

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**  
**дисциплины «Анализ данных»**

Выполнил:  
Данилецкий Дмитрий Витальевич  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р. А., канд. технических  
наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

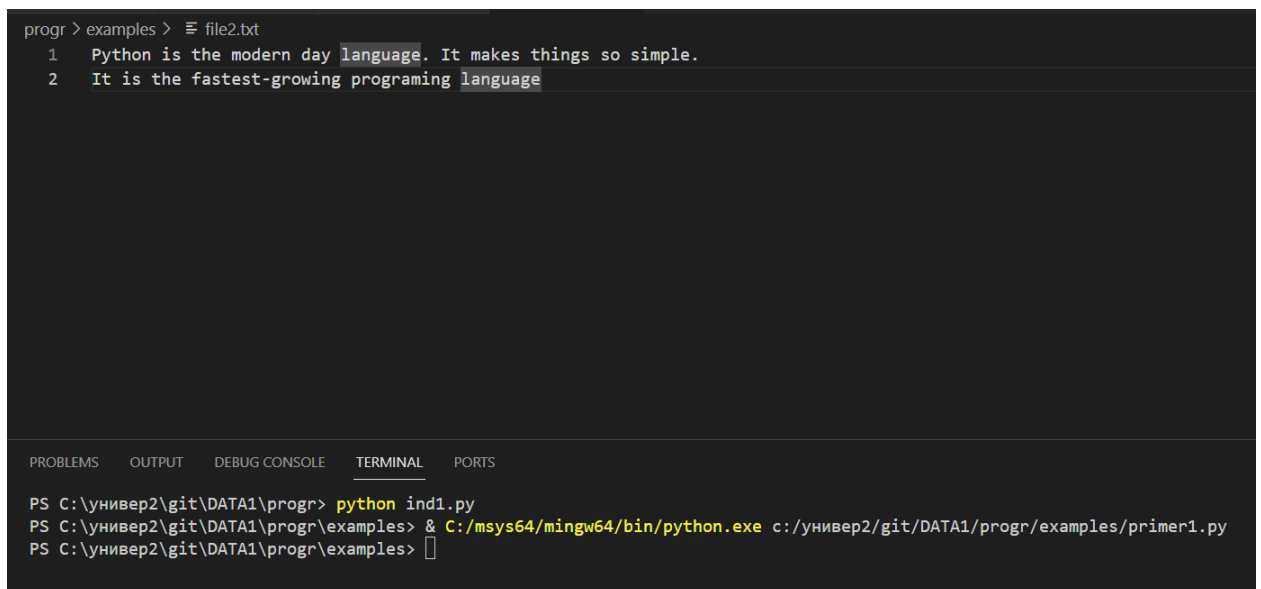
Ставрополь, 2024 г.

Тема: работа с файлами в языке Python

Цель: приобретение навыков по работе с текстовыми файлами при написании программ с помощью языка программирования Python версии 3.x, изучение основных методов модуля os для работы с файловой системой, получение аргументов командной строки.

#### Ход работы

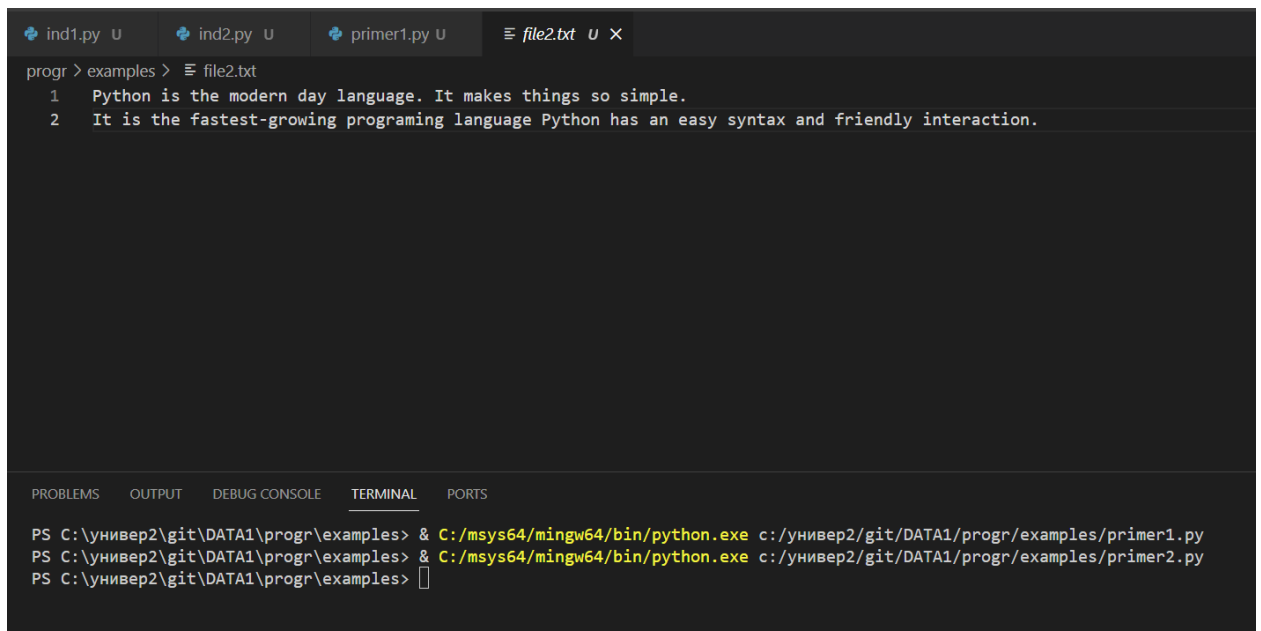
1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл .gitignore необходимыми правилами.
3. Организовал созданный репозиторий в соответствие с моделью ветвления git-flow.
4. Проработал примеры лабораторной работы. Создал для них отдельные модули языка Python. Привел в отчете скриншоты результата выполнения программ примеров при различных исходных данных, вводимых с клавиатуры.



The screenshot shows a code editor with a dark theme. The top part displays a text file named 'file2.txt' with two lines of text: '1 Python is the modern day language. It makes things so simple.' and '2 It is the fastest-growing programing language'. The bottom part shows a terminal window with the following commands and output:

```
PS C:\универ2\git\DATA1\progr> python ind1.py
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer1.py
PS C:\универ2\git\DATA1\progr\examples>
```

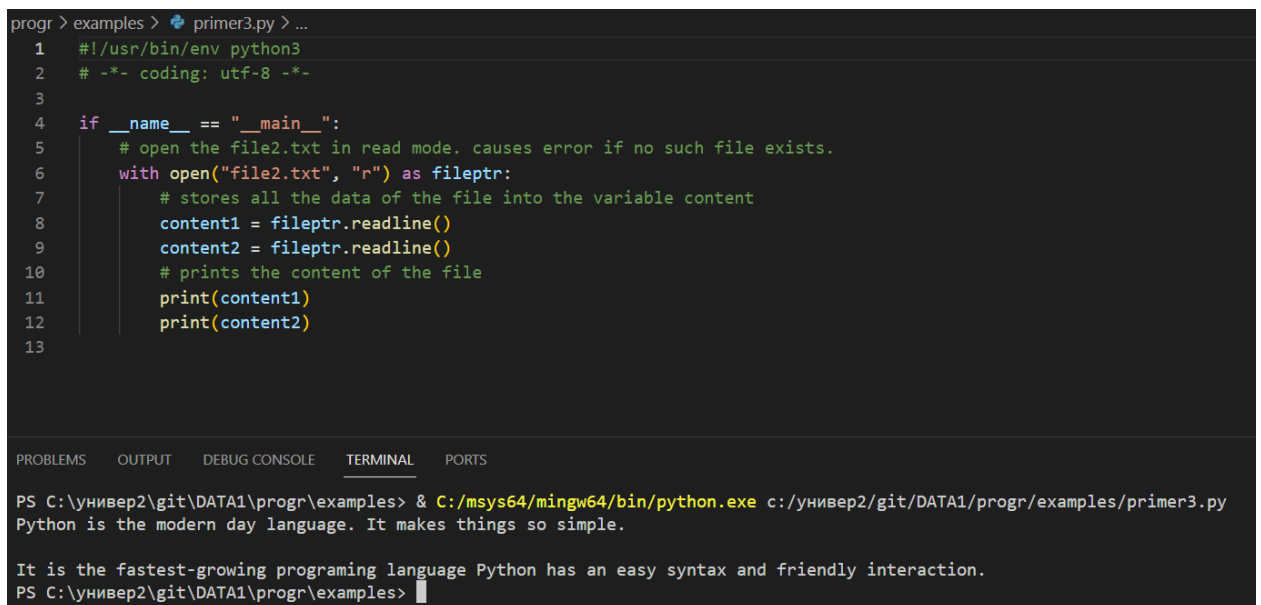
Рисунок 1. Результат работы программы из примера 1



```
ind1.py U ind2.py U primer1.py U file2.txt U X
progr > examples > file2.txt
1 Python is the modern day language. It makes things so simple.
2 It is the fastest-growing programing language Python has an easy syntax and friendly interaction.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer1.py
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer2.py
PS C:\универ2\git\DATA1\progr\examples> 
```

Рисунок 2. Результат работы программы из примера 2



```
progr > examples > primer3.py > ...
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == "__main__":
5     # open the file2.txt in read mode. causes error if no such file exists.
6     with open("file2.txt", "r") as fileptr:
7         # stores all the data of the file into the variable content
8         content1 = fileptr.readline()
9         content2 = fileptr.readline()
10        # prints the content of the file
11        print(content1)
12        print(content2)
13
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer3.py
Python is the modern day language. It makes things so simple.

It is the fastest-growing programing language Python has an easy syntax and friendly interaction.
PS C:\универ2\git\DATA1\progr\examples> 
```

Рисунок 3. Результат работы программы из примера 3

```
progr > examples > primer4.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      # open the file2.txt in read mode. causes error if no such file exists.
6      with open("file2.txt", "r") as fileptr:
7          # stores all the data of the file into the variable content
8          content = fileptr.readlines()
9          # prints the content of the file
10         print(content)
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer4.py  
['Python is the modern day language. It makes things so simple.\n', 'It is the fastest-growing programing language Python ha  
n easy syntax and friendly interaction.']  
PS C:\универ2\git\DATA1\progr\examples>

Рисунок 4. Результат работы программы из примера 4

```
progr > examples > primer5.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5
6      # open the newfile.txt in read mode. causes error if no such file exists.
7      with open("newfile.txt", "x") as fileptr:
8          print(fileptr)
9
10         if fileptr:
11             print("File created successfully")
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer5.py  
<\_io.TextIOWrapper name='newfile.txt' mode='x' encoding='cp1251'>  
File created successfully  
PS C:\универ2\git\DATA1\progr\examples>

Рисунок 5. Результат работы программы из примера 5

```
progr > examples > text.txt
1  UTF-8 is a variable-width character encoding used for electronic communication.
2  UTF-8 is capable of encoding all 1,112,064 valid character code points.
3  In Unicode using one to four one-byte (8-bit) code units.
4
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer6.py  
PS C:\универ2\git\DATA1\progr\examples>

Рисунок 6. Результат работы программы из примера 6

```
progr > examples > primer7.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      with open("text.txt", "r", encoding="utf-8") as fileptr:
5          sentences = fileptr.readlines()
6
7      # Вывод предложений с запятыми.
8      for sentence in sentences:
9          if "," in sentence:
10             print(sentence)
11
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer7.py
UTF-8 is capable of encoding all 1,112,064 valid character code points.

PS C:\универ2\git\DATA1\progr\examples> █
```

Рисунок 7. Результат работы программы из примера 7

```
progr > examples > primer8.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  if __name__ == "__main__":
4      # open the file file2.txt in read mode
5      with open("file2.txt", "r") as fileptr:
6          # initially the filepointer is at 0
7          print("The filepointer is at byte :", fileptr.tell())
8          # changing the file pointer location to 10
9          fileptr.seek(10)
10         # tell() returns the location of the fileptr.
11         print("After reading, the filepointer is at:", fileptr.tell())
12
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer8.py
The filepointer is at byte : 0
After reading, the filepointer is at: 10
PS C:\универ2\git\DATA1\progr\examples> █
```

Рисунок 8. Результат работы программы из примера 8

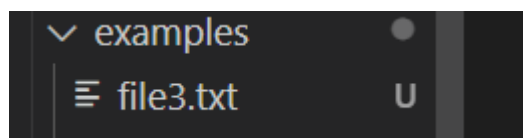


Рисунок 9. Результат работы программы из примера 9

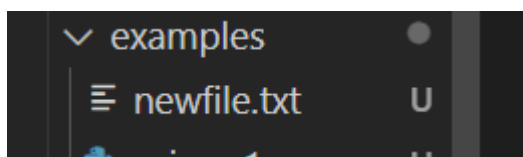


Рисунок 10. Результат работы программы из примера 10

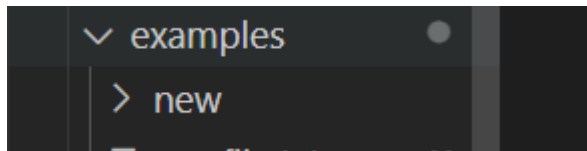


Рисунок 11. Результат работы программы из примера 11

```
progr > examples > primer12.py > ...
1  import os
2
3  # !usr/bin/env python3
4  # -*- coding: utf-8 -*-
5
6  if __name__ == "__main__":
7      path = os.getcwd()
8      print(path)
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer12.py
C:\универ2\git\DATA1\progr\examples
PS C:\универ2\git\DATA1\progr\examples>
```

Рисунок 12. Результат работы программы из примера 12

```
progr > examples > primer13.py
1  import os
2
3  # !usr/bin/env python3
4  # -*- coding: utf-8 -*-
5  if __name__ == "__main__":
6      # Changing current directory with the new directory
7      os.chdir("C:\универ2\git\DATA1\progr\examples\new")
8      # It will display the current working directory
9      print(os.getcwd())
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer13.py
C:\универ2\git\DATA1\progr\examples\new
PS C:\универ2\git\DATA1\progr\examples>
```

Рисунок 13. Результат работы программы из примера 13

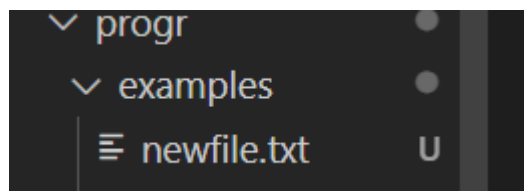


Рисунок 14. Результат работы программы из примера 14

```
progr > examples > primer15.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4
5  if __name__ == "__main__":
6      print("Number of arguments:", len(sys.argv), "arguments")
7      print("Argument List:", str(sys.argv))
8
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer15.py
Number of arguments: 1 arguments
Argument List: ['c:/универ2/git/DATA1/progr/examples/primer15.py']
PS C:\универ2\git\DATA1\progr\examples> █
```

Рисунок 15. Результат работы программы из примера 15

```
progr > examples > primer16.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import sys
4
5  if __name__ == "__main__":
6      for idx, arg in enumerate(sys.argv):
7          print(f"Argument #{idx} is {arg}")
8      print("No. of arguments passed is ", len(sys.argv))
9
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\универ2\git\DATA1\progr\examples> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/examples/primer16.py
Argument #0 is c:/универ2/git/DATA1/progr/examples/primer16.py
No. of arguments passed is 1
PS C:\универ2\git\DATA1\progr\examples> █
```

Рисунок 16. Результат работы программы из примера 16

```
progr > examples > primer17.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import secrets
6  import string
7  import sys
8
9  if __name__ == "__main__":
10     if len(sys.argv) != 2:
11         print("The password length is not given!", file=sys.stderr)
12         sys.exit(1)
13
14     chars = string.ascii_letters + string.punctuation + string.digits
15     length_pwd = int(sys.argv[1])
16
17     result = []
18     for _ in range(length_pwd):
19         idx = secrets.SystemRandom().randrange(len(chars))
20         result.append(chars[idx])
21
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\универ2\git\DATA1\progr\examples> python ./primer17.py 21
Secret Password: n68#b%mcS$Qa=1A]JQC"
```

Рисунок 17. Результат работы программы из примера 17

5. Выполнил индивидуальные задания, согласно варианту 8. Привёл в отчете скриншоты работы программ.

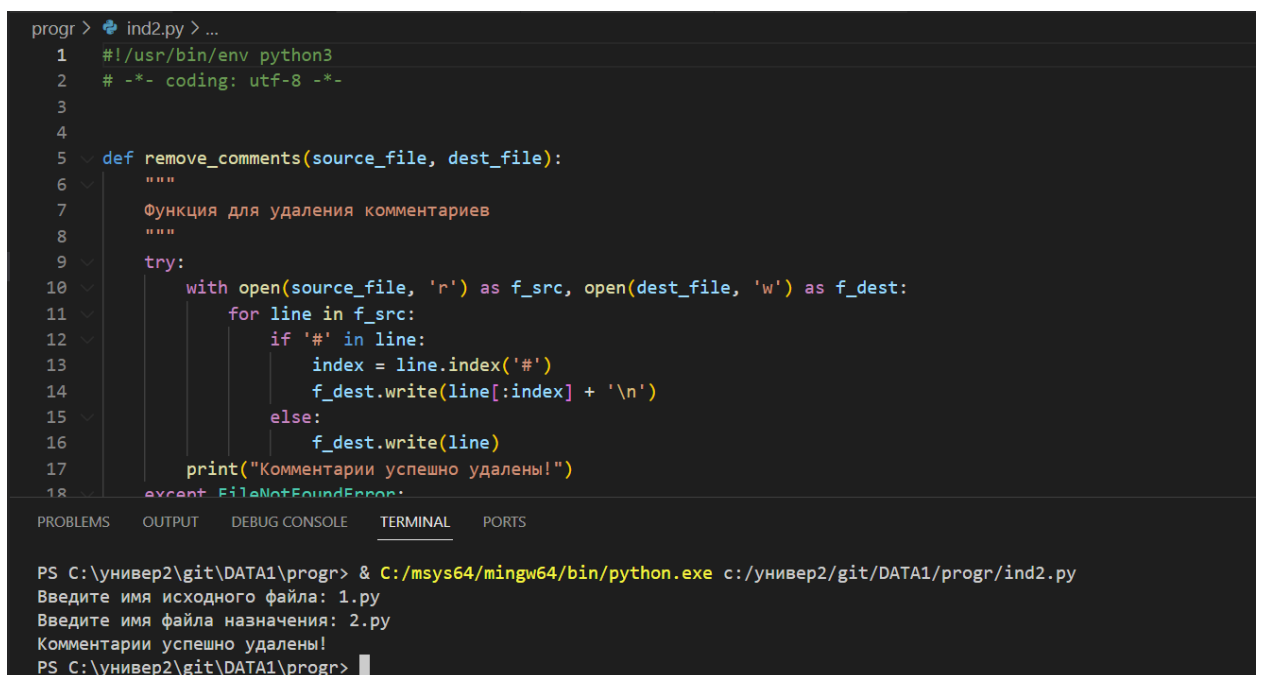
Задание. Написать программу, которая считывает текст из файла и выводит на экран толь ко цитаты, то есть предложения, заключенные в кавычки.

```
progr > ind1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      with open("citati.txt", "r") as file:
6          text = file.read()
7
8      quotes = []
9      quote = ""
10     in_quote = False
11
12     for char in text:
13         if char == '"':
14             if in_quote:
15                 quotes.append(quote)
16                 quote = ""
17                 in_quote = False
18             else:
19                 in_quote = True
20         elif in_quote:
21             quote += char
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\универ2\git\DATA1\progr\examples> cd ..
PS C:\универ2\git\DATA1\progr> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/ind1.py
The best dreams happen when youre awake.
Accept who you are. Unless youre a serial killer.
PS C:\универ2\git\DATA1\progr>
```

Рисунок 18. Результат работы программы из индивидуального задания 1



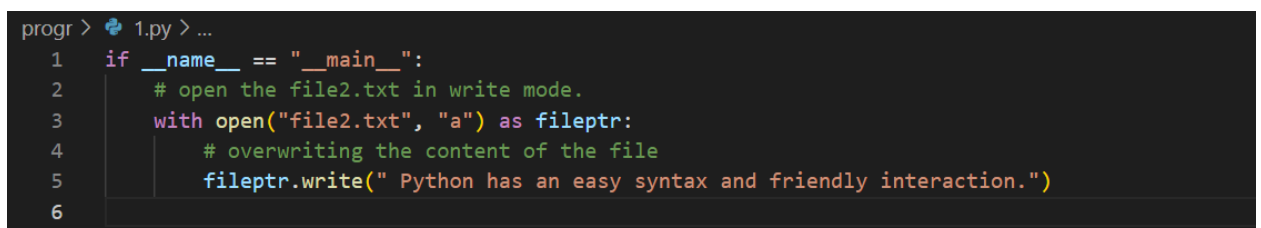
Задание. Как вы знаете, в языке Python для создания комментариев в коде используется символ #. Комментарий начинается с этого символа и продолжается до конца строки – без возможности остановить его раньше. В данном упражнении вам предстоит написать программу, которая будет удалять все комментарии из исходного файла с кодом на языке Python. Пройдите по всем строкам в файле на предмет поиска символа #. Обнаружив его, программа должна удалить все содержимое, начиная с этого символа и до конца строки. Для простоты не будем рассматривать ситуации, когда знак решетки встречается в середине строки. Сохраните новое содержимое в созданном файле. Имена файла источника и файла назначения должны быть запрошены у пользователя. Удостоверьтесь в том, что программа корректно обрабатывает возможные ошибки при работе с обоими файлами.



```
progr > ind2.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  def remove_comments(source_file, dest_file):
6      """
7      Функция для удаления комментариев
8      """
9      try:
10         with open(source_file, 'r') as f_src, open(dest_file, 'w') as f_dest:
11             for line in f_src:
12                 if '#' in line:
13                     index = line.index('#')
14                     f_dest.write(line[:index] + '\n')
15                 else:
16                     f_dest.write(line)
17             print("Комментарии успешно удалены!")
18     except FileNotFoundError:
```

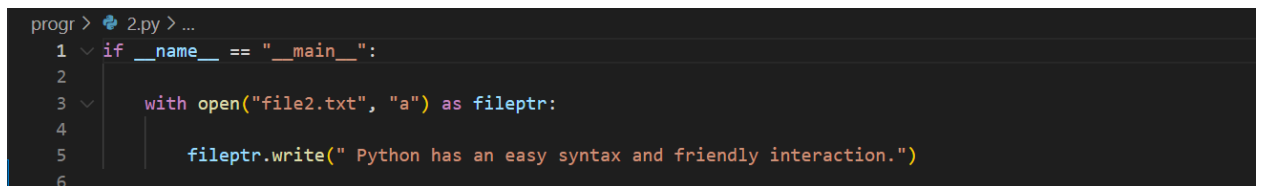
```
PS C:\универ2\git\DATA1\progr> & C:/msys64/mingw64/bin/python.exe c:/универ2/git/DATA1/progr/ind2.py
Введите имя исходного файла: 1.py
Введите имя файла назначения: 2.py
Комментарии успешно удалены!
PS C:\универ2\git\DATA1\progr>
```

Рисунок 19. Результат работы программы из индивидуального задания 2



```
progr > 1.py > ...
1  if __name__ == "__main__":
2      # open the file2.txt in write mode.
3      with open("file2.txt", "a") as fileptr:
4          # overwriting the content of the file
5          fileptr.write(" Python has an easy syntax and friendly interaction.")
6
```

Рисунок 19. Файл до запуска программы



```
progr > 2.py > ...
1  if __name__ == "__main__":
2
3  with open("file2.txt", "a") as fileptr:
4
5      fileptr.write(" Python has an easy syntax and friendly interaction.")
6
```

Рисунок 20. Файл после запуска программы

### Контрольные вопросы

1. Как открыть файл в языке Python только для чтения?

В Python для открытия файла только для чтения используется функция `open()` с указанием режима доступа `r`.

2. Как открыть файл в языке Python только для записи?

Для открытия файла только для записи в Python используется функция `open()` с указанием режима доступа `'w'`. Если файл с указанным именем не существует, он будет создан. Если файл уже существует, его содержимое будет перезаписано.

3. Как прочитать данные из файла в языке Python?

Для чтения данных из файла в языке Python вы можете использовать встроенную функцию `open()` в сочетании с методами чтения, такими как `read()`, `readline()`, или `readlines()`.

4. Как записать данные в файл в языке Python?

В языке Python для записи данных в файл можно воспользоваться функцией `write()` объекта файла или методом `writelines()`.

5. Как закрыть файл в языке Python?

В языке Python закрыть файл можно с помощью метода `close()`. Этот метод следует вызывать после завершения всех операций с файлом, чтобы освободить ресурсы операционной системы. Однако, более предпочтительным способом управления файлом является использование конструкции `with`, которая автоматически закрывает файл после завершения работы с ним.

6. Изучите самостоятельно работу конструкции `with ... as`. Каково ее назначение в языке Python? Где она может быть использована еще, помимо работы с файлами?

Конструкция `with ... as` в языке Python используется для создания контекстных менеджеров. Ее основное назначение - обеспечить правильное управление ресурсами в блоке кода, гарантируя выполнение определенных действий до и после выполнения блока кода.

В контексте работы с файлами, конструкция `with ... as` используется для автоматического закрытия файла после завершения работы с ним. Но помимо работы с файлами, она может быть использована в различных сценариях, где требуется управление ресурсами или выполнение каких-то действий до и после выполнения определенного блока кода.

Кроме работы с файлами, конструкцию `with ... as` можно использовать для работы с другими ресурсами, которые требуют явного закрытия, например, сетевыми соединениями или базами данных. При использовании конструкции `with ... as` с такими ресурсами, можно быть уверенным, что они будут корректно закрыты, даже при возникновении исключений или ошибок.

7. Изучите самостоятельно документацию Python по работе с файлами. Какие помимо рассмотренных существуют методы записи/чтения информации из файла?

`writelines()`: Этот метод используется для записи списка строк в файл. Он принимает список строк в качестве аргумента и записывает каждую строку в конец файла.

`readline()`: Этот метод используется для чтения одной строки из файла. При каждом вызове метода он возвращает следующую строку файла.

`readlines()`: Этот метод используется для чтения всех строк из файла и возвращает список строк, где каждая строка представляет собой элемент списка.

Метод `seek()` используется для изменения позиции указателя файла. Он позволяет переместить указатель на определенное смещение `offset` относительно начала, текущей позиции или конца файла в зависимости от значения аргумента `whence`.

Аргументы `offset` и `whence` являются необязательными:

- `offset` - целочисленное значение, указывающее смещение в байтах.

Значение `offset` может быть положительным, отрицательным или нулем.

- `whence` - указывает, как интерпретировать аргумент `offset`. Доступные значения `whence`:

- 0 (по умолчанию): смещение относительно начала файла.
- 1: смещение относительно текущей позиции.
- 2: смещение относительно конца файла.

`tell()`: Метод `tell()` используется для получения текущей позиции указателя файла. Он возвращает целочисленное значение, представляющее текущую позицию указателя в байтах от начала файла.

`truncate([size])`: Метод `truncate()` используется для изменения размера файла до указанного размера `size`. Если аргумент `size` не указан, то размер файла обрезается до текущей позиции указателя. Метод возвращает новый размер файла после изменения.

8. Какие существуют, помимо рассмотренных, функции модуля `os` для работы с файловой системой?

Функция `listdir()` возвращает список файлов и директорий в указанном пути `path`. Она принимает один аргумент - путь к директории. Возвращается список строк, представляющих имена файлов и директорий в указанной директории.

Функция `makedirs()` создает все директории в указанном пути `path`. Если указан аргумент `mode`, то устанавливает права доступа для всех созданных директорий в соответствии с указанным значением `mode`.

Функция `removedirs()` используется для удаления директории по указанному пути `path` и всех пустых поддиректорий. Если какая-либо поддиректория не является пустой, будет вызвано исключение `OSError`.

Функция `join()` объединяет один или несколько компонентов пути в один путь. Она автоматически добавляет разделитель пути, соответствующий операционной системе.

Функция `exists()` проверяет, существует ли файл или директория по указанному пути `path`. Возвращает `True`, если файл или директория существуют, и `False` в противном случае.

Функция `isdir()` проверяет, является ли заданный путь директорией. Возвращает `True`, если путь является директорией, и `False` в противном случае.