

Модуль 3 Занятие 5

Работа с командной строкой. Argparse



**Командная
строка**

**Работа
с командной
строкой**

Модуль argparse

Командная строка



Графический интерфейс (Graphical user interface — GUI) — система средств для взаимодействия пользователя с программами. Графический интерфейс основан на представлении всех доступных пользователю системных объектов в виде графических элементов экрана: окна, значки, меню, кнопки, списки и так далее.



Командная строка (терминал, консоль) (Command line interface — CLI) — способ взаимодействия между пользователем и компьютером с помощью текстовых команд.



Командная строка — это прародитель графических интерфейсов.

Командная строка на моем компьютере

Для выполнения различных действий с операционной системой из командной строки используются специальные программы — командные оболочки. В каждой операционной системе по умолчанию есть своя командная оболочка:



Windows: PowerShell и cmd



Linux: Bash



macOS: Zsh (версия Bash)

Набор команд для разных операционных систем отличается, поэтому пользователям Windows рекомендуется установить приложение Git Bash, которое поддерживает необходимую систему команд, а также потребуется нам в будущем.

Начало работы

Если вы работаете на Windows — запустите Git Bash, если на macOS или Linux, запустите программу «Терминал».

MINGW64:/c/Users/User



User@DESKTOP-PVGUDRB MINGW64 ~

\$

Где я?

Чтобы узнать, в какой директории вы находитесь, введите в терминал команду **pwd** (print working directory).

MINGW64:/c/Users/User



User@DESKTOP-PVGUDRB MINGW64 ~

\$ pwd

/c/Users/User

User@DESKTOP-PVGUDRB MINGW64 ~

\$

Что здесь?

Чтобы узнать, что находится в директории, введите в терминал команду `ls` (list directory contents).

MINGW64:/c/Users/User

User@DESKTOP-PVGUDRB MINGW64 ~

```
$ ls
'3D Objects'/
anse1/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/
go/
IntelGraphicsProfiles/
Links/
'Local Settings'@
Music/
NetHood@
NTUSER.DAT
ntuser.dat.LOG1
ntuser.dat.LOG2
NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TM.blf
NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer00000000000000000001.regtrans-ms
NTUSER.DAT{53b39e88-18c4-11ea-a811-000d3aa4692b}.TMContainer00000000000000000002.regtrans-ms
```

Ключи

Находясь в любой директории, введите команду и ключ **--help**.

Ключи пишутся через пробел после команды и начинаются с одного или двух дефисов.

MINGW64:/c/Users/User

User@DESKTOP-PVGUDRB MINGW64 ~

```
$ ls --help
```

```
Usage: ls [OPTION]... [FILE]...
```

```
List information about the FILES (the current directory by default).
```

```
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

```
Mandatory arguments to long options are mandatory for short options too.
```

```
-a, --all                do not ignore entries starting with .
```

```
-A, --almost-all        do not list implied . and ..
```

```
    --author              with -l, print the author of each file
```

```
-b, --escape              print C-style escapes for nongraphic characters
```

```
    --block-size=SIZE     with -l, scale sizes by SIZE when printing them;
```

```
                          e.g., '--block-size=M'; see SIZE format below
```

```
-B, --ignore-backups      do not list implied entries ending with ~
```

```
-c                        with -lt: sort by, and show, ctime (time of last  
                          modification of file status information);
```

```
                          with -l: show ctime and sort by name;
```

```
                          otherwise: sort by ctime, newest first
```

```
-C                        list entries by columns
```

```
    --color[=WHEN]        colorize the output; WHEN can be 'always' (default  
                          if omitted), 'auto', or 'never'; more info below
```


Ключи

Ключи также называют «аргументами». Аргументы можно вводить в короткой форме с одним дефисом, например **ls -a** и в длинной форме, с двумя дефисами, например **ls --all**.

При использовании ключа важен регистр: **ls -a** и **ls -A** вернут разный результат.

Ключи также можно комбинировать, например: **ls -at** покажет файлы, в том числе скрытые (-a), и отсортирует результат по времени изменения (-t).

Навигация в командной строке

Для перемещения из одной папки в другую используется команда `cd` (change directory).

```
cd <имя директории>
```

Для перемещения в домашнюю директорию выполните команду:

```
cd ~
```

Для перемещения в директорию уровнем выше выполните команду:

```
cd ..
```

MINGW64:/c/Users

```
User@DESKTOP-PVGUDRB MINGW64 ~
```

```
$ cd Desktop/
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/Desktop
```

```
$ pwd
```

```
/c/Users/User/Desktop
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/Desktop
```

```
$ cd ~
```

```
User@DESKTOP-PVGUDRB MINGW64 ~
```

```
$ pwd
```

```
/c/Users/User
```

```
User@DESKTOP-PVGUDRB MINGW64 ~
```

```
$ cd ..
```

```
User@DESKTOP-PVGUDRB MINGW64 /c/Users
```

```
$ pwd
```

```
/c/Users
```

```
User@DESKTOP-PVGUDRB MINGW64 /c/Users
```

```
$
```

Новые папки и файлы

Чтобы создать новую папку в текущей директории, используется команда `mkdir` (make directory) и имя папки:

```
mkdir programs
```

Перейдите в созданную директорию и создайте файл. Чтобы создать файл, введите команду `touch`:

```
touch program.py
```

MINGW64:/c/Users/User/programs

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$ mkdir programs
```

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$ cd programs/
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs  
$ touch program.py
```

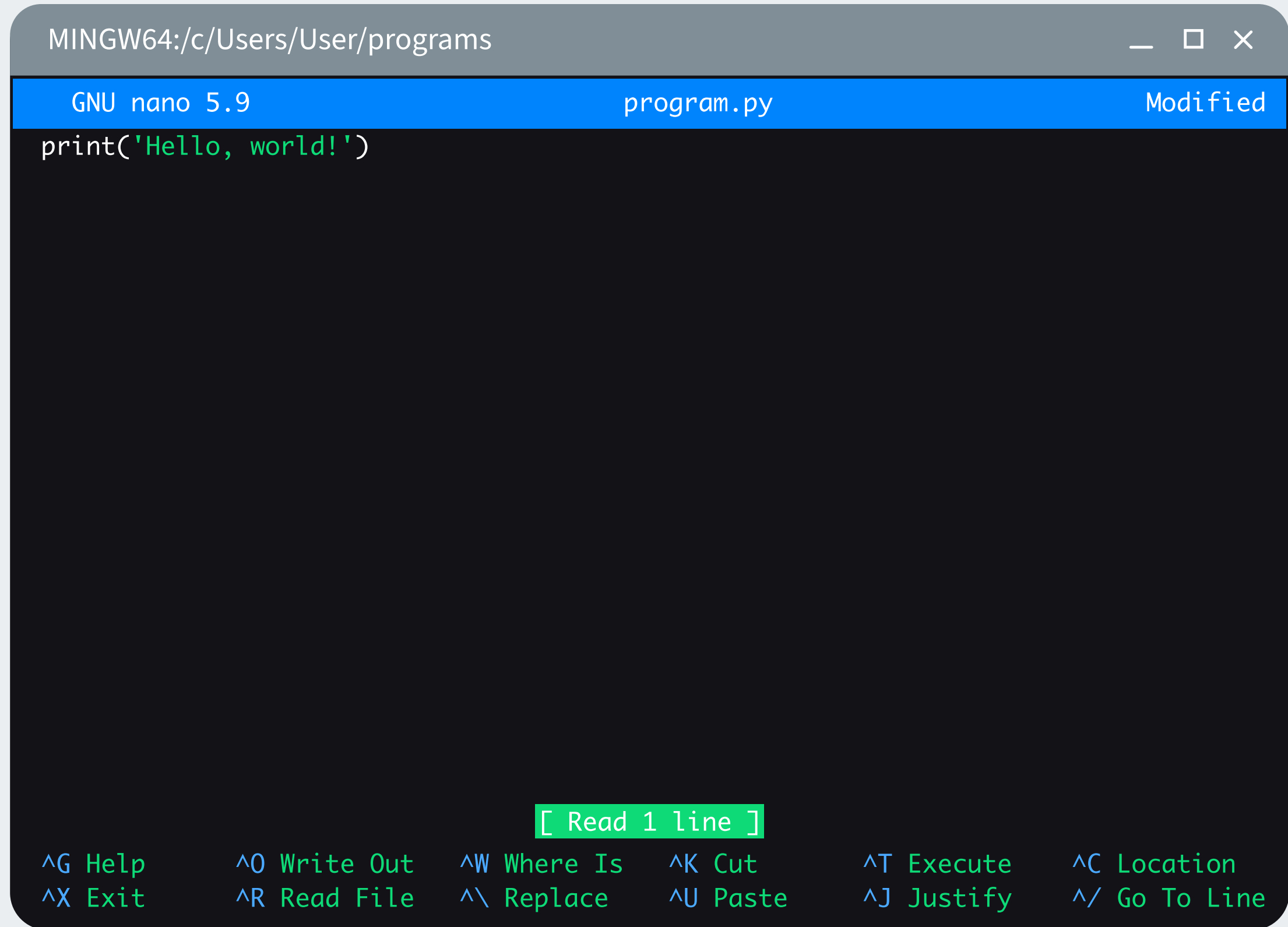
```
User@DESKTOP-PVGUDRB MINGW64 ~/programs  
$ ls  
program.py
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs  
$
```

Редактирование файлов

При работе на сервере у вас не будет программ с графическим интерфейсом, но при этом иногда необходимо иметь возможность отредактировать файл прямо в терминале. Воспользуемся встроенным текстовым редактором Nano. Введите команду:

```
nano program.py
```



```
MINGW64:/c/Users/User/programs
```

GNU nano 5.9	program.py	Modified
print('Hello, world!')		

```
[ Read 1 line ]
```

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^\ Replace	^U Paste	^J Justify	^/ Go To Line

Запуск программы через терминал

Чтобы запустить программу в терминале, нужно через интерпретатор Python вызвать программу по имени. Для этого используйте команду:

```
python program.py
```

MINGW64:/c/Users/User/programs

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs  
$ python program.py  
Hello, world!
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs  
$
```

Передача аргументов

Получить список аргументов поможет функция `argv` модуля `sys`.

Измените код `program.py` и запустите программу, пробуя передавать произвольные аргументы, например:

```
python program.py -a --arg 1 2 3
```

MINGW64:/c/Users/User/programs



GNU nano 5.9

program.py

Modified

```
import sys

print('program name:', sys.argv[0])
print('args:', sys.argv[1:])
```

[Read 4 lines]

[^]G Help
[^]X Exit

[^]O Write Out
[^]R Read File

[^]W Where Is
[^]\ Replace

[^]K Cut
[^]U Paste

[^]T Execute
[^]J Justify

[^]C Location
[^]/ Go To Line

1

Задача

Измените код `program.py` так, чтобы при запуске программы с двумя целочисленными аргументами программа выводила бы их сумму.

MINGW64:/c/Users/User/programs

User@DESKTOP-PVGUDRB MINGW64 ~/programs

\$ python program.py 2 3

5

User@DESKTOP-PVGUDRB MINGW64 ~/programs

\$

1

Задача

Решение:

MINGW64:/c/Users/User/programs

GNU nano 5.9

program.py

```
import sys
```

```
print(int(sys.argv[1]) + int(sys.argv[2]))
```

^G Help

^X Exit

^O Write Out

^R Read File

^W Where Is

^\ Replace

^K Cut

^U Paste

^T Execute

^J Justify

^C Location

^/ Go To Line

2

Задача

Создайте две папки в домашней директории:



first_folder — оставьте пустой.



second_folder — в ней создайте два файла first.txt и second.txt.

Должна получиться следующая структура:

~

└─ first_folder

└─ second_folder

│ └─ first.txt

│ └─ second.txt

2

Задача

Решение:

MINGW64:/c/Users/User/second_folder

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$ mkdir first_folder
```

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$ mkdir second_folder
```

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$ cd second_folder/
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/second_folder  
$ touch first.txt
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/second_folder  
$ touch second.txt
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/second_folder  
$ ls  
first.txt second.txt
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/second_folder  
$
```

Удаление файлов и директорий

Чтобы удалить файл, используется команда `rm` (remove) и имя файла. Находясь в папке `second_folder`, удалите файл `first`:

```
rm first
```

Теперь перейдите в директорию выше и удалите папку `first_folder`. Для этого используется команда `rmdir` (remove directory) и имя папки:

```
rmdir first_folder
```

MINGW64:/c/Users/User

```
User@DESKTOP-PVGUDRB MINGW64 ~/second_folder  
$ rm first.txt
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/second_folder  
$ cd ..
```

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$ rmdir first_folder/
```

```
User@DESKTOP-PVGUDRB MINGW64 ~  
$
```

Удаление файлов и директорий

Теперь попробуйте удалить папку `second_folder`.

Так как она не пуста, то для удаления папки со всем ее содержимым необходимо воспользоваться командой `rm` с ключом `-r`:

```
rm -r second_folder
```

MINGW64:/c/Users/User

User@DESKTOP-PVGUDRB MINGW64 ~

```
$ rmdir second_folder/
```

rmdir: failed to remove 'second_folder/': Directory not empty

User@DESKTOP-PVGUDRB MINGW64 ~

```
$ rm -r second_folder/
```

User@DESKTOP-PVGUDRB MINGW64 ~

```
$
```

Итоги

Вспомним основные команды в терминале:

Действие

Команда



Удалить директорию

`rmdir <имя папки>`



Создать файл

`touch <имя файла>`



Создать новую папку

`mkdir <имя папки>`



Выводит в терминал путь к текущей папке

`pwd`



Удалить файл

`rm <имя файла>`



Удалить директорию если она не пуста

`rm -r <имя папки>`



Используется для перемещения по папкам

`cd`



Выводит список директорий и файлов в текущей директории

`ls`

argparse

Для удобной работы с аргументами командной строки можно воспользоваться встроенной библиотекой argparse.

Модуль argparse позволяет:



Анализировать аргументы `sys.argv`.



Конвертировать строковые аргументы в Python объекты.



Указывать обязательные и необязательные аргументы.



Указывать типы аргументов, значения по умолчанию.



Отображать сообщения с ошибками и подсказками.



Привязывать выполнение действий к аргументам.



И многое другое...

argparse

Чтобы использовать argparse в вашей программе, необходимо сделать четыре действия:

- 1 Импортировать argparse.
- 2 Создать экземпляр класса ArgumentParser.
- 3 Добавить аргументы и параметры в парсер с помощью метода add_argument().
- 4 Вызвать метод parse_args(), чтобы получить пространство имен аргументов и работать с ними как с атрибутами объекта.

Пример



✓ Измените код program.py в привычной IDE:

```
import argparse

parser = argparse.ArgumentParser(
    description='Print arg'
)
parser.add_argument('arg')
args = parser.parse_args()
print(args.arg)
```

✓ Попробуйте запустить программу в терминале:

```
python program.py
```

✓ Попробуйте запустить программу с ключом --help:

```
python program.py --help
```

MINGW64:/c/Users/User/programs

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$ python program.py
usage: program.py [-h] arg
program.py: error: the following arguments are required: arg
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$ python program.py --help
usage: program.py [-h] arg

Print arg

positional arguments:
  arg

options:
  -h, --help    show this help message and exit
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$ python program.py 1
1
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$
```


Позиционные и опциональные аргументы

Документация модуля определяет два разных типа аргументов командной строки:

- 1 Позиционные аргументы (аргументы)
- 2 Опциональные аргументы (ключи)

В нашем примере `arg` — это позиционный аргумент, т. к. его положение в списке аргументов определяет его назначение. Опциональные аргументы не являются обязательными и позволяют изменить поведение команды.

Пример



✓ Расширим функционал нашей программы:

```
import argparse

parser = argparse.ArgumentParser(
    description='Print sum or avg of the number'
)
parser.add_argument('numbers', nargs='+', type=int)
parser.add_argument(
    '--avg',
    action='store_true',
    help='Print avg, default sum'
)
args = parser.parse_args()
total = sum(args.numbers)
if args.avg:
    print(total / len(args.numbers))
else:
    print(total)
```

MINGW64:/c/Users/User/programs

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$ python program.py --help
usage: program.py [-h] [--avg] numbers [numbers ...]
```

Print sum or avg of the number

positional arguments:
numbers

options:
-h, --help show this help message and exit
--avg Print avg, default sum

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$ python program.py 1 2 3
6
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$ python program.py 1 2 3 --avg
2.0
```

```
User@DESKTOP-PVGUDRB MINGW64 ~/programs
$
```

Параметры `add_argument()`

Чтобы определить, как анализировать конкретный аргумент, можно использовать дополнительные параметры `add_argument()`. Рассмотрим некоторые из них:

★ `name` — имя (или имена) позиционного или опционального аргумента, например, `arg` или `-arg`, `--arg`

★ `action` — тип действия, которое должно быть выполнено, если этот аргумент встречается в командной строке, может быть: `'store'`, `'store_const'`, `'store_true'`, `'append'`, `'append_const'`, `'count'`, `'help'`, `'version'`

★ `nargs` — количество аргументов командной строки, которые будут использоваться, может быть: конкретным числом, `'?'`, `'*'`, `'+'`

★ `default` — значение по умолчанию, если аргумент отсутствует в командной строке

Параметры `add_argument()`

Чтобы определить, как анализировать конкретный аргумент, можно использовать дополнительные параметры `add_argument()`. Рассмотрим некоторые из них:

- ✦ `type` — тип, в который должен быть преобразован аргумент командной строки
- ✦ `choices` — список допустимых значений для аргумента
- ✦ `required` — можно ли пропустить параметр командной строки (только для дополнительных параметров)
- ✦ `help` — краткое описание того, что делает аргумент

Документация по модулю

Со всеми возможностями модуля argparse можно ознакомиться на странице официальной документации.

