1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности
5. **Кафедра «Высшая школа кибербезопасности»**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Управление ФС/реестром посредством CLI-приложения на ЯП Python**»
2. по дисциплине «Цифровая культура»
3. Выполнил
4. студент гр. 5131001/30003 Шевчук Н.Е.

<*подпись*>

Руководитель

ассистент ВШК ИКНК Иванов М.С.

<*подпись*>

1. Санкт-Петербург
2. 2025

**Оглавление**

[1 Цель работы 3](#_Toc191936671)

[2 Задачи 3](#_Toc191936672)

[3 Ход работы 3](#_Toc191936673)

[3.1 Функции работы с файловой системой 3](#_Toc191936674)

[3.2 Функции работы с реестром 6](#_Toc191936675)

[3.3 Обработка входных параметров 8](#_Toc191936676)

[4 Ответы на контрольные вопросы 9](#_Toc191936677)

[5 Вывод 10](#_Toc191936678)

[Приложение 12](#_Toc191936679)

# Цель работы

Получить навыки управления файловой системой и реестром с использованием языка программирования Python.

# Задачи

1. Реализовать программу для работы с файловой системой и реестром операционной системы Windows.
2. Реализовать взаимодействие с командами через аргументы командной строки.

# Ход работы

Для создания консольного приложения сначала были подключены все модули, указанные в задании: argsparse, os, shutil, winreg.

## Функции работы с файловой системой

Были реализованы базовые функции работы с файловой системой. Для обработки ошибок использовался блок try/except.

1. Создание файла. С примером создания файла через командную строку можно ознакомиться на рисунке 1. Результат примера создания файла представлен на рисунке 2.

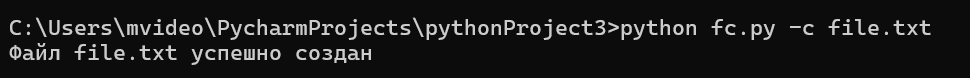


Рисунок 1. Пример создания файла через командную строку

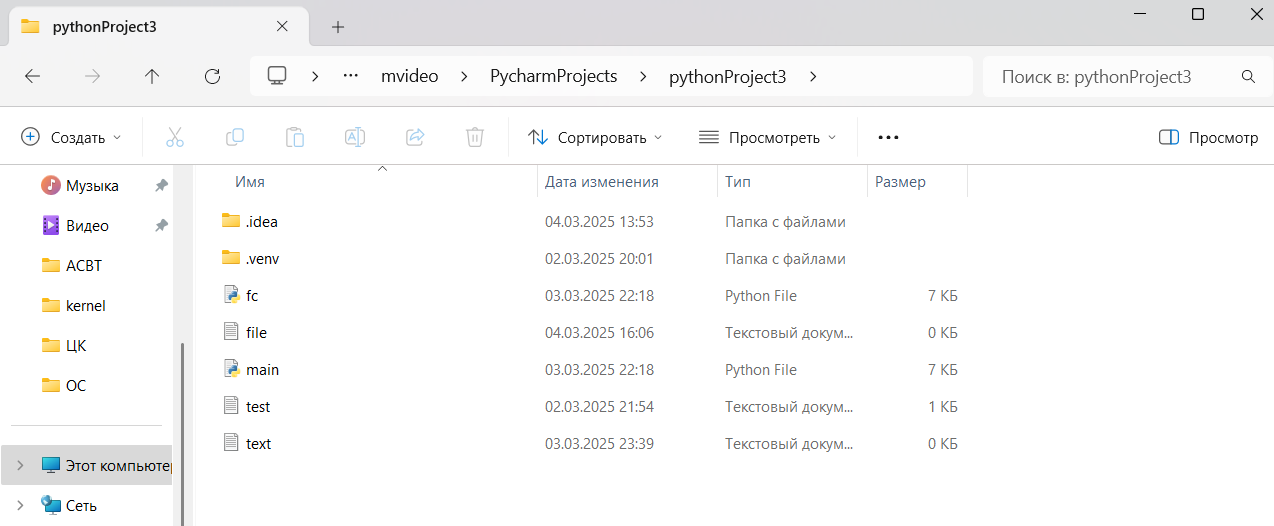


Рисунок 2. Результат создания файла

1. Удаление файла. С примером удаления файла через командную строку можно ознакомиться на рисунке 3. Результат примера удаления файла представлен на рисунке 4.

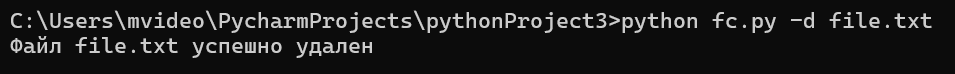


Рисунок 3. Пример удаления файла через командную строку

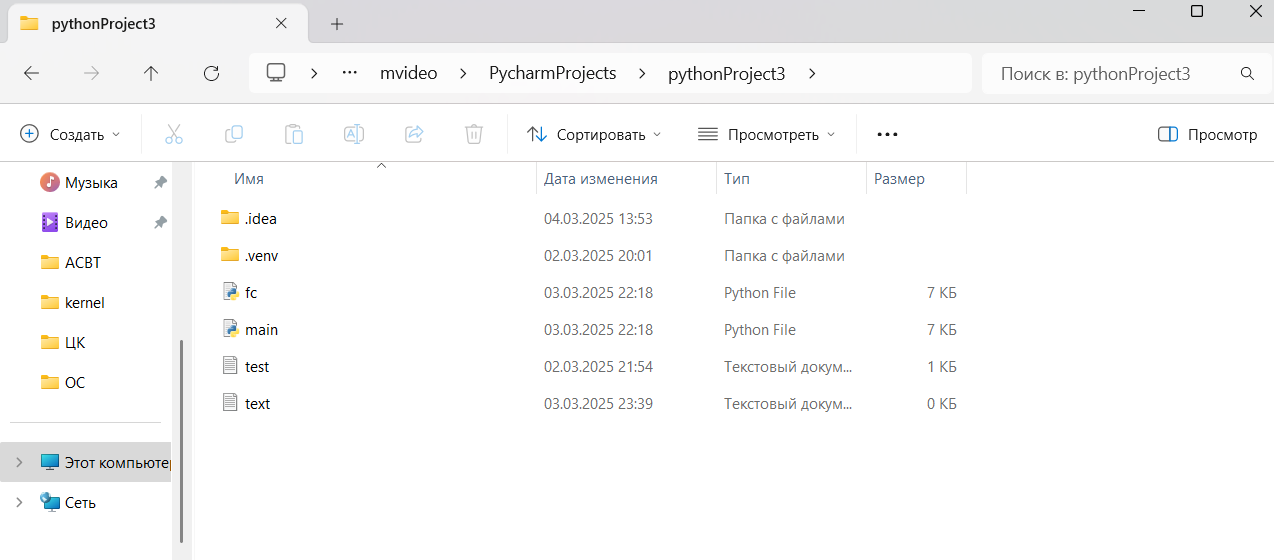


Рисунок 4. Результат удаления файла

1. Запись в файл. С примером записи в файл через командную строку можно ознакомиться на рисунке 5. Результат примера записи в файл представлен на рисунке 6.

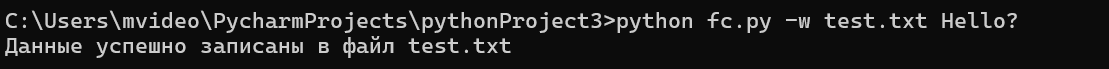


Рисунок 5. Пример записи в файл через командную строку

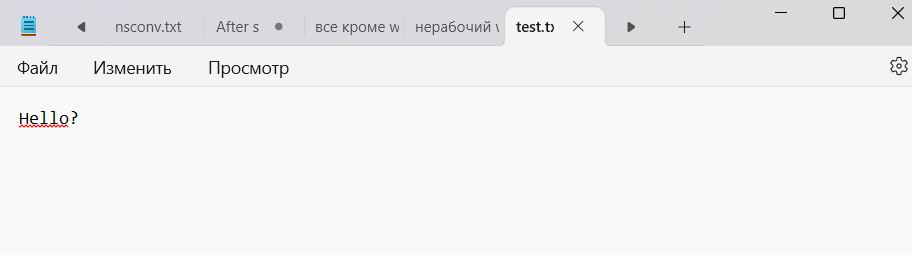


Рисунок 6. Результат записи в файл

1. Чтение из файла. С примером чтения файла через командную строку и результатом чтения можно ознакомиться на рисунке 7.

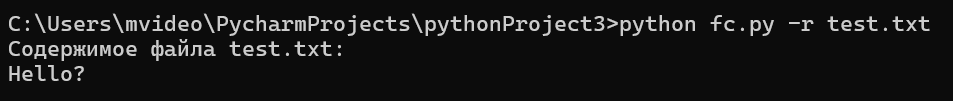


Рисунок 7. Результат чтения из файла

1. Копирование файла из одной директории в другую. С примером копирования файла через командную строку можно ознакомиться на рисунке 8. Результат примера копирования файла представлен на рисунке 9.

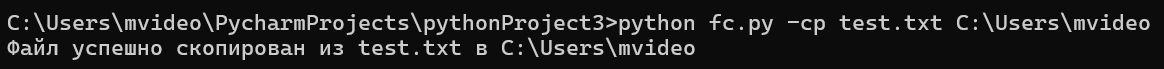


Рисунок 8. Пример копирования файла через командную строку

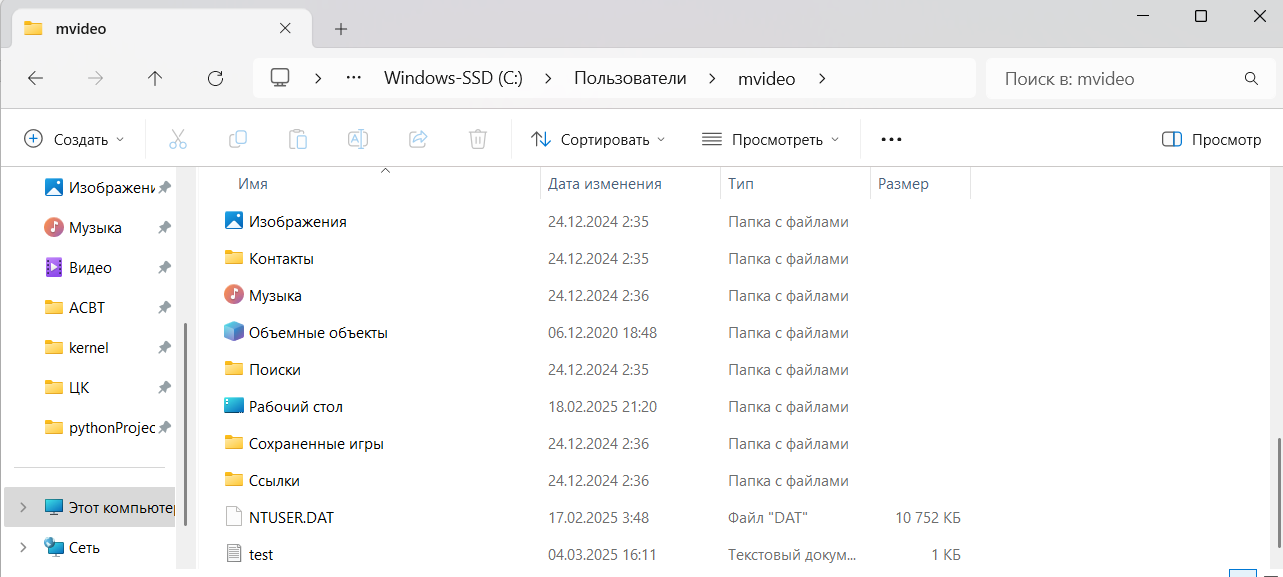


Рисунок 9. Результат копирования файла

1. Переименование файла. С примером переименования файла через командную строку можно ознакомиться на рисунке 10. Результат примера переименования файла представлен на рисунке 11.

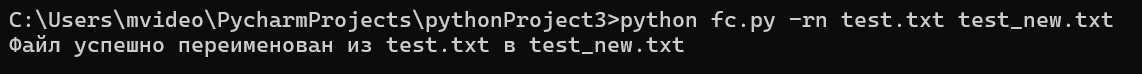


Рисунок 10. Пример переименования файла через командную строку

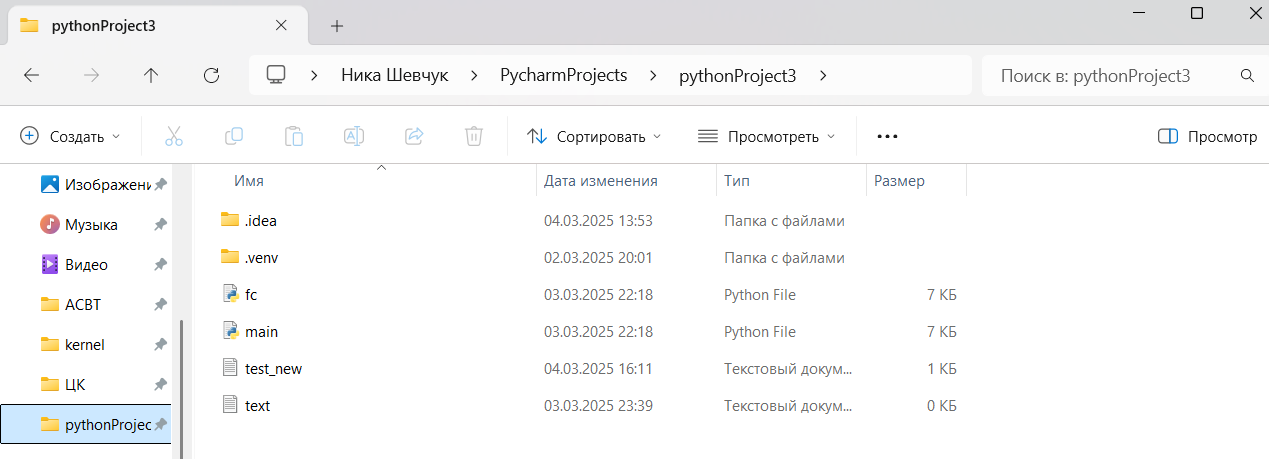


Рисунок 11. Результат переименования файла

## Функции работы с реестром

Реестр Windows — иерархически построенная база данных параметров и настроек в большинстве операционных систем семейства Microsoft Windows.

Сначала был прописан словарь для сопоставления сокращений корневых регистров с их объектами в модуле winger. Затем были реализованы следующие функции работы с реестром:

1. Создание ключа. С примером создания ключа через командную строку можно ознакомиться на рисунке 12. Результат создания ключа представлен на рисунке 13.

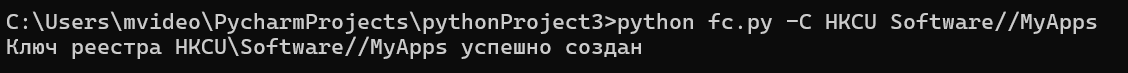


Рисунок 12. Пример создания ключа через командную строку

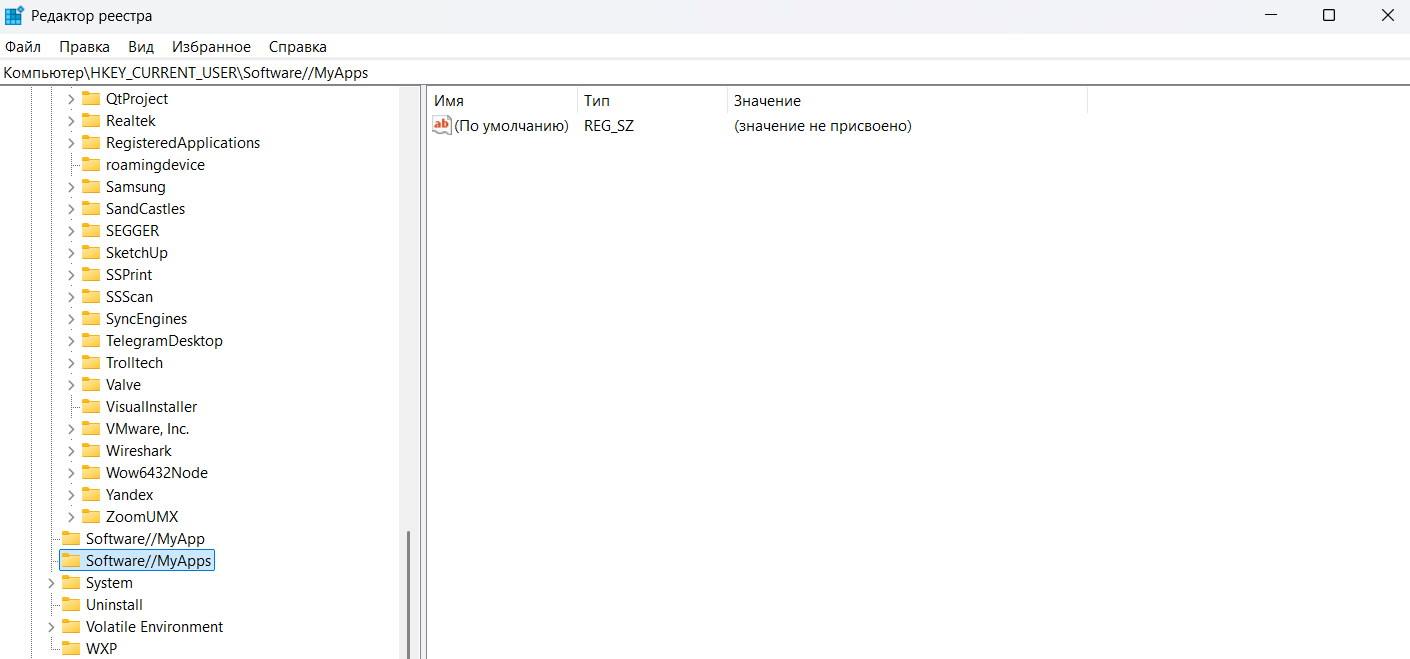


Рисунок 13. Результат создания ключа

1. Удаление ключа. С примером удаления ключа через командную строку можно ознакомиться на рисунке 14. Результат удаления ключа представлен на рисунке 15.

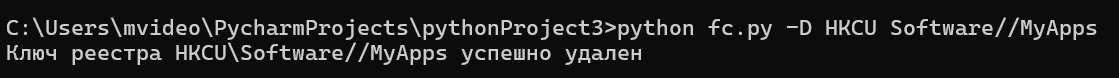


Рисунок 14. Пример удаления ключа через командную строку

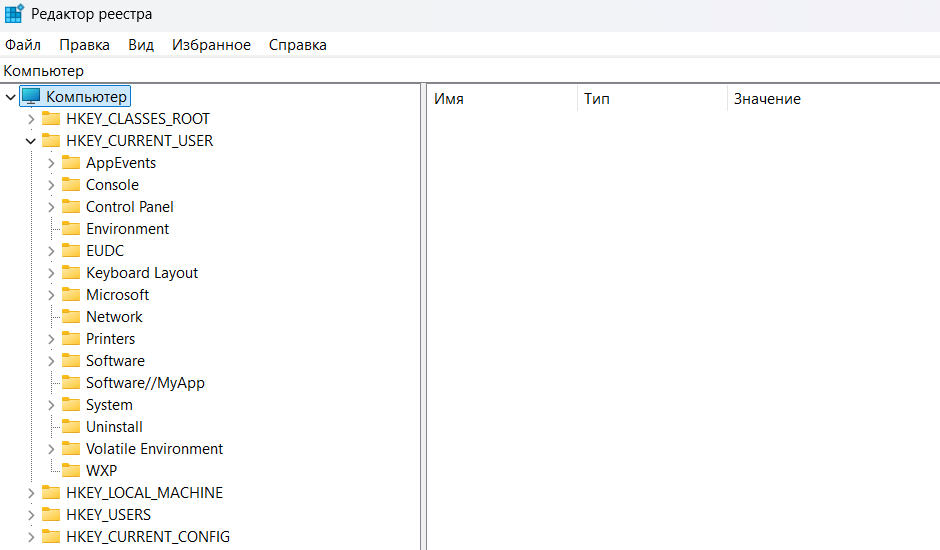


Рисунок 15. Результат удаления ключа

1. Запись значения в ключ. С примером записи значения в ключ через командную строку можно ознакомиться на рисунке 16. Результат записи значения в ключ представлен на рисунке 17.

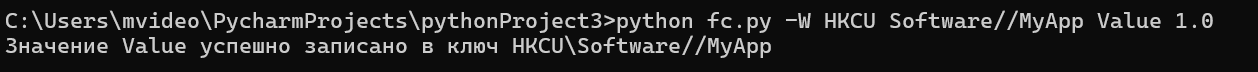


Рисунок 16. Пример записи значения в ключ через командную строку

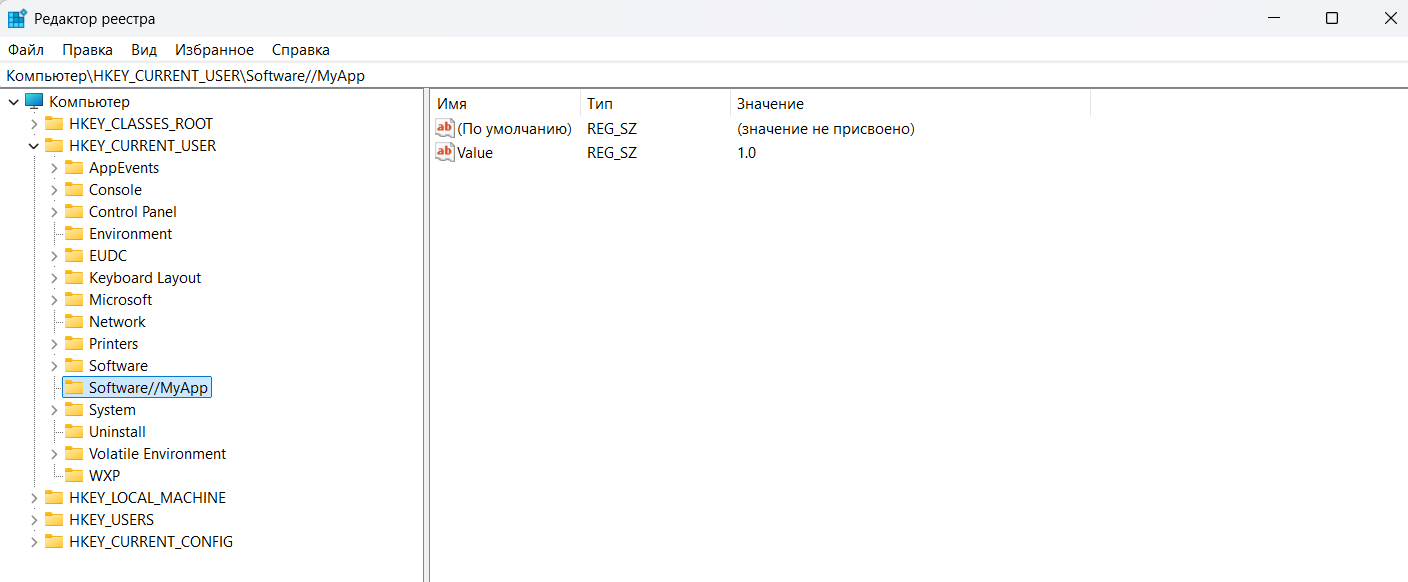


Рисунок 17. Результат записи значения в ключ

## Обработка входных параметров

Обработка входных параметров должна производиться с помощью модуля argparse командной строкой. Для этого сначала создаем объект парсера аргументов командной строки, затем с помощью метода add\_argument() добавляем опции аргументов командной строки. После этого с помощью метода parse\_args() сохраняем аргументы командной строки в объект args (с атрибутами, соответствующими именам опций). Затем системой из if находим необходимую опцию и вызываем соответствующую функцию.

В результате получен набор опций аргументов командной строки для различных функций. С полным набором опций аргументов командной строки можно ознакомиться на рисунке 18.

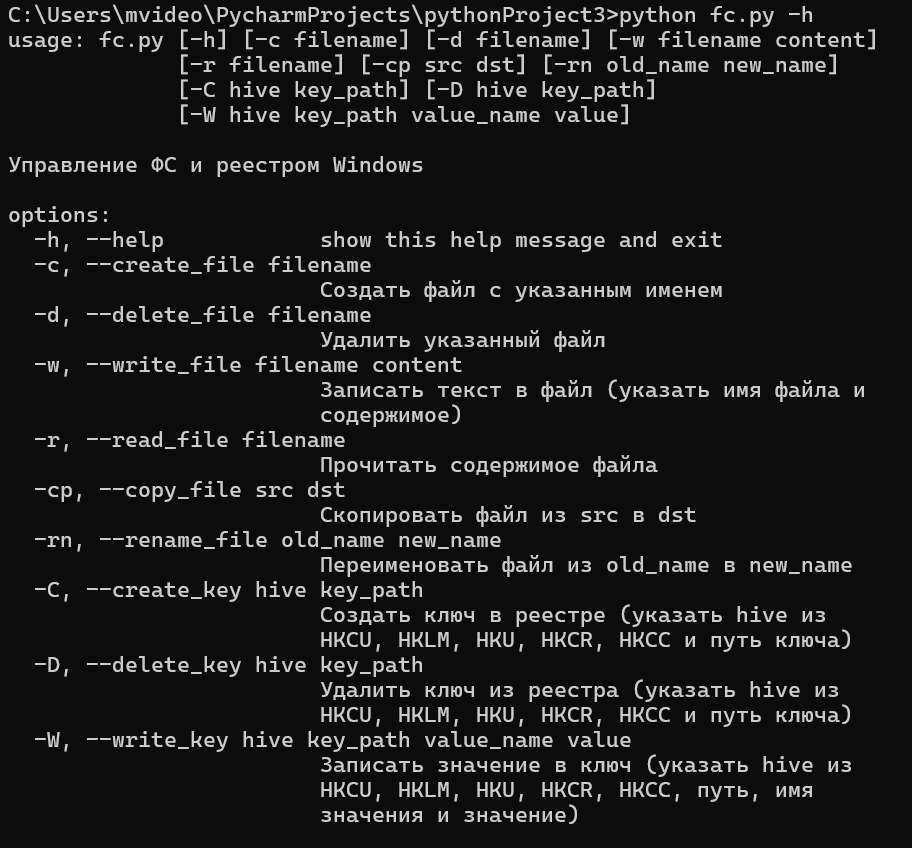


Рисунок 18. Опции аргументов командной строки

# Ответы на контрольные вопросы

1. *Чем компилируемый язык программирования отличается от интерпретируемого? Приведите примеры языков программирования для обоих типов.*

Компилируемые – язык программирования, исходный код которых преобразуется компилятором в машинный код.

Интерпретируемые – язык программирования, в которых операторы программы друг за другом отдельно транслируются и сразу выполняются (интерпретируются) с помощью программы-интерпретатора.

Примеры компилируемых языков: C/C++/Go. Примеры интерпретируемых языков: Ruby, Python и JavaScript.

1. *Каким образом могут быть считаны входные данные для консольного приложения помимо использования модуля argparse?*

Модуль argparse построен на модуле sys. Через команду sys.argv можно получить командную строку целиком, она будет не разделена на аргументы, это нужно будет делать вручную. Также возможно использование других модулей, например, optparse или click.

1. *В чем разница открытия файла конструкцией:*

*fd = open(‘test.txt’, ‘r’)*

*…*

*fd.close()*

*от конструкции:*

*with open(‘test.txt’, ‘r’) as fd:*

*…*

Без использования with open() необходимо использовать функцию close() после функции open() для закрытия файла после использования. Конструкция позволяет интерпретатору закрыть файл самому.

1. *Каким образом могут быть установлены модули для Python?*

Подключение модуля в конкретном файле происходит через конструкцию import. Установка самих модулей может производиться через скрипт get\_pip.py, который следует предварительно скачать. Pip – одно из программ, с помощью которой можно установить другие модули. Начиная с Python 3.4, он по умолчанию включен в установщики двоичных файлов Python. Его управление реализуется через терминал.

# Вывод

В ходе лабораторной работы были получены навыки управления файловой системой и реестром с использованием языка программирования Python. Были изучены методы модулей argparse, os, shutil, winreg, а также получены навыки работы с командной строкой.

# Приложение

Листинг исходного кода:

import argparse *# Модуль для обработки аргументов командной строки*import os *# Модуль для работы с файловой системой*import shutil *# Модуль для операций копирования файлов*import winreg *# Модуль для работы с реестром Windows  
  
  
# Словарь для сопоставления текстовых названий корневых разделов реестра с их объектами в winreg*REGISTRY\_HIVES = {  
 'HKCU': winreg.HKEY\_CURRENT\_USER,  
 'HKLM': winreg.HKEY\_LOCAL\_MACHINE,  
 'HKU': winreg.HKEY\_USERS,  
 'HKCR': winreg.HKEY\_CLASSES\_ROOT,  
 'HKCC': winreg.HKEY\_CURRENT\_CONFIG  
}  
  
  
*# Функции для работы с файловой системой*def create\_file(filename):  
 try:  
 directory = os.path.dirname(filename)  
 if directory and not os.path.exists(directory):  
 os.makedirs(directory)  
 with open(filename, 'w') as f:  
 print(f"Файл {filename} успешно создан")  
 except Exception as e:  
 print(f"Ошибка при создании файла: {e}")

def delete\_file(filename):  
 try:  
 os.remove(filename)  
 print(f"Файл {filename} успешно удален")  
 except Exception as e:  
 print(f"Ошибка при удалении файла: {e}")  
  
  
def write\_file(filename, content):  
 try:  
 with open(filename, 'w', encoding='utf-8') as f:  
 f.write(content + '\n')  
 print(f"Данные успешно записаны в файл {filename}")  
 except Exception as e:  
 print(f"Ошибка при записи в файл: {e}")  
  
  
def read\_file(filename):  
 try:  
 with open(filename, 'r', encoding='utf-8') as f:  
 content = f.read()  
 print(f"Содержимое файла {filename}:\n{content}")  
 except Exception as e:  
 print(f"Ошибка при чтении файла: {e}")  
  
  
def copy\_file(src, dst):  
 try:  
 shutil.copy2(src, dst)  
 print(f"Файл успешно скопирован из {src} в {dst}")  
 except Exception as e:  
 print(f"Ошибка при копировании файла: {e}")  
  
  
def rename\_file(old\_name, new\_name):  
 try:  
 os.rename(old\_name, new\_name)  
 print(f"Файл успешно переименован из {old\_name} в {new\_name}")  
 except Exception as e:  
 print(f"Ошибка при переименовании файла: {e}")  
  
  
*# Функции для работы с реестром*def create\_registry\_key(hive, key\_path):  
 try:  
 key = winreg.CreateKey(REGISTRY\_HIVES[hive], key\_path)  
 winreg.CloseKey(key)  
 print(f"Ключ реестра {hive}\\{key\_path} успешно создан")  
 except Exception as e:  
 print(f"Ошибка при создании ключа реестра: {e}")  
  
  
def delete\_registry\_key(hive, key\_path):  
 try:  
 winreg.DeleteKey(REGISTRY\_HIVES[hive], key\_path)  
 print(f"Ключ реестра {hive}\\{key\_path} успешно удален")  
 except Exception as e:  
 print(f"Ошибка при удалении ключа реестра: {e}")  
  
  
def write\_registry\_value(hive, key\_path, value\_name, value):  
 try:  
 key = winreg.OpenKey(REGISTRY\_HIVES[hive], key\_path, 0, winreg.KEY\_SET\_VALUE)  
 winreg.SetValueEx(key, value\_name, 0, winreg.REG\_SZ, value)  
 winreg.CloseKey(key)  
 print(f"Значение {value\_name} успешно записано в ключ {hive}\\{key\_path}")  
 except Exception as e:  
 print(f"Ошибка при записи значения в реестр: {e}")  
  
def validate\_hive(hive, option):  
 if hive not in REGISTRY\_HIVES:  
 parser.error(  
 f"argument {option}: invalid hive value: '{hive}' (choose from {', '.join(REGISTRY\_HIVES.keys())})")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 parser = argparse.ArgumentParser(description='Управление ФС и реестром Windows')  
  
 *# Операции с файловой системой* parser.add\_argument('-c', '--create\_file', metavar='filename', help='Создать файл с указанным именем')  
 parser.add\_argument('-d', '--delete\_file', metavar='filename', help='Удалить указанный файл')  
 parser.add\_argument('-w', '--write\_file', nargs=2, metavar=('filename', 'content'), help='Записать текст в файл (указать имя файла и содержимое)')  
 parser.add\_argument('-r', '--read\_file', metavar='filename', help='Прочитать содержимое файла')  
 parser.add\_argument('-cp', '--copy\_file', nargs=2, metavar=('src', 'dst'), help='Скопировать файл из src в dst')  
 parser.add\_argument('-rn', '--rename\_file', nargs=2, metavar=('old\_name', 'new\_name'), help='Переименовать файл из old\_name в new\_name')  
  
 *# Операции с реестром* parser.add\_argument('-C', '--create\_key', nargs=2, metavar=('hive', 'key\_path'), help='Создать ключ в реестре (указать hive из HKCU, HKLM, HKU, HKCR, HKCC и путь ключа)')  
 parser.add\_argument('-D', '--delete\_key', nargs=2, metavar=('hive', 'key\_path'), help='Удалить ключ из реестра (указать hive из HKCU, HKLM, HKU, HKCR, HKCC и путь ключа)')  
 parser.add\_argument('-W', '--write\_key', nargs=4, metavar=('hive', 'key\_path', 'value\_name', 'value'), help='Записать значение в ключ (указать hive из HKCU, HKLM, HKU, HKCR, HKCC, путь, имя значения и значение)')  
  
 args = parser.parse\_args()  
  
 if args.create\_file:  
 create\_file(args.create\_file)  
 elif args.delete\_file:  
 delete\_file(args.delete\_file)  
 elif args.write\_file:  
 write\_file(args.write\_file[0], args.write\_file[1])  
 elif args.read\_file:  
 read\_file(args.read\_file)  
 elif args.copy\_file:  
 copy\_file(args.copy\_file[0], args.copy\_file[1])  
 elif args.rename\_file:  
 rename\_file(args.rename\_file[0], args.rename\_file[1])  
 elif args.create\_key:  
 validate\_hive(args.create\_key[0], '-C/--create\_key')  
 create\_registry\_key(args.create\_key[0], args.create\_key[1])  
 elif args.delete\_key:  
 validate\_hive(args.delete\_key[0], '-D/--delete\_key')  
 delete\_registry\_key(args.delete\_key[0], args.delete\_key[1])  
 elif args.write\_key:  
 validate\_hive(args.write\_key[0], '-W/--write\_key')  
 write\_registry\_value(args.write\_key[0], args.write\_key[1], args.write\_key[2], args.write\_key[3])  
 else:  
 parser.print\_help()