**pip install python-constraint**

**Example: 1**

import constraint

problem = constraint.Problem()

problem.addVariable('x', [1,2,3])

problem.addVariable('y', range(10))

def our\_constraint(x, y):

if x + y >= 5:

return True

problem.addConstraint(our\_constraint, ['x','y'])

solutions = problem.getSolutions()

for solution in solutions:

print(solution)

**Example: 2**

import constraint

problem = constraint.Problem()

# (coin\_value\*num\_of\_coints) <= 60

problem.addVariable("1 rupee", range(61))

problem.addVariable("2 rupee", range(31))

problem.addVariable("5 rupee", range(13))

problem.addVariable("10 rupee", range(7))

problem.addVariable("20 rupee", range(4))

problem.addConstraint(

constraint.ExactSumConstraint(60,[1,2,5,10,20]),["1 rupee", "2 rupee", "5 rupee","10 rupee", "20 rupee"])

def custom\_constraint(a, b, c, d, e):

if a + 2\*b + 5\*c + 10\*d + 20\*e == 60:

return True

problem.addConstraint(o, ["1 rupee", "2 rupee", "5 rupee","10 rupee", "20 rupee"])

def print\_solutions(solutions):

for s in solutions:

print("---")

print("""

1 rupee: {0:d}

2 rupee: {1:d}

5 rupee: {2:d}

10 rupee: {3:d}

20 rupee: {4:d}""".format(s["1 rupee"], s["2 rupee"], s["5 rupee"], s["10 rupee"], s["20 rupee"]))

print("Total:", s["1 rupee"] + s["2 rupee"]\*2 + s["5 rupee"]\*5 + s["10 rupee"]\*10 + s["20 rupee"]\*20)

print("---")

solutions = problem.getSolutions()

print\_solutions(solutions)

print("Total number of ways: {}".format(len(solutions)))

**Example: 3**

import constraint

problem = constraint.Problem()

problem.addVariable('A', range(31))

problem.addVariable('B', range(45))

problem.addVariable('C', range(76))

problem.addVariable('D', range(101))

def weight\_constraint(a, b, c, d):

if (a\*100 + b\*45 + c\*10 + d\*25) <= 3000:

return True

def volume\_constraint(a, b, c, d):

if (a\*8\*2.5\*0.5 + b\*6\*2\*0.5 \* c\*2\*2\*0.5 + d\*3\*3\*0.5) <= 1000:

return True

def value\_constraint(a, b, c, d):

if (a\*8 + b\*6.8 + c\*4 + d\*3) < 300:

return True

problem.addConstraint(weight\_constraint, "ABCD")

problem.addConstraint(volume\_constraint, "ABCD")

problem.addConstraint(value\_constraint, "ABCD")

maximum\_sweetness = 0

solution\_found = {}

solutions = problem.getSolutions()

for s in solutions:

current\_sweetness = s['A']\*10 + s['B']\*8 + s['C']\*4.5 + s['D']\*3.5

if current\_sweetness > maximum\_sweetness:

maximum\_sweetness = current\_sweetness

solution\_found = s

print("""

The maximum sweetness we can bring is: {}

We'll bring:

{} A Chocolates,

{} B Chocolates,

{} C Chocolates,

{} D Chocolates

""".format(maximum\_sweetness, solution\_found['A'], solution\_found['B'], solution\_found['C'], solution\_found['D']))