

УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ АДМИНИСТРАЦИИ

ГОРОДСКОГО ОКРУГА БАЛАШИХА

МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДСКОГО ОКРУГА БАЛАШИХА «ГИМНАЗИЯ № 11»

(МБОУ «ГИМНАЗИЯ № 11»)

Индивидуальный проект

‘’Создание кастомных QR-кодов’’

Выполнила: Захарова Таисия Александровна

Учащаяся 10/04 класса

Руководитель:

Крылов Виталий Витальевич,

учитель первой кв. категории ГБОУ Школа №1373 г. Москва

**Балашиха**

**2025**

**Содержание**

[Введение 3](#_Toc180700829)

[Глава 1. Генерация простого QR-кода: 4](#_Toc180700830)

[1.1. Язык программирования Python 4](#_Toc180700831)

[1.2. Генерация простого QR-кодов средствами языка программирования Python. 4](#_Toc180700832)

[Глава 2. Кастомизация QR-кода 10](#_Toc180700833)

[2.1. Изменение цвета 10](#_Toc180700834)

[2.2. Добавление логотипа 10](#_Toc180700835)

[Глава 3. Превращение изображения в QR-код 11](#_Toc180700836)

[3.1. Превращение изображения в QR-код 11](#_Toc180700837)

[Глава 4. Генерация QR-кода в форме произвольной фигуры 12](#_Toc180700838)

[4.1. Создание QR-кода в форме сердца 12](#_Toc180700839)

[Приложение 1: Установка необходимых библиотек 13](#_Toc180700840)

[Приложение 2: Настройка рабочего пространства проекта 16](#_Toc180700841)

[Заключение 24](#_Toc180700842)

[Используемая литература 25](#_Toc180700843)

**Введение**

QR-коды (Quick Response codes) — это двухмерные штрих-коды, которые содержат закодированную информацию, которую можно быстро считать с помощью смартфонов и других устройств. За последние годы они стали популярным инструментом для передачи данных, от ссылок на веб-ресурсы до информации о продуктах. Использование кастомных QR-кодов позволяет не только передавать информацию, но и добавить уникальный визуальный стиль, что делает их привлекательными для брендинга и маркетинга. В данном проекте мы рассмотрим, как сгенерировать кастомные QR-коды на языке программирования Python, а также добавим возможности для изменения их дизайна.

**Цели и задачи проекта**

Цель проекта — изучить и разработать программное решение для генерации кастомных QR-кодов с использованием языка программирования Python.

**Задачи проекта:**

1. Изучить базовые принципы генерации QR-кодов.

2. Исследовать библиотеку Python для работы с QR-кодами.

3. Реализовать генерацию QR-кодов с кастомным дизайном.

4. Настроить возможность изменения цвета, размера и логотипов в QR-кодах.

5. Провести тестирование сгенерированных QR-кодов на совместимость с различными устройствами.

**Глава 1.** Генерация простого QR-кода

1.1. Язык программирования Python

**Python** — это **мультипарадигмальный высокоуровневый язык программирования общего назначения** с динамической строгой типизацией и автоматическим управлением памятью.[1]

**Особенности языка**:

* выделение блоков кода отступами;
* минималистичный синтаксис;
* ориентирован на повышение производительности разработчика, читаемости кода и его качества.

**Python применяется в следующих сферах**:

* **Веб-разработка.**   
   На нём обычно пишут серверные части приложений — это код, который запускается непосредственно на сервере, а не на устройствах пользователей.
* **Тестирование.**   
   Python часто используют для автоматизации тестирования.
* **Data Science**.   
   С его помощью пишут аналитические приложения.
* **Машинное обучение.**   
   Помогает создавать алгоритмы, которые среди массива входных данных обнаруживают знакомые шаблоны.

1.2. Генерация простого QR-кодов средствами языка программирования Python.

В этом пункте рассмотрим создание простейших Qr–кодов средствами программирования языка Python. Под простейшими Qr-кодами будет понимать Qr-код, который содержит базовые настройки (настройки из листинга 0 без изменений) из библиотеки Qrcode языка программирования Python.

Приведем пример участка кода на языке программирования Python, который отражает базовые настройки Qr-кода:

Листинг 0:ГенерацияQr-кода с помощью библиотеки qrcode.

qr0 = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)

**Описание структуры и параметров:**

* **qrcode.QRCode:**

Это класс, который используется для создания и настройки объекта QR-кода.

Класс позволяет задать параметры, которые определяют размеры, коррекцию ошибок, и внешний вид QR-кода. После создания объекта QR-кода можно добавлять данные (например, текст или URL) и генерировать окончательное изображение.

Параметры конструктора qrcode.QRCode:

* **version=1:**

Параметр version определяет размер QR-кода, то есть количество модулей (черных и белых квадратов) по каждой стороне.

Значение может быть от 1 до 40. Версия 1 — это наименьший QR-код, состоящий из 21x21 модуля, и с увеличением версии количество модулей увеличивается.

Более высокие значения версии позволяют кодировать больше данных.

error\_correction=qrcode.constants.ERROR\_CORRECT\_L:

* **error\_correction** определяет уровень коррекции ошибок, используемый QR-кодом.

У **error\_correction** есть допустимые возможные значения:

* ERROR\_CORRECT\_L: ~7% данных можно восстановить при повреждении.
* ERROR\_CORRECT\_M: ~15% данных можно восстановить (уровень по умолчанию).
* ERROR\_CORRECT\_Q: ~25% данных можно восстановить.
* ERROR\_CORRECT\_H: ~30% данных можно восстановить (наивысший уровень защиты).

Уровень коррекции ошибок влияет на плотность QR-кода. Высокий уровень коррекции позволяет восстановить данные, даже если часть QR-кода будет повреждена или закрыта, но при этом увеличивает размер кода.

box\_size=10:

* **Параметр box\_size** определяет размер одного модуля (квадрата) QR-кода в пикселях.

Например, при значении box\_size=10 каждый модуль будет 10x10 пикселей. Это влияет на общий размер QR-кода, но не на его содержимое.

Параметр полезен, если вы хотите изменить масштаб QR-кода, чтобы он был больше или меньше на изображении.

border=4:

* **Параметр border** задает ширину границы QR-кода (в количестве модулей).

Граница необходима, чтобы обеспечить правильное считывание QR-кода, так как сканеру нужно "видеть" контраст между началом QR-кода и фоном.

Стандартное значение — 4, это минимальная рекомендуемая ширина границы.

Далее создадим 5 Qr–кодов из 5 URL–адресов одновременно с помощью библиотеки Qrcode на основе параметров из литсинга 0, который упрощает и автоматизирует процесс создания Qr–кодов по следующему алгоритму:

1. Передача URL-адресов;
2. Указание места сохранения Qr–кодов;
3. Генерация Qr–кодов;
4. Передача ссылки в сегрегированный Qr–код;
5. Сборка Qr–кода на основе полученных данных;
6. Создание изображения из собранного Qr–кода;
7. Сохранение созданного изображения в соответствующую папку проекта.

Результат работы алгоритма можно увидеть в листинге 1.

Листинг 1: Генерация простых Qr–кодов из URL–адресов.

import qrcode  
  
*# Данные для кодирования*data0 = "http://yandex.ru"  
data1 = "http://ya.ru"  
data2 = "http://rbc.ru"  
data3 = "http://vc.ru"  
data4 = "https://inf-ege.sdamgia.ru/"  
data5 = "https://inf-oge.sdamgia.ru/"  
  
name\_qr\_code0 = 'img/qr\_0.png'  
name\_qr\_code1 = 'img/qr\_1.png'  
name\_qr\_code2 = 'img/qr\_2.png'  
name\_qr\_code3 = 'img/qr\_3.png'  
name\_qr\_code4 = 'img/qr\_4.png'  
name\_qr\_code5 = 'img/qr\_5.png'  
  
*# Генерация QR-кода*qr0 = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
  
qr1 = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
qr2 = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
  
qr3 = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
  
qr4 = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
  
qr5= qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
  
  
qr0.add\_data(data0)  
qr1.add\_data(data1)  
qr2.add\_data(data2)  
qr3.add\_data(data3)  
qr4.add\_data(data4)  
qr5.add\_data(data5)  
  
qr0.make(fit=True)  
qr1.make(fit=True)  
qr2.make(fit=True)  
qr3.make(fit=True)  
qr4.make(fit=True)  
qr5.make(fit=True)  
  
*# Создание и сохранение изображения*img0 = qr0.make\_image(fill='black', back\_color='white')  
img1 = qr1.make\_image(fill='black', back\_color='white')  
img2 = qr2.make\_image(fill='black', back\_color='white')  
img3 = qr3.make\_image(fill='black', back\_color='white')  
img4 = qr4.make\_image(fill='black', back\_color='white')  
img5 = qr5.make\_image(fill='black', back\_color='white')  
  
img0.save(name\_qr\_code0)  
img1.save(name\_qr\_code1)  
img2.save(name\_qr\_code2)  
img3.save(name\_qr\_code3)  
img4.save(name\_qr\_code4)  
img5.save(name\_qr\_code5)

1.2.Генерация цветных QR-кодов средствами языка программирования Python.

import qrcode  
  
*# Данные для кодирования*data = "https://example.com"  
  
*# Генерация QR-кода*qr = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
qr.add\_data(data)  
qr.make(fit=True)  
  
*# Создание и сохранение изображения*img = qr.make\_image(fill='blue', back\_color='yellow')  
img.save('img-color/qr\_code-color.png')

1.3. Генерация кастомных QR-кодов средствами языка программирования Python.

from MyQR import myqr  
  
myqr.run(  
 words="http://ege-drive.ru",  
 version=20, level="H",  
 picture="foto/img\_1.png",  
 colorized=True,  
 save\_name='img-from-img/samurai.png',  
 contrast=3.0,  
 brightness=10.0)  
*# Dynamic pictures should use gif format pictures  
# words parameter is to specify the content of the QR code  
# version parameter is the control side length, it is an int type, the range is 1-40, the larger the number, the larger the side length  
# level is the error correction level, the range is L, M, Q, H, increasing from left to right  
# picture parameter is to specify the name of the picture file to be used, the picture here is in the directory that this python file belongs to  
# colorized parameter is to specify that the generated QR code image is colored, if it is False or not set, the generated image is black and white  
# The contrast parameter is to set the contrast of the picture  
# The brightness parameter is to adjust the brightness of the picture*

1.4. Генерация кастомных QR-кодов с логотипом средствами языка программирования Python.

from MyQR import myqr as mq  
  
mq.run(words='https://ege-drive.ru',  
 version=6,  
 picture='logo/logo.png',  
 colorized=True,  
 save\_name='img-with-logo/ege-drive.png')

**Глава 2. Кастомизация QR-кода**

2.1. Изменение цвета

Теперь изменим цвет QR-кода, что сделает его более интересным:

Листинг 0:

img = qr.make\_image(fill='blue', back\_color='yellow')  
img.save('img-color/qr\_code-color.png')

Структура кода следующая:

1. Определение переменной img. Переменная img используется для хранения созданного изображения QR-кода.
2. Вызов функции qr.make\_image(). Функция qr.make\_image() вызывается для создания изображения QR-кода. Она принимает два аргумента: fill=‘blue’ (заполнение синим цветом) и back\_color=‘yellow’ (фон жёлтый).
3. Сохранение созданного изображения в файл ‘img-color/qr\_code-color.png’. Метод img.save() используется для сохранения созданного изображения в файл ‘img-color/qr\_code-color.png’.

Листинг 1:

import qrcode  
  
*# Данные для кодирования*data = "https://example.com"  
  
*# Генерация QR-кода*qr = qrcode.QRCode(  
 version=1,  
 error\_correction=qrcode.constants.ERROR\_CORRECT\_L,  
 box\_size=10,  
 border=4  
)  
qr.add\_data(data)  
qr.make(fit=True)  
  
*# Создание и сохранение изображения*img = qr.make\_image(fill='blue', back\_color='yellow')  
img.save('img-color/qr\_code-color.png')

Код создает цветной QR-код с синими элементами и желтым фоном, после чего сохраняет его в указанное местоположение.

2.2. Добавление логотипа

Чтобы сделать QR-код более персонализированным, добавим логотип:

Листинг 1:

from PIL import Image

*# Открываем сгенерированный QR-код*img = qr.make\_image(fill='black', back\_color='white')  
img = img.convert("RGB")  
  
*# Открываем изображение логотипа*logo = Image.open('logo.png')  
  
*# Изменяем размер логотипа под QR-код*box = (img.size[0] // 4, img.size[1] // 4)  
logo = logo.resize(box)  
  
*# Вставляем логотип в центр QR-кода*pos = ((img.size[0] - logo.size[0]) // 2, (img.size[1] - logo.size[1]) // 2)  
img.paste(logo, pos)  
  
*# Сохраняем QR-код с логотипом*img.save('qr\_with\_logo.png')

Давайте разберем представленный код на Python, который использует библиотеку Pillow (PIL) для работы с изображениями и генерации QR-кодов.

Структура кода:

1. Импорт необходимых библиотек: здесь мы импортируем класс Image из библиотеки Pillow, который позволяет работать с изображениями и их манипуляциями.

2. Создание QR-кода:

qr.make\_image(): Метод, который генерирует изображение QR-кода. Параметры:

- fill='black': Цвет, которым будут закрашены элементы QR-кода (в данном случае — черный).

- back\_color='white': Цвет фона QR-кода (в данном случае — белый).

- img.convert("RGB"): Преобразует изображение в режим RGB, что позволяет работать с цветами.

3. Открытие изображения логотипа: Этот код загружает изображение логотипа из файла 'logo.png' и создает объект logo.

4. Изменение размера логотипа:

- img.size[0]: Ширина QR-кода.

- img.size[1]: Высота QR-кода.

- box: Кортеж, представляющий новые размеры логотипа (в данном случае — 1/4 от ширины и высоты QR-кода).

- logo.resize(box): Изменяет размер логотипа до размеров box.

5. Вставка логотипа в центр QR-кода:

- pos: Кортеж, задающий координаты (x, y) для размещения логотипа, чтобы он оказался в центре QR-кода.

- img.paste(logo, pos): Вставляет логотип на QR-код, начиная с позиции pos.

6. Сохранение итогового изображения: этот код сохраняет финальное изображение QR-кода с логотипом в файл 'qrwithlogo.png'.

Параметры:

1. Цвета:

- fill: Определяет цвет, наполненный в QR-коде (по умолчанию черный).

- back\_color: Определяет цвет фона QR-кода (по умолчанию белый).

2. Размер логотипа: логотип изменяется до 1/4 размера QR-кода, что делает его достаточно крупным, но не доминирующим над QR-кодом.

3. Позиционирование: логотип размещается в центре QR-кода, что делает его визуально привлекательным.

4. Файловые операции:

- Image.open('logo.png'): Открывает изображение логотипа.

- img.save('qr\_with\_logo.png'): Сохраняет итоговое изображение QR-кода с логотипом.

Заключение:

Этот код позволяет эффективно генерировать QR-коды с интегрированным логотипом, обеспечивая наглядность и идентифицируемость вашего бренда или ссылки.

**Глава 3. Превращение изображения в QR-код**

3.1. Превращение изображения в QR-код

С помощью библиотеки Pillow мы можем закодировать изображение в QR-код. Для этого изображение будет преобразовано в битовый массив и закодировано в QR-код.

**Глава 4. Генерация QR-кода в форме произвольной фигуры**

4.1. Создание QR-кода в форме сердца**Приложение 1:**

Установка необходимых библиотек для генерации QR-кодов

Для начала работы с проектом необходимо предварительно установить следующие библиотеки:

1. Qrcode – библиотека для Python, позволяет генерировать QR-коды с помощью нескольких строк кода;
2. Pillow - библиотека для Python для работы с изображениями в Python;
3. Numpy - это открытая бесплатная **библиотека** для **Python**. Она ускоряет работу с многомерными массивами и матрицами, а также позволяет вычислять много высокоуровневых математических функций при работе с массивами данных;
4. Matplotlib – это библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой.

Установим библиотеку qrcode для Python в среде программирования PyCharm IDE, для введем в командной строке команду:

* pip install qrcode.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Результат выполнения установки библиотеки qrcode в среде программирования PyCharm IDE показан на рис. 1.

Установим библиотеку pillow для Python в среде программирования PyCharm IDE, для введем в командной строке команду:

* pip install pillow.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Результат выполнения установки библиотеки pillow в среде программирования PyCharm IDE показан на рис. 2.

Установим библиотеку numpy для Python в среде программирования PyCharm IDE, для введем в командной строке команду:

* pip install numpy.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Результат выполнения установки библиотеки numpy в среде программирования PyCharm IDE показан на рис. 3.

Установим библиотеку matplotlib для Python в среде программирования PyCharm IDE, для введем в командной строке команду:

* pip install matplotlib.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Результат выполнения установки библиотеки matplotlib в среде программирования PyCharm IDE показан на рис. 4.

**Приложение 2:**

Настройка рабочего пространства проекта.

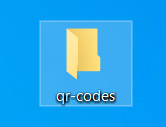


Рис. 1. Создадим папку

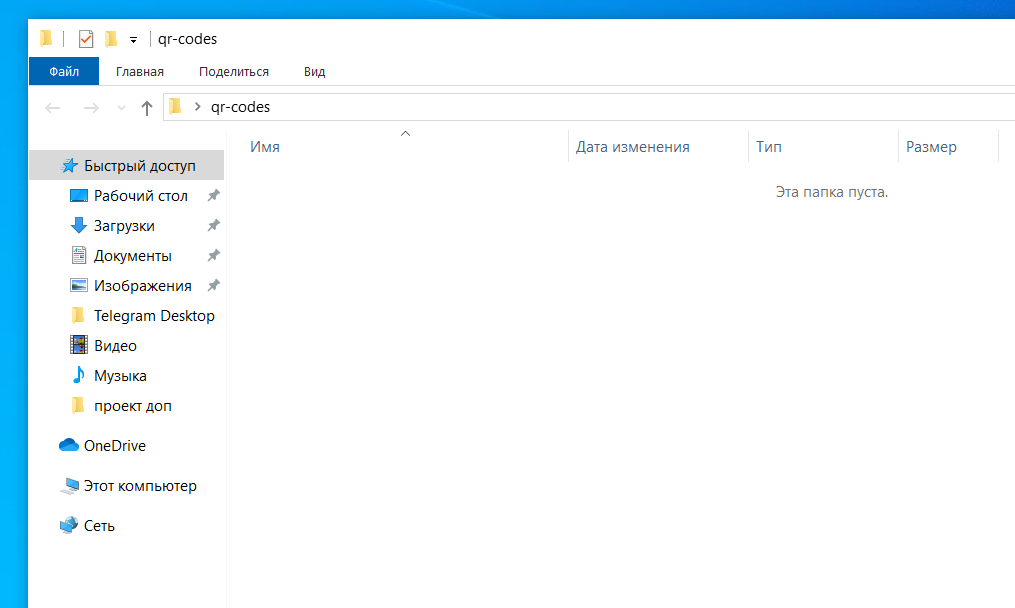


Рис. 2. Папка пустая внутри

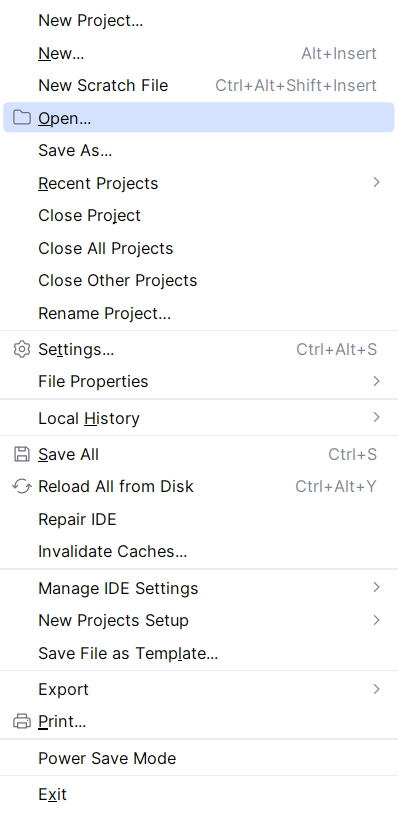


Рис. 3. Откроем среду программирования **PyCharm** **IDE** и выберем “Open”

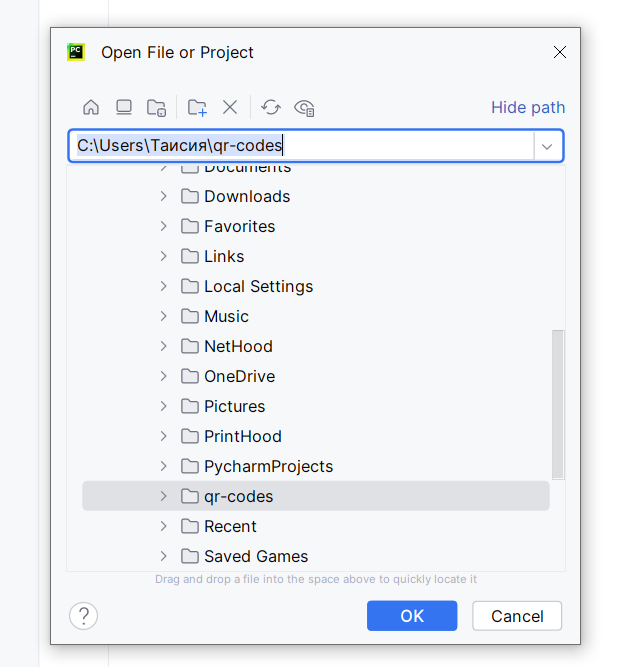
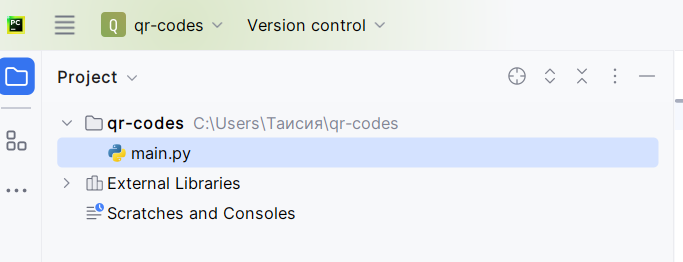


Рис. 4. Выберем папку qr-codes



На рис. 5 показан результат открытия папки

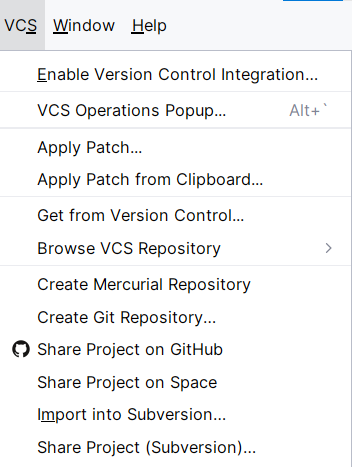


Рис. 6. Выбираем “VCS” и затем “Get from Version Control”

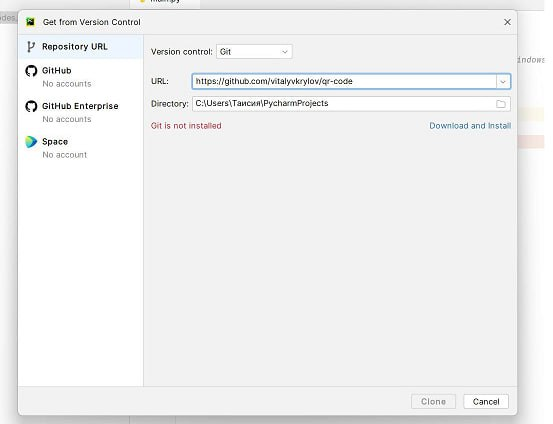


Рис. 7. Появится окно, Git не установлен

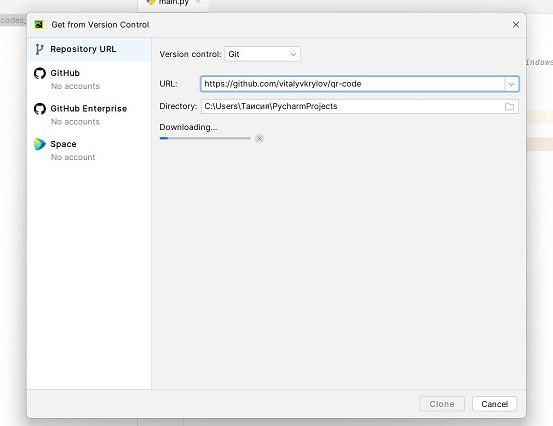


Рис. 8. Загрузим Git

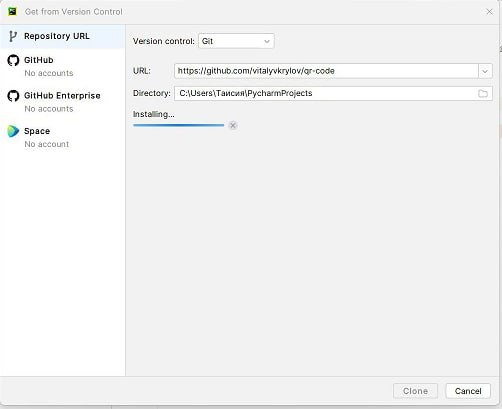


Рис. 9. Git устанавливается

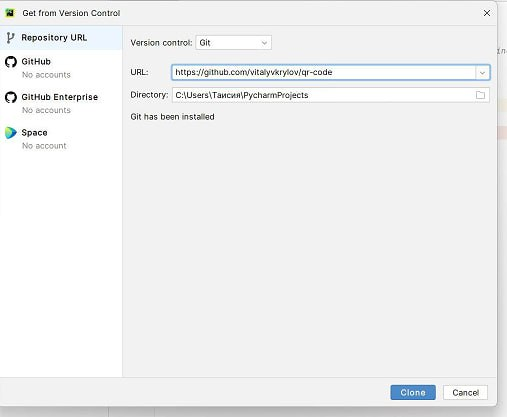


Рис 10. Git установлен, перейдём по введенной ссылке на проект на сайте **GitHub**

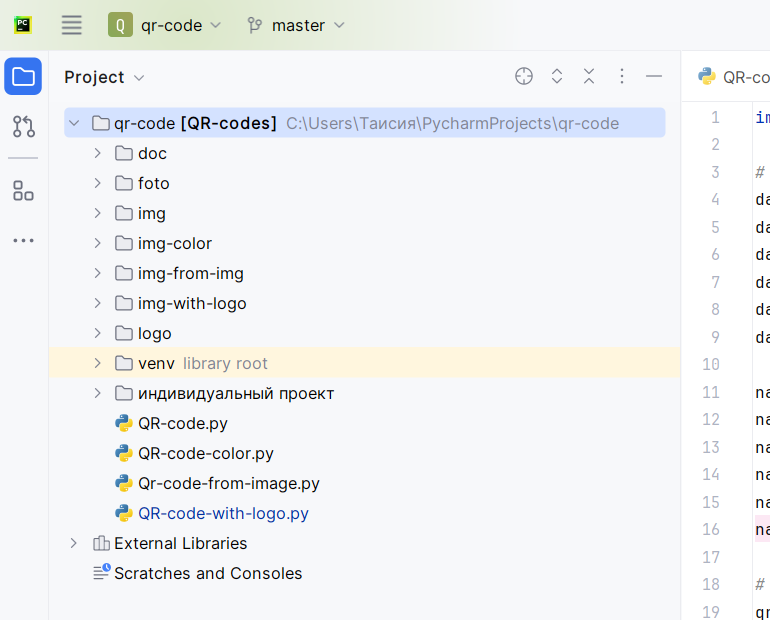


Рис. 11. Проект появился автоматически в PyCharm IDE

Модули **Git** языка Python – это распределенная система контроля версий. Система контроля версий – это набор инструментов, используемых для отслеживания прошлого набора файлов проектов.

Пользователи могут легко сохранить состояние файлов в любой момент, указав систему контроля версий **Git**. После этого пользователь может продолжить редактирование файлов проекта и сохранить проект в любом со-стоянии.

**GitHub** — это облачный хостинг репозиториев **Git**, то есть папок, где **Git** отслеживает изменения.

Эта платформа предлагает:

* безопасную авторизацию по токенам
* инструменты тестирования и анализа кода
* сервисы деплоя проектов — GitHub Actions.

В **GitHub** можно:

* подсматривать за чужим кодом в опенсорсных проектах,
* комментировать, копировать то, что понравилось,
* предлагать свои решения.

**Заключение**

В данном проекте был рассмотрен процесс генерации QR-кодов на языке программирования Python с использованием библиотек **qrcode**, **myqr**. Мы рассмотрели, как создать простой QR-код, а также как кастомизировать его, изменяя цвета, добавляя логотипы, создания qr-кода по изображению. Это позволяет использовать QR-коды не только для передачи информации, но и для маркетинговых и брендовых задач. Будущие улучшения могут включать генерацию динамических QR-кодов и оптимизацию для более сложных дизайнерских решений.

**Используемая литература:**

[1] - **https://ru.wikipedia.org/wiki/Python**