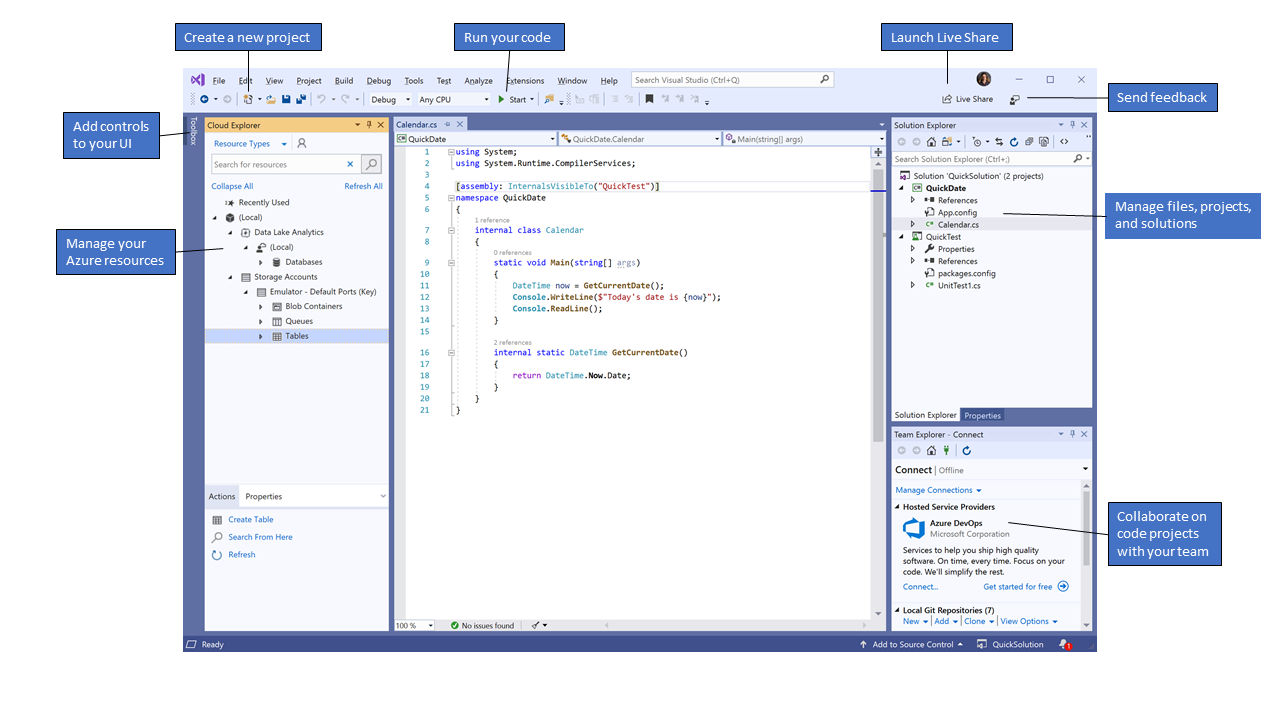
**Лабораторная работа №1**

Создание консольного приложения в IDE Visual Studio. Работа с системой контроля версий

**Цель**: изучение IDE Visual Studio, создание простейшего приложения на Python, подключение Git.

**Краткие теоретические сведения**

[](https://docs.microsoft.com/ru-ru/visualstudio/get-started/media/vs-2019/ide-overview.png?view=vs-2019#lightbox)Интегрированная среда разработки **Visual Studio** — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред, IDE Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

Подробнее <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019>

На рисунке показана среда Visual Studio с открытым проектом и несколькими окнами основных инструментов, которые вам наиболее полезны:

* [Обозреватель решений](https://docs.microsoft.com/ru-ru/visualstudio/ide/solutions-and-projects-in-visual-studio?view=vs-2019) (вверху справа) позволяет просматривать файлы кода, перемещаться по ним и управлять ими. **Обозреватель решений** позволяет упорядочить код путем объединения файлов в [решения и проекты](https://docs.microsoft.com/ru-ru/visualstudio/get-started/tutorial-projects-solutions?view=vs-2019).
* В [окне редактора](https://docs.microsoft.com/ru-ru/visualstudio/ide/writing-code-in-the-code-and-text-editor?view=vs-2019) (центр), где вы, скорее всего, будете проводить большую часть времени, отображается содержимое файла. Здесь вы можете редактировать код или разрабатывать пользовательский интерфейс, например окно с кнопками или текстовые поля.
* [Team Explorer](https://docs.microsoft.com/ru-ru/azure/devops/user-guide/work-team-explorer) (правый нижний угол) позволяет отслеживать рабочие элементы и использовать код совместно с другими пользователями с помощью технологий управления версиями, таких как [Git](https://git-scm.com/) и [система управления версиями Team Foundation (TFVC)](https://docs.microsoft.com/ru-ru/azure/devops/repos/tfvc/overview).

**Проекты**

При создании в Visual Studio приложения или веб-сайта вы начинаете с *проекта*. С логической точки зрения проект содержит все файлы, которые будут скомпилированы в исполняемую программу, библиотеку или веб-сайт. Сюда входят файлы с исходным кодом, значками, изображениями, данными и т. д. Проект также содержит параметры компилятора и другие файлы конфигурации, которые могут потребоваться разным службам или компонентам, с которыми взаимодействует программа.

**Файл проекта**

Visual Studio использует [MSBuild](https://docs.microsoft.com/ru-ru/visualstudio/msbuild/msbuild?view=vs-2019) для создания каждого проекта в решении, и каждый проект содержит файл проекта MSBuild. Расширение имени файла отражает тип проекта, например проект C# (CSPROJ), проект Visual Basic (VBPROJ) или проект базы данных (DBPROJ). Файл проекта представляет собой XML-документ, который содержит все сведения и инструкции, необходимые MSBuild для сборки проекта, включая содержимое, требования к платформе, сведения об управлении версиями, параметры веб-сервера или сервера баз данных, а также выполняемые задачи.

Файлы проекта основаны на [схеме XML MSBuild](https://docs.microsoft.com/ru-ru/visualstudio/msbuild/msbuild-project-file-schema-reference?view=vs-2019). Чтобы просмотреть содержимое [файлов проекта в новом стиле SDK](https://docs.microsoft.com/ru-ru/visualstudio/msbuild/how-to-use-project-sdk?view=vs-2019) в Visual Studio, щелкните правой кнопкой мыши узел проекта в **обозревателе решений** и выберите пункт **Изменить <projectname>** . Чтобы просмотреть содержимое проекта .NET Framework или другого проекта в этом стиле, сначала выгрузите проект (щелкните правой кнопкой мыши узел проекта в **обозревателе решений** и выберите пункт **Выгрузить проект**). Затем щелкните проект правой кнопкой мыши и выберите пункт **Изменить <projectname>** .

***Примечание***

Использовать решения или проекты в Visual Studio для редактирования, сборки и отладки кода необязательно. Можно просто открыть папку с файлами исходного кода в Visual Studio и начать редактирование. Дополнительные сведения см. в статье [Разработка кода в Visual Studio без использования проектов и решений](https://docs.microsoft.com/ru-ru/visualstudio/ide/develop-code-in-visual-studio-without-projects-or-solutions?view=vs-2019).

**Создать новые проекты**

Самый простой способ создать новый проект — использовать шаблон проекта с нужным типом проекта. Шаблон проекта — это базовый набор предварительно созданных файлов кода, файлов конфигурации, активов и параметров. Для выбора шаблона проекта нажмите **Файл** > **Создать** > **Проект**. Дополнительные сведения см. в разделе [Создание нового проекта](https://docs.microsoft.com/ru-ru/visualstudio/ide/create-new-project?view=vs-2019).

Вы можете также создать пользовательский шаблон проекта, чтобы затем использовать его для создания новых проектов. Дополнительные сведения см. в статье [Создание шаблонов проектов и элементов](https://docs.microsoft.com/ru-ru/visualstudio/ide/creating-project-and-item-templates?view=vs-2019).

При создании нового проекта Visual Studio сохраняет его в расположении по умолчанию — *%USERPROFILE%\source\repos*. Чтобы изменить это расположение, перейдите в раздел **Средства** > **Параметры** > **Проекты и решения** > **Расположения**. Дополнительные сведения см. в [диалоговом окне параметров в разделе "Проекты и решения" > "Расположения"](https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/projects-solutions-locations-options?view=vs-2019).

**Решения**

Проект находится внутри *решения*. Несмотря на название, под решением не подразумевается "решение вопроса". Это просто контейнер для одного или нескольких связанных проектов вместе с информацией о сборке, параметрами окна Visual Studio и любыми прочими файлами, которые не относятся к какому-либо конкретному проекту.

**Файл решения**

В Visual Studio используются два типа файлов (*SLN* и *SUO*) для хранения параметров, связанных с решениями:

| Файл решения | | |
| --- | --- | --- |
| **Расширение** | **name** | **Описание** |
| .SLN | Решение Visual Studio | Организует проекты, элементы проектов и решений в решении. |
| SUO | Параметры пользователя решения | Сохраняет настройки и параметры уровня пользователя, например точки останова. |

***Важно!***

Решение описывается текстовым файлом (*SLN*) в собственном уникальном формате; его обычно не изменяют вручную. И наоборот, *SUO*-файл является скрытым и не отображается, если проводник имеет параметры по умолчанию. Для отображения скрытых файлов в меню **Вид** проводника установите флажок **Скрытые элементы**.

**Папка решения**

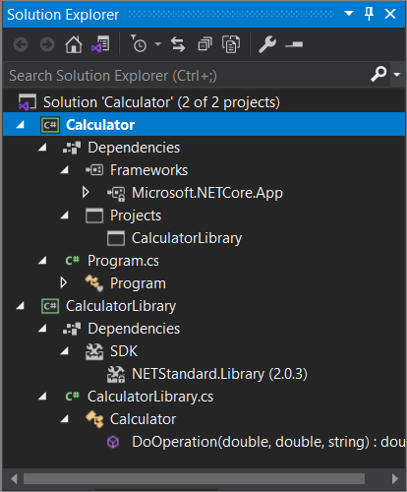
Папка решения — это виртуальная папка, которая доступна только в **Обозревателе решений**, где с ее помощью можно группировать проекты в решении. Если нужно найти файл решения на компьютере, перейдите к разделу **Средства** > **Параметры** > **Проекты и решения** > **Расположения**. Дополнительные сведения см. в [диалоговом окне параметров в разделе "Проекты и решения" > "Расположения"](https://docs.microsoft.com/ru-ru/visualstudio/ide/reference/projects-solutions-locations-options?view=vs-2019).

***Совет***

Пример проекта и решения, созданного с нуля, а также пошаговые инструкции и пример кода см. в статье [Общие сведения о проектах и решениях](https://docs.microsoft.com/ru-ru/visualstudio/get-started/tutorial-projects-solutions?view=vs-2019).

**Обозреватель решений**

После создания проекта можно просматривать проект, решение и связанные с ними элементы, а также для управлять ими с помощью **обозревателя решений**. На следующем рисунке показан **обозреватель решений** с решением C#, включающим два проекта:



На панели инструментов в верхней части **Обозревателя решений** есть кнопки для переключения из представления решения в представление папки, фильтрации ожидающих изменений, отображения всех файлов, свертывания всех узлов, просмотра страниц [свойств](https://docs.microsoft.com/ru-ru/visualstudio/ide/managing-project-and-solution-properties?view=vs-2019), просмотра кода в [редакторе кода](https://docs.microsoft.com/ru-ru/visualstudio/ide/writing-code-in-the-code-and-text-editor?view=vs-2019) и т. д.

Многие команды меню доступны в контекстном меню различных элементов в **Обозревателе решений**. К таким командам относится сборка проекта, управление пакетами NuGet, добавление ссылок, переименование файла и запуск тестов. Это лишь некоторые из них.

В **обозревателе решений** можно настроить способ вложения файлов для проектов ASP.NET Core. Дополнительные сведения см. в статье [Настройка вложения файлов в обозревателе решений](https://docs.microsoft.com/ru-ru/visualstudio/ide/file-nesting-solution-explorer?view=vs-2019)

***Совет***

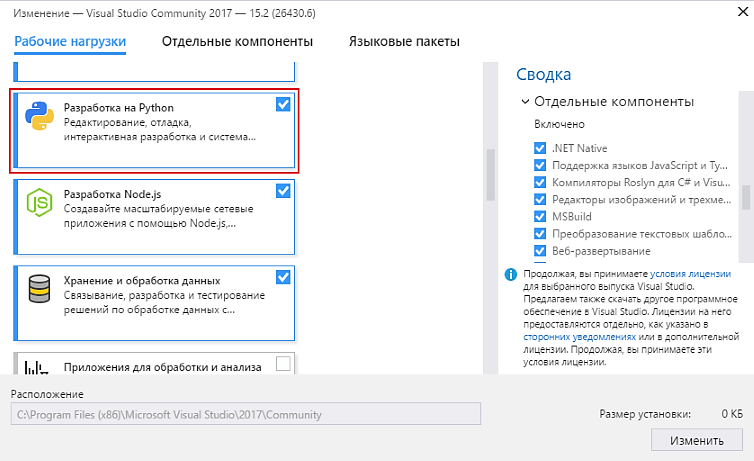
Если вы закрыли Обозреватель решений и хотите вновь открыть его, выберите **Вид** > **Обозреватель решений** в строке меню или нажмите клавиши **CTRL**+**ALT**+**L**. Если же вы закрыли боковые вкладки и хотите восстановить их в расположениях по умолчанию, выберите **Окно** > **Сброс макета окон** в строке меню.

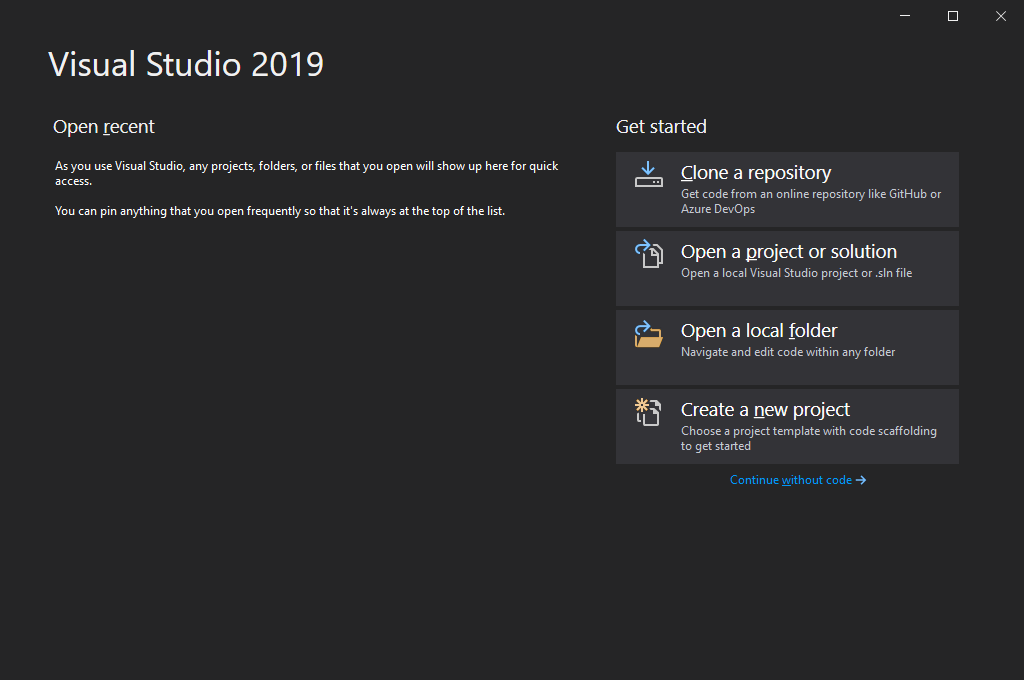
***Примечание***

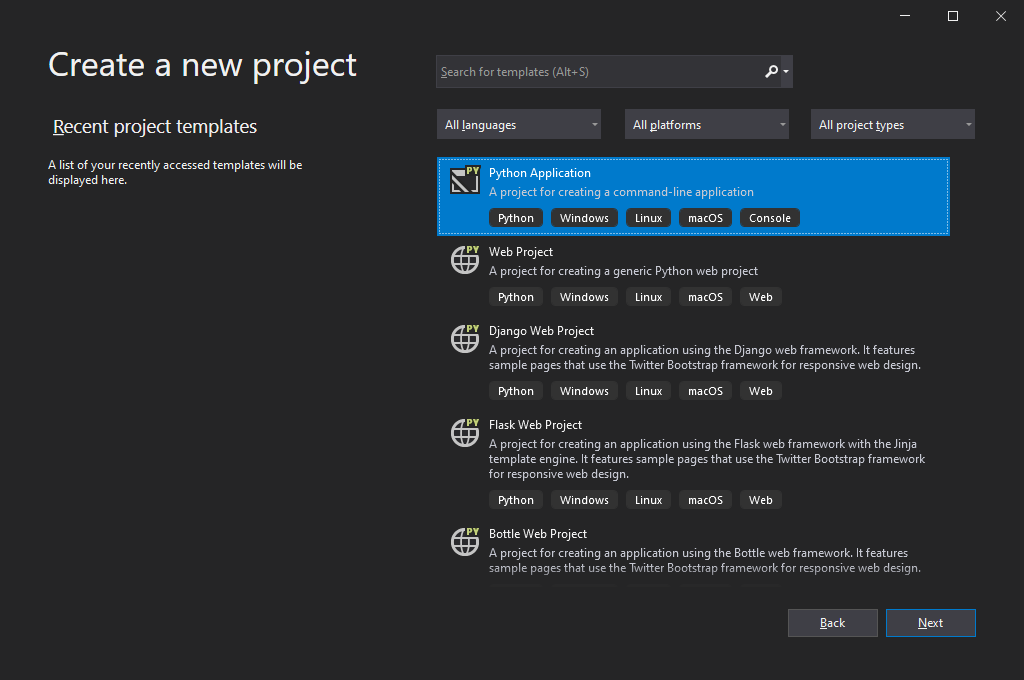
Чтобы просмотреть изображения и значки приложений, отображаемые в Visual Studio, скачайте [**библиотеку изображений Visual Studio**](https://www.microsoft.com/download/details.aspx?id=35825).

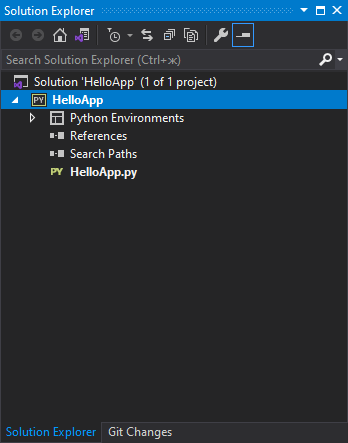
Преимуществом IDE Visual Studio как среды разработки, которая позволяет работать с *Python*, по сравнению, скажем, с PyCharm, следует отметить прежде всего то, что в ее бесплатной редакции VS 2019 Community бесплатно доступны ряд функций и возможностей, которые в том же PyCharm доступны только в платной версии Professional Edition. Например, это веб-разработка, в том числе с помощью различных *фреймворков* (ПО, облегчающее разработку и объединение разных компонентов большого программного проекта, отличается от библиотеки тем, что та может быть использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру ПП и не накладывая на неё никаких ограничений, а фреймворк диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять).

**Выполнение работы:**

* 1. Установка Visual Studio 2019 Community. Загрузить установочный файл Visual Studio 2019 Community по ссылке <https://visualstudio.microsoft.com/downloads/>. После запуска установочного файла выбрать среди устанавливаемых опций Python:
  2. После установки Visual Studio запустить ее. В меню выбрать пункт **Create a new Project** (Создать проект), откроется окно создания нового проекта.



В этом окне в перечне основного меню перейти к **Python Application**:

Выбрав Python, в центральной части окна можно увидеть богатую палитру типов проектов, которые возможно создавать для разработки на данном языке программирования. Это и веб-разработка, и машинное обучение, и проекты для работы с облаком, проекты настольных приложений и т.д. В данном же случае выбрать в качестве типа проекта Python Application, то есть тип простых консольных приложений, и назовем новый проект HelloApp. Нажать на кнопку **Create**, и Visual Studio создаст новый проект:

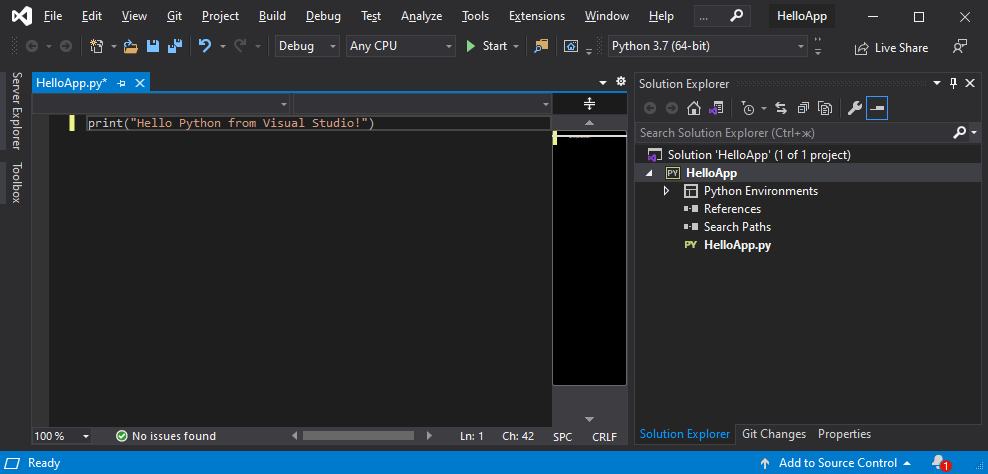
По умолчанию проекты сохраняются по пути C:\Users\Me\source\repos.

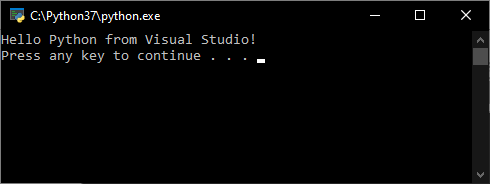
* 1. Справа в окне Solution Explorer (Обозреватель решений) можно увидеть структуру проекта. По умолчанию здесь мы можем увидеть следующие элементы:
* Python Environments: здесь можно увидеть все используемые среды, в частности, здесь можно найти сведения о компиляторе, который используется.
* References: в этот узел помещаются все внешние зависимости, которые используются текущим проектом
* Search Paths: этот узел позволяет указать пути поиска для модулей Python
* HelloApp.py: собственно, файл Python с исходным кодом

По умолчанию в Visual Studio уже открыт файл HelloApp.py, но он пока пуст. Добавим в него строку кода на Python:

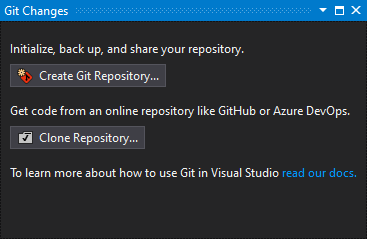
|  |  |
| --- | --- |
|  | print("Hello Python from Visual Studio!") |

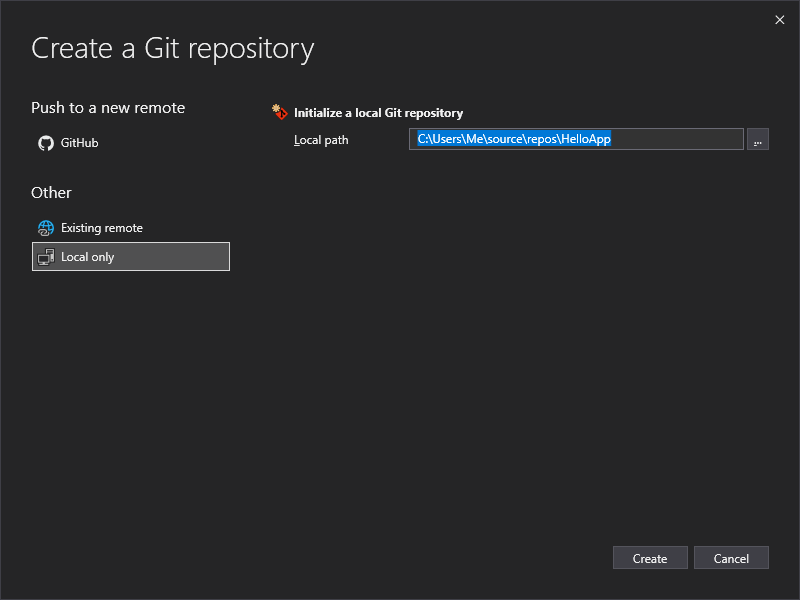
* 1. Затем в панели инструментов нажмем на зеленую стрелочку для запуска:

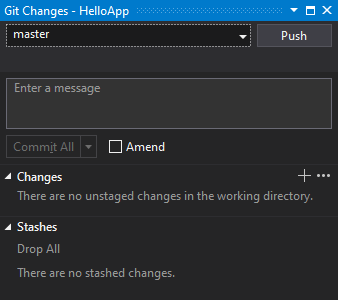


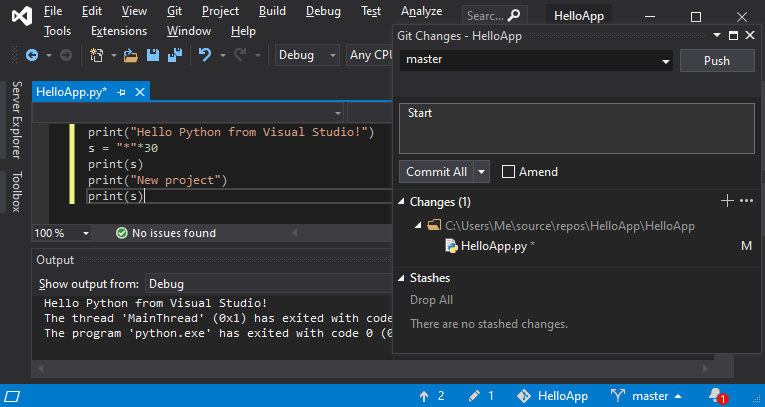
В результате запуска отобразится консоль, которая выведет нужную строку:

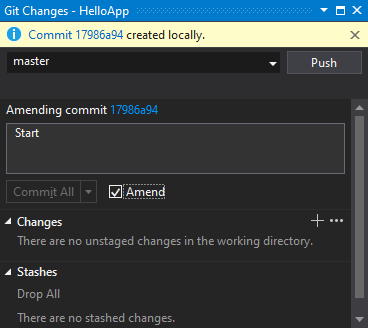
* 1. Для того, чтобы файлы приложения отслеживались системой контроля версий Git, выбрать **Git** **Changes** и нажать **Create Git Repository**:



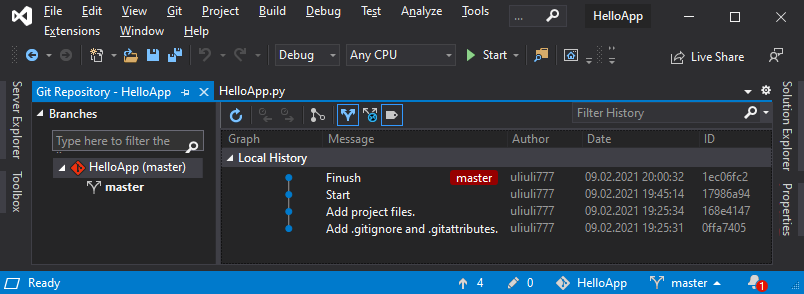
По умолчанию в качестве репозитория создаётся одноимённая с текущим проектом директория, путь к которой C:\Users\Me\source\repos\HelloApp. Пока работа выполняется локально, выбрать **Local Only** и нажать **Create**.

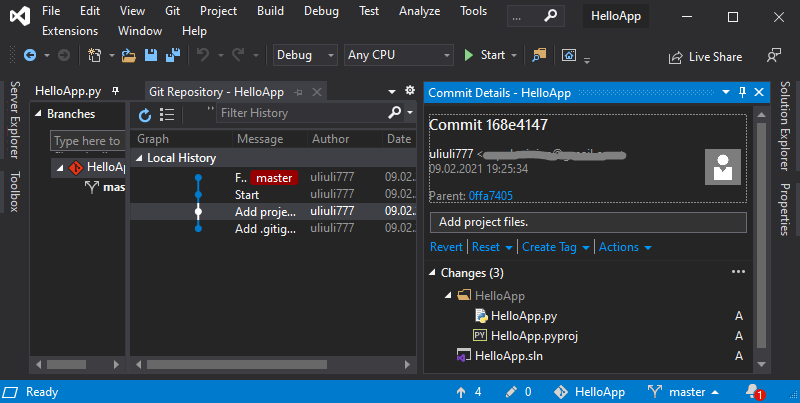
В окне Git Changes появилось название репозитория и главная ветвь разработки **master**.

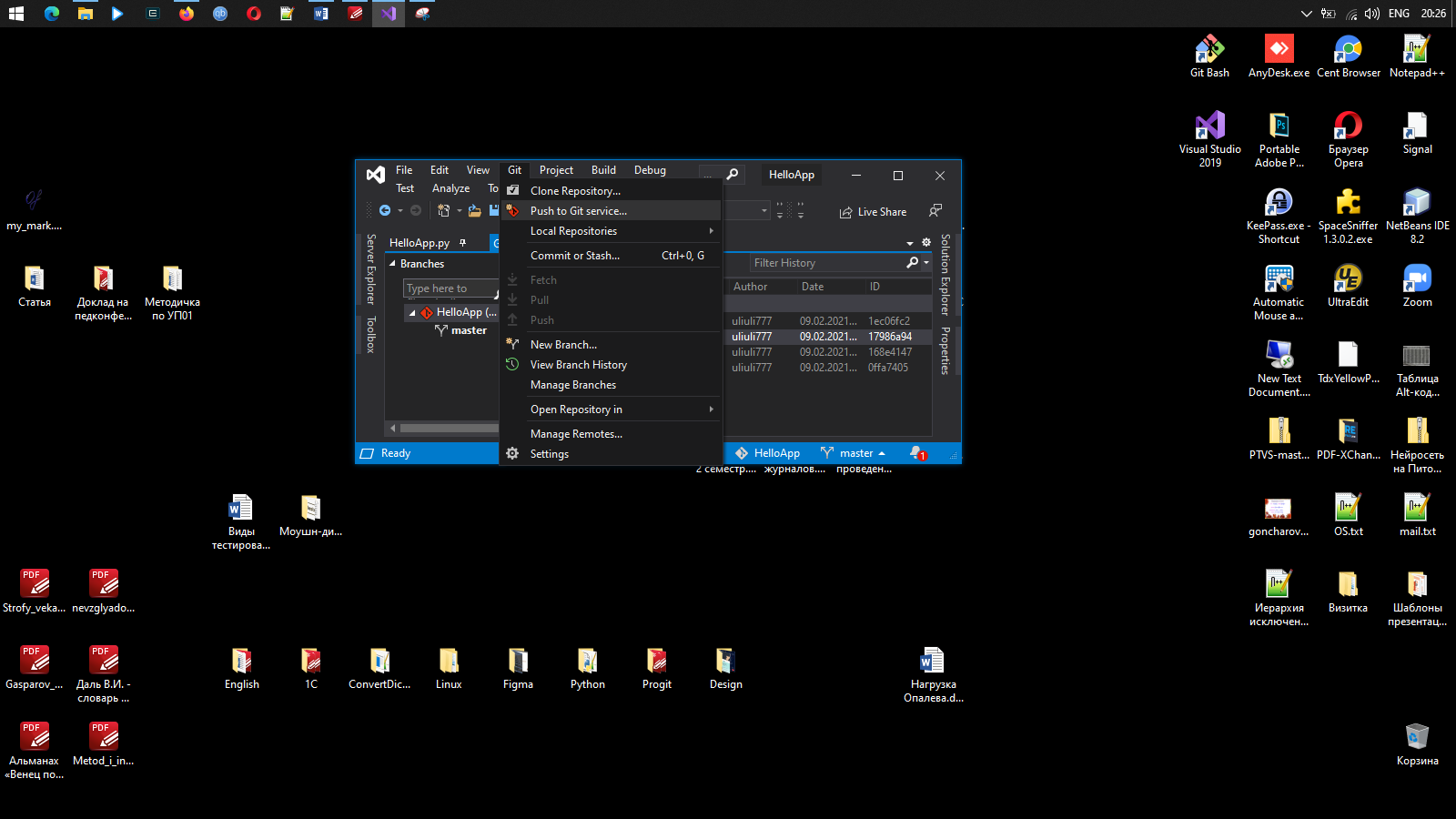
* 1. Далее добавить несколько строк кода и выполнить **коммит** с комментарием (например, «Start»):

В заголовке Git Changes высветится сообщение о создании коммита. Если вдруг потребуется изменить комментарий к нему, то можно это сделать, выбрав опцию **Amend**:

* 1. Добавить строки кода одной из решаемых ранее по своему варианту задач на Python. Выполнить коммит. Просмотреть историю коммитов, нажав на цифру со стрелкой вверх:



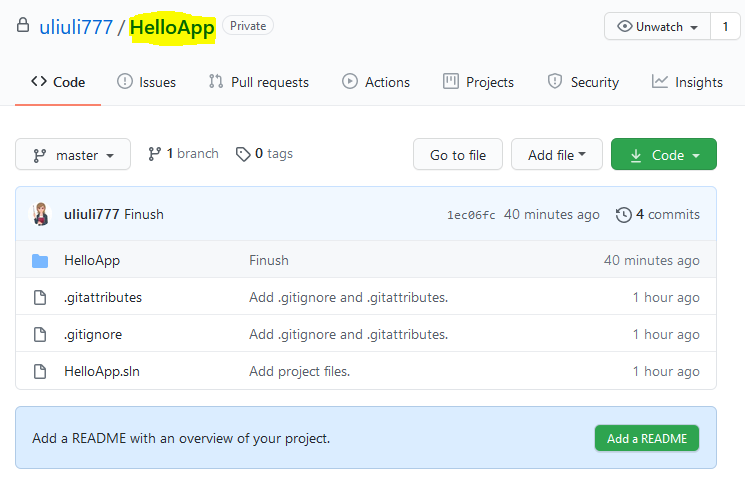
Выбрав один из коммитов, просмотреть его детали с помощью команды контекстного меню:



* 1. Запушить изменения (передать в удалённый репозиторий), воспользовавшись командой главного меню **Git > Push to Git service**.

Потребуется авторизация на **Github** посредством имеющегося у вас аккаунта.

***Примечание:*** обязательно изучите основные возможности, пройдя последовательно по пунктам главного меню Git.

* 1. Перейти на Github и продемонстрировать результат. Должен быть создать новый репозиторий:
  2. Просмотреть содержимое файла **.gitignore**. Внести в отчёт вывод о том, какого рода файла не отслеживаются системой Git при работе с IDE Visual Studio.
  3. Создать новый проект в VS. Написать код генерирующий список элементов случайным образом из диапазона от 5 до № по журналу \* 100 (число элементов № по журналу + 10). Выполнить коммит (содержание должно соответствовать задаче). Оформить код в виде функции, вызвав её с указанным числом элементов. Добавить коммит. Запушить на GitHub. На веб-сервисе создать файл README с описанием задачи, перечнем, включающим среду и язык реализации, используемые библиотеки, фамилию разработчика.

**Контрольные вопросы:**

* 1. Что такое система контроля версий Git? Требуется ли её установка при работе с VS?
  2. Какие основные возможности предоставляет Git в среде VS?
  3. Что из настроек Git является обязательным при работе с удалённым репозиторием?
  4. Какую систему защиты и сертификации данных использует Git по умолчанию?
  5. Можно ли вернуться к прежней версии файла с помощью Git? Каким образом?
  6. Что нужно сделать, если требуется изменить сообщение последнего коммита?
  7. Как называется главная ветвь разработки? Можно ли её переименовать?
  8. Зачем нужен файл .gitignore и каким образом он создаётся?