**Запреты**

1. **Запрещается использовать глобальные переменные без крайней необходимости.**  
   Глобальные переменные усложняют понимание кода и могут привести к ошибкам. Вместо этого используйте классы или передавайте данные через параметры функций.  
   **Пример запрещенного использования:**

cache = [] # Глобальная переменная

self.cache = {}

**Пример правильного использования:**

**class MainWindow(QMainWindow):**

cache: dict = {}

**Запрещается использовать "магические числа" (числовые константы без объяснения).**Все числовые значения должны быть заменены на именованные константы или переменные. **Пример запрещенного использования:**

if size >= 1024:

size = round(float(size) / 1024, 2)

index += 1

size = f'{size} {sizes[index]}'

**Пример правильного использования:**

Twopow10 = 1024

if size >= Twopow10:

size = round(float(size) / Twopow10, 2)

index += 1

size = f'{size} {sizes[index]}'

#### ****Требования****

1. **Все функции и методы должны иметь docstrings.**  
   Каждая функция должна быть документирована с использованием docstrings для описания её назначения, параметров и возвращаемых значений.  
   **Пример:**

def open\_item(self, item: QTreeWidgetItem):

Открывает файл в системе для дальнейших действий и отрисовывает его в интерфейсе

:param item: QTreeWidgetItem: открываемый файл

Self переменная для всего окна

""

# Логика функции

1. **Имена переменных, функций и классов должны быть осмысленными и соответствовать PEP 8.**Имена должны быть написаны в snake\_case для переменных и функций, и в PascalCase для классов.

**Пример:**

**class NewRemoteWindow(QDialog):**

**def new\_remote(self, edit\_mode: bool = False, remote\_name: str = None):**

**pass**

1. **Использование типизации (type hints) для функций и методов.  
   Типизация улучшает читаемость и помогает избежать ошибок.**

**Пример:**

**def open\_folder(self, remote\_name: str, path\_dir: str = ''): """**

**Открывает папку.**

**:param remote\_name: str тип стринг.**

**: param path\_dir: str тип стринг.**

**"""**

**# Логика функции**

#### ****Рекомендации****

* 1. **Рекомендуется использовать контекстные менеджеры для работы с ресурсами.**  
     Например, при работе с файлами или базами данных.  
     **Пример:**

**with open('file.txt', 'r') as file:**

**data = file.read()**

* 1. **Рекомендуется разделять код на модули и пакеты.**  
     Например, создать отдельные модули для моделей, сервисов и контроллеров.  
     **Пример структуры:**

**project/**

**│ ├── forms/**

**│ │ └── main\_window.ui**

**│ │ └── new\_remote\_window.ui**

**│ ├── main\_window.py**

**│ ├── main.py**

**│ ├── new\_remote\_window.py**

**│ ├── rclone.exe**

**│ ├── requirements.txt**

* 1. **Рекомендуется использовать логгирование вместо print для отладки.**  
     Логгирование позволяет гибко управлять уровнем вывода информации.  
     **Пример:**

**import logging**

**logging.basicConfig(level=logging.INFO)**

**logging.info("file created successfully!”)**

### ****Чек-лист для проверки корректности кода****

1. **Проверка соответствия PEP 8:**

* Используется ли snake\_case для переменных и функций?
* Используется ли PascalCase для классов?
* Соответствует ли длина строки 79 символам?

1. **Проверка docstrings:**

* Есть ли docstrings у всех функций и методов?
* Описаны ли параметры и возвращаемые значения?

1. **Проверка типизации:**

* Используются ли type hints для функций и методов?

1. **Проверка на "магические числа":**

* Все ли числовые константы заменены на именованные переменные?

1. **Проверка на использование глобальных переменных:**

* Отсутствуют ли глобальные переменные (кроме констант)?

1. **Проверка структуры проекта:**

* Разделен ли код на модули и пакеты?
* Соответствует ли структура проекта логике приложения?

1. **Проверка логгирования:**

* Используется ли логгирование вместо print для отладки?

1. **Проверка обработки ошибок:**

* Есть ли обработка исключений в критических местах?

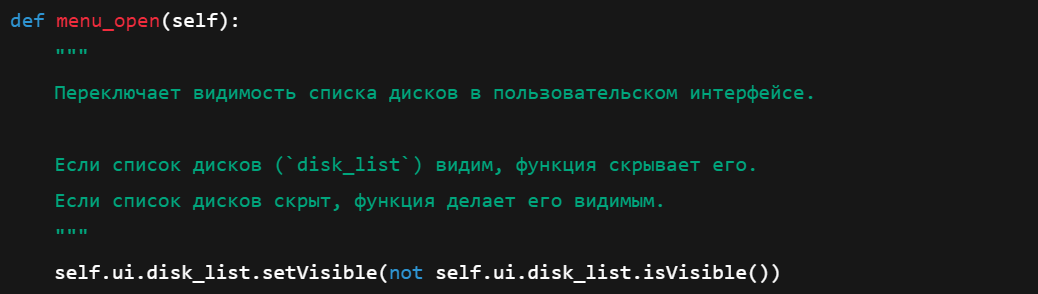
1. **Проверка читаемости кода:**

* Используются ли meaningful names (осмысленные имена)?
* Соответствует ли код принципам DRY (Don't Repeat Yourself)?

1. **Проверка тестов:**

* Есть ли unit-тесты для ключевых функций и методов?

### ****Пример оформления кода****

****