

Работа с командной строкой. Linux, Windows

Пересунько Евгения Олеговна, преподаватель МГТУ им. Н.Э.
Баумана

Сегодня на занятии

- История появления терминалов в операционных системах
- Особенности работы с командной строкой в Linux и Windows
- Основные возможности командной строки Windows
- Разработка программ с интерфейсом командной строки

О себе



Пересунько Евгения Олеговна, преподаватель МГТУ им. Н.Э. Баумана

- стипендиат Правительства РФ и Президента РФ,
- лауреат Премии Главы города молодым талантам,
- лауреат стипендии имени академика М. Ф. Решетнева,

- веду курсы «Теория вычислительного обучения», «Нейронные сети и их приложения», «Разработка и анализ требований» в Сибирском федеральном университете

- преподаватель программы профессиональной переподготовки Data Science в МГТУ им. Н.Э. Баумана.
- Machine Learning Engineer (ООО «ИСС Арт»)

История терминала в операционных системах

- В 80-х гг. пользователям ОС Unix нужно было взаимодействовать с системой.
- Самый простой способ – использование команд. Пользователь вводит команду, система возвращает ответ.
- Подобный способ ввода использовался во многих ОС, в том числе DOS и OS/2 от Apple, пока не был придуман графический интерфейс (*англ.* graphical user interface, GUI)
- **Терминал** – это окружение, где можно вводить команды и получать на них ответ. Может быть физический терминал или терминал на компьютере.

Что такое командная строка?

Командная строка – это инструмент для управления операционной системой компьютера с помощью ряда зарезервированных команд и набора символов

Управление происходит с помощью **внутренних** и **внешних** команд:

- внутренние команды — команды, встроенные в операционную систему;
- внешние команды — программы, которые пользователь устанавливает на компьютер

Работа с командной строкой

Плюсы:

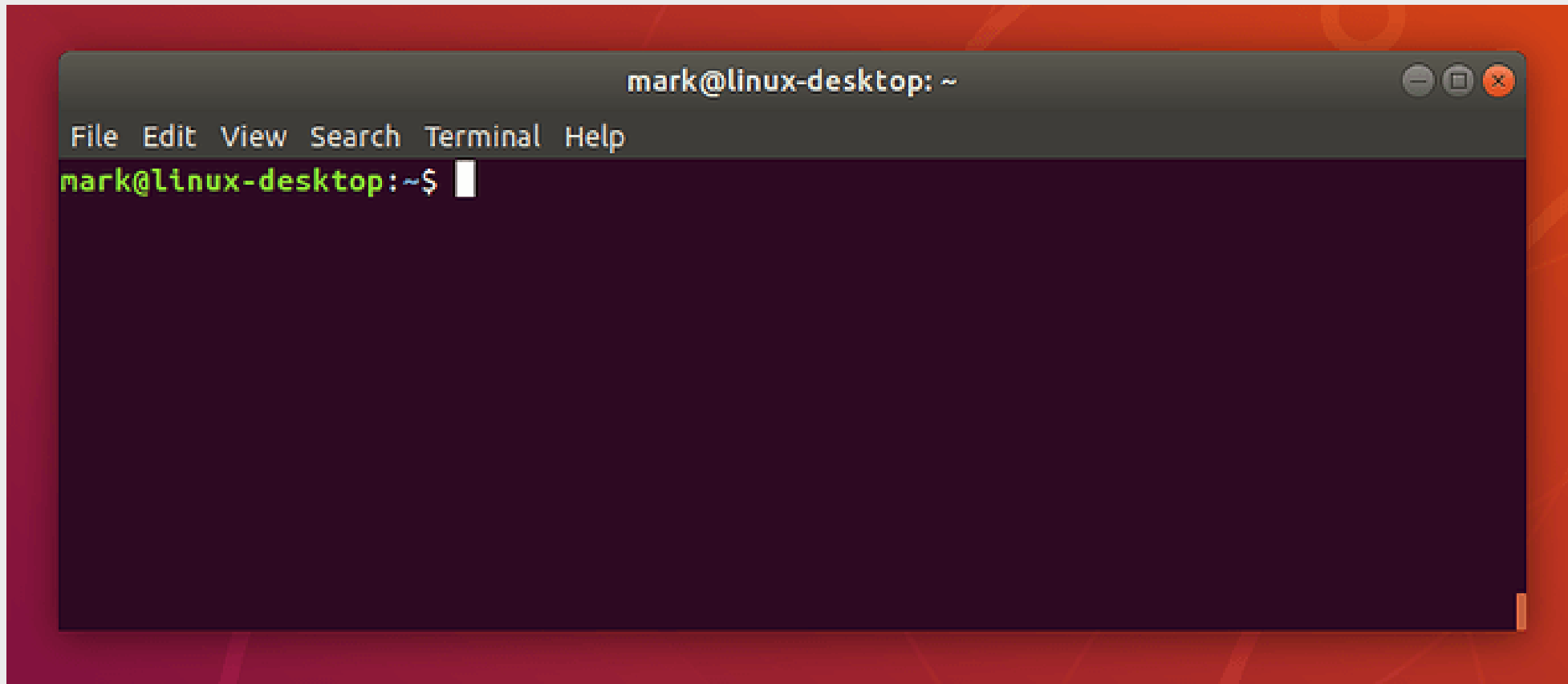
- Низкая ресурсоемкость по сравнению с приложениями с графическим интерфейсом
- Нет необходимости в использовании мышки
- Можно автоматизировать процессы
- Удаленный доступ

Минусы:

- Отсутствие привычного графического интерфейса
- Необходимость помнить большое число команд и ориентироваться в них

Терминал Linux (Ubuntu)

Терминал Linux может быть открыт как в текстовом режиме Linux, так и в графическом режиме окна терминала.



Запуск текстового терминала Linux

Система инициализации по умолчанию создает 12 виртуальных терминалов.

В одном из терминалов (*обычно седьмом*) запущена графическая оболочка Linux, но все другие могут быть свободно использованы.

Для переключения между терминалами можно использовать сочетания **Ctrl+Alt+F***

*(*в зависимости от номера терминала клавиши F1-F12).*

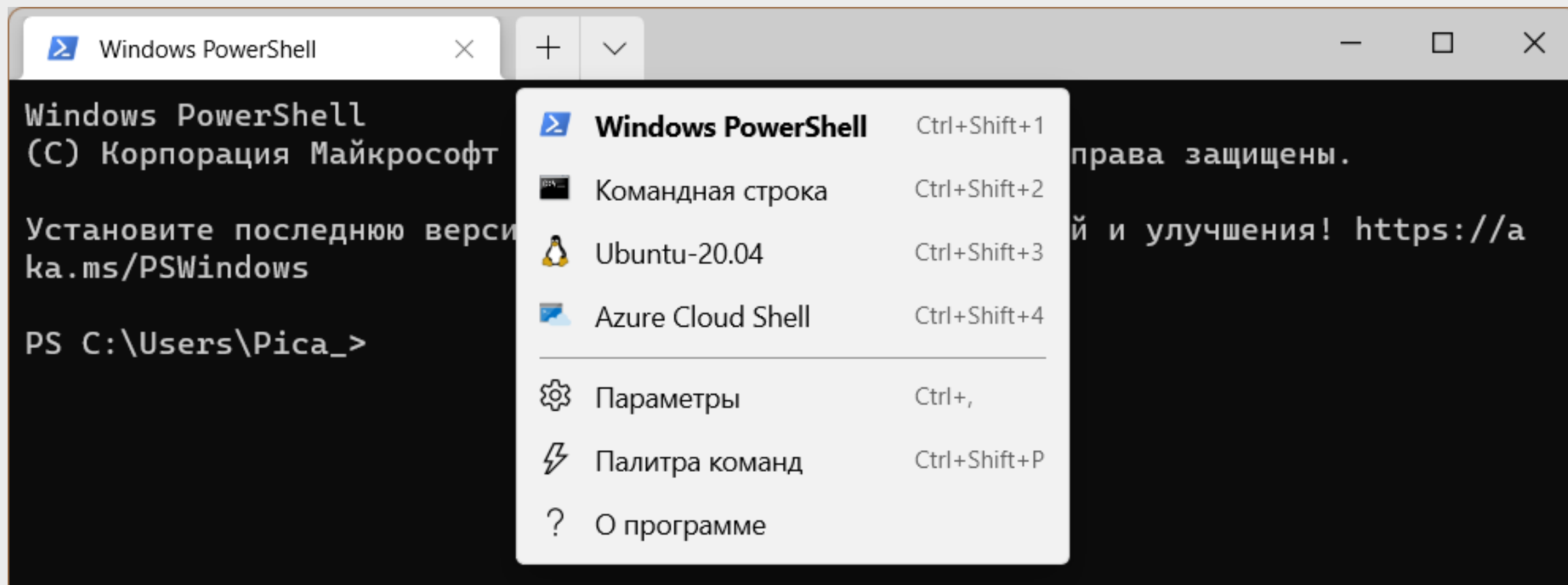
Для авторизации нужно будет ввести логин и пароль.

Запуск терминала Linux с графическим интерфейсом

- Второй способ позволяет открыть виртуальный терминал прямо в графическом интерфейсе с помощью эмулятора терминала. Эмулятор терминала Linux работает с файлами в каталоге `/dev/pts/*` и еще называется псевдотерминалом.
- В Ubuntu вы можете запустить терминал Linux нажав сочетание клавиш **Ctrl+Alt+T**

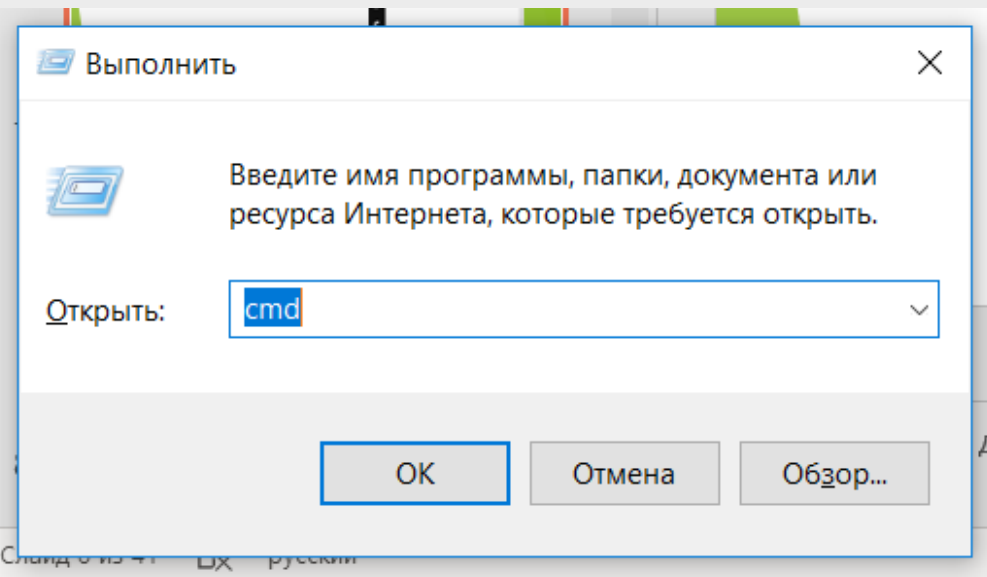
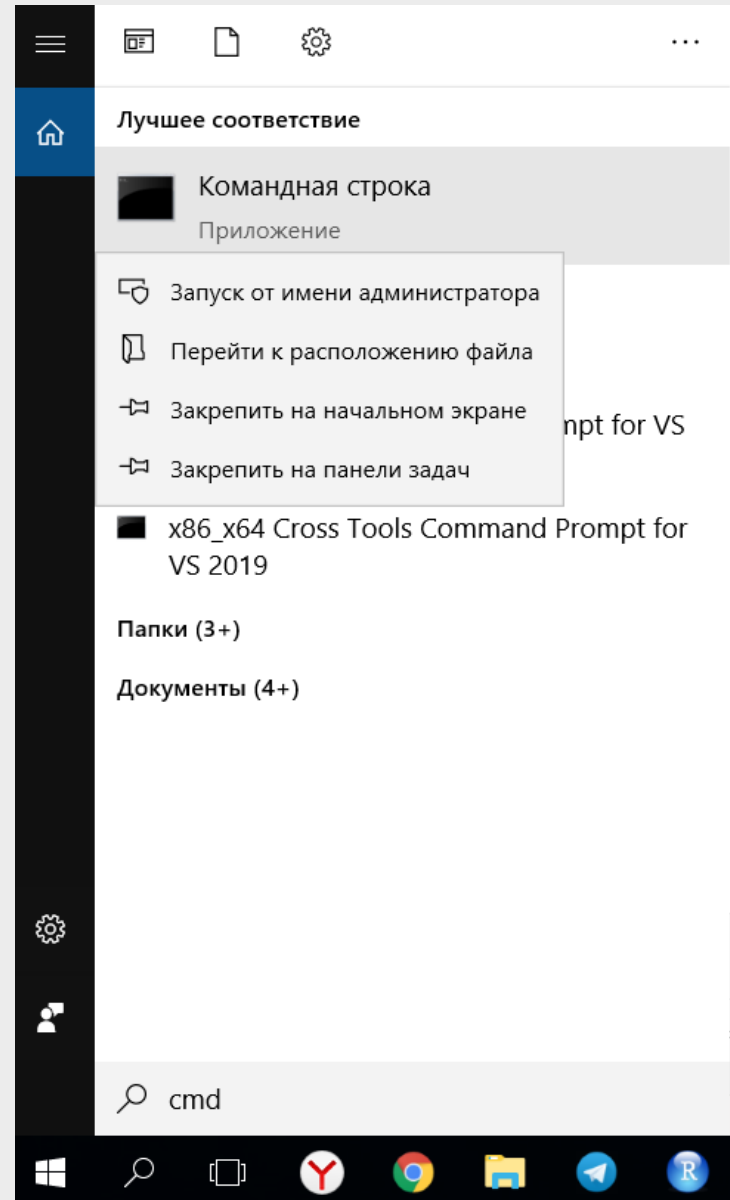
Терминал Windows

Windows Terminal позволяет работать как с командной строкой, так и с PowerShell или даже с терминалами других сред и ОС



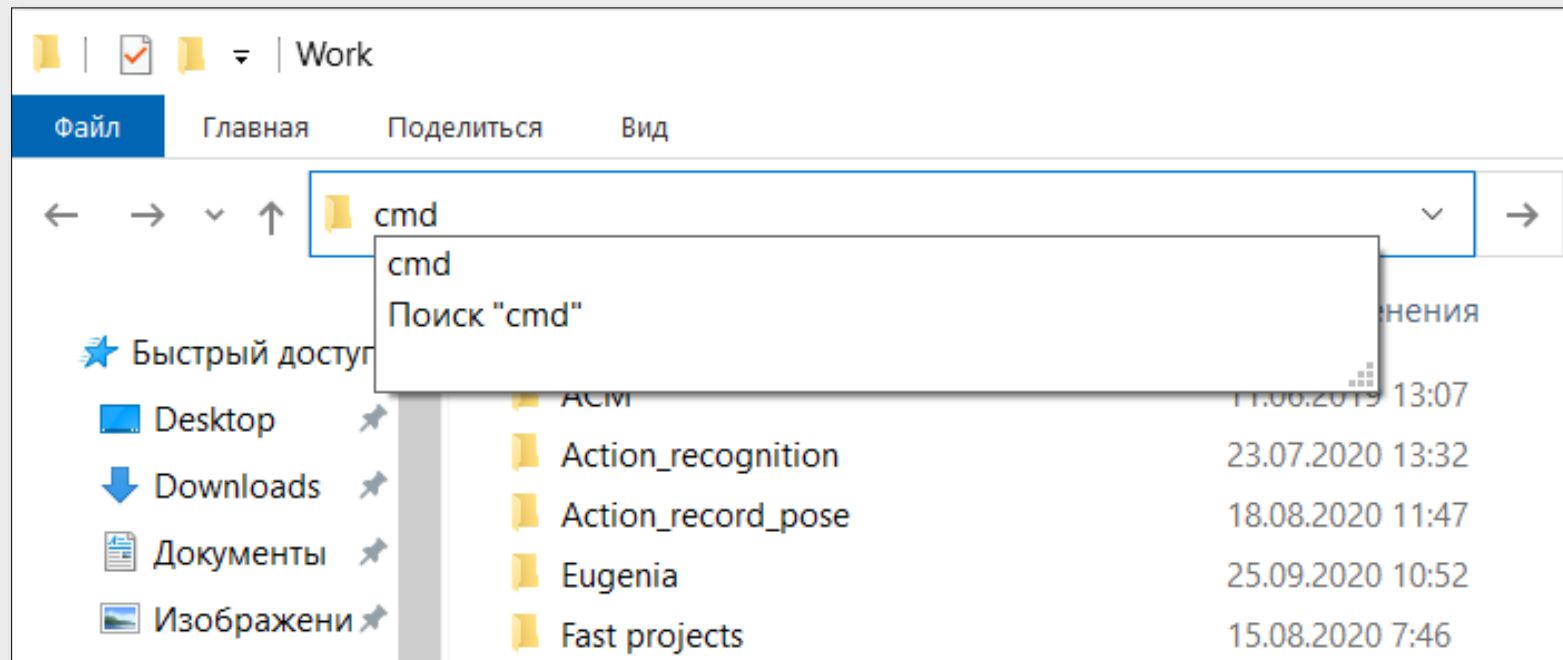
Запуск командной строки Windows

- Ввести в поиск cmd
- Windows + R -> cmd



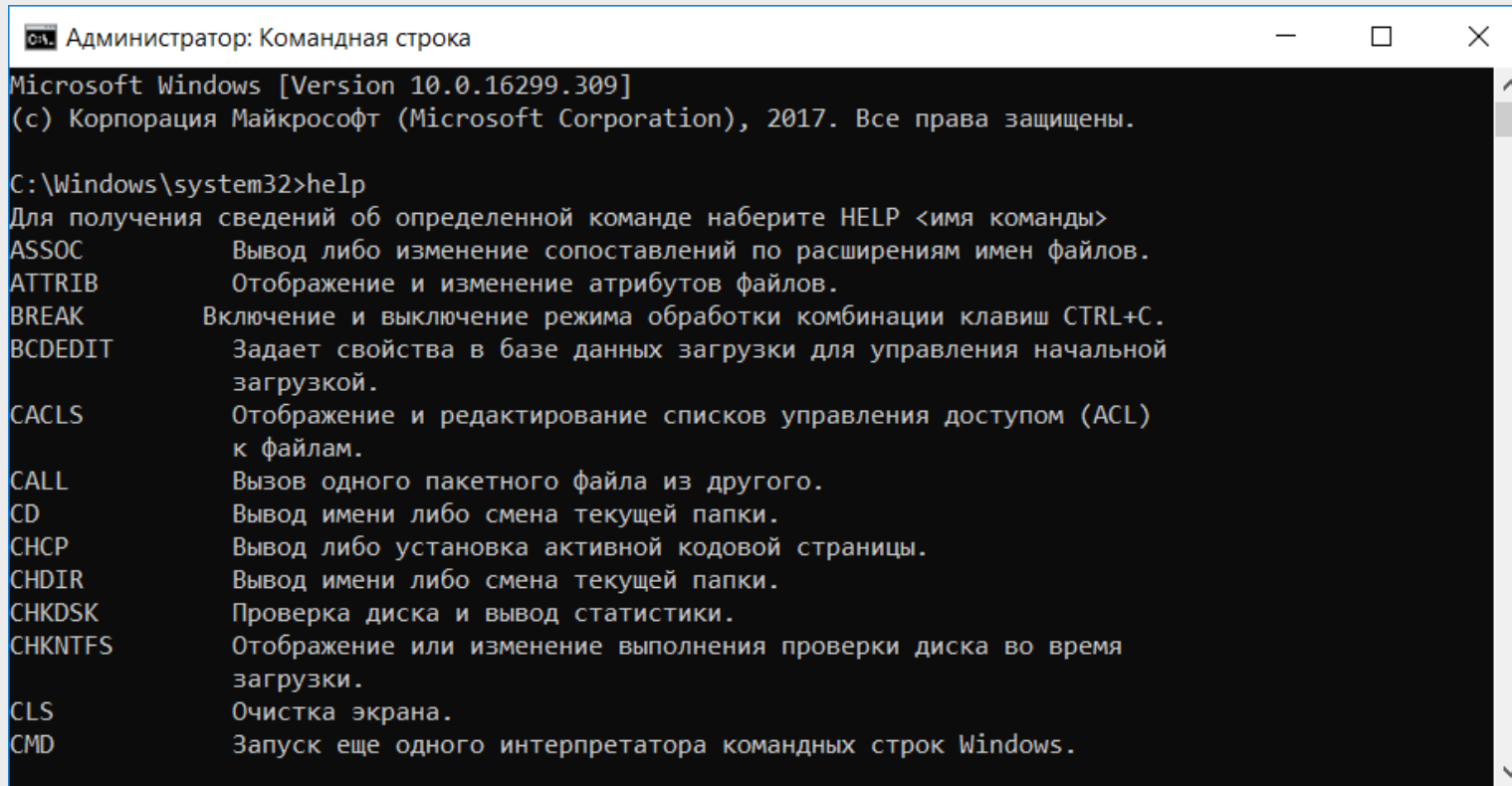
Запуск командной строки в Проводнике Windows

В проводнике вводим в адресной строке нужной папки **cmd**



Работа с командной строкой

- Пользователь вводит команду
- CMD выполняет команду



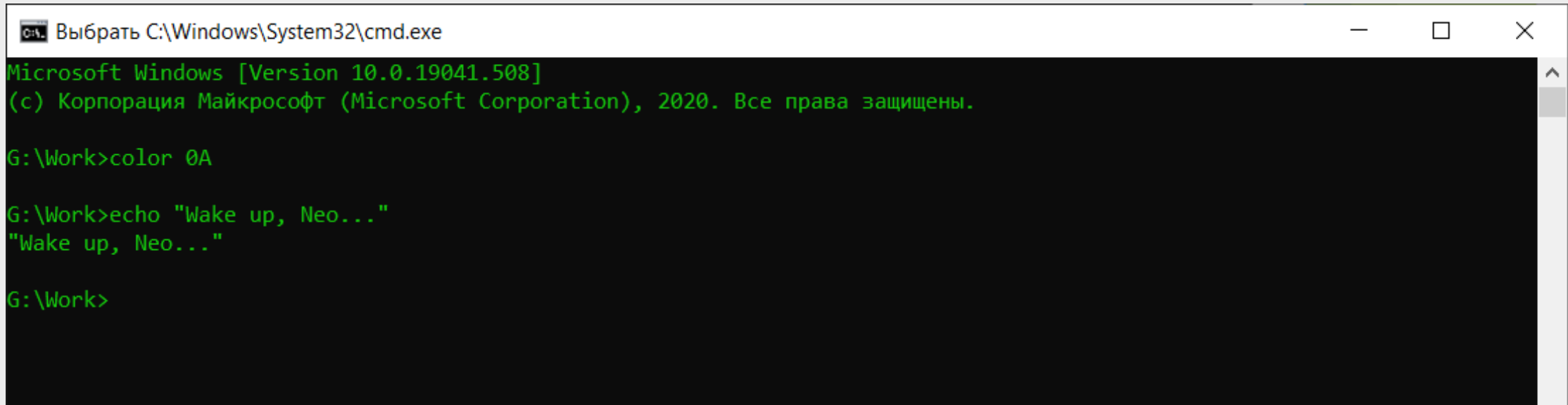
```
Администратор: Командная строка
Microsoft Windows [Version 10.0.16299.309]
(c) Корпорация Майкрософт (Microsoft Corporation), 2017. Все права защищены.

C:\Windows\system32>help
Для получения сведений об определенной команде наберите HELP <имя команды>
ASSOC          Вывод либо изменение сопоставлений по расширениям имен файлов.
ATTRIB         Отображение и изменение атрибутов файлов.
BREAK          Включение и выключение режима обработки комбинации клавиш CTRL+C.
BCDEDIT        Задаёт свойства в базе данных загрузки для управления начальной
                загрузкой.
CACLS          Отображение и редактирование списков управления доступом (ACL)
                к файлам.
CALL           Вызов одного пакетного файла из другого.
CD             Вывод имени либо смена текущей папки.
CHCP           Вывод либо установка активной кодовой страницы.
CHDIR          Вывод имени либо смена текущей папки.
CHKDSK         Проверка диска и вывод статистики.
CHKNTFS        Отображение или изменение выполнения проверки диска во время
                загрузки.
CLS            Очистка экрана.
CMD            Запуск ещё одного интерпретатора командных строк Windows.
```

Пример: команда **help**

Работа с командной строкой

- Команда **color** позволяет перекрасить цвет шрифта и фона
- Команда **echo** позволяет вывести сообщение



The screenshot shows a Windows Command Prompt window titled "Выбрать C:\Windows\System32\cmd.exe". The window has a black background and green text. The text inside the window is as follows:

```
Microsoft Windows [Version 10.0.19041.508]
(c) Корпорация Майкрософт (Microsoft Corporation), 2020. Все права защищены.

G:\Work>color 0A

G:\Work>echo "Wake up, Neo..."
"Wake up, Neo..."

G:\Work>
```

Работа с файловой системой

- **Путь** (англ. path) — набор символов, показывающий расположение файла или каталога в файловой системе
- В операционных системах UNIX разделительным знаком при записи пути является «/»
- В Windows — «\». Эти знаки служат для разделения названия каталогов, составляющих путь к файлу.

Абсолютный и относительный путь

Абсолютный путь – полный путь к файлу (каталогу), начиная с метки тома

```
D:\work\datasets>cd D:\work\example  
D:\work\example>dir  
Том в устройстве D не имеет метки.  
Серийный номер тома: 54FD-0D31  
  
Содержимое папки D:\work\example
```

Относительный путь – путь к файлу (каталогу) относительно текущего каталога

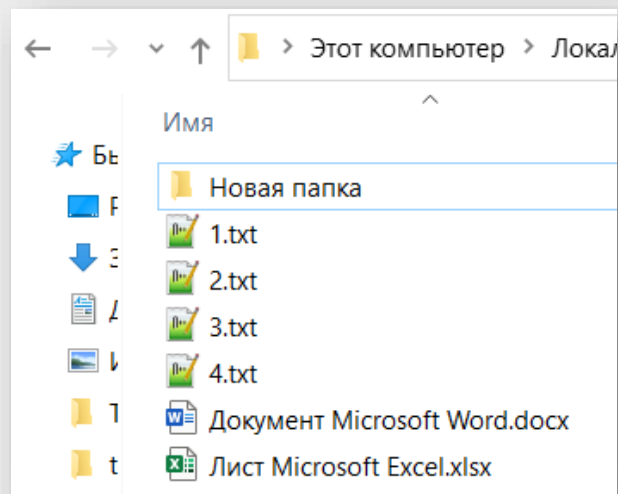
«Подняться» на уровень выше можно с помощью двух точек (..)

```
D:\work\example>cd ../datasets  
D:\work\datasets>cd ..  
D:\work>
```

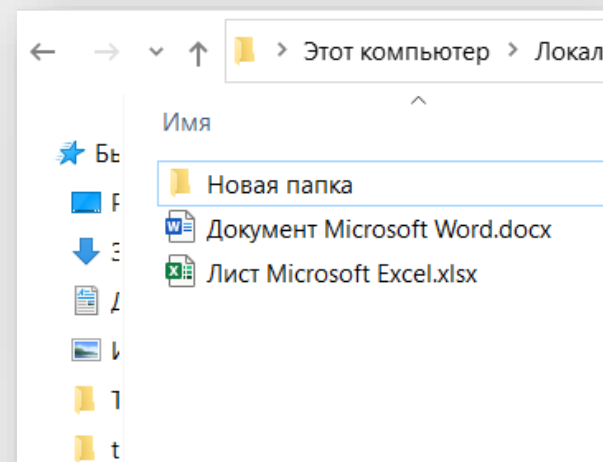

Маска

Символ «звездочка» (*) заменяет последовательность символов произвольной длины

- Пример: удалить все файлы с расширением .txt

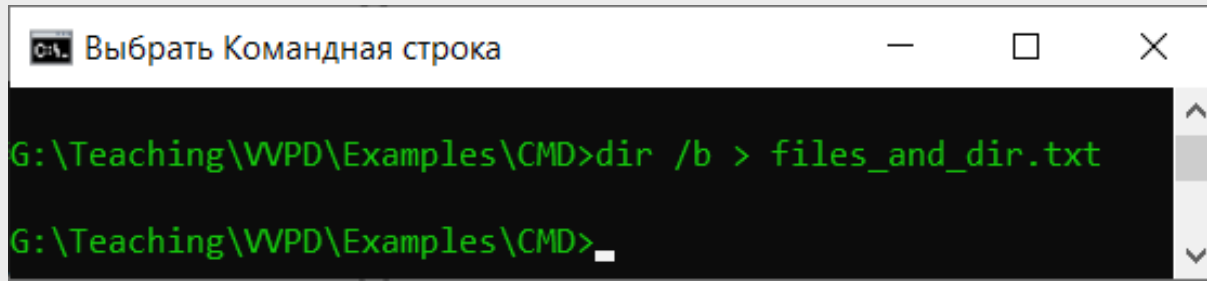


```
D:\>cd tmp  
D:\tmp>del "*.txt"
```

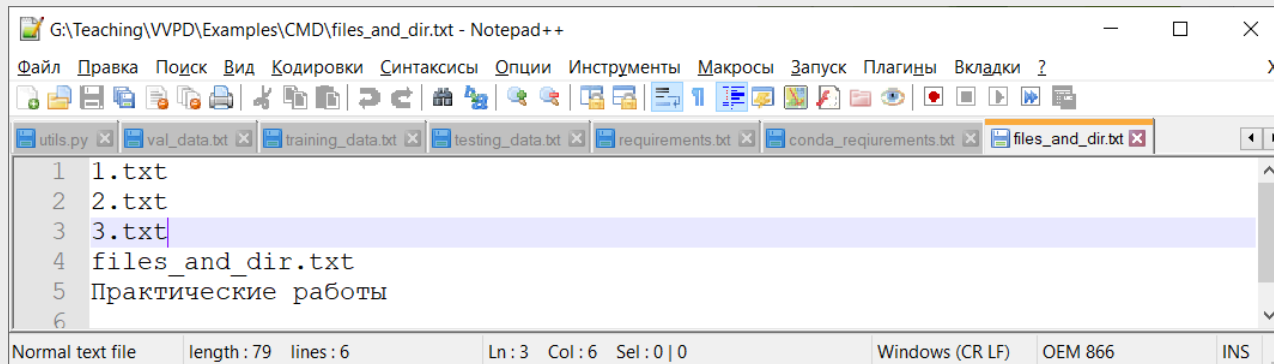


Поток ввода-вывода

- **Поток вывода** – это то, куда происходит вывод символов.
- По умолчанию – на экран
- С помощью оператора `>` можно изменить поток вывода



```
G:\Teaching\WVPD\Examples\CMD>dir /b > files_and_dir.txt
G:\Teaching\WVPD\Examples\CMD>
```



```
1 1.txt
2 2.txt
3 3.txt
4 files_and_dir.txt
5 Практические работы
6
```

Normal text file | length : 79 | lines : 6 | Ln : 3 | Col : 6 | Sel : 0 | 0 | Windows (CR LF) | OEM 866 | INS

Операторы совместного запуска

- Оператор «>>»: если такого файла нет, то он создается, а если файл существует, то данные добавляются в него

команда1 >> имя_файла

- Оператор «&»: сначала выполнятся Команда1, а уже потом Команда2

команда1 & команда2

- Оператор «&&»: выполнятся команда2, только если выполнилась команда1

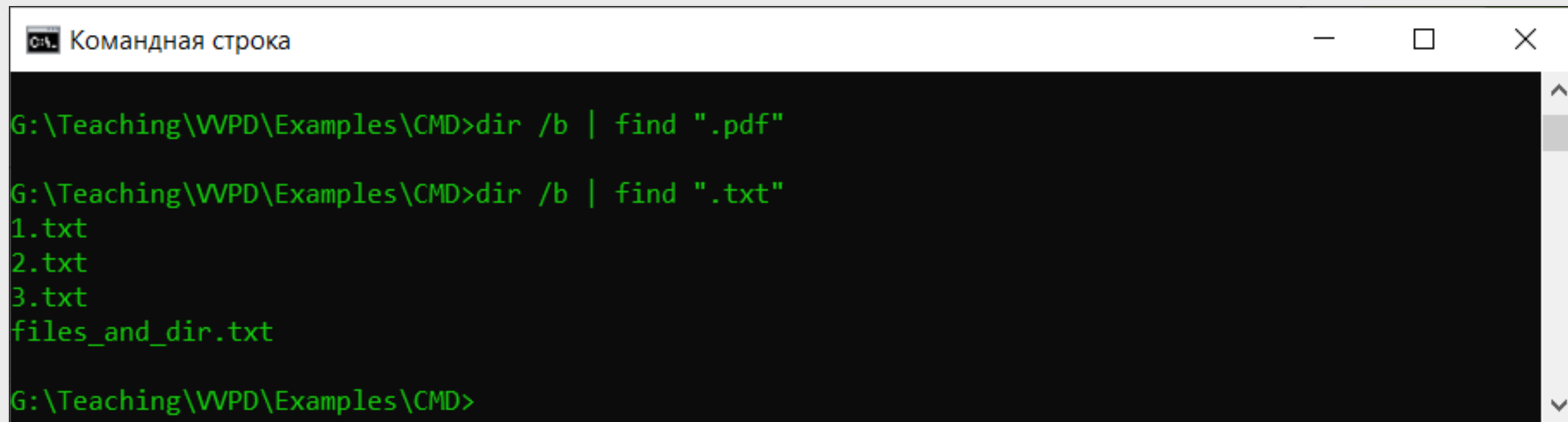
команда1 && команда2

- Оператор «||»: команда2 будет выполняться только в том случае, если команда1 не смогла выполниться

команда1 || команда2

Конвейер

- Оператор «|»: результат, полученный после выполнения команды1 будет служить как входной параметр для команды2
команда1 | команда2



```
Командная строка

G:\Teaching\WPD\Examples\CMD>dir /b | find ".pdf"

G:\Teaching\WPD\Examples\CMD>dir /b | find ".txt"
1.txt
2.txt
3.txt
files_and_dir.txt

G:\Teaching\WPD\Examples\CMD>
```

Интерфейс командной строки

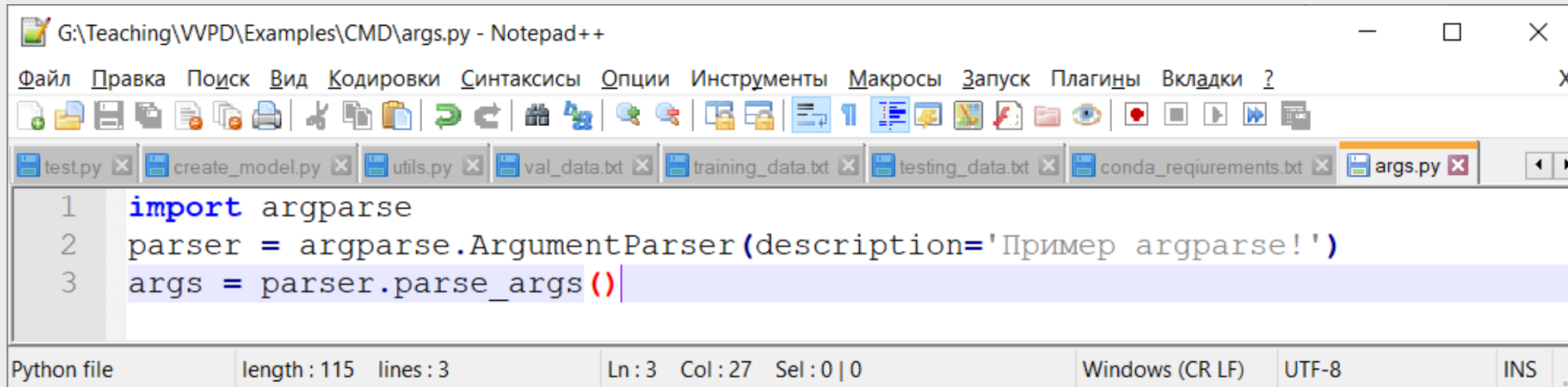
CLI – Command Line Interface – интерфейс командной строки

Основные особенности использования:

- Запускается одна команда, а ей передаются аргументы
- Эта команда отработает, ничего от пользователя не ждет и закрывается

Пример CLI на Python

1. Создадим файл args.py
2. Импортируется модуль argparse
3. Пользователю дается подсказка через аргументы
4. Программа может получить аргументы из командной строки и использовать их

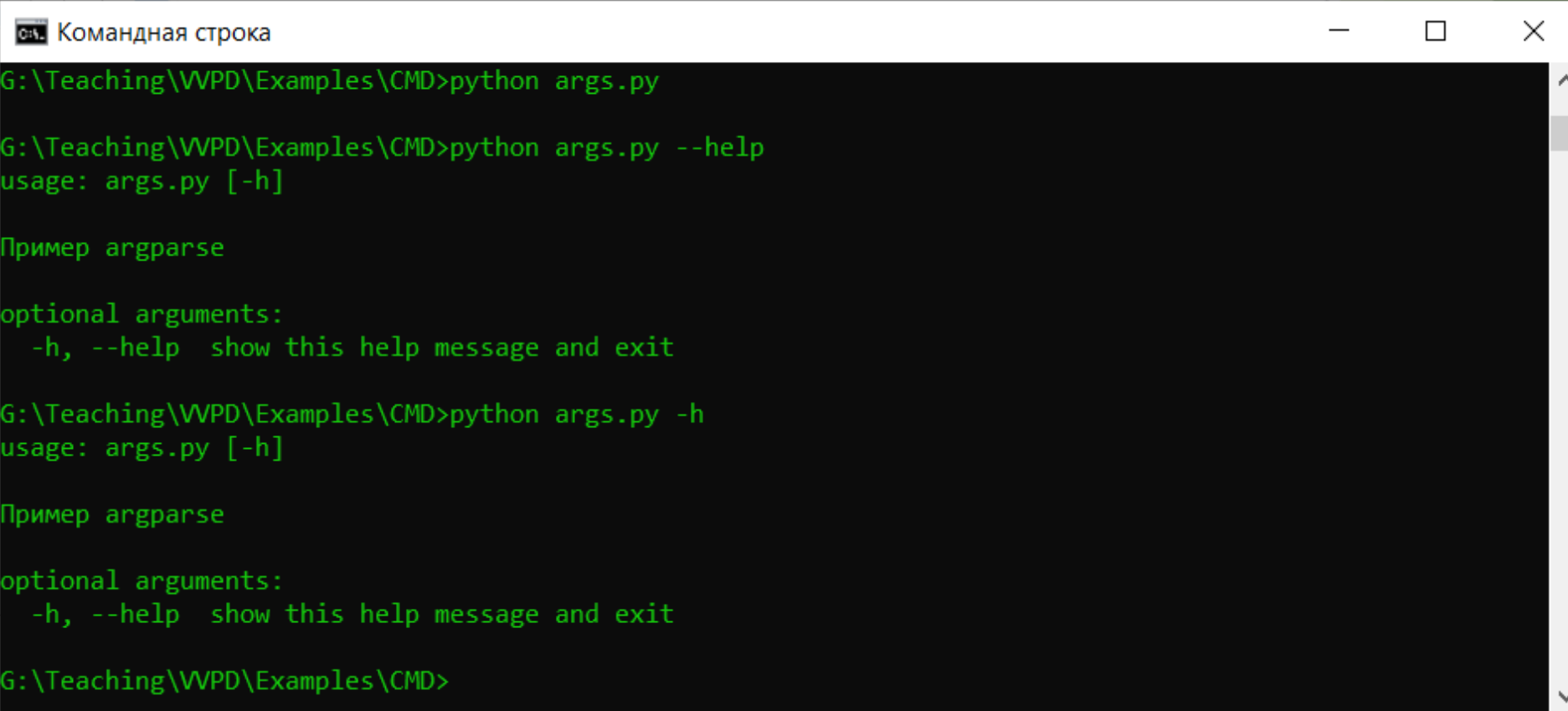


```
G:\Teaching\VVPD\Examples\CMD\args.py - Notepad++
Файл  Правка  Поиск  Вид  Кодировки  Синтаксисы  Опции  Инструменты  Макросы  Запуск  Плагины  Вкладки  ?
test.py x create_model.py x utils.py x val_data.txt x training_data.txt x testing_data.txt x conda_requirements.txt x args.py x
1  import argparse
2  parser = argparse.ArgumentParser(description='Пример argparse!')
3  args = parser.parse_args()

Python file    length : 115  lines : 3    Ln : 3  Col : 27  Sel : 0 | 0    Windows (CR LF)  UTF-8  INS
```

Результат работы программы

- Запуск файла происходит с помощью Python
- Пишем --help, чтоб увидеть описание
- Можем передать тоже самое с помощью -h



```
Командная строка
G:\Teaching\WVPD\Examples\CMD>python args.py

G:\Teaching\WVPD\Examples\CMD>python args.py --help
usage: args.py [-h]

Пример argparse

optional arguments:
  -h, --help  show this help message and exit

G:\Teaching\WVPD\Examples\CMD>python args.py -h
usage: args.py [-h]

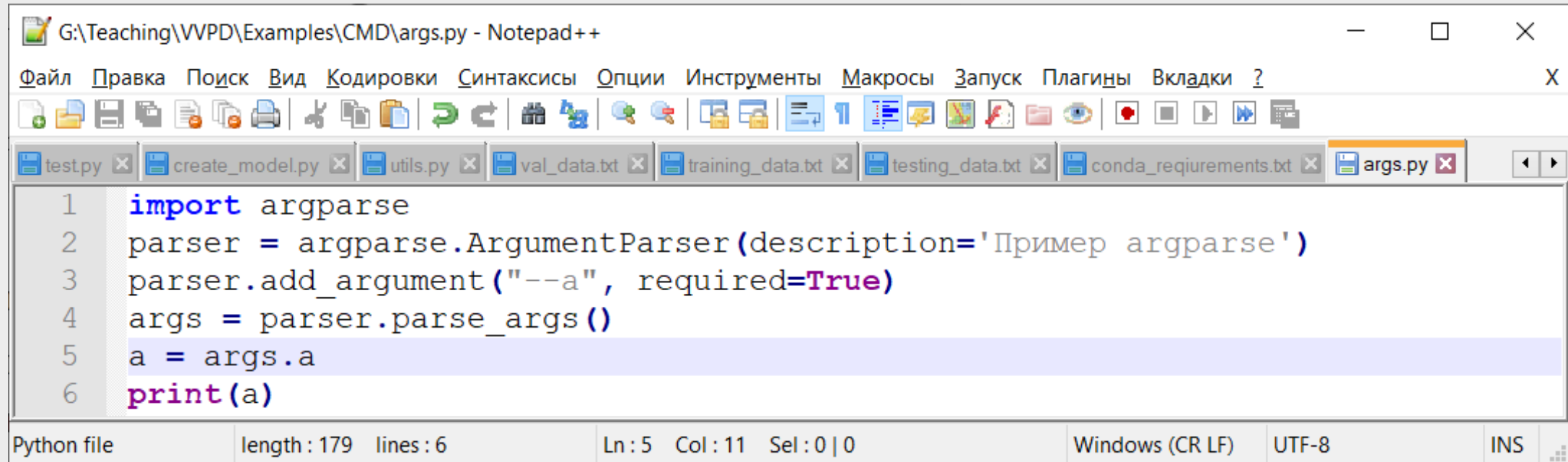
Пример argparse

optional arguments:
  -h, --help  show this help message and exit

G:\Teaching\WVPD\Examples\CMD>
```

Модифицируем программу...

- Добавим обязательный аргумент **--a**
- Добавим печать на экран с помощью **print()**




The screenshot shows a Notepad++ window titled "G:\Teaching\VVPD\Examples\CMD\args.py - Notepad++". The menu bar includes "Файл", "Правка", "Поиск", "Вид", "Кодировки", "Синтаксисы", "Опции", "Инструменты", "Макросы", "Запуск", "Плагины", "Вкладки", and "?". The toolbar contains various icons for file operations and editing. The tab bar shows several open files: "test.py", "create_model.py", "utils.py", "val_data.txt", "training_data.txt", "testing_data.txt", "conda_requirements.txt", and "args.py". The main text area displays the following Python code:

```
1 import argparse
2 parser = argparse.ArgumentParser(description='Пример argparse')
3 parser.add_argument("--a", required=True)
4 args = parser.parse_args()
5 a = args.a
6 print(a)
```

The status bar at the bottom indicates "Python file", "length : 179 lines : 6", "Ln : 5 Col : 11 Sel : 0 | 0", "Windows (CR LF)", "UTF-8", and "INS".

Тестирование программы

Запуск №1 – без аргументов. Выдаст ошибку и сообщит, что аргумент «а» – обязательный

 Выбрать Командная строка

```
G:\Teaching\WVPD\Examples\CMD>python args.py
usage: args.py [-h] --a A
args.py: error: the following arguments are required: --a
```

Запуск №2 – ввод числа. Программа напечатала введенное число

```
G:\Teaching\WVPD\Examples\CMD>python args.py --a 10
10
```

Запуск №3 – ввод строки. Программа напечатала введенную строку

```
G:\Teaching\WVPD\Examples\CMD>python args.py --a "вава"
вава
```

Требования к приложению с CLI

- Приложение должно быть простым и иметь четкую цель
- Пользователь должен иметь простой доступ к справке о том, как что делает приложение и как (help)
- Стандартные сценарии использования должны быть доступны без указания большого числа опций
- В случае ошибки в вызове приложения, оно должно сообщить, в чем была ошибка и как её исправить
- Вывод приложения должен быть приятным глазу. Возможно интерактивное взаимодействие с пользователем

Завершающая отбивка

Итак, сегодня мы с вами изучили тему Работа с командной строкой Linux и Windows

Я познакомила вас с историей возникновения командной строки и основными командами.

Теперь вы знаете, что можно сделать в командной строке Windows

Умеете создавать свои собственные приложения с интерфейсом командной строки

Это только начало. Изучайте курс дальше. До встречи на следующих уроках



edu.bmstu.ru

+7 (495) 120-99-76

edu@bmstu.ru

Москва, 2-ая Бауманская, д. 5, с. 1