Example

Most of the following is copied from https://github.com/benalexkeen/Introduction-to-linear-programming/blob/master/Introduction%20to%20Linear%20Programming%20with%20Python%20-%20Part%202.ipynb

LP Problem

Maximize $3x + 5y + 5$

Subject to

$2x + 3y \leq 12$

$-x + y <= 3$

$x \leq 4$

$y \leq 3$

$2y \leq 25 - x$

$4y \geq 2x - 8$

$y \leq 2x - 5$

and $x, y \geq 0$

We import pulp, in the following two cells.

In [1]:

```
import sys
!{sys.executable} -m pip install pulp
```

```
Requirement already satisfied: pulp in c:\users\muntakim\appdata\local\programs\python\python38-32\lib\site-packages (2.4)
Requirement already satisfied: amply>=0.1.2 in c:\users\muntakim\appdata\local\programs\python\python38-32\lib\site-packages (from pulp) (0.1.4)
Requirement already satisfied: pyparsing in c:\users\muntakim\appdata\local\programs\python\python38-32\lib\site-packages (from amply>=0.1.2->pulp) (2.4.7)
Requirement already satisfied: docutils>=0.3 in c:\users\muntakim\appdata\local\programs\python\python38-32\lib\site-packages (from amply>=0.1.2->pulp) (0.17.1)
```

In [2]:

```
import pulp
```

To use the PuLP package, do the previous steps first before proceeding.

After installing PuLP, we can set up our problem to solve. First, we define it.

In [3]:

```
# Create a LP Minimization problem.
Lp_prob = pulp.LpProblem('Your_LP_Problem', pulp.LpMaximize)
# We set up the problem using the command LpProblem in the PuLP package.
```

`#` makes the righthand side a commment, which doesn't run as code.

pulp.LpProblme <-- pulp is the package, `pulp.LpProblem` means we are using the class `LpProblem` in the pulp package.

For minimization problems, use `pulp.LpMinimize` .

Here, `Your_LP_Problem` is the name of the problem which shows up when we display the problem. We used `_` as spaces are not permitted in the name.

In [4]:
```
# Create problem decision variables.

# Create a variable x >= 0.  "x" means we put `x' when printing this variable.
x = pulp.LpVariable("x")
# Create a variable y >= 0.
y = pulp.LpVariable("y")
```

We used the `LpVariable` class.

Lower and Upper bounds can be assigned using the 'lowBound' and 'upBound' parameter instead.

For example, `x = pulp.LpVariable("x", lowBound = 0)` creates a variable x >= 0 and `y = pulp.LpVariable("y", upBound = 10)` creates a variable y <= 10.

We now set up our LP problem.

In [5]:
```
# Objective Function
Lp_prob += 3 * x + 5 * y +5

# We put objective function first then constraints.

# Constraints:
Lp_prob += 2 * x + 3 * y <= 12
Lp_prob += -x + y <= 3
Lp_prob += x <= 4
Lp_prob += y <= 3
Lp_prob += 2 * y <= 25 - x
Lp_prob += 4 * y >= 2 * x - 8
Lp_prob += y <= 2 * x - 5
Lp_prob += x >= 0
Lp_prob += y >= 0
```

The objective function and constraints are added using the += operator to our model. The objective function is added first, then the individual constraints.

In [6]:
```
# Display the problem
print(Lp_prob)
```

```
Your_LP_Problem:
MAXIMIZE
3*x + 5*y + 5
SUBJECT TO
_C1: 2 x + 3 y <= 12
```

```
_C2: - x + y <= 3

_C3: x <= 4

_C4: y <= 3

_C5: x + 2 y <= 25

_C6: - 2 x + 4 y >= -8

_C7: - 2 x + y <= -5

_C8: x >= 0

_C9: y >= 0

VARIABLES
x free Continuous
y free Continuous
```

You realize that the inequalities are rearranged to put numbers only in the right hand side.

We can solve this LP using the `solve` function. `Lp_prob.solve` means apply the `solve` function to the `Lp_prob` object we defined.

In [7]:

```python
Lp_prob.solve()
pulp.LpStatus[Lp_prob.status]
```

Out[7]: 'Optimal'

It solved the LP problem and gave the result: There are 5 status codes:

- Not Solved: Status prior to solving the problem.
- Optimal: An optimal solution has been found.
- Infeasible: There are no feasible solutions.
- Unbounded: The constraints are not bounded, maximising the solution will tend towards infinity.
- Undefined: The optimal solution may exist but may not have been found.

We can now view our optimal variable values and the optimal value of Z.

In [8]:

```python
# Printing the final solution
print("x=", pulp.value(x), "y=", pulp.value(y), "z=", pulp.value(Lp_prob.objective))
```

```
x= 3.375 y= 1.75 z= 23.875
```

Another way to show the solutions:

In [9]:

```python
for variable in Lp_prob.variables():
        print(variable.name, "=", variable.varValue)
print("Optimal value is z = ", pulp.value(Lp_prob.objective))
```

```
x = 3.375
y = 1.75
Optimal value is z =  23.875
```