Linear Optimization Sample 1

- A store has requested a manufacturer to produce Pants and Sports Jackets
- For Materials, the manufacturer has
  - Cotton - 750 m^2
  - Polyester - 1000 m^2
- Every pair of Pants needs
  - Cotton - 1 m^2
  - Polyster - 2 m^2
- Every pair of Jacket needs
  - Cotton - 1.5 m^2
  - Polyster - 1 m^2
- Price of Pant is $50
- Price of Jacket is $40

What is the number of pants and jackets that the manufacturer must give to the stores so that these obtain a maximum sale?

Objective Function

Maximize the Revenue of the Store

$$J = 50 * n_p + 40 * n_j$$

- where J is the revenue of the store
- n_p is the number of Pants sold
- n_j is the number of Jackets sold

Decision Variables

- Number of Jackets - n_j
- Number of Pants - n_p

Constraints

- Total Amount of Cotton is 750 m^2

  $$1 * n_p + 1.5 * n_j <= 750$$

- Total Amount of Polyester is 1000 m^2

$$2 * n_p + 1 * n_j <= 1000$$

- Number of Pants and Jackets is greater than equal to 0

$$n_p >= 0$$
$$n_j >= 0$$

In [79]:

```
from ortools.linear_solver import pywraplp
```

In [80]:

```
def DisplaySolution():
    if status == pywraplp.Solver.OPTIMAL:
        print('Objective value =', solver.Objective().Value())
        for j in range(data['num_vars']):
            print(x[j].name(), ' = ', x[j].solution_value())
        print()
        print('Problem solved in %f milliseconds' % solver.wall_time())
        print('Problem solved in %d iterations' % solver.iterations())
        print('Problem solved in %d branch-and-bound nodes' % solver.nodes())
    else:
        print('The problem does not have an optimal solution.')
```

In [81]:

```
def create_data_model():
    data = {}
    data['constraint_coeffs'] = [
        [1, 1.5],
        [2, 1],
        [-1, 0],
        [0, -1],
    ]

    data['bounds'] = [750, 1000, 0, 0]
    data['obj_coeffs'] = [50, 40]
    data['num_vars'] = 2
    data['num_constraints'] = 4

    return data
```

In [82]:

```
# Get the data and Create the Solver
data = create_data_model()
solver = pywraplp.Solver.CreateSolver('SCIP')
```

In [83]:

```
# Define the decision variables
infinity = solver.infinity()
x = {}
for j in range(data['num_vars']):
    x[j] = solver.IntVar(0, infinity, 'x[%i]' % j)
print('Number of Variables = ', solver.NumVariables())
```

```
Number of Variables =  2
```

In [84]:

```
# Define the Constraints
for i in range(data['num_constraints']):
    constraint_expr = [data['constraint_coeffs'][i][j] * x[j] for j in range(
    solver.Add(sum(constraint_expr) <= data['bounds'][i])


print('Number of Constraints = ', solver.NumConstraints())
```

```
Number of Constraints =  4
```

In [85]:

```
# Define the Objective
objective = solver.Objective()

obj_expr = [data['obj_coeffs'][j] * x[j] for j in range(data['num_vars'])]
solver.Maximize(solver.Sum(obj_expr))
```

In [86]:

```
status = solver.Solve()
```

In [87]:

```
DisplaySolution()
```

```
Objective value = 28750.000000000004
x[0]  =  375.0
x[1]  =  250.0

Problem solved in 240.000000 milliseconds
Problem solved in 2 iterations
Problem solved in 1 branch-and-bound nodes
```