

ТЕМА 1. МОДУЛИ QT ДЛЯ СОЗДАНИЯ ПРИЛОЖЕНИЙ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ.

ЛЕКЦИЯ 1. СОСТАВ QT.



ОБО МНЕ

- ФОТО

- Алексеев Владислав Алексеевич, 26 лет.
- 2012 – 2017 гг. ВГТУ. Факультет Радиотехники и Электроники.
- 2014 – 2018 гг. Фриланс.
- 2017 – 2018 гг. АО «НИИ СВТ». Программист.
- 2018 – 2021 гг. АО «НИИ СВТ». Старший программист.

УЧЕБНЫЕ ВОПРОСЫ

1. Основные компоненты Qt
2. Основные фреймворки для Qt на Python, их различия
3. Установка и настройка PySide2

ИСТОЧНИКИ

- Официальная документация: <https://doc.qt.io/qtforpython>
- Прохоренок Н. А., Дронов В. А. Python 3 и PyQt 5. Разработка приложений. 2019 г.

1. ОСНОВНЫЕ КОМПОНЕНТЫ QT

| <u>QtCore</u> | <u>QtGui</u> | <u>QtWidgets</u> |
|-------------------------|--------------------------|------------------------|
| <u>QtConcurrent</u> | <u>QtHelp</u> | <u>QtNetwork</u> |
| <u>QtOpenGL</u> | <u>QtOpenGLFunctions</u> | <u>QtOpenGLWidgets</u> |
| <u>QtPrintSupport</u> | <u>QtQml</u> | <u>QtQuick</u> |
| <u>QtQuickControls2</u> | <u>QtQuickWidgets</u> | <u>QtSql</u> |
| <u>QtSvg</u> | <u>QtSvgWidgets</u> | <u>QtTest</u> |
| <u>QtUiTools</u> | <u>QtXml</u> | |

- **QtCore** – основные функции, не связанные с графическим интерфейсом.
- **QtGui** – расширяет функциональность графического интерфейса.
- **QtWidgets** – работа с виджетами Qt.
- **QtConcurrent** – высокоуровневое для работы с потоками.
- **QtHelp** – интеграция онлайн-документации в приложения.
- **QtNetwork** – позволяет писать клиент-серверные (TCP/IP) приложения.
- **QtOpenGL/QtOpenGLFunctions/QtOpenGLWidgets** – работа с 2D/3D графикой.
- **QtPrintSupport** – класс обеспечивающий поддержку печати.
- **QtQml/QtQuick/QtQuickControls2/QtQuickWidgets** – API для использования Qt QML (Qt Meta/Modeling Language) и создания настраиваемых высокодинамичных графических пользовательских интерфейсов с плавными переходами и эффектами.

- **QtSql** – содержит драйвера для обеспечения интеграции БД приложением.
- **QtSvg/QtSvgWidgets** – для работы с файлами SVG.
- **QtTest** – классы для модульного тестирования.
- **QtUiTools** – служит для обработки форм, созданных с помощью Qt Designer.
- **QtXml** – обеспечивает работу с потоками чтения и записи XML документов и реализацию их в форме SAX (Simple API for XML) и DOM (Document Object Model)

2.ОСНОВНЫЕ ФРЕЙМВОРКИ ДЛЯ QT НА PYTHON, ИХ РАЗЛИЧИЯ

2.1. Различия в поставке.

| Категория/Фреймворк | PyQt5 | PySide2 |
|--------------------------|-------------------------------|--|
| Лицензия | GPL или коммерческая лицензия | LGPL |
| Версия Qt на xx.xx.xxxx | v5.15.3/v6.0.2 (PyQt6) | v5.15.2/v5.15.2+(PySide6) |
| Платформа | Python 3+ | Python 3 и Python 2.7 (только для Linux и MacOS) |
| Первый стабильный выпуск | Апрель 2016 | Июль 2018 |
| Состав пакета | Основное ядро ~50Mb | Полный пакет со всеми инструментами ~120Mb |

Таблица 1. Различия в поставке.

2.2. РАЗЛИЧИЯ В КОДЕ.

- 2.2.1. Конвертация Ui – файлов в Py:

```
pyside2-uic mainwindow.ui -o MainWindow.py  o Pyside2
pyuic5 mainwindow.ui -o MainWindow.py      o PyQt5
```

- 2.2.2. Использование `exec()` или `exec_()`:

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    myapp = FirstForm()
    myapp.show()
    sys.exit(app.exec_())
```

- PySide2 – только `exec_()`, т.к. присутствует поддержка Python 2;
- PyQt5 – возможны оба варианта, т.к. в Python 3 “`exec`” не является ключевым/зарезервированным словом.

- 2.2.3. Слоты и сигналы:

```
from PySide2.QtCore import Signal, Slot # для PySide2
from PyQt5.QtCore import pyqtSignal, pyqtSlot # для PyQt5

myCustomPySide2Signal = Signal() # для PySide2
myCustomPyQt5Signal = pyqtSignal() # для PyQt5

@Slot # для PySide2
def myCustomPySide2Slot():
    pass

@pyqtSlot # для PyQt5
def myCustomPyQt5Slot():
    pass
```



Обеспечение обратной совместимости:

```
import pip._internal.operations.freeze

for requirement in pip._internal.operations.freeze.freeze(local_only=True):
    print(requirement)
    if 'PyQt5' in requirement: # PyQt5
        print("Work with PyQt5")
        from PyQt5 import QtGui, QtWidgets, QtCore
        from PyQt5.QtCore import pyqtSignal as Signal, pyqtSlot as Slot
        break

    elif 'PySide2' in requirement: # PySide2
        print("Work with PySide2")
        from PySide2 import QtGui, QtWidgets, QtCore
        from PySide2.QtCore import Signal, Slot
        break
```

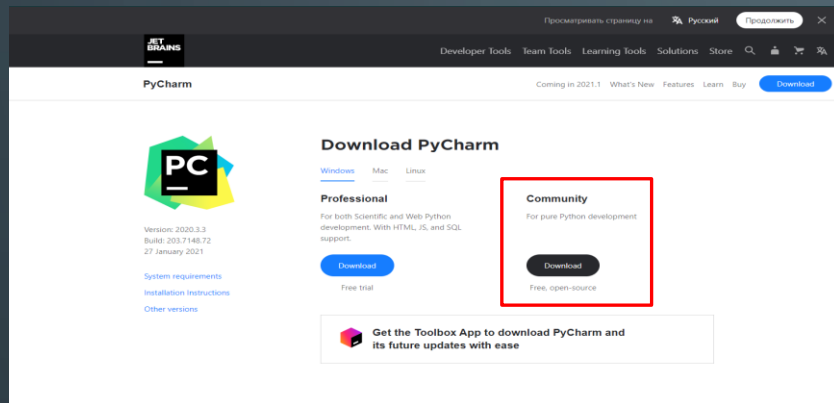
3. УСТАНОВКА И НАСТРОЙКА PYSIDE2

Используемые в курсе инструменты для разработки

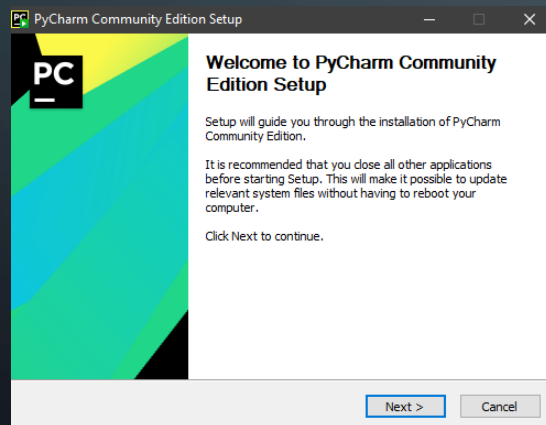
| | | |
|------------------------|------------|---|
| IDE | PyCharm CE | https://www.jetbrains.com/pycharm/download |
| Окружение | Virtualenv | https://docs.python.org/3/library/venv.html |
| VSC (рекомендовано) | GIT/GitHub | https://git-scm.com |
| Фреймворк | PySide2 | https://doc.qt.io/qtforpython/ |

3.1. УСТАНОВКА PYCHARM

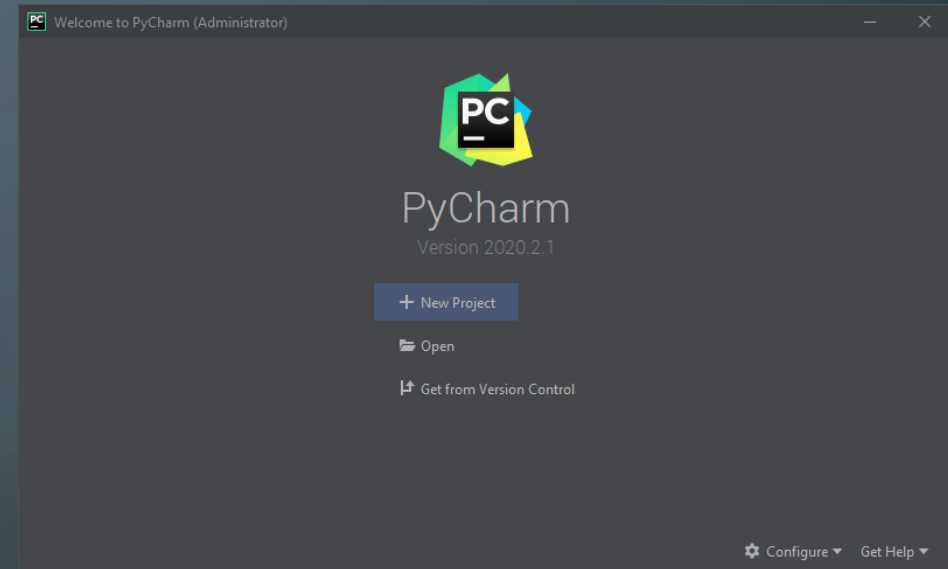
- 3.1.1. Загружаем ~350Mb.



- 3.1.2. Устанавливаем.



- 3.1.3. Используем.



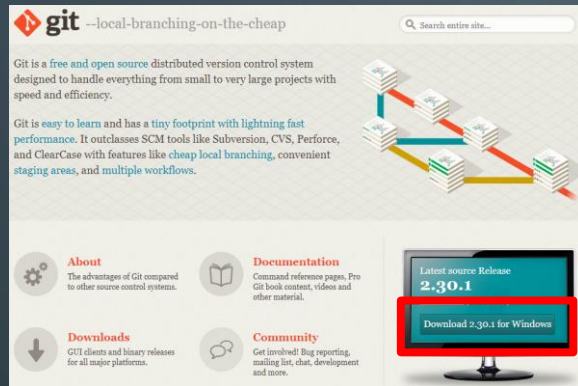
Примечание:

горячие клавиши Pycharm:

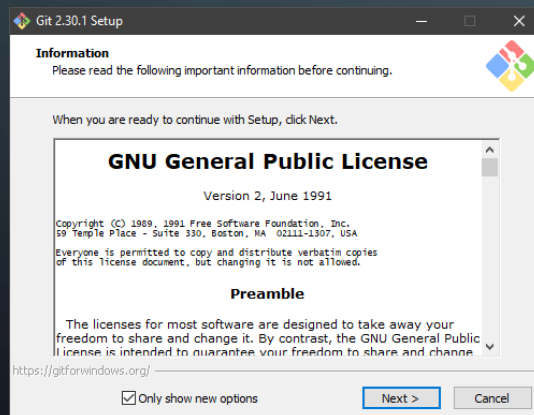
<https://www.jetbrains.com/help/pycharm/mastering-keyboard-shortcuts.html>

3.2. УСТАНОВКА GIT

• 3.2.1. Загружаем ~50Mb.



• 3.2.2. Устанавливаем.



3.2.3. Проверяем установку

```
C:\Users\vlad>git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone          Clone a repository into a new directory
  init           Create an empty Git repository or reinitialize an existing one


work on the current change (see also: git help everyday)
  add            Add file contents to the index
  mv             Move or rename a file, a directory, or a symlink
  restore        Restore working tree files
  rm             Remove files from the working tree and from the index
  sparse-checkout Initialize and modify the sparse-checkout


examine the history and state (see also: git help revisions)
  bisect        Use binary search to find the commit that introduced a bug
  diff          Show changes between commits, commit and working tree, etc
  grep          Print lines matching a pattern
  log           Show commit logs
  show          Show various types of objects
  status        Show the working tree status

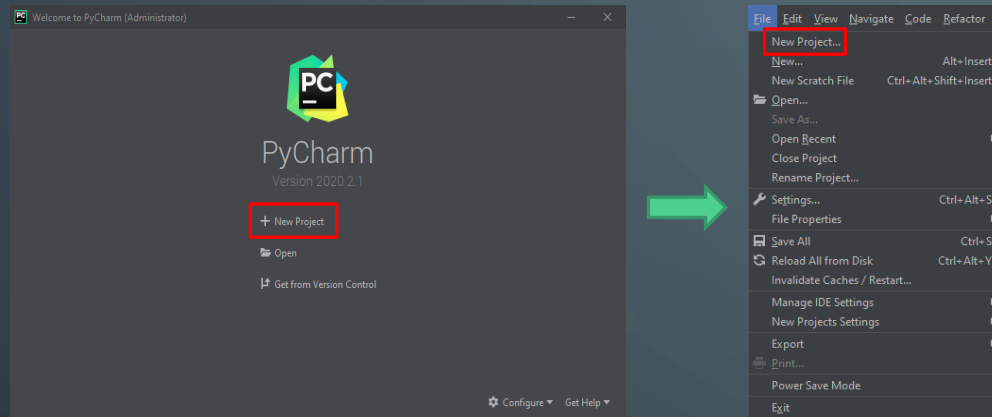

grow, mark and tweak your common history
  branch        List, create, or delete branches
  commit        Record changes to the repository
  merge         Join two or more development histories together
  rebase        Reapply commits on top of another base tip
  reset         Reset current HEAD to the specified state
  switch        Switch branches
  tag           Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
  fetch         Download objects and refs from another repository
  pull          Fetch from and integrate with another repository or a local branch
  push          Update remote refs along with associated objects

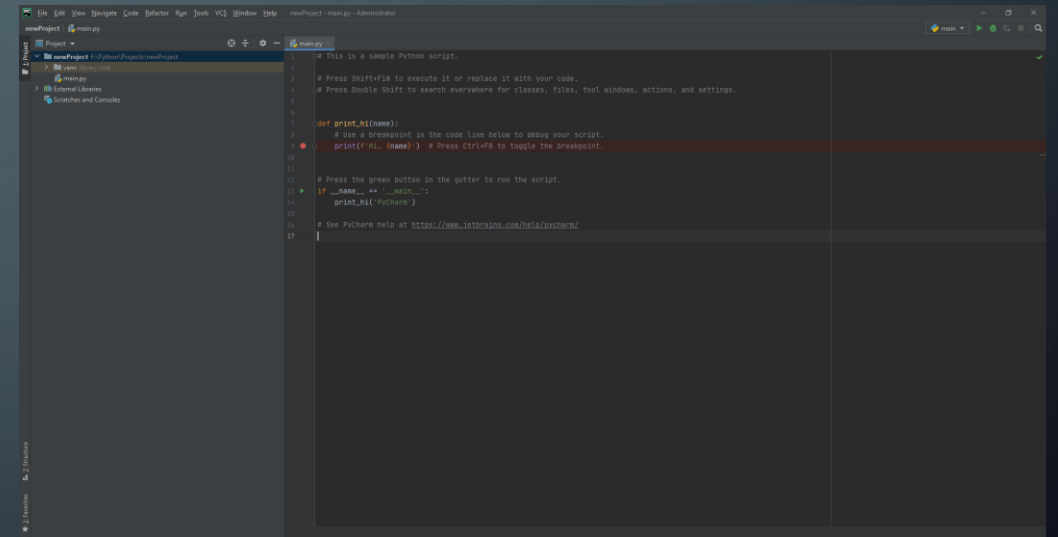
'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

3.3. СОЗДАНИЕ ПРОЕКТА PYCHARM

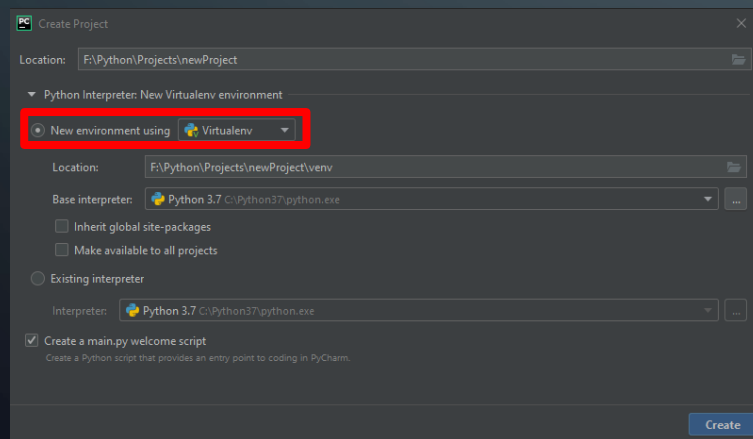
• 3.3.1. New Project



3.3.3. Минимальное рабочее приложение

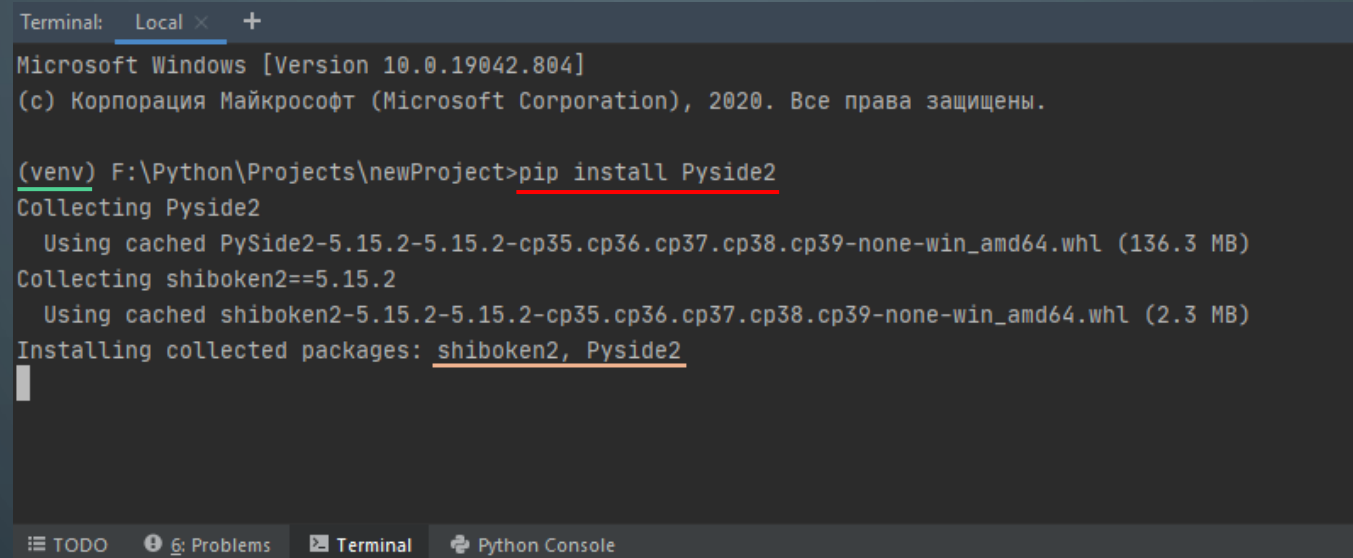


• 3.3.2. New environment using



3.4. УСТАНОВКА PYSIDE2 И СОЗДАНИЕ ПРОЕКТА

- 3.4.1. Устанавливаем PySide2.



```
Terminal: Local x +
Microsoft Windows [Version 10.0.19042.804]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

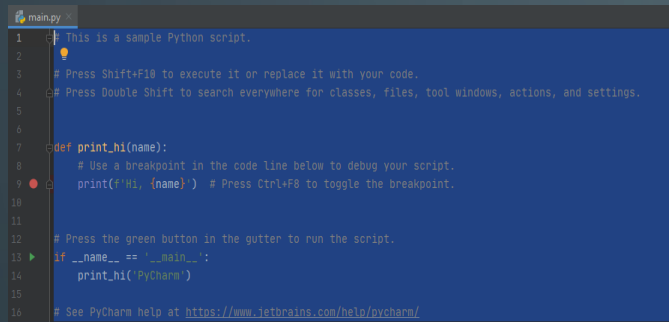
(venv) F:\Python\Projects\newProject>pip install Pyside2
Collecting Pyside2
  Using cached PySide2-5.15.2-5.15.2-cp35.cp36.cp37.cp38.cp39-none-win_amd64.whl (136.3 MB)
Collecting shiboken2==5.15.2
  Using cached shiboken2-5.15.2-5.15.2-cp35.cp36.cp37.cp38.cp39-none-win_amd64.whl (2.3 MB)
Installing collected packages: shiboken2, Pyside2
```

Примечание:

Пакет *shiboken2* – генератор привязок (binding generator), используемый для обеспечения связи между классами Qt (C++) и кодом PySide2.

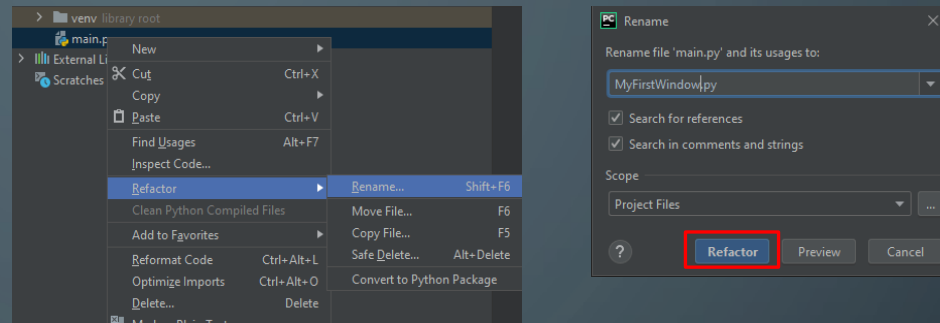
• 3.4.2. Создаём минимальное работающее PySide2 приложение.

3.4.2.1. Удаляем код шаблона

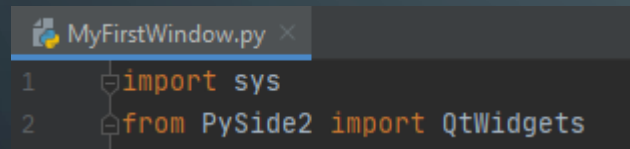


```
1 # This is a sample Python script.  
2  
3 # Press Shift+F10 to execute it or replace it with your code.  
4 # Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.  
5  
6 def print_hi(name):  
7     # Use a breakpoint in the code line below to debug your script.  
8     print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.  
9  
10  
11 # Press the green button in the gutter to run the script.  
12 if __name__ == '__main__':  
13     print_hi('PyCharm')  
14  
15 # See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

3.4.2.2. Переименовываем основной файл проекта

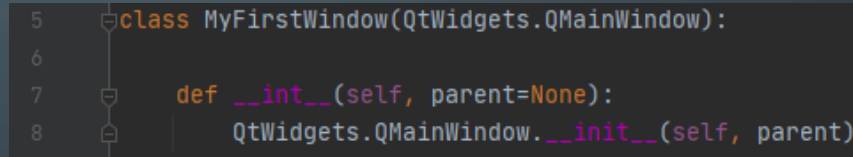


3.4.2.3. Импортируем пакеты



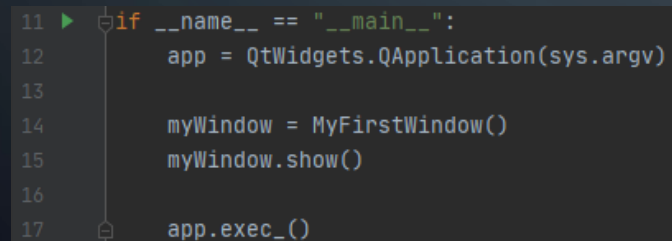
```
1 import sys  
2 from PySide2 import QtWidgets
```

3.4.2.4. Пишем код класса



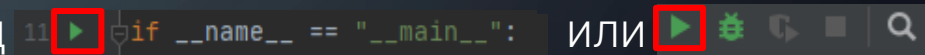
```
5 class MyFirstWindow(QtWidgets.QMainWindow):  
6  
7     def __init__(self, parent=None):  
8         QtWidgets.QMainWindow.__init__(self, parent)
```





3.4.2.5. Создаём приложение Qt

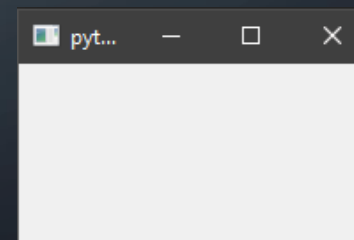


```
11 if __name__ == '__main__':  
12     app = QtWidgets.QApplication(sys.argv)  
13  
14     myWindow = MyFirstWindow()  
15     myWindow.show()  
16  
17     app.exec_()
```

3.4.2.6. Запускаем код



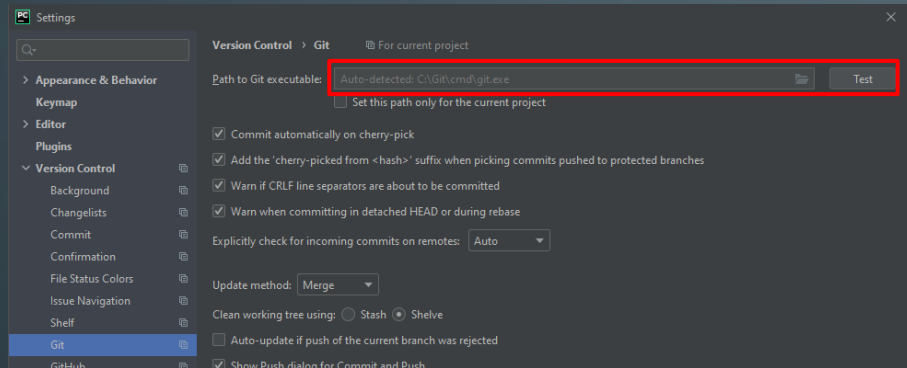
или    



3.5. НАСТРОЙКА GIT В ПРОЕКТЕ

3.5.1. Настройка в PyCharm

File -> Settings... -> Version Control -> Git



3.5.2. Инициализируем Git в проекте

```
(venv) F:\Python\Projects\newProject>git init
Initialized empty Git repository in F:/Python/Projects/newProject/.git/
```

3.5.3. Проверка статуса проекта

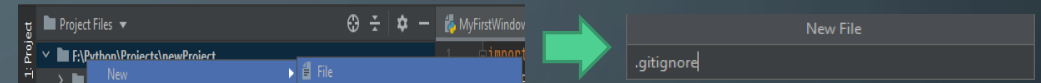
```
(venv) F:\Python\Projects\newProject>git status
On branch master

No commits yet

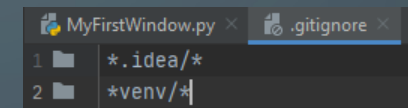
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .idea/
    MyFirstWindow.py
    venv/

nothing added to commit but untracked files present (use "git add" to track)
```

3.5.4. Создаём файл .gitignore



3.5.5. Редактируем .gitignore



3.5.6. Проверка статуса проекта

```
(venv) F:\Python\Projects\newProject>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    MyFirstWindow.py

nothing added to commit but untracked files present (use "git add" to track)
```

3.5.7. Создаём файл requirements.txt

```
(venv) F:\Python\Projects\newProject>pip freeze > requirements.txt
```


3.5. НАСТРОЙКА GIT В ПРОЕКТЕ (ПРОДОЛЖЕНИЕ)

3.5.8. Добавляем файлы для отслеживания и проверяем статус проекта

```
(venv) F:\Python\Projects\newProject>git add .

(venv) F:\Python\Projects\newProject>git status
On branch master

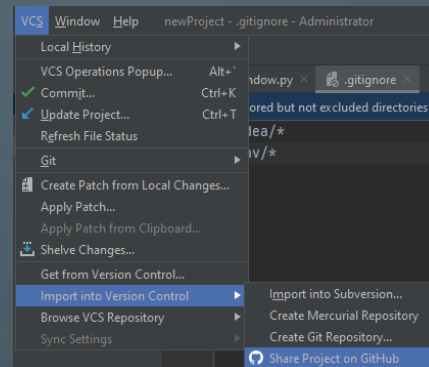
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   MyFirstWindow.py
        new file:   requirements.txt
```

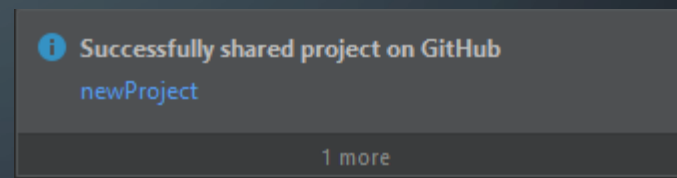
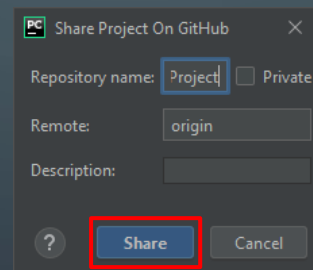
3.5.9. Делаем «снимок» проекта

```
(venv) F:\Python\Projects\newProject>git commit -m "init commit"
[master (root-commit) a554a83] init commit
3 files changed, 21 insertions(+)
create mode 100644 .gitignore
create mode 100644 MyFirstWindow.py
create mode 100644 requirements.txt
```

*3.5.10. Отправляем проект на GitHub



*3.5.11. Указываем параметры и отправляем



*3.5.12. Проект размещён

| | | | | |
|-----------------------|------------------|-------------|-----------------------|---------------|
| vladuxa23 init commit | | | a554a83 7 minutes ago | 1 commit |
| | .gitignore | init commit | | 7 minutes ago |
| | MyFirstWindow.py | init commit | | 7 minutes ago |
| | requirements.txt | init commit | | 7 minutes ago |

Спасибо за внимание!