**:

1. Создать новый проект
2. Открыть существующий проект или файл
3. Клонировать существующий проект, воспользовавшись системой контроля версий



# PyCharm: создание нового проекта



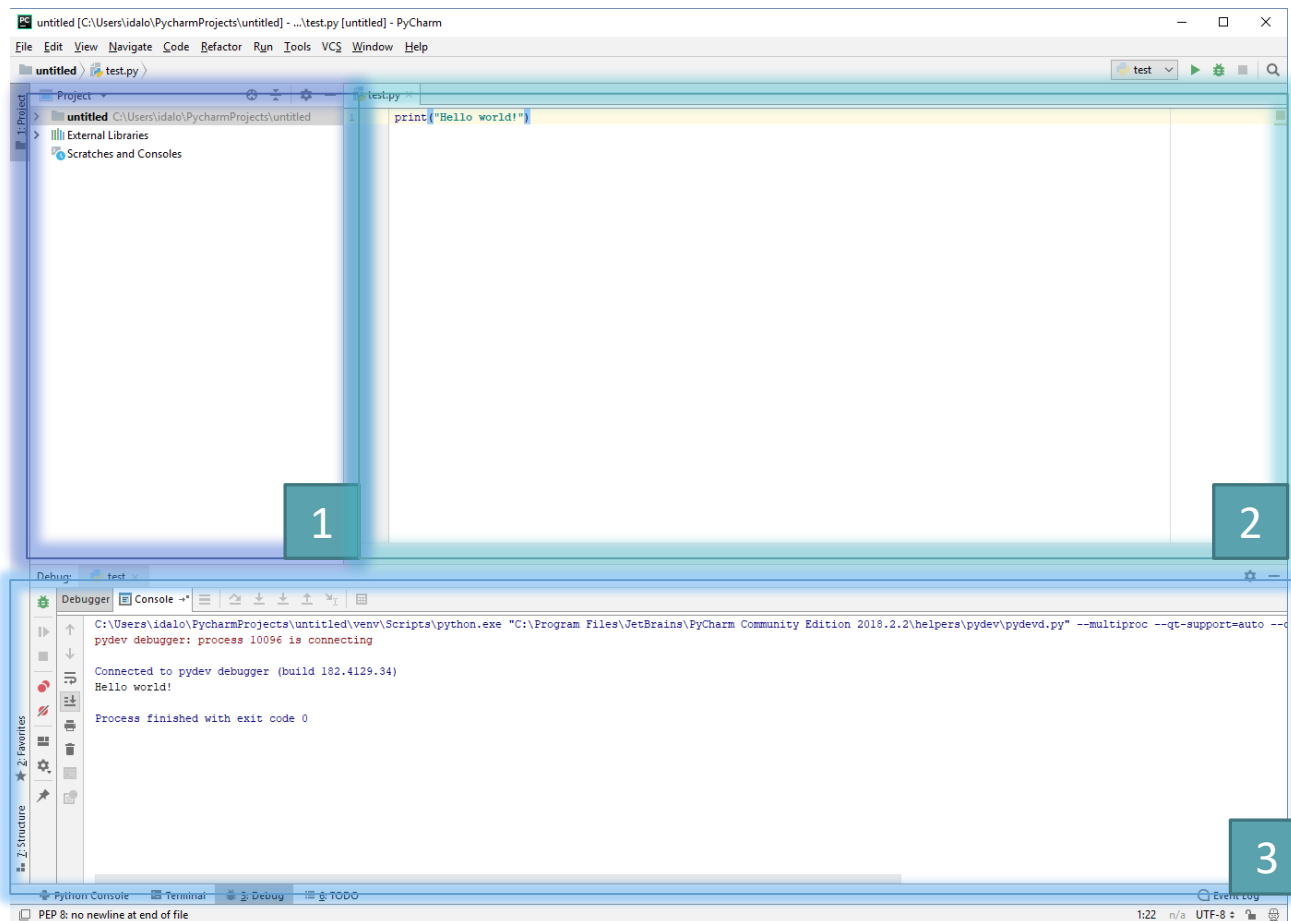
В данном окне задаются настройки нового проекта:

1. Путь к рабочей директории проекта
2. Дополнительные настройки проекта
3. Путь к используемому в качестве основы интерпретатору.
4. Путь к уже существующей настроенной копии интерпретатора

*Примечание: в различных проектах может использован различный набор модулей. В данном случае будет использован набор модулей по умолчанию*



# Рабочее окно PyCharm



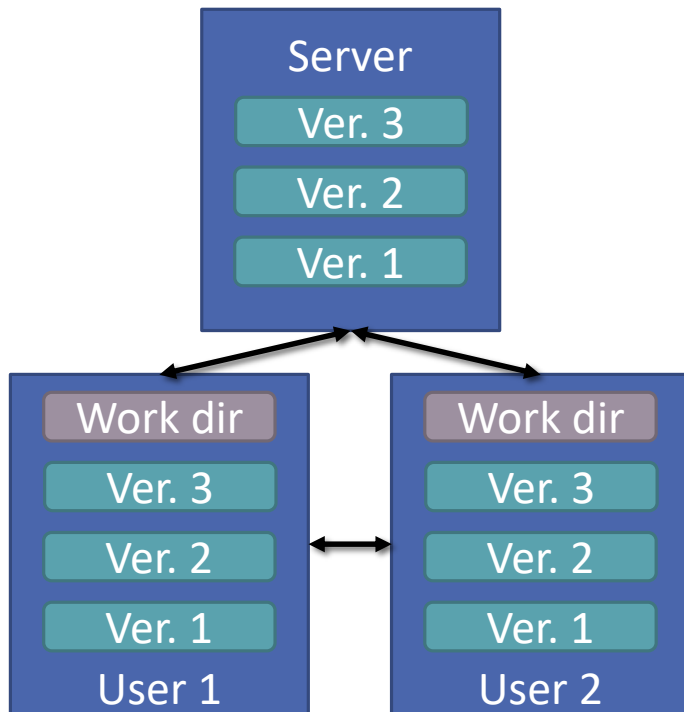
Структура основного окна среды разработки **PyCharm**:

1. Дерево каталогов проекта
2. Главная рабочая область (область в которой можно редактировать активные файлы)
3. Вспомогательные окна (в этот список входит Debug, terminal, рабочая консоль вывода, дерево системы контроля версий и т.п.)



# Система контроля версий Git

**Репозиторий** – это хранилище, где хранятся и поддерживаются какие – либо данные, а также история их изменений

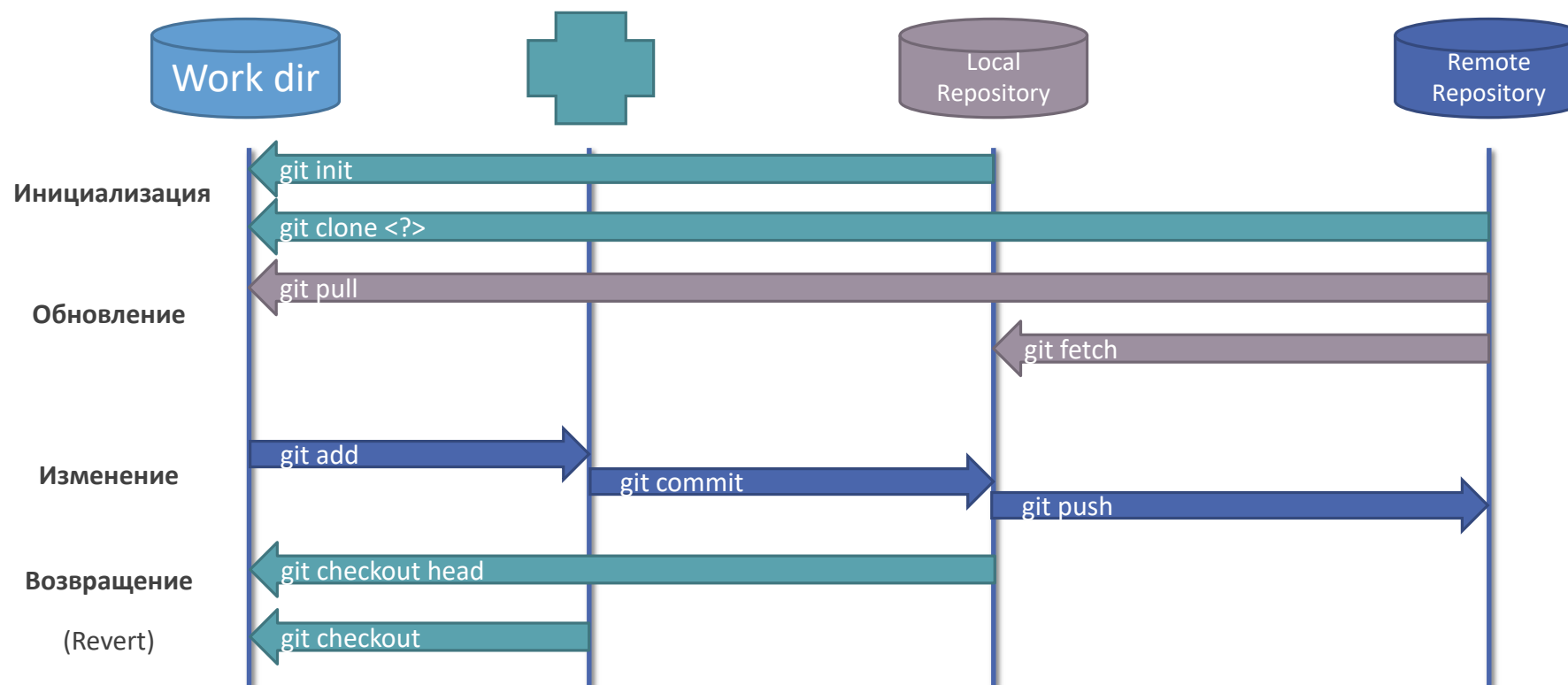


**Репозиторий** – это хранилище, где хранятся и поддерживаются какие – либо данные, а также история их изменений.

**Git-** программное обеспечение для управления версиями. Git относится к классу систем распределённого контроля версий.

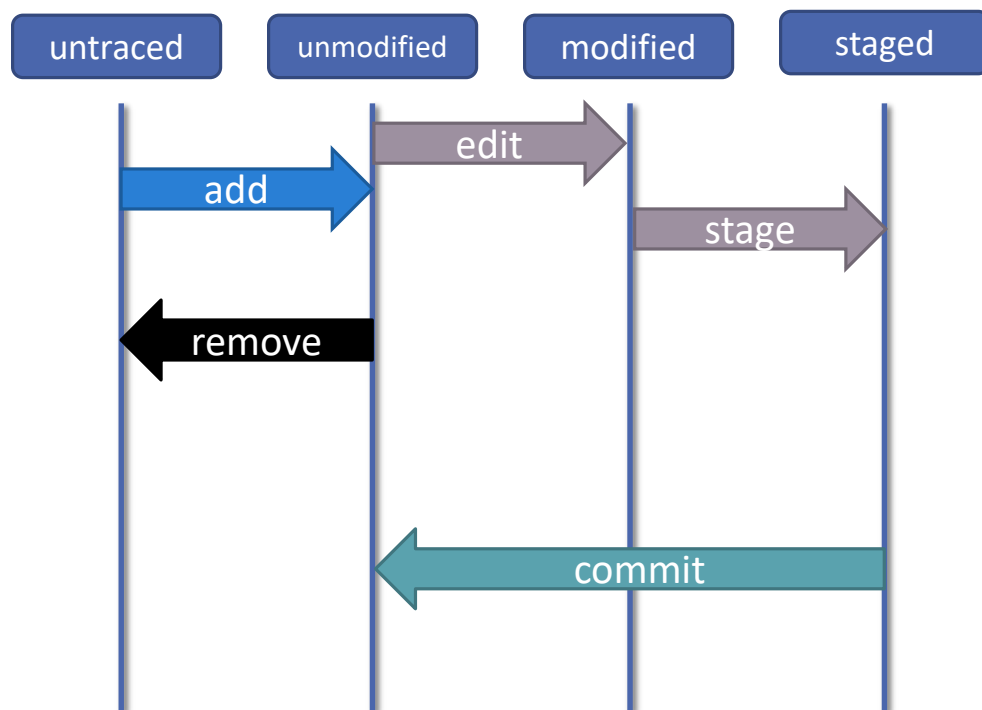
**Распределенный контроль версий** – система, в которой у каждого разработчика храниться полная копия репозитория и всех изменений, и каждый разработчик непосредственно работает со своей копией.

# Жизненный цикл Git





# Жизненный цикл файлов в Git



Все файлы в процессе работы с контролем версий делятся на два типа: *отслеживаемые* и *не отслеживаемые*.

**Отслеживаемые** файлы – это файлы находящиеся под контролем git. При этом они могут быть **не модифицированными**, **модифицированными** или **индексированными**, т.е. готовыми к «сохранению» (а точнее *коммиту*)

**Не отслеживаемые** файлы не хранятся в репозитории, и содержатся только в рабочем каталоге



# Основные операции Git

---

*Примечание: обычно принято работать с git либо через командную строку, однако в рамках данного курса работа с git будет рассмотрена непосредственно в среде фирмы JetBrains (во всех средах разработки данной фирмы работа с git аналогична)*

## **Инициализация:**

- ☐ Инициализация (git init) – перевод текущего проекта под контроль версий Git
- ☐ Клонирование (git clone) – копирование удаленного репозитория с созданием локальной копии и переводом в рабочее состояние

## **Обновление:**

- ☐ Fetch (git fetch) – получает информации обо всех изменениях и *ветках* на удаленном репозитории
- ☐ Pull (git pull) – скачивает изменения в текущий рабочий каталог, одновременно пытаюсь их *соединить*



# Основные операции Git ( продолжение)

---

## Изменение:

- ☐ **Add/Remove** (git add/ git remove) – добавление/исключение в систему контроля версий файла
- ☐ **Commit** (git commit) – создание новой «контрольной точки» в репозитории, добавляя в нее все проиндексированные файлы

## Возвращение:

- ☐ **Checkout** (git checkout) – переход к сохраненной точке, с возможностью отката всех не индексированных изменений, перехода на другую «ветку проекта», и т.д.



# GitHub

---

В качестве веб-сервиса для хранения проектов и учебных материалов в рамках данного курса используется **GitHub**

<https://github.com/dep24>

Стоит отметить следующие курсы:

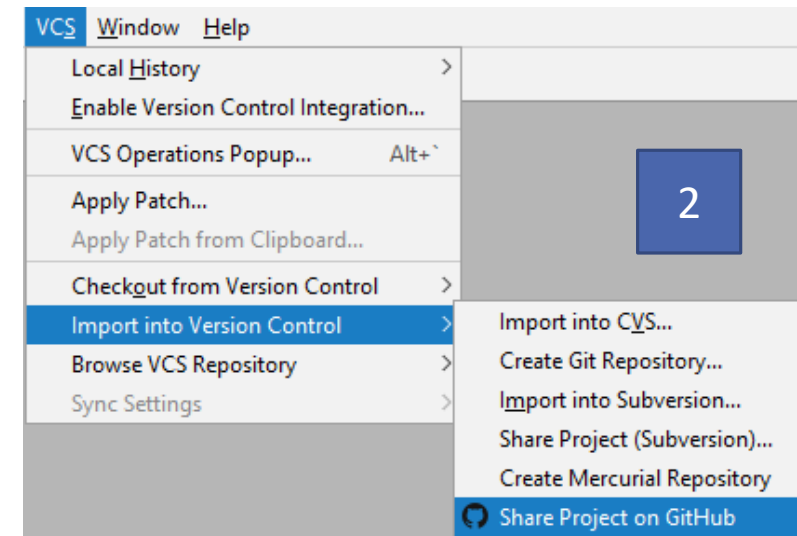
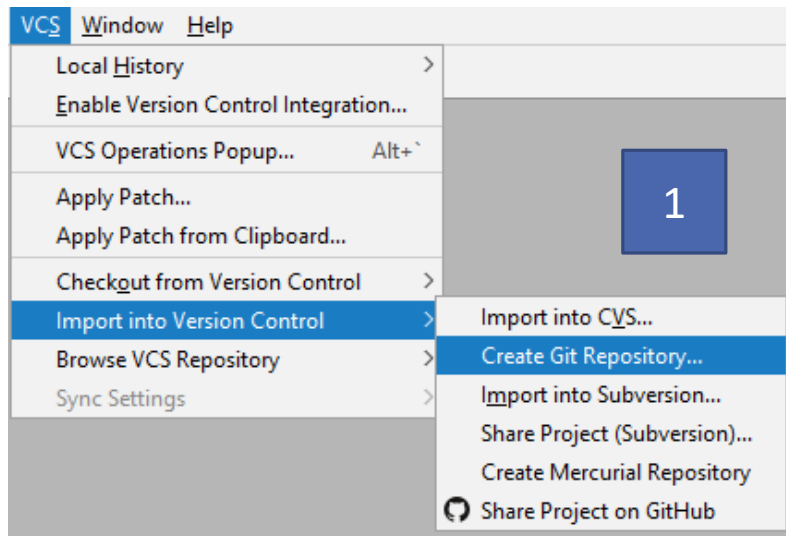
- ☐ **B\_INFO**: базовый курс Си по программе бакалавриата
- ☐ **B\_INFO\_P**: базовый курс Python по программе бакалавриата
- ☐ **M\_INFO\_OOP**: базовый курс ООП на основе C++ магистры
- ☐ **M\_INFO**: Курс Geant4

# Основные операции Git в средах JetBrains на примере PyCharm

## Инициализация:

Для того чтобы перевести проект под распределённую систему контроля версий Git нужно:

- ❑ Во вкладке VCS выбрать либо создать локальный Git-репозиторий (1), либо сразу опубликовать проект в своем аккаунте на сервере, к примеру, **GitHub** (2)



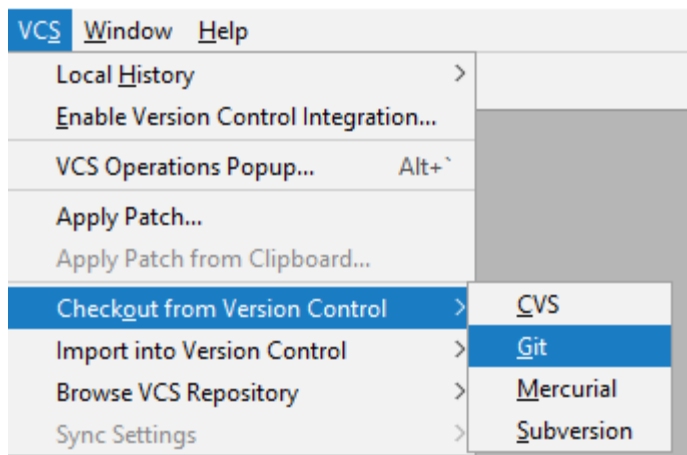




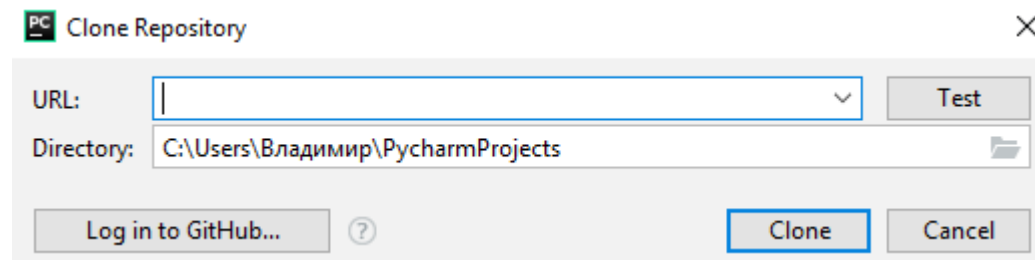
# Основные операции Git в средах JetBrains на примере PyCharm

## Клонирование:

Для того чтобы клонировать любой существующий проект под контролем версий Git нужно выбрать:



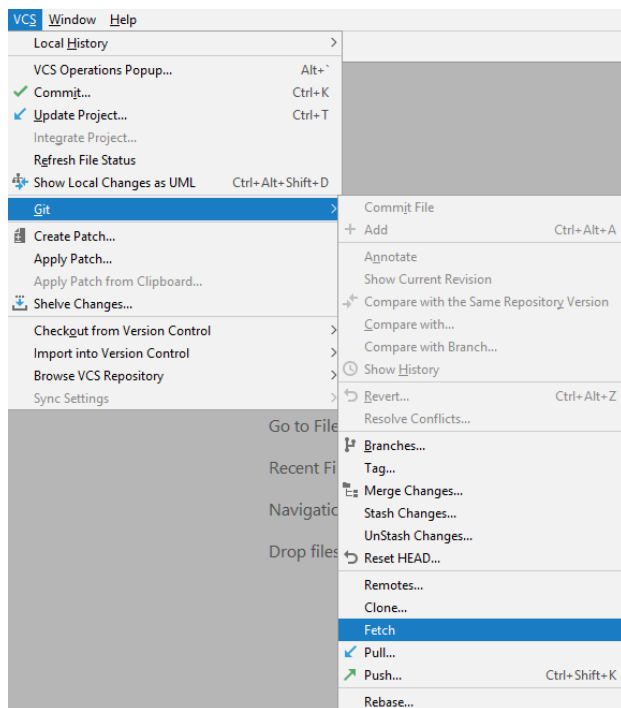
☐ В открывшемся окне следует указать путь до репозитория





# Основные операции Git в средах JetBrains на примере PyCharm

**Fetch** – данная операция обновляет информацию обо всех версиях и ветках на удаленном репозитории

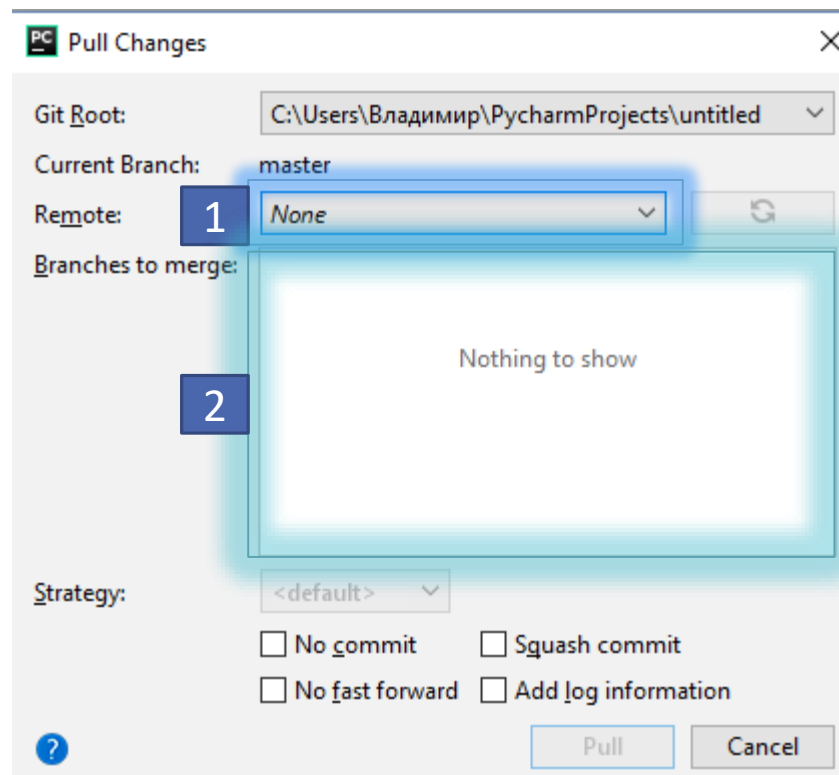
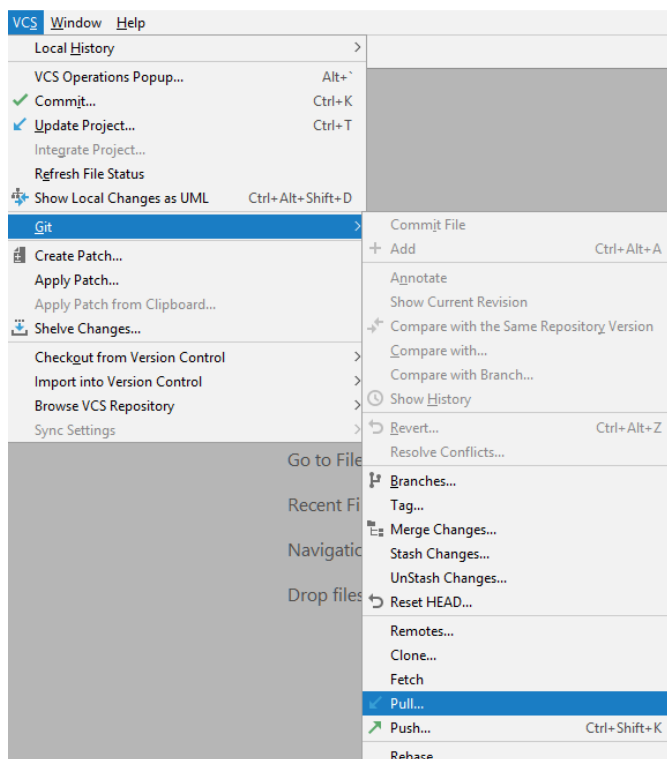


*Примечание: обратите внимание что все операции с Git доступны только для проектов находящихся под контролем версий*



# Основные операции Git в средах JetBrains на примере PyCharm

**Pull** – позволяет загрузить файлы из удаленного.



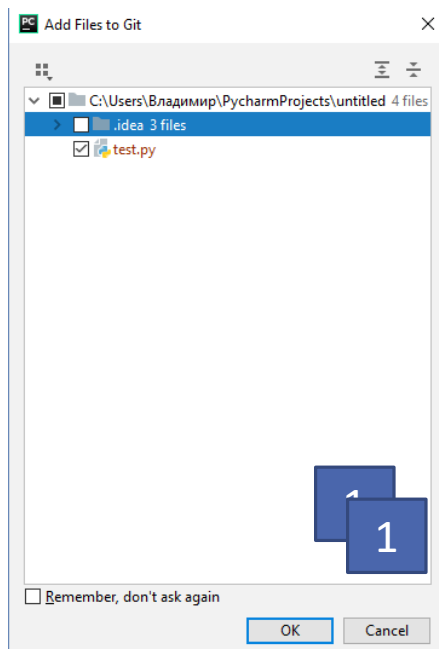
1 – имя (индекс) удаленного репозитория

2 – список доступных для загрузки (и одновременно сравнения и слияния) веток

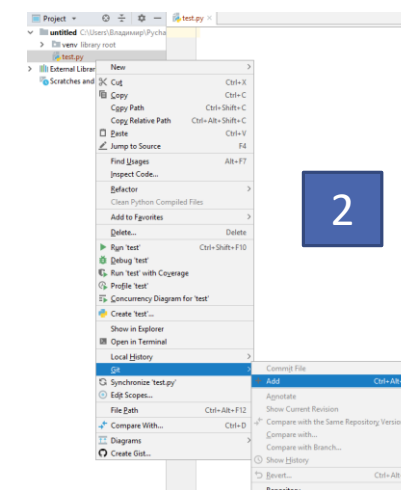


# Основные операции Git над файлами

При добавлении нового файла в проект **(Add)**, находящийся под управлением Git, среда разработки предлагает его *отслеживать* (Рисунок 1)



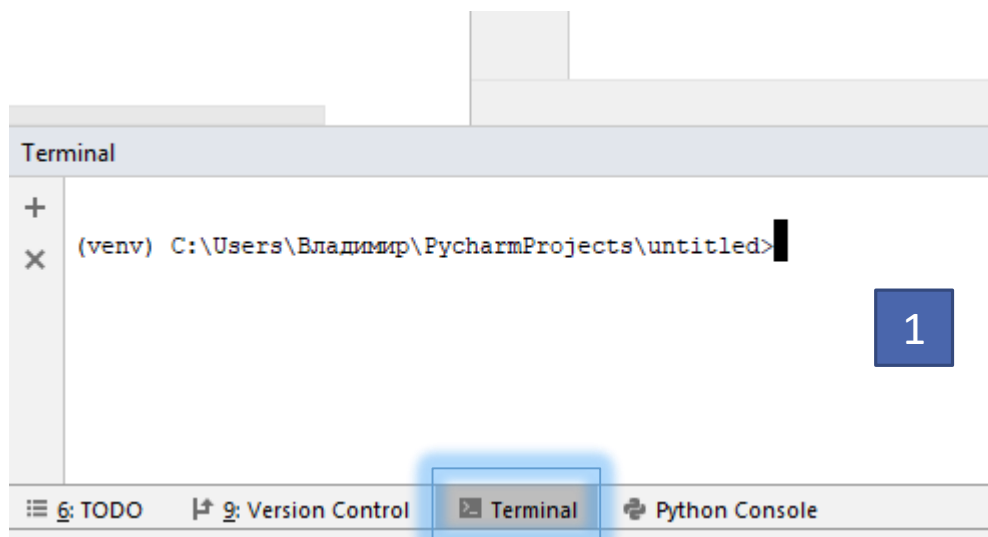
Кроме того можно добавить файл в любой момент просто нажав на него правой клавишей и выбрав соответствующий пункт **Add** в меню Git





# Основные операции Git над файлами

В средах **JetBrains** нет специальной кнопки чтобы не отслеживать файл системой контроля версий, однако эту процедуру можно при необходимости сделать с помощью встроенной консоли расположенной в нижней части рабочей области (рисунок 1):



Команда для удаления файла:

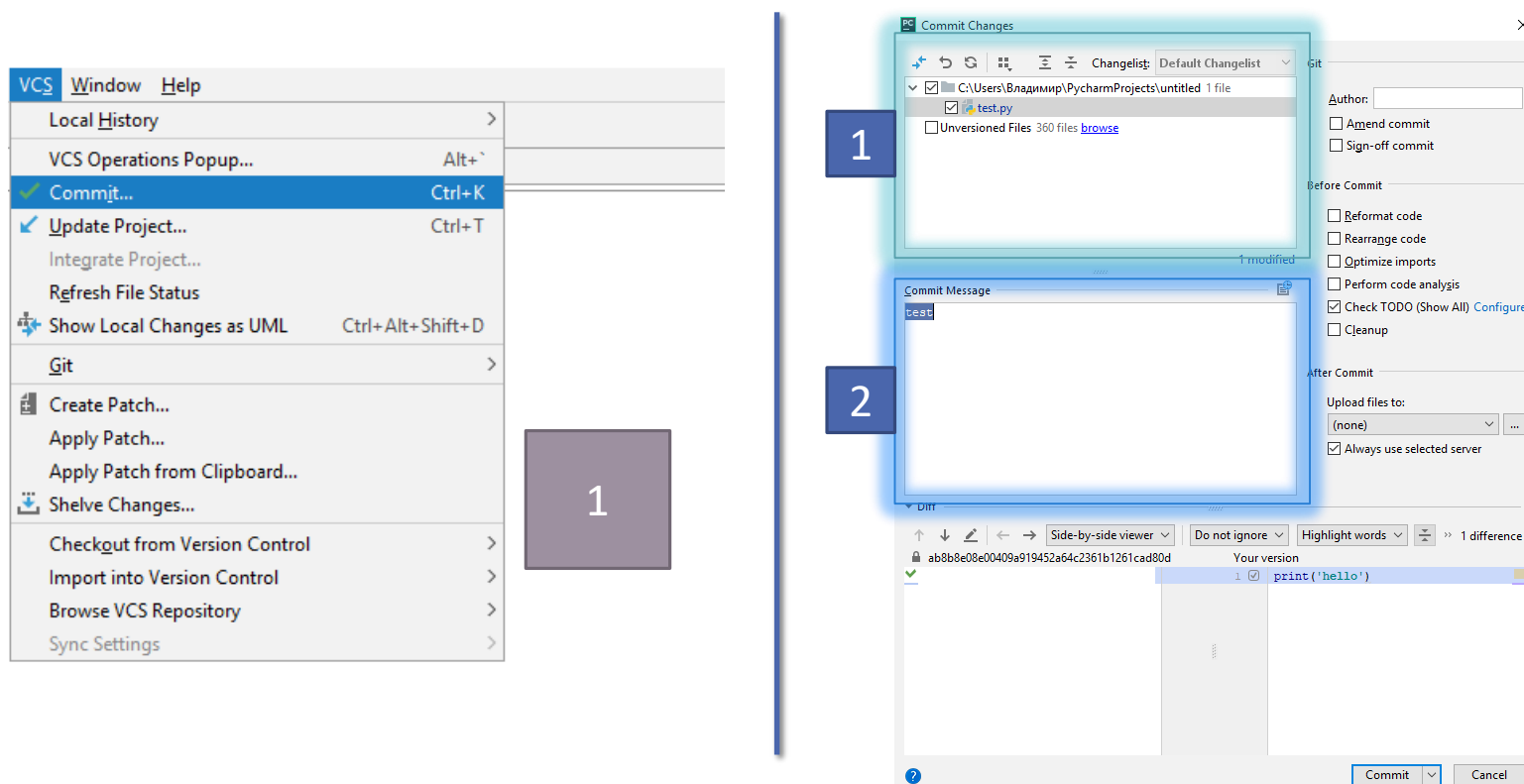
```
git rm --cached {file_name}
```

Где вместо **{file\_name}** следует указать исключаемый файл

*Примечание: через терминал можно вызывать все остальные процедуры которые не реализованы в интерфейсе*

# Commit

Когда завершены все планируемые работы по модификации файлов следует создать новую контрольную точку или версию. Для этого предусмотрена процедура commit (рис 1)



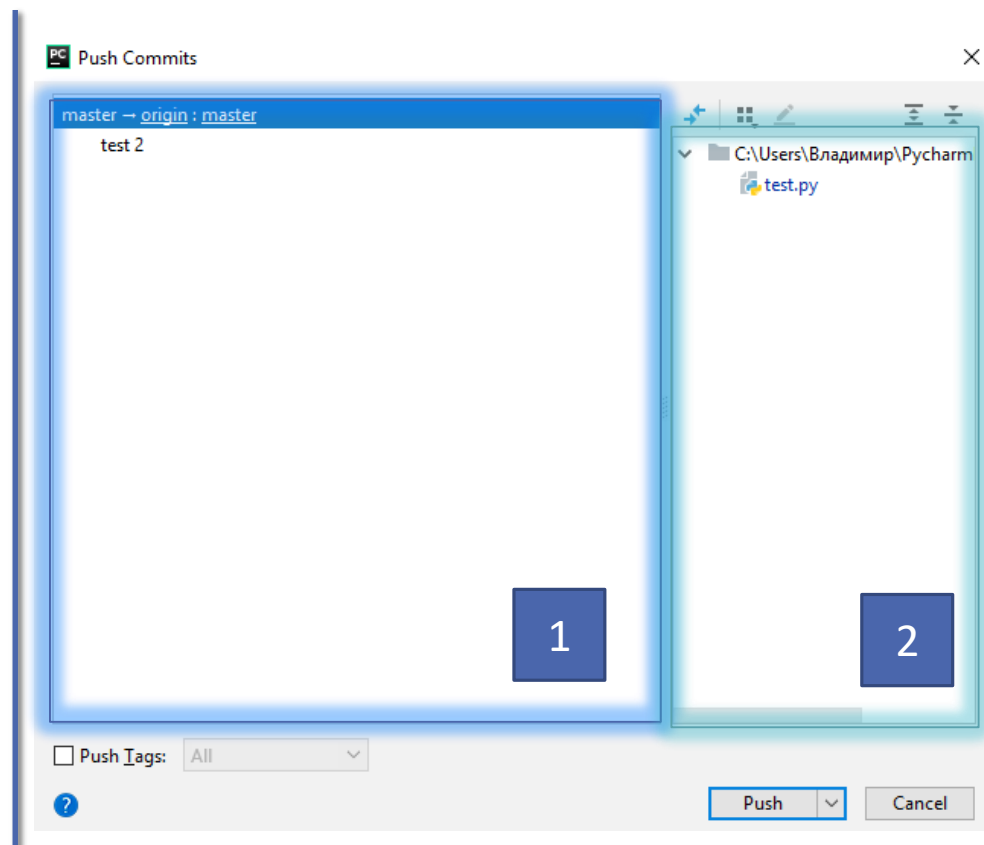
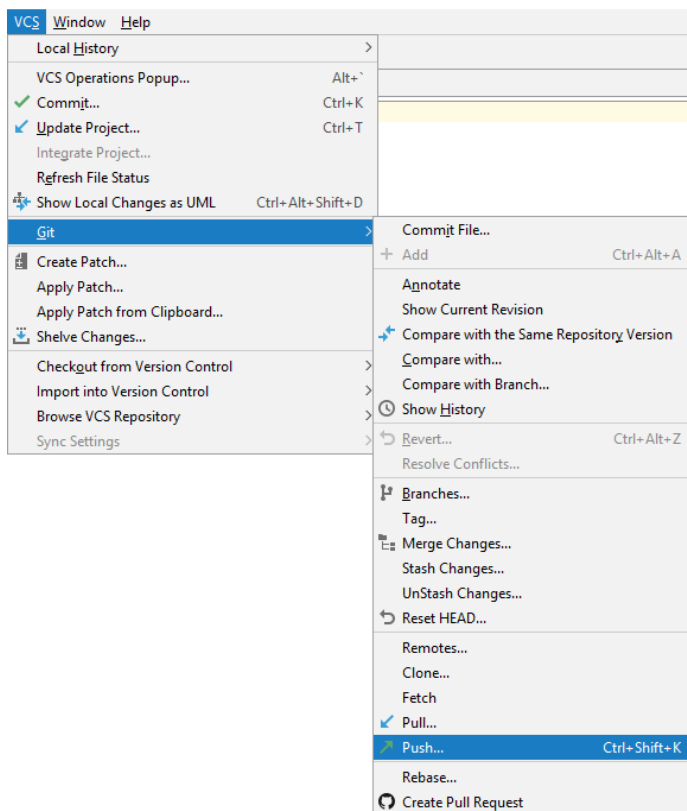
**1** – в этом окне индексируются все файлы необходимые для сохранения в данной версии

**2** – здесь пишется комментарий для текущих изменений



# Основные операции Git в средах JetBrains на примере PyCharm

**Push** – позволяет выгрузить все сохраненные изменения в репозитории



1 – список всех добавляемых версий (контрольных точек)

2 – изменяемые файлы