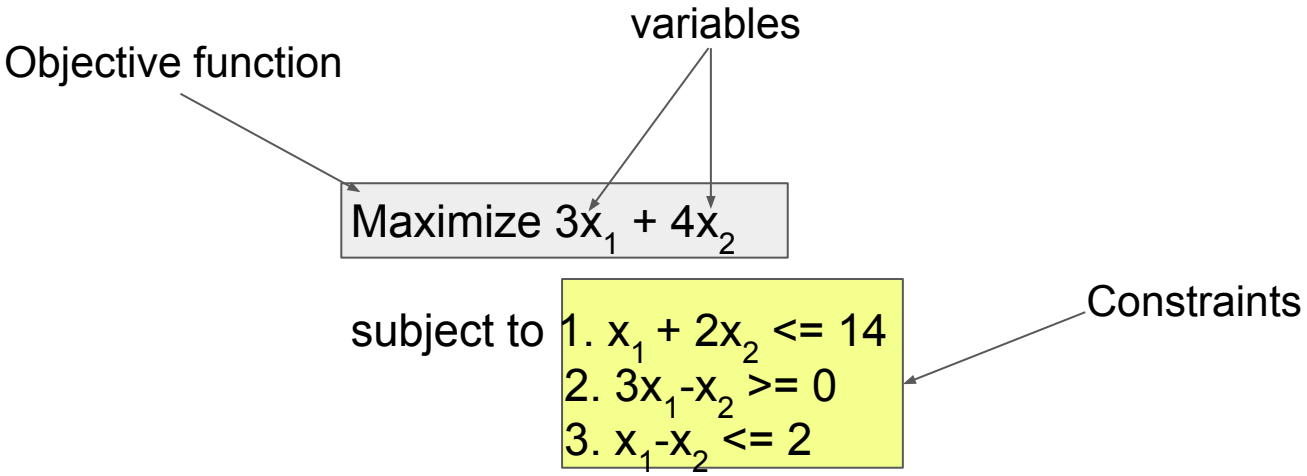
Computing Lab

Linear Programming and Integer Programming

Linear programming?



Tutorial on using OR-Tools to solve linear programming problems in Python

Installations

You will need to use the pywraplp module from the ortools package.

```
pip install ortools
```

Import the required libraries

You will need to import the pywraplp module from the ortools.linear_solver package.

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

Declare the solver

- Create a solver variable that will contain all the necessary items to solve the problem.

- You can do this by calling the CreateSolver() method of the pywraplp solver class and passing in the name of the solver you want to use. For linear programming problems, you can use the GLOP solver.

```
solver = pywraplp.Solver.CreateSolver('GLOP')
```

Create the variables

- Next, you will need to create variables for your problem.

- You can do this by calling the NumVar() method of the solver object and passing in the lower and upper bounds for the variable, as well as a name for the variable.

```
x = solver.NumVar(0, solver.infinity(), 'x')
y = solver.NumVar(0, solver.infinity(), 'y')
```

Create the variables (Contd.)

- If you need to create variables for a Integer Optimization problem.

- You can define integer variables using the IntVar() method of the solver object and passing in the lower and upper bounds for the variable, as well as a name for the variable.

```
# x and y are integer non-negative variables.
x = solver.IntVar(0.0, infinity, "x")
y = solver.IntVar(0.0, infinity, "y")
```

Define the constraints

- After creating the variables, you will need to define the constraints for your problem.

- You can do this by calling the Add() method of the solver object and passing in a linear expression that represents the constraint.

```
# Constraint 0: x + 2y <= 14.
solver.Add(x + 2 * y <= 14.0)

# Constraint 1: 3x - y >= 0.
solver.Add(3 * x - y >= 0.0)

# Constraint 2: x - y <= 2.
solver.Add(x - y <= 2.0)
```

Define the constraints (Contd.)

- Alternately, You can do this by calling the Constraint() and SetCoefficient() method of the solver object that represents the constraint.

```
# Create a linear constraint, 0 <= x + y <= 2.
ct = solver.Constraint(0, 2, "ct")
ct.SetCoefficient(x, 1)
ct.SetCoefficient(y, 1)
```

Define the objective function

- Next, you will need to define the objective function for your problem.

- You can do this by calling either the Maximize() or Minimize() method of the solver object, depending on whether you want to maximize or minimize the objective function, and passing in a linear expression that represents the objective function.

```
# Objective function: 3x + 4y.
solver.Maximize(3 * x + 4 * y)
```

Define the objective function (Contd.)

- Alternately, you can represent the objective function using Objective() and SetCoefficient() methods of the solver object.

```
# Create the objective function, 3 * x + y.
objective = solver.Objective()
objective.SetCoefficient(x, 3)
objective.SetCoefficient(y, 1)
objective.SetMaximization()
```

Invoke the solver

- After defining all of the necessary components of your problem, you can solve it by calling the Solve method of the solver object.

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

Display the solution

- Finally, you can display the solution to your problem by accessing the values of your variables using their solution_value() attribute for the variables.

```
# objective function value
print(solver.Objective().Value())
# x and y values respectively
print("x =", x.solution_value())
print("y =", y.solution_value())
```

Thank You