1)

import argparse  
  
parser = argparse.ArgumentParser()  
parser.add\_argument("user\_input", type=str)  
args = parser.parse\_args()  
  
if args.user\_input:  
 try:  
 print(eval(args.user\_input))  
 except SyntaxError:  
 print("Input error (Syntax error)")

2)

import operator  
import argparse  
  
parser = argparse.ArgumentParser()  
parser.add\_argument("Operator", type=str)  
parser.add\_argument("Number1", type=str)  
parser.add\_argument("Number2", type=str)  
args = parser.parse\_args()  
  
ds = {'add': operator.add, 'sub': operator.sub, 'div': operator.truediv, "mul": operator.mul}  
first\_num = int(args.Number1)  
operation = str(args.Operator)  
second\_num = int(args.Number1)  
print(ds[operation](first\_num, second\_num))

3)

import operator  
import argparse  
  
parser = argparse.ArgumentParser()  
parser.add\_argument("Formula", type=str)  
args = parser.parse\_args()  
  
def check(string, curr\_pos):  
 if not string[curr\_pos].isdigit() and (string[curr\_pos] not in oper\_list or string[curr\_pos - 1] in oper\_list):  
 return False  
 elif not curr\_pos:  
 if string[len(string) - 1] not in oper\_list:  
 return True  
 else:  
 return check(string, curr\_pos - 1)  
  
  
user\_input = args.Formula  
# parameter list  
oper\_list = ['+', '-']  
# list of operation signs  
  
if not user\_input:  
 # check if there is an input  
 print("No parameters")  
else:  
 user\_input\_str = "".join(user\_input)  
 # converting a list of parameters into string  
 if check(user\_input\_str, len(user\_input\_str) - 1):  
 # if string is correct, printing the answer  
 # else, printing the "False" message  
 print("True", eval(user\_input\_str), sep=", ")  
 else:  
 print("False, None")

4)

import argparse  
  
parser = argparse.ArgumentParser()  
parser.add\_argument("-W", type=int)  
parser.add\_argument("-w", nargs='+', type=int)  
parser.add\_argument("-n", type=int)  
args = parser.parse\_args()  
  
  
def knapSack(W, w, n):  
 n = len(w)  
 table = [[0 for x in range(W + 1)] for x in range(n + 1)]  
 for i in range(n + 1):  
 for j in range(W + 1):  
 if i == 0 or j == 0:  
 table[i][j] = 0  
 elif w[i - 1] <= j:  
 table[i][j] = max(w[i - 1] + table[i - 1][j - w[i - 1]], table[i - 1][j])  
 else:  
 table[i][j] = table[i - 1][j]  
  
 return table[n][W]  
  
knapSack(args.W, args.w, args.n)

(knapSack(W, wt))