# Optimal Solution of Linear Programming Problem Using Python and TORA: A Comparative Analysis

**Mayuri B. Babar*, Varsha J. Patil**

**Department of Mathematics, Dr. D. Y. Patil Arts, Commerce and Science College, Pimpri, Pune-18, Maharashtra, India**

**DOI: https://doi.org/10.51583/IJLTEMAS.2025.1413SP040**

**Received: 26 June 2025; Accepted: 30 June 2025; Published: 25 October 2025**

**Abstract:** Linear Programming (LP) is a mathematical method to optimize results under certain constraints. With the improvement of computational aids, LP problems have become easy to solve. This study compares two well-known aids: Python's PULP Library and the TORA software. Applying a new LP problem, both aids are compared for accuracy, usability and application for various user groups. The research aims to guide students, teachers and practitioners on choosing the best tool depending on their requirements.

**Keywords:** Linear Programming Problem, TORA Software, Python Software

## I. Introduction

Linear Programming problem (LPP) has wide applications in logistics, finance, operations management and production. Solving LP problems calculative is time-consuming and prone to errors. Thus, software tools such as Python and TORA have gained popularity. As real-world LP problems often involve numerous variables and constraints, solving them manually not only become tedious and time-consuming but also increase the likelihood of errors in calculation and interpretation. To overcome these limitations, software tools have become integral in solving LP models efficiently and accurately. Among these tools, Python and TORA have emerged as popular choices for both academic and practical applications. Python, a versatile programming language, offers powerful libraries like PULP. On the other hand, TORA provides a user-friendly graphical interface, primarily used in educational settings to illustrate the steps of the Simplex method and other operations research algorithms.

This paper evaluates both tools by solving the same LP problem using Python's programmatic approach using PULP library and TORA's Simplex method interface.

### Problem Formulation

The new Linear Programming Problem is:

Maximize $\boldsymbol{z = 3x_1 + 6x_2 + 2x_3}$

Subject to constraints:

$3x_1 + 3x_2 + x_3 \leq 10$

$x_1 + 2x_2 + 3x_3 \leq 15$

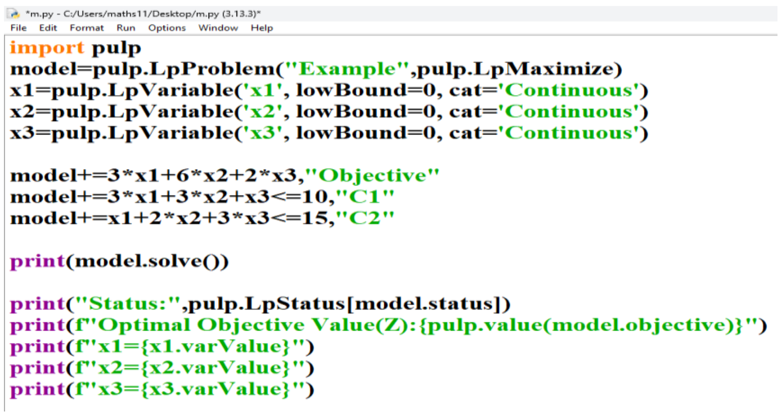
$x_1, x_2, x_3 \geq 0$

This problem represents an optimization scenario such as resource allocation or production planning.

## II. Methodology

### Solving with Python

### Input


*m.py - C:/Users/maths11/Desktop/m.py (3.13.3)*
File Edit Format Run Options Window Help

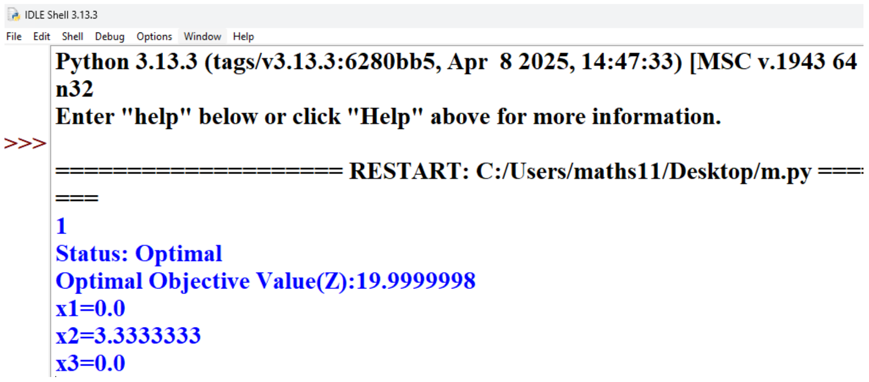
```
import pulp
model=pulp.LpProblem("Example",pulp.LpMaximize)
x1=pulp.LpVariable('x1', lowBound=0, cat='Continuous')
x2=pulp.LpVariable('x2', lowBound=0, cat='Continuous')
x3=pulp.LpVariable('x3', lowBound=0, cat='Continuous')

model+=3*x1+6*x2+2*x3,"Objective"
model+=3*x1+3*x2+x3<=10,"C1"
model+=x1+2*x2+3*x3<=15,"C2"

print(model.solve())

print("Status:",pulp.LpStatus[model.status])
print(f"Optimal Objective Value(Z):{pulp.value(model.objective)}")
print(f"x1={x1.varValue}")
print(f"x2={x2.varValue}")
print(f"x3={x3.varValue}")
```

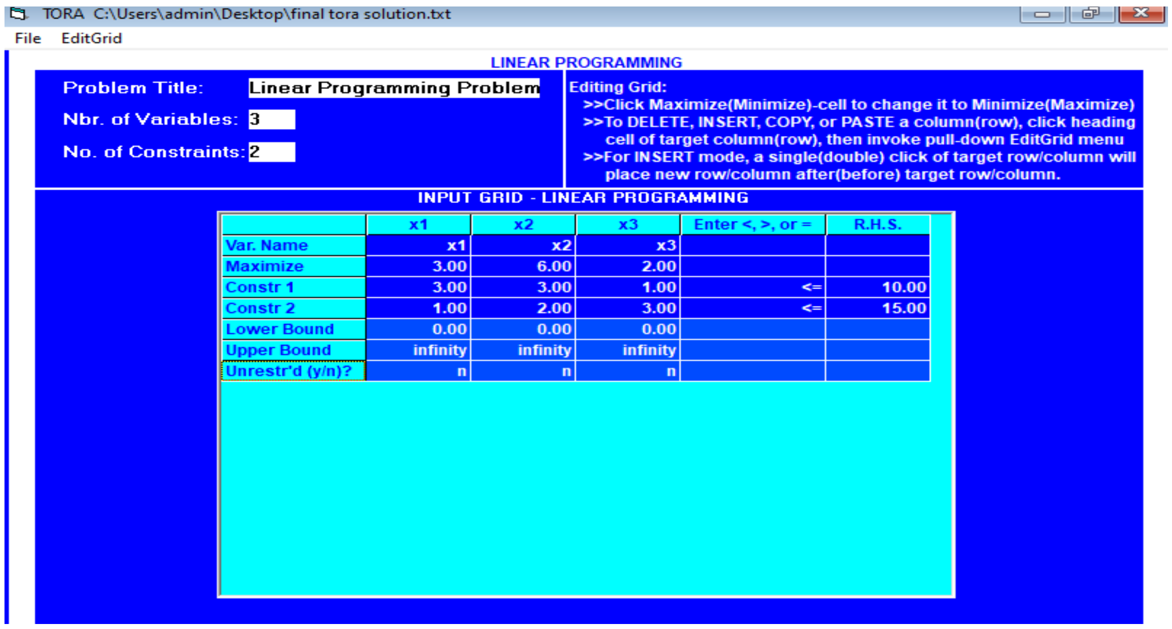
**Output**



```
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64
n32
Enter "help" below or click "Help" above for more information.
>>>
==================== RESTART: C:/Users/maths11/Desktop/m.py ===
===
1
Status: Optimal
Optimal Objective Value(Z):19.9999998
x1=0.0
x2=3.3333333
x3=0.0
```

**Solving with TORA**

**Input**



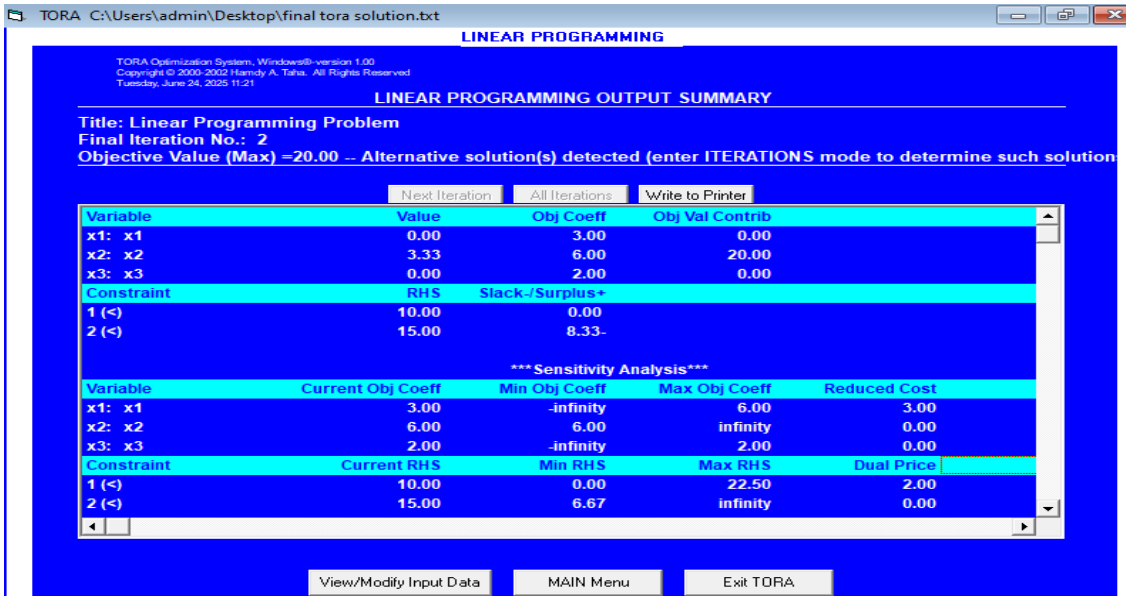
Problem Title: Linear Programming Problem
Nbr. of Variables: 3
No. of Constraints: 2

Editing Grid:
>>Click Maximize(Minimize)-cell to change it to Minimize(Maximize)
>>To DELETE, INSERT, COPY, or PASTE a column(row), click heading cell of target column(row), then invoke pull-down EditGrid menu
>>For INSERT mode, a single(double) click of target row/column will place new row/column after(before) target row/column.

INPUT GRID - LINEAR PROGRAMMING

| | x1 | x2 | x3 | Enter <, >, or = | R.H.S. |
|---|---|---|---|---|---|
| Var. Name | x1 | x2 | x3 | | |
| Maximize | 3.00 | 6.00 | 2.00 | | |
| Constr 1 | 3.00 | 3.00 | 1.00 | <= | 10.00 |
| Constr 2 | 1.00 | 2.00 | 3.00 | <= | 15.00 |
| Lower Bound | 0.00 | 0.00 | 0.00 | | |
| Upper Bound | infinity | infinity | infinity | | |
| Unrestr'd (y/n)? | n | n | n | | |

**Output**



```
TORA Optimization System, Windows®-version 1.00
Copyright © 2000-2002 Hamdy A. Taha. All Rights Reserved
Tuesday, June 24, 2025 11:21

                    LINEAR PROGRAMMING OUTPUT SUMMARY

Title: Linear Programming Problem
Final Iteration No.:  2
Objective Value (Max) =20.00 -- Alternative solution(s) detected (enter ITERATIONS mode to determine such solution

Variable                    Value      Obj Coeff    Obj Val Contrib
x1:  x1                      0.00           3.00               0.00
x2:  x2                      3.33           6.00              20.00
x3:  x3                      0.00           2.00               0.00
Constraint                    RHS   Slack-/Surplus+
1 (<)                       10.00           0.00
2 (<)                       15.00           8.33-

                         ***Sensitivity Analysis***
Variable         Current Obj Coeff  Min Obj Coeff   Max Obj Coeff    Reduced Cost
x1:  x1                      3.00       -infinity            6.00            3.00
x2:  x2                      6.00            6.00        infinity            0.00
x3:  x3                      2.00       -infinity            2.00            0.00
Constraint            Current RHS         Min RHS         Max RHS      Dual Price
1 (<)                       10.00            0.00           22.50            2.00
2 (<)                       15.00            6.67        infinity            0.00
```

## III. Results:

Both tools gives the same optimal solution:

**Optimal Solution by using Python**

$x_1$ = 0.00, $x_2$ = 3.3333333, $x_3$ = 0.00

$z_{max} = 19.9999998 \sim 20$

**Optimal Solution by using TORA**

$x_1$ = 0.00, $x_2$ = 3.33, $x_3$ = 0.00

$z_{max} = 20$

**Comparison Table:**

| Criteria | Python (PULP) | TORA Software |
|---|---|---|
| Ease of Use | Requires programming Skills | Simple GUI, ideal for beginners |
| Output Detail | Code-based output | Step-by-step tableau |
| Speed | Very fast | Fast for small problems |
| Flexibility | High, allows for automation/scripting | Limited to small-scale problems |
| Graphical Support | Needs extra libraries (e.g., matplotlib) | Built-in for 2-variable cases |
| Suitability | Research, automation, large datasets | Education, small manual problems |

## IV. Discussion:

Both software show efficacy in identifying the best solution. Python's programmatic style provides us with speed, automation, and interfacing into real-world systems such as logistics systems and business application.

TORA, however, is useful for teaching and learning. It decomposes every step of the Simplex method so that a learner can see how the algorithm unfolds internally.

Python does need minimum programming skill but has high scalability and customizability. TORA is ideal for low-scale Problems.

## V. Conclusion:

This comparative analysis concludes that Python and TORA are both great tools to solve LP Problems. Yet, their usability is context-dependent: For students and teachers, TORA is perfect as it involves visual, step-by-step learning of the Simplex method. Python is better for researchers, engineers and analysts who require

Scalability and integration with larger systems. Overall, Python is the best choice for professional and industrial use, while TORA is best used in academic environments.

## References:

1. Hamdy A. TAHA, Operations Research (Seventh Edition)
2. Introduction to Machine Learning with Python