Chapter - 7: File Processing

Command Line Arguments

argparse Module

- It implements the parser for command line arguments.
- To use this module, it is required to import the **argparse Module. e.g. import argparse.**
- It has ArgumentParser( ) method to create the parser for the arguments.
- The parser has .add_argument( ) method define the argument and store the value.
- The parser has the parse_args( ) method which parse the sequence argument taken from sys.argv[1:]. And it returns the namespace containing the argument with its attribute .

Argument Action Supported in Add_Argument( ) Method

SNo	Action Name	Description
1	store	Save the value, after optionally converting it to a different type. This is the default action taken if none is specified explicitly.
2.	store_const	Save a value defined as part of the argument specification, rather than a value that comes from the arguments being parsed. This is typically used to implement command-line flags that are not Booleans.
3.	store_true / store_false	Save the appropriate Boolean value. These actions are used to implement Boolean switches.
4	append	Save the value to a list. Multiple values are saved if the argument is repeated.
5	append_const	Save a value defined in the argument specification to a list.
6	version	Prints version details about the program and then exits.

Syntax

```
parser.add_argument("-s", action='store',
                    dest="square",
                    help="Display a square of a given number",
                    default=0
                    type=int)
```

Example

Program #1

```
# cli_py.py script file to print the square of the command line argument
import argparse
parser=argparse.ArgumentParser( )
parser.add_argument("square", help="Display a square of a given number",
type=int)

args=parser.parse_args()
print("Square= ",args.square**2)
```

Output

Set-1

```
C:\Python38> python cli_ap.py 4
Square= 16
```

Set-2

```
C:\Python38>python cli_ap.py -h
usage: cli_ap.py [-h] square

positional arguments:
  square      Display a square of a given number

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

Set-3

```
C:\Python38>python cli_ap.py
usage: cli_ap.py [-h] square
cli_ap.py: error: the following arguments are required: square
```

Set-4

```
C:\Python38>python cli_ap.py 4 5
usage: cli_ap.py [-h] square
cli_ap.py: error: unrecognized arguments: 5
```

Program #2

```
# cli_py.py script file the square and cube of the command line argument passed

import argparse
parser=argparse.ArgumentParser( )

parser.add_argument("-s", action='store',
```

```
                dest="square",
                help="Display a square of a given number",
                type=int)

parser.add_argument("-c", action='store',
                dest="cube",
                help="Display a cube of a given number",
                type=int)

args=parser.parse_args()

print(args)

if  args.square :
   print("Square= ",args.square**2)

if  args.cube :
    print("Cube= ",args.cube**3)
```

Output

Set-1

```
C:\Python38>python cli_ap.py -h
usage: cli_ap.py [-h] [-s SQUARE] [-c CUBE]

optional arguments:
  -h, --help  show this help message and exit
  -s SQUARE   Display a square of a given number
  -c CUBE     Display a cube of a given number
```

Set-2

```
C:\Python38>python cli_ap.py
Namespace(cube=None, square=None)
```

Set-3

```
C:\Python38>python cli_ap.py
Namespace(cube=None, square=None)
```

Set-4

```
C:\Python38>python cli_ap.py  -s 4 -c 5
Namespace(cube=5, square=4)
Square=  16
Cube=  125
```

Set-5

```
C:\Python38>python cli_ap.py  -s 4
Namespace(cube=5, square=4)
Square=  16
```

Program #3

```
# cli_py.py script to print the sum of numbers passed as command line argument

import argparse
parser=argparse.ArgumentParser( )

parser.add_argument("-a", action='append',
                dest="Number",
                help="Display a sum of number",
                type=int)

args=parser.parse_args()
print(args)

s=0
for arg in args.Number:
   s= s+ arg

if  args.Number :
  print("Sum= ",s)
```

Output

Set-1

```
C:\Python38>python cli_ap.py -h
usage: cli_ap.py [-h] [-a NUMBER]

optional arguments:
  -h, --help  show this help message and exit
  -a NUMBER   Display a sum of number
```

Set-2

```
C:\Python38>python cli_ap.py -a 10 -a 20 -a 30
Namespace(Number=[10, 20, 30])
Sum=  60
```

Set-3

```
C:\Python38>python cli_ap.py -a 10 -a 20 -a 30 -a 40
Namespace(Number=[10, 20, 30, 40])
Sum=  100
```