**https://python.hotexamples.com/fr/examples/pulp/LpProblem/-/python-lpproblem-class-examples.html**

**Python LpProblem - 30 exemples trouvés. Ce sont les exemples réels les mieux notés de pulp.LpProblem extraits de projets open source. Vous pouvez noter les exemples pour nous aider à en améliorer la qualité.**

**Langage de programmation: Python**

**Espace de nommage/Pack: pulp**

**Class/Type:** [**LpProblem**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/-/python-lpproblem-class-examples.html)

**Exemples au hotexamples.com: 30**

**Méthodes fréquemment utilisées**

[**solve (30)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/solve/python-lpproblem-solve-method-examples.html)

[**writeLP (8)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/writeLP/python-lpproblem-writelp-method-examples.html)

[**variables (7)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/variables/python-lpproblem-variables-method-examples.html)

[**setObjective (2)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/setObjective/python-lpproblem-setobjective-method-examples.html)

[**sortedVariables (2)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/sortedVariables/python-lpproblem-sortedvariables-method-examples.html)

[**add (1)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/add/python-lpproblem-add-method-examples.html)

[**addConstraint (1)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/addConstraint/python-lpproblem-addconstraint-method-examples.html)

[**objective (1)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/objective/python-lpproblem-objective-method-examples.html)

[**setSolver (1)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/setSolver/python-lpproblem-setsolver-method-examples.html)

[**writeMPS (1)**](https://python.hotexamples.com/fr/examples/pulp/LpProblem/writeMPS/python-lpproblem-writemps-method-examples.html)

**Related**

[**exp**](https://python.hotexamples.com/fr/examples/screw.screw/-/exp/python-exp-function-examples.html)

[**get\_branch\_name**](https://python.hotexamples.com/fr/examples/sponge.utils.repo/-/get_branch_name/python-get_branch_name-function-examples.html)

[**standardize**](https://python.hotexamples.com/fr/examples/limix.utils.preprocess/-/standardize/python-standardize-function-examples.html)

[**getSetting**](https://python.hotexamples.com/fr/examples/resources.lib.utils/-/getSetting/python-getsetting-function-examples.html)

[**numerical\_theta**](https://python.hotexamples.com/fr/examples/vollib.helper.numerical_greeks/-/numerical_theta/python-numerical_theta-function-examples.html)

[**pol2cart**](https://python.hotexamples.com/fr/examples/utils/-/pol2cart/python-pol2cart-function-examples.html)

[**setup\_vmedia**](https://python.hotexamples.com/fr/examples/ironic.drivers.modules.ilo.common/-/setup_vmedia/python-setup_vmedia-function-examples.html)

[**is\_tabular**](https://python.hotexamples.com/fr/examples/src.utils/-/is_tabular/python-is_tabular-function-examples.html)

[**is\_entity\_constructor\_command\_name**](https://python.hotexamples.com/fr/examples/trustedanalytics.meta.names/-/is_entity_constructor_command_name/python-is_entity_constructor_command_name-function-examples.html)

[**ArgumentRegion**](https://python.hotexamples.com/fr/examples/redis_lua.regions/ArgumentRegion/-/python-argumentregion-class-examples.html)

**report this ad**

**Related in langs**

[**extGetParam (PHP)**](https://hotexamples.com/fr/examples/-/-/extGetParam/php-extgetparam-function-examples.html)

[**Category (PHP)**](https://hotexamples.com/fr/examples/jorenvanhocht.blogify.models/Category/-/php-category-class-examples.html)

[**Header (C#)**](https://csharp.hotexamples.com/fr/examples/ActionInnocence.P2PScan.Plugins.Gnutella2.G2Packets/Header/-/php-header-class-examples.html)

[**Resource (C#)**](https://csharp.hotexamples.com/fr/examples/Poseidon/Resource/-/php-resource-class-examples.html)

[**initLanguages (C++)**](https://cpp.hotexamples.com/fr/examples/-/-/initLanguages/cpp-initlanguages-function-examples.html)

[**openfile (C++)**](https://cpp.hotexamples.com/fr/examples/-/-/openfile/cpp-openfile-function-examples.html)

[**GenerateName (Go)**](https://golang.hotexamples.com/fr/examples/github.com.bunin.flotilla.flotilla-server.daemon.broker/-/GenerateName/golang-generatename-function-examples.html)

[**GetBuildLast (Go)**](https://golang.hotexamples.com/fr/examples/github.com.ciscocloud.drone.store/-/GetBuildLast/golang-getbuildlast-function-examples.html)

[**DefaultFeatureCollection (Java)**](https://java.hotexamples.com/fr/examples/org.geotools.feature/DefaultFeatureCollection/-/java-defaultfeaturecollection-class-examples.html)

[**ITextBuilder (Java)**](https://java.hotexamples.com/fr/examples/-/ITextBuilder/-/java-itextbuilder-class-examples.html)

[**default (JS)**](https://javascript.hotexamples.com/fr/examples/client.isscrolledintoview/-/default/javascript-default-function-examples.html)

[**addFilter (JS)**](https://javascript.hotexamples.com/fr/examples/%40wordpress.hooks/-/addFilter/javascript-addfilter-function-examples.html)

**AddThis Sharing Buttons**

**Share to Google Bookmark**

**Google BookmarkShare to FacebookFacebookShare to TwitterTwitterShare to ImprimerImprimerShare to EmailEmailShare to Plus d'options...**

**Plus d'options...**

**Exemple #1**

**0**

**Fichier :** [**V\_bar\_search.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fpegahani%2FAdvance-Project%2Fblob%2Fmaster%2FFinal_project%2FV_bar_search.py) **Projet :** [**pegahani/Advance-Project**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fpegahani%2FAdvance-Project)

**def K\_dominnace\_check\_2(self, u\_d, v\_d, \_inequalities):**

**"""**

**:param u\_d: a d-dimensional vector(list) like [ 8.53149891 3.36436796]**

**:param v\_d: tha same list like u\_d**

**:param \_inequalities: list of constraints on d-dimensional Lambda Polytope like**

**[[0, 1, 0], [1, -1, 0], [0, 0, 1], [1, 0, -1], [0.0, 1.4770889, -3.1250839]]**

**:return: True if u is Kdominance to v regarding given \_inequalities otherwise False**

**"""**

**\_d = len(u\_d)**

**prob = LpProblem("Kdominance", LpMinimize)**

**lambda\_variables = LpVariable.dicts("l", range(\_d), 0)**

**for inequ in \_inequalities:**

**prob += lpSum([inequ[j + 1] \* lambda\_variables[j] for j in range(0, \_d)]) + inequ[0] >= 0**

**prob += lpSum([lambda\_variables[i] \* (u\_d[i]-v\_d[i]) for i in range(\_d)])**

**#prob.writeLP("show-Ldominance.lp")**

**status = prob.solve()**

**LpStatus[status]**

**result = value(prob.objective)**

**if result < 0:**

**return False**

**return True**

**Exemple #2**

**0**

**Fichier :** [**simplex\_test.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Ftkralphs%2FGiMPy%2Fblob%2Fmaster%2FGiMPy-master%2Fsrc%2Fgimpy%2Fsimplex_test.py) **Projet :** [**tkralphs/GiMPy**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Ftkralphs%2FGiMPy)

**def solve(g):**

**el = g.get\_edge\_list()**

**nl = g.get\_node\_list()**

**p = LpProblem('min\_cost', LpMinimize)**

**capacity = {}**

**cost = {}**

**demand = {}**

**x = {}**

**for e in el:**

**capacity[e] = g.get\_edge\_attr(e[0], e[1], 'capacity')**

**cost[e] = g.get\_edge\_attr(e[0], e[1], 'cost')**

**for i in nl:**

**demand[i] = g.get\_node\_attr(i, 'demand')**

**for e in el:**

**x[e] = LpVariable("x"+str(e), 0, capacity[e])**

**# add obj**

**objective = lpSum (cost[e]\*x[e] for e in el)**

**p += objective**

**# add constraints**

**for i in nl:**

**out\_neig = g.get\_out\_neighbors(i)**

**in\_neig = g.get\_in\_neighbors(i)**

**p += lpSum(x[(i,j)] for j in out\_neig) -\**

**lpSum(x[(j,i)] for j in in\_neig)==demand[i]**

**p.solve()**

**return x, value(objective)**

**Exemple #3**

**0**

**Fichier :** [**soccer.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fmwalton%2Freinforcement-learning%2Fblob%2Fmaster%2Fsrc%2Frldm%2Ffinal%2Fsoccer.py) **Projet :** [**mwalton/reinforcement-learning**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fmwalton%2Freinforcement-learning)

**def pi\_solve(pi, Q, s):**

**# The 'prob' variable will contain the problem data.**

**prob = LpProblem('FindPi', LpMinimize)**

**a0 = LpVariable('a0', 0.0) # the minimum is 0.0**

**a1 = LpVariable('a1', 0.0) # the minimum is 0.0**

**a2 = LpVariable('a2', 0.0) # the minimum is 0.0**

**a3 = LpVariable('a3', 0.0) # the minimum is 0.0**

**a4 = LpVariable('a4', 0.0) # the minimum is 0.0**

**v = LpVariable('v', 0.0)**

**# The objective function is added to 'prob' first**

**prob += v, "to minimize"**

**# constraints**

**prob += a0 \* Q[s,0,0] + a1 \* Q[s,1,0] + a2 \* Q[s,2,0] + a3 \* Q[s,3,0] + a4 \* Q[s,4,0] <= v, 'constraint 1'**

**prob += a0 \* Q[s,0,1] + a1 \* Q[s,1,1] + a2 \* Q[s,2,1] + a3 \* Q[s,3,1] + a4 \* Q[s,4,1] <= v, 'constraint 2'**

**prob += a0 \* Q[s,0,2] + a1 \* Q[s,1,2] + a2 \* Q[s,2,2] + a3 \* Q[s,3,2] + a4 \* Q[s,4,2] <= v, 'constraint 3'**

**prob += a0 \* Q[s,0,3] + a1 \* Q[s,1,3] + a2 \* Q[s,2,3] + a3 \* Q[s,3,3] + a4 \* Q[s,4,3] <= v, 'constraint 4'**

**prob += a0 \* Q[s,0,4] + a1 \* Q[s,1,4] + a2 \* Q[s,2,4] + a3 \* Q[s,3,4] + a4 \* Q[s,4,4] <= v, 'constraint 5'**

**prob += 1.0\*a0 + 1.0\*a1 + 1.0\*a2 + 1.0\*a3 + 1.0\*a4 == 1, 'constraint 6'**

**prob.solve()**

**pi\_prime = [a.varValue for a in prob.variables()[:5]]**

**return np.array(pi\_prime)**

**Exemple #4**

**1**

**Fichier :** [**V\_bar\_search.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fpegahani%2FAdvance-Project%2Fblob%2Fmaster%2FFinal_project%2FV_bar_search.py) **Projet :** [**pegahani/Advance-Project**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fpegahani%2FAdvance-Project)

**def K\_dominance\_check(self, \_V\_best\_d, Q\_d):**

**"""**

**:param \_V\_best\_d: a list of d-dimension**

**:param Q\_d: a list of d-dimension**

**:return: True if \_V\_best\_d is prefered to Q\_d regarding self.Lambda\_inequalities and using Kdominance**

**other wise it returns False**

**"""**

**\_d = len(\_V\_best\_d)**

**prob = LpProblem("Ldominance", LpMinimize)**

**lambda\_variables = LpVariable.dicts("l", range(\_d), 0)**

**for inequ in self.Lambda\_ineqalities:**

**prob += lpSum([inequ[j + 1] \* lambda\_variables[j] for j in range(0, \_d)]) + inequ[0] >= 0**

**prob += lpSum([lambda\_variables[i] \* (\_V\_best\_d[i]-Q\_d[i]) for i in range(\_d)])**

**#prob.writeLP("show-Ldominance.lp")**

**status = prob.solve()**

**LpStatus[status]**

**result = value(prob.objective)**

**if result < 0:**

**return False**

**return True**

**Exemple #5**

**0**

**Fichier :** [**process\_path.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FMagdoll%2FCogent%2Fblob%2Fmaster%2Fprocess_path.py) **Projet :** [**Magdoll/Cogent**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FMagdoll%2FCogent)

**def make\_into\_lp\_problem(good\_for, N):**

**"""**

**Helper function for solve\_with\_lp\_and\_reduce()**

**N --- number of isoform sequences**

**good\_for --- dict of <isoform\_index> --> list of matched paths index**

**"""**

**prob = LpProblem("The Whiskas Problem",LpMinimize)**

**# each good\_for is (isoform\_index, [list of matched paths index])**

**# ex: (0, [1,2,4])**

**# ex: (3, [2,5])**

**variables = [LpVariable(str(i),0,1,LpInteger) for i in xrange(N)]**

**# objective is to minimize sum\_{Xi}**

**prob += sum(v for v in variables)**

**# constraints are for each isoform, expressed as c\_i \* x\_i >= 1**

**# where c\_i = 1 if x\_i is matched for the isoform**

**# ex: (0, [1,2,4]) becomes t\_0 = x\_1 + x\_2 + x\_4 >= 1**

**for t\_i, p\_i\_s in good\_for:**

**#c\_i\_s = [1 if i in p\_i\_s else 0 for i in xrange(N)]**

**prob += sum(variables[i]\*(1 if i in p\_i\_s else 0) for i in xrange(N)) >= 1**

**prob.writeLP('cogent.lp')**

**return prob**

**Exemple #6**

**0**

**Fichier :** [**views.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fvccabral%2Fbalancedself%2Fblob%2Fmaster%2Fbalancedself-master%2Fviews.py) **Projet :** [**vccabral/balancedself**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fvccabral%2Fbalancedself)

**def get\_linear\_program\_solution(self, c, b, A, x):**

**prob = LpProblem("myProblem", LpMinimize)**

**prob += lpSum([xp\*cp for xp, cp in zip(x, c)]), "Total Cost of Ingredients per can"**

**for row, cell in zip(A, b):**

**prob += lpSum(ap\*xp for ap, xp in zip(row, x)) <= cell**

**solved = prob.solve()**

**return prob**

**Exemple #7**

**0**

**Fichier :** [**fuelsolver.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fjackdreilly%2Fjetfuel%2Fblob%2Fmaster%2Fjetfuel-master%2Ffuelsolver.py) **Projet :** [**jackdreilly/jetfuel**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fjackdreilly%2Fjetfuel)

**def fuelSolution(airports):**

**for i in range(len(airports)):**

**airports[i].previous = airports[i-1]**

**problem = LpProblem('Flight', LpMinimize)**

**problem+=sum(airport.fuelcost\*airport.refuel for airport in airports), 'Cost'**

**for ap in airports:**

**problem+= ap.min\_fuel + ap.refuel <= ap.startingFuel, 'Min Fuel %s' % ap.startingFuel**

**problem+= ap.startingFuel == ap.previous.startingFuel + ap.refuel - ap.distance\*(1 + (ap.startingFuel + ap.previous.startingFuel - ap.refuel)/(2.\*rate)), 'Takeoff Fuel Level %s' % ap.startingFuel**

**problem.solve()**

**return problem**

**Exemple #8**

**0**

**Fichier :** [**demo.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Ffcostin%2Ftracking%2Fblob%2Fmaster%2Fdemo.py) **Projet :** [**fcostin/tracking**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Ffcostin%2Ftracking)

**def main():**

**x = LpVariable("x", 0, 3)**

**y = LpVariable("y", 0, 1)**

**prob = LpProblem("myProblem", LpMinimize)**

**prob += x + y <= 2**

**prob += -4\*x + y**

**status = prob.solve(COIN(msg = 0))**

**print LpStatus[status]**

**print value(x)**

**print value(y)**

**Exemple #9**

**0**

**Fichier :** [**facility\_location.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FJan-David%2FDip%2Fblob%2Fmaster%2FDip-master%2FDip%2Fsrc%2Fdippy%2Fexamples%2Ffacility_location.py) **Projet :** [**Jan-David/Dip**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FJan-David%2FDip)

**def knapsack01(obj, weights, capacity):**

**""" 0/1 knapsack solver, maximizes profit. weights and capacity integer """**

**debug\_subproblem = False**

**assert len(obj) == len(weights)**

**n = len(obj)**

**if n == 0:**

**return 0, []**

**if debug\_subproblem:**

**relaxation = LpProblem('relaxation', LpMaximize)**

**relax\_vars = [str(i) for i in range(n)]**

**var\_dict = LpVariable.dicts("", relax\_vars, 0, 1, LpBinary)**

**relaxation += (lpSum(var\_dict[str(i)] \* weights[i] for i in range(n))**

**<= capacity)**

**relaxation += lpSum(var\_dict[str(i)] \* obj[i] for i in range(n))**

**relaxation.solve()**

**relax\_obj = value(relaxation.objective)**

**solution = [i for i in range(n) if var\_dict[str(i)].varValue > tol ]**

**print relax\_obj, solution**

**c = [[0]\*(capacity+1) for i in range(n)]**

**added = [[False]\*(capacity+1) for i in range(n)]**

**# c [items, remaining capacity]**

**# important: this code assumes strictly positive objective values**

**for i in range(n):**

**for j in range(capacity+1):**

**if (weights[i] > j):**

**c[i][j] = c[i-1][j]**

**else:**

**c\_add = obj[i] + c[i-1][j-weights[i]]**

**if c\_add > c[i-1][j]:**

**c[i][j] = c\_add**

**added[i][j] = True**

**else:**

**c[i][j] = c[i-1][j]**

**# backtrack to find solution**

**i = n-1**

**j = capacity**

**solution = []**

**while i >= 0 and j >= 0:**

**if added[i][j]:**

**solution.append(i)**

**j -= weights[i]**

**i -= 1**

**return c[n-1][capacity], solution**

**Exemple #10**

**0**

**Fichier :** [**solutions(181-190).py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FColdHumour%2FProjectEulerSolutions%2Fblob%2Fmaster%2Fsolutions%28181-190%29.py) **Projet :** [**ColdHumour/ProjectEulerSolutions**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FColdHumour%2FProjectEulerSolutions)

**def pe185():**

**"""**

**Modelling as an integer programming problem.**

**Then using PuLP to solve it. It's really fast, just 0.24 seconds.**

**For details, see https://pythonhosted.org/PuLP/index.html**

**"""**

**from pulp import LpProblem, LpVariable, LpMinimize, LpInteger, lpSum, value**

**constraints = [**

**('2321386104303845', 0),**

**('3847439647293047', 1),**

**('3174248439465858', 1),**

**('8157356344118483', 1),**

**('6375711915077050', 1),**

**('6913859173121360', 1),**

**('4895722652190306', 1),**

**('5616185650518293', 2),**

**('4513559094146117', 2),**

**('2615250744386899', 2),**

**('6442889055042768', 2),**

**('2326509471271448', 2),**

**('5251583379644322', 2),**

**('2659862637316867', 2),**

**('5855462940810587', 3),**

**('9742855507068353', 3),**

**('4296849643607543', 3),**

**('7890971548908067', 3),**

**('8690095851526254', 3),**

**('1748270476758276', 3),**

**('3041631117224635', 3),**

**('1841236454324589', 3)**

**]**

**VALs = map(str, range(10))**

**LOCs = map(str, range(16))**

**choices = LpVariable.dicts("Choice", (LOCs, VALs), 0, 1, LpInteger)**

**prob = LpProblem("pe185", LpMinimize)**

**prob += 0, "Arbitrary Objective Function"**

**for s in LOCs:**

**prob += lpSum([choices[s][v] for v in VALs]) == 1, ""**

**for c, n in constraints:**

**prob += lpSum([choices[str(i)][v] for i,v in enumerate(c)]) == n, ""**

**prob.writeLP("pe185.lp")**

**prob.solve()**

**res = int(''.join(v for s in LOCs for v in VALs if value(choices[s][v])))**

**# answer: 4640261571849533**

**return res**

**Exemple #11**

**0**

**Fichier :** [**fseconomy.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fpostalservice14%2FFSEconomy%2Fblob%2Fmaster%2Ffseconomy.py) **Projet :** [**postalservice14/FSEconomy**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fpostalservice14%2FFSEconomy)

**def get\_best\_assignments(self, row):**

**df = self.assignments[(self.assignments.FromIcao == row['FromIcao']) &**

**(self.assignments.ToIcao == row['ToIcao']) & (self.assignments.Amount <= row['Seats'])]**

**if not len(df):**

**return None**

**prob = LpProblem("Knapsack problem", LpMaximize)**

**w\_list = df.Amount.tolist()**

**p\_list = df.Pay.tolist()**

**x\_list = [LpVariable('x{}'.format(i), 0, 1, 'Integer') for i in range(1, 1 + len(w\_list))]**

**prob += sum([x \* p for x, p in zip(x\_list, p\_list)]), 'obj'**

**prob += sum([x \* w for x, w in zip(x\_list, w\_list)]) <= row['Seats'], 'c1'**

**prob.solve()**

**return df.iloc[[i for i in range(len(x\_list)) if x\_list[i].varValue]]**

**Exemple #12**

**0**

**Fichier :** [**granularity.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FChadi-akel%2Fcere%2Fblob%2Fmaster%2Fcere-master%2Fsrc%2Fgranularity.py) **Projet :** [**Chadi-akel/cere**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FChadi-akel%2Fcere)

**def solve\_under\_coverage(graph, min\_coverage=80):**

**prob = LpProblem("granularity selection", LpMinimize)**

**codelet\_vars = LpVariable.dicts("codelet",**

**graph,**

**lowBound=0,**

**upBound=1,**

**cat=LpInteger)**

**# Objective function: minimize the total replay cost of selected codelets**

**# Compute replay time**

**for n,d in graph.nodes(data=True):**

**d['\_total\_replay\_cycles'] = 0**

**for inv in d['\_invocations']:**

**d['\_total\_replay\_cycles'] = d['\_total\_replay\_cycles'] + float(inv["Invivo (cycles)"])**

**prob += lpSum([codelet\_vars[n]\*d['\_total\_replay\_cycles'] for n,d in graph.nodes(data=True)])**

**# and with good coverage**

**prob += (lpSum([codelet\_vars[n]\*d['\_coverage'] for n,d in graph.nodes(data=True)]) >= min\_coverage)**

**# selected codelets should match**

**for n,d in graph.nodes(data=True):**

**if not d['\_matching']:**

**prob += codelet\_vars[n] == 0**

**# Finally we should never include both the children and the parents**

**for dad in graph.nodes():**

**for son in graph.nodes():**

**if not dad in nx.ancestors(graph, son):**

**continue**

**# We cannot select dad and son at the same time**

**prob += codelet\_vars[dad] + codelet\_vars[son] <= 1**

**#prob.solve(GLPK())**

**prob.solve()**

**if (LpStatus[prob.status] != 'Optimal'):**

**raise Unsolvable()**

**for v in prob.variables():**

**assert v.varValue == 1.0 or v.varValue == 0.0**

**if v.varValue == 1.0:**

**for n,d in graph.nodes(data=True):**

**if ("codelet\_"+str(n)) == v.name:**

**d["\_selected"] = True**

**yield n**

**Exemple #13**

**0**

**Fichier :** [**matrixexpansion.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Feswald%2FRanked-Voting%2Fblob%2Fmaster%2FRanked-Voting-master%2Fmatrixexpansion.py) **Projet :** [**eswald/Ranked-Voting**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Feswald%2FRanked-Voting)

**def setup(self, statement):**

**self.problem = LpProblem(statement, LpMinimize)**

**self.problem += sum(self.variables.values())**

**for a, b in combinations(self.candidates, 2):**

**ab = a + b**

**ba = b + a**

**if ab in statement and ba in statement:**

**# The statement itself contains the ordering information.**

**pass**

**elif ab in statement:**

**self.constrainGreater(ab, ba)**

**elif ba in statement:**

**self.constrainGreater(ba, ab)**

**#else:**

**# self.constrainEqual(ab, ba)**

**prev = None**

**for rank in statement.split(">"):**

**pairs = rank.split("=")**

**if prev:**

**self.constrainGreater(prev, pairs[0])**

**for n in range(len(pairs) - 1):**

**self.constrainEqual(pairs[n], pairs[n+1])**

**prev = pairs[-1]**

**Exemple #14**

**0**

**Fichier :** [**utils.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Flouisdang%2Fgovhack2016%2Fblob%2Fmaster%2Fgovhack2016-master%2Futils.py) **Projet :** [**louisdang/govhack2016**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Flouisdang%2Fgovhack2016)

**def get\_optimal\_routes(sources, destinations):**

**sources = collections.OrderedDict([(x['id'], x) for x in sources])**

**destinations = collections.OrderedDict([(x['id'], x) for x in destinations])**

**sources\_points = [{'lat': x['lat'], 'lng': x['lng']} for x in sources.values()]**

**destinations\_points = [{'lat': x['lat'], 'lng': x['lng']} for x in destinations.values()]**

**source\_ids = [str(x['id']) for x in sources.values()]**

**dest\_ids = [str(x['id']) for x in destinations.values()]**

**demand = {str(x['id']): convert\_int(x['num\_students']) for x in sources.values()}**

**supply = {str(x['id']): convert\_int(x['num\_students']) for x in destinations.values()}**

**log.info("Calling gmaps api...")**

**distances = gmaps.distance\_matrix(origins=sources\_points, destinations=destinations\_points, mode='walking')**

**costs = {}**

**for i, origin in enumerate(distances['rows']):**

**origin\_costs = {}**

**for j, entry in enumerate(origin['elements']):**

**origin\_costs[dest\_ids[j]] = entry['duration']['value']**

**costs[source\_ids[i]] = origin\_costs**

**prob = LpProblem("Evaucation Routing for Schools",LpMinimize)**

**routes = [(s,d) for s in source\_ids for d in dest\_ids]**

**route\_lookup = {'Route\_{}\_{}'.format(x.replace(' ','\_'),y.replace(' ','\_')):(x,y) for (x,y) in routes}**

**route\_vars = LpVariable.dicts("Route",(source\_ids,dest\_ids),0,None,LpInteger)**

**prob += lpSum([route\_vars[w][b]\*(costs[w][b]\*\*2) for (w,b) in routes])**

**for dest in dest\_ids:**

**prob += lpSum([route\_vars[source][dest] for source in source\_ids]) <= supply[dest], "Students going to {} is <= {}".format(dest, supply[dest])**

**for source in source\_ids:**

**prob += lpSum([route\_vars[source][dest] for dest in dest\_ids]) == demand[source], "Students leaving {} is {}".format(source, demand[source])**

**log.info("Optimizing routes...")**

**prob.solve()**

**if prob.status != 1:**

**raise Exception("Algorithm could not converge to a solution")**

**result = []**

**for v in prob.variables():**

**src, dst = route\_lookup[v.name]**

**value = v.value()**

**result.append({'src': sources[src], 'dst': destinations[dst], 'value': int(value)})**

**return result**

**Exemple #15**

**0**

**Fichier :** [**workdist.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FBaljan%2Fcafesys%2Fblob%2Fmaster%2Fcafesys-master%2Fcafesys%2Fworkdist.py) **Projet :** [**Baljan/cafesys**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FBaljan%2Fcafesys)

**def \_recurse(work\_pairs, mae, grace=None):**

**req = grace or 1.0**

**demands = {**

**'morning': mae[0],**

**'afternoon': mae[1],**

**'exam': mae[2],**

**}**

**total\_avg = float(sum([demands[i] for i in SHIFTS])) / work\_pairs**

**total\_low, total\_high = floor(total\_avg), ceil(total\_avg)**

**work\_pair\_count = work\_pairs**

**avgs = [float(demands[i]) / work\_pair\_count for i in SHIFTS]**

**lows = [floor(a) for a in avgs]**

**highs = [ceil(a) for a in avgs]**

**target = req \* total\_avg \* float(sum([COSTS[i] for i in SHIFTS])) / len(SHIFTS)**

**prob = LpProblem("Work Distribution", LpMinimize)**

**var\_prefix = "shift"**

**shift\_vars = LpVariable.dicts(var\_prefix, SHIFTS, 0, cat=LpInteger)**

**prob += lpSum([COSTS[i] \* shift\_vars[i] for i in SHIFTS]), "cost of combination"**

**prob += lpSum([COSTS[i] \* shift\_vars[i] for i in SHIFTS]) >= target, "not too good"**

**prob += lpSum([shift\_vars[i] for i in SHIFTS]) >= total\_low, "low TOTAL"**

**prob += lpSum([shift\_vars[i] for i in SHIFTS]) <= total\_high, "high TOTAL"**

**for shift, low, high in zip(SHIFTS, lows, highs):**

**prob += lpSum([shift\_vars[shift]]) >= low, "low %s" % shift**

**prob += lpSum([shift\_vars[shift]]) <= high, "high %s" % shift**

**prob.solve(GLPK\_CMD(msg=0))**

**if not LpStatus[prob.status] == 'Optimal':**

**next\_grace = req - 0.1**

**assert 0.0 < next\_grace**

**return \_recurse(work\_pairs, mae, next\_grace)**

**new\_mae = [0, 0, 0]**

**solution = [0, 0, 0]**

**for v in prob.variables():**

**for pos, name in enumerate(SHIFTS):**

**if v.name == "%s\_%s" % (var\_prefix, name):**

**solution[pos] = v.varValue**

**new\_mae[pos] = mae[pos] - solution[pos]**

**return (PairAlloc(solution), work\_pairs - 1) + tuple(new\_mae)**

**Exemple #16**

**0**

**Fichier :** [**optknock.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Feladnoor%2Foptslope%2Fblob%2Fmaster%2Foptslope-master%2Foptknock.py) **Projet :** [**eladnoor/optslope**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Feladnoor%2Foptslope)

**def create\_prob(self, sense=LpMaximize, use\_glpk=False):**

**# create the LP**

**self.prob = LpProblem('OptKnock', sense=sense)**

**if use\_glpk:**

**self.prob.solver = solvers.GLPK()**

**else:**

**self.prob.solver = solvers.CPLEX(msg=self.verbose)**

**if not self.prob.solver.available():**

**raise Exception("CPLEX not available")**

**Exemple #17**

**0**

**Fichier :** [**approxLCM.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fgkehne%2Fapproximate-LCM%2Fblob%2Fmaster%2FapproxLCM.py) **Projet :** [**gkehne/approximate-LCM**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fgkehne%2Fapproximate-LCM)

**def runLPP(lines):**

**# reads the data from the CSV file into the LPP**

**variables, ingredients, servingsPerBlock, costsPerServing = [], [], [], []**

**for row in lines: # read in the variables from the lines**

**variables.append(LpVariable("x\_" + row[1], 0, None, LpInteger))**

**ingredients.append(row[0])**

**servingsPerBlock.append(float(row[2]))**

**costsPerServing.append(float(row[3]))**

**variables.append(LpVariable("s", 0))**

**# read in the upper and lower bounds for aLCM**

**minServe = lines[0][4]**

**maxServe = lines[0][5]**

**# makes the new LP Problem**

**problem = LpProblem("Approximate LCM", LpMinimize)**

**# serving constraints: specifies the interval in which the approximate**

**# LCM may lie**

**min\_constraint = variables[-1] >= float(minServe)**

**max\_constraint = variables[-1] <= float(maxServe)**

**problem += min\_constraint**

**problem += max\_constraint**

**# block constraints: there must be enough of each ingredient for the**

**# ultimate optimal number**

**for v in range(len(variables) - 1):**

**min\_constraint = variables[v] \* servingsPerBlock[v] >= variables[-1]**

**problem += min\_constraint**

**pps = 0.0 # price per serving**

**for i in range(len(variables) - 1):**

**pps += costsPerServing[i]**

**# add objective function to be minimized (waste)**

**obj\_fn = 0 \* variables[0]**

**for i in range(len(variables) - 1):**

**obj\_fn += variables[i] \* costsPerServing[i] \* servingsPerBlock[i]**

**obj\_fn -= variables[-1] \* pps**

**problem += obj\_fn # add the objective funciton to the problem**

**problem.solve() # runs the linear programming problem**

**return [variables, costsPerServing, obj\_fn]**

**Exemple #18**

**0**

**Fichier :** [**strategy.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fbogdan-kulynych%2Fmrf-in-economics%2Fblob%2Fmaster%2Fmrf-in-economics-master%2Fstrategy.py) **Projet :** [**bogdan-kulynych/mrf-in-economics**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fbogdan-kulynych%2Fmrf-in-economics)

**def find\_optimal\_strategy(states, controls, costs, kernels, solver=None):**

**"""**

**:param states: Number of system states (X)**

**:param controls: Number of system controls (U)**

**:param сosts: Cost matrix |X| x |U|**

**:param kernels: Transition kernels. Dimensionality |X| x |X| x |U|**

**"""**

**tolerance = 10e-15**

**X = range(states)**

**U = range(controls)**

**R = costs**

**Q = kernels**

**# Check costs**

**# Check num of rows**

**assert(len(R) == states)**

**for row in R:**

**# Check num of cols**

**assert(len(row) == controls)**

**# Check kernels**

**# Check num of rows**

**assert(len(Q) == states)**

**for row in Q:**

**# Check num of cols**

**assert(len(row) == states)**

**for items in row:**

**# Check num of items**

**assert(len(items) == controls)**

**# Check if distribution is normed**

**for dist in zip(\*row):**

**assert(sum(dist)-1 < tolerance)**

**# LP object**

**optm = LpProblem("Optimal strategy", sense=LpMinimize)**

**# Variables (continuous in range [0, 1])**

**Z = [[LpVariable("z({},{})".format(x, u), 0, 1) \**

**for u in U] for x in X]**

**# Objective**

**optm.objective = sum(np.dot(Z[x], R[x]) for x in X)**

**# Constraints**

**for x in X:**

**cn = (sum(Z[x]) == sum(Q[y][x][u]\*Z[y][u] for u in U for y in X))**

**optm.add(cn)**

**cn = sum(Z[x][u] for u in U for x in X) == 1**

**optm.add(cn)**

**optm.solve(solver)**

**return [(x, u) for u in U for x in X if value(Z[x][u]) != 0]**

**Exemple #19**

**0**

**Fichier :** [**problem5.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fnyy7%2Fdata2_hw3%2Fblob%2Fmaster%2Fproblem5.py) **Projet :** [**nyy7/data2\_hw3**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fnyy7%2Fdata2_hw3)

**def lp(self):**

**a\_name = []**

**p\_name = []**

**n\_name = []**

**a = []**

**p = []**

**n = []**

**for i in range(self.att):**

**a\_name.append('a\_' + str(i))**

**for h in range(self.h):**

**p\_name.append('p\_' + str(h))**

**for m in range(self.m):**

**n\_name.append('n\_' + str(m))**

**a = [LpVariable(a\_name[i]) for i in range(self.att)]**

**b = LpVariable('b')**

**p = [LpVariable(p\_name[i],0) for i in range(self.h)]**

**n = [LpVariable(n\_name[i],0) for i in range(self.m)]**

**#set objective function and constrains**

**prob = LpProblem("myProblem",LpMinimize)**

**sumP = 0**

**sumN = 0**

**for i in range(self.h):**

**sumX = 0**

**for j in range(self.att):**

**sumX -= a[j] \* self.H[i,j]**

**prob += sumX + b + 1 <= p[i]**

**sumP += p[i]**

**for i in range(self.m):**

**sumX = 0**

**for j in range(self.att):**

**sumX += a[j] \* self.M[i,j]**

**prob += sumX - b + 1 <= n[i]**

**sumN += n[i]**

**prob += (1/h) \* sumP + (1/m) \* sumN**

**status = prob.solve()**

**for i in range(self.att):**

**self.a.append(pulp.value(a[i]))**

**self.b = pulp.value(b)**

**Exemple #20**

**0**

**Fichier :** [**CuttingPlane.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fkhour%2Fpynumeric%2Fblob%2Fmaster%2FCuttingPlane.py) **Projet :** [**khour/pynumeric**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fkhour%2Fpynumeric)

**def pulp\_solve(A, b, c):**

**problem = LpProblem(sense = pulp.constants.LpMaximize)**

**x = [LpVariable('x' + str(i + 1),**

**0,**

**None,**

**LpInteger)**

**for i in xrange(len(c))]**

**problem += lpSum(ci \* xi for ci, xi in zip(c, x))**

**for ai, bi in zip(A, b):**

**problem += lpSum(aij \* xj for aij, xj in zip(ai, x)) == bi**

**status = problem.solve()**

**return (pulp.constants.LpStatus[status],**

**[variable.varValue for variable in problem.sortedVariables()],**

**problem.objective.value())**

**Exemple #21**

**0**

**Fichier :** [**BranchAndBound.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fkhour%2Fpynumeric%2Fblob%2Fmaster%2FBranchAndBound.py) **Projet :** [**khour/pynumeric**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fkhour%2Fpynumeric)

**def pulp\_solve(A, b, c, bounds):**

**problem = LpProblem(sense = pulp.constants.LpMaximize)**

**x = [LpVariable('x' + str(i + 1),**

**bound['low\_bound'],**

**bound['up\_bound'],**

**LpInteger)**

**for i, bound in enumerate(bounds)]**

**problem += lpSum(ci \* xi for ci, xi in zip(c, x))**

**for ai, bi in zip(A, b):**

**problem += lpSum(aij \* xj for aij, xj in zip(ai, x)) == bi**

**status = problem.solve()**

**return (pulp.constants.LpStatus[status],**

**[variable.varValue for variable in problem.sortedVariables()],**

**problem.objective.value())**

**Exemple #22**

**0**

**Fichier :** [**pulp\_wrapper.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fwilligott%2Fframed%2Fblob%2Fmaster%2Fsrc%2Fframed%2Fpulp_wrapper.py) **Projet :** [**willigott/framed**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fwilligott%2Fframed)

**def build\_problem(self, model):**

**""" Create and store solver-specific internal structure for the given model.**

**Arguments:**

**model : ConstraintBasedModel**

**"""**

**problem = LpProblem(sense=LpMaximize)**

**problem.setSolver(SELECTED\_SOLVER)**

**#create variables**

**lpvars = {r\_id: LpVariable(r\_id, lb, ub) for r\_id, (lb, ub) in model.bounds.items()}**

**#create constraints**

**table = model.metabolite\_reaction\_lookup\_table()**

**for m\_id in model.metabolites:**

**problem += lpSum([coeff \* lpvars[r\_id] for r\_id, coeff in table[m\_id].items()]) == 0, m\_id**

**self.problem = problem**

**self.var\_ids = model.reactions.keys()**

**self.constr\_ids = model.metabolites.keys()**

**Exemple #23**

**0**

**Fichier :** [**linprog.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FArafatk%2Ffraciso%2Fblob%2Fmaster%2Ffraciso-master%2Flinprog.py) **Projet :** [**Arafatk/fraciso**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2FArafatk%2Ffraciso)

**def pulp\_solver(G, h, A, b, c, n):**

**# First, create a variable for each of the columsn of G and A.**

**#**

**# pre-condition: G and A have the same number of columns.**

**#**

**# The second argument specifies a lower bound for the variable, so we can**

**# safely ignore the inequality constraints given by G and h.**

**variables = [LpVariable('s{}'.format(i), 0) for i in range(G.shape[1])]**

**# LpVariable has a second argument that allows you to specify a lower bound**

**# for the variable (for example, x1 >= 0). We don't specify nonnegativity**

**# here, because it is already specified by the inequality constraints G and**

**# h.**

**#variables = [LpVariable('s{}'.format(i)) for i in range(G.shape[1])]**

**# Next, create a problem context object and add the objective function c to**

**# it. The first object added to LpProblem is implicitly interpreted as the**

**# objective function.**

**problem = LpProblem('fraciso', LpMinimize)**

**# The np.dot() function doesn't like mixing numbers and LpVariable objects,**

**# so we compute the dot product ourselves.**

**#**

**#problem += np.dot(variables, c), 'Dummy objective function'**

**problem += \_pulp\_dot\_product(c, variables), 'Dummy objective function'**

**# Add each equality constraint to the problem context.**

**for i, (row, b\_value) in enumerate(zip(A, b)):**

**#problem += np.dot(row, variables), 'Constraint {}'.format(i)**

**# Convert the row to a list so pulp has an easier time dealing with it.**

**row\_as\_list = np.asarray(row).flatten().tolist()**

**dot\_product = \_pulp\_dot\_product(row\_as\_list, variables)**

**problem += dot\_product == b\_value, 'Constraint {}'.format(i)**

**solver\_backend = GLPK()**

**#solver\_backend = COIN()**

**problem.solve(solver\_backend)**

**if problem.status == LpStatusOptimal:**

**# PuLP is silly and sorts the variables by name before returning them,**

**# so we need to re-sort them in numerical order.**

**solution = [s.varValue for s in sorted(problem.variables(),**

**key=lambda s: int(s.name[1:]))]**

**return True, solution**

**# TODO status could be unknown here, but we're currently ignoring that**

**return False, None**

**Exemple #24**

**0**

**Fichier :** [**LPSolver.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fbionomicron%2FRedirector%2Fblob%2Fmaster%2FRedirector-master%2Fcore%2FLPSolver.py) **Projet :** [**bionomicron/Redirector**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fbionomicron%2FRedirector)

**def \_\_init\_\_(self,name='analysis',verbose=False):**

**self.lpProblem = LpProblem()**

**self.lpVariables = {}**

**self.lpObjective = {}**

**self.predictionMap = {}**

**self.objectiveValue = None**

**self.rowNames= []**

**self.columnNames= []**

**self.configFile = ''**

**self.statusCode = None**

**self.verbose = verbose**

**self.useLimitTag = False**

**self.mpsLogFile = False**

**self.ignoreBadReferences = False**

**self.Mip = True**

**self.scip\_path = ":" + os.environ["HOME"] + "/bin"**

**Exemple #25**

**0**

**Fichier :** [**hitchcock.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fwlxiong%2FPyMarkovActv%2Fblob%2Fmaster%2FPyMarkovActv-master%2Fhitchcock.py) **Projet :** [**wlxiong/PyMarkovActv**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fwlxiong%2FPyMarkovActv)

**def \_\_init\_\_(self, home\_list, work\_list, util\_matrix):**

**""" Input a list of utils**

**utils = [ #Works**

**#1 2 3 4 5**

**[2,4,5,2,1],#A Homes**

**[3,1,3,2,3] #B**

**]**

**"""**

**self.util\_matrix = util\_matrix**

**self.homes = dict((home, home.houses) for home in home\_list)**

**self.works = dict((work, work.jobs) for work in work\_list)**

**self.utils = makeDict([home\_list, work\_list], util\_matrix, 0)**

**# Creates the 'prob' variable to contain the problem data**

**self.prob = LpProblem("Residential Location Choice Problem", LpMinimize)**

**# Creates a list of tuples containing all the possible location choices**

**self.choices = [(h, w) for h in self.homes for w in self.works.keys()]**

**# A dictionary called 'volumes' is created to contain the referenced variables(the choices)**

**self.volumes = LpVariable.dicts("choice", (self.homes, self.works), 0, None, LpContinuous)**

**# The objective function is added to 'prob' first**

**self.prob += (**

**lpSum([self.volumes[h][w] \* self.utils[h][w] for (h, w) in self.choices]),**

**"Sum\_of\_Transporting\_Costs",**

**)**

**# The supply maximum constraints are added to prob for each supply node (home)**

**for h in self.homes:**

**self.prob += (**

**lpSum([self.volumes[h][w] for w in self.works]) <= self.homes[h],**

**"Sum\_of\_Products\_out\_of\_Home\_%s" % h,**

**)**

**# The demand minimum constraints are added to prob for each demand node (work)**

**for w in self.works:**

**self.prob += (**

**lpSum([self.volumes[h][w] for h in self.homes]) >= self.works[w],**

**"Sum\_of\_Products\_into\_Work%s" % w,**

**)**

**Exemple #26**

**0**

**Fichier :** [**asistente.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fgmroldan%2Funiversitary_advisor%2Fblob%2Fmaster%2Fasistente.py) **Projet :** [**gmroldan/universitary\_advisor**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fgmroldan%2Funiversitary_advisor)

**def \_\_generar\_restricciones(self):**

**X = LpVariable.dicts('X', self.variables, cat = LpBinary)**

**self.cursado = LpProblem("Cursado",LpMaximize)**

**# "\n\n Restricciones de dia-modulo\n\n"**

**i = 1**

**for r in self.r1.values():**

**self.cursado += lpSum([X["".join(x)] for x in r]) <= 1, \**

**"\_1C" + str(i).zfill(3)**

**i += 1**

**#"\n\n Restricciones de cursado completo\n\n"**

**for r in self.r2.values():**

**self.cursado += int(r[0][0]) \*X[r[0][1]+r[0][2]] == lpSum([X["".join(x[1:])] for x in r]), \**

**"\_2C" + str(i).zfill(3)**

**i += 1**

**#"""**

**#"\n\n Restricciones 1 sola comision\n\n"**

**for r in self.r3.values():**

**self.cursado += int(r[0][0]) \*X[r[0][1]] == lpSum([X["".join(x[1:])] for x in r]), \**

**"\_3C" + str(i).zfill(3)**

**i += 1**

**#"\n\n Restricciones de Colision\n\n"**

**for r in self.r4.values():**

**self.cursado += lpSum([X["".join(x)] for x in r]) <= 1, \**

**"\_4C" + str(i).zfill(3)**

**i += 1**

**#"\n\n Restricciones de franja horaria\n\n"**

**for r in [x + "" for x in ["".join(x) for x in self.r5]]:**

**self.cursado += X[r] == 0, \**

**"\_5C" + str(i).zfill(3)**

**i += 1**

**self.X = X**

**Exemple #27**

**0**

**Fichier :** [**genericSeparation.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fashutoshmahajan%2FCuPPy%2Fblob%2Fmaster%2Fsrc%2FgenericSeparation.py) **Projet :** [**ashutoshmahajan/CuPPy**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fashutoshmahajan%2FCuPPy)

**class GenericSeparation(object):**

**def \_\_init\_\_(self, origProb, x0):**

**self.prob = origProb**

**self.iter = 1**

**self.x0 = x0**

**self.var = origProb.variablesDict()**

**self.dim = len(self.var)**

**self.currentXtremePoint = {}**

**self.solver = None**

**self.c = dict(list((v.name,c) for v,c in origProb.objective.iteritems()))**

**for v in self.var:**

**if v in self.c:**

**continue**

**else:**

**self.c[v] = 0.0**

**self.generate\_separation\_problem()**

**self.piCurrent = dict([(v.name, 0) for v in self.var.values()])**

**self.piPrevious = dict([(v.name, 0) for v in self.var.values()])**

**self.extremePoints = []**

**self.f = Figure()**

**def generate\_separation\_problem(self):**

**self.sepProb = LpProblem (name='separation '+self.prob.name, sense=LpMinimize)**

**self.pi = dict(list((v,LpVariable('pi'+v, None, None, LpContinuous,**

**None))**

**for v in self.var))**

**self.sepProb += lpSum(self.pi[v]\*self.x0[v] for v in self.var)**

**def generate\_xtreme\_point(self):**

**obj = LpConstraintVar()**

**for v in self.prob.variables():**

**obj.addVariable(v, self.piCurrent[v.name])**

**self.prob.setObjective(obj)**

**self.prob.solve()**

**#solvers.COIN\_CMD.solve(self.prob)**

**for v in self.var:**

**self.currentXtremePoint[v] = self.var[v].value()**

**if self.output == 1:**

**currentXtremePointList = self.currentXtremePoint.items()**

**currentXtremePointList.sort()**

**for v in currentXtremePointList:**

**print v[0]+'\t', v[1]**

**self.extremePoints.append(self.currentXtremePoint.values())**

**return self.prob.objective.value()**

**def add\_inequality(self):**

**# change this, you should not access sense directly call a method**

**self.sepProb += lpSum (self.currentXtremePoint[v]\*self.pi[v] for v in self.var) >= 1**

**def separate(self, output = False, p = None):**

**self.output = output**

**while True:**

**print 'Iteration ', self.iter**

**if self.generate\_xtreme\_point() >= 1-EPS:**

**break**

**self.add\_inequality()**

**if self.output:**

**print "Separation problem solution status:", LpStatus[self.sepProb.solve()]**

**for v in self.var:**

**if self.pi[v].value() is not None:**

**print self.pi[v].name+'\t\t', self.pi[v].value()**

**else:**

**print self.pi[v].name+'\t\t', 0**

**self.piPrevious = deepcopy(self.piCurrent)**

**for v in self.var:**

**if self.pi[v].value() is not None:**

**self.piCurrent[v] = self.pi[v].value()**

**else:**

**self.piCurrent[v] = 0**

**self.iter += 1**

**if p is not None:**

**self.f.initialize()**

**self.f.add\_polyhedron(p, label = 'Polyhedron P')**

**xList = (self.x0.values()[0], self.x0.values()[1])**

**if len(self.extremePoints) > 2:**

**pp = Polyhedron2D(points = self.extremePoints)**

**self.f.add\_polyhedron(pp, color = 'red', linestyle = 'dashed',**

**label = 'Convex Hull of Generated Points')**

**elif len(self.extremePoints) == 1:**

**self.f.add\_point(self.extremePoints[0], radius = 0.05,**

**color = 'green')**

**self.f.add\_text(self.extremePoints[0][0]-0.5,**

**self.extremePoints[0][1]-0.08, '$x^0$')**

**else:**

**self.f.add\_line\_segment(self.extremePoints[0],**

**self.extremePoints[1],**

**color = 'red',**

**linestyle = 'dashed',**

**label = 'Convex Hull of Generated Points')**

**self.f.set\_xlim(p.plot\_min[0], p.plot\_max[0])**

**self.f.set\_ylim(p.plot\_min[1], p.plot\_max[1])**

**self.f.add\_point(xList, radius = 0.05, color = 'red')**

**self.f.add\_text(xList[0]-0.5, xList[1]-0.08, '$x^\*$')**

**dList = (self.piCurrent.values()[0], self.piCurrent.values()[1])**

**self.f.add\_line(dList, 1,**

**p.plot\_max, p.plot\_min, color = 'green',**

**linestyle = 'dashed', label = 'Separating Hyperplane')**

**self.f.show()**

**if self.output:**

**print self.sepProb.objective.value()**

**Exemple #28**

**0**

**Fichier :** [**genericSeparation.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fashutoshmahajan%2FCuPPy%2Fblob%2Fmaster%2Fsrc%2FgenericSeparation.py) **Projet :** [**ashutoshmahajan/CuPPy**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fashutoshmahajan%2FCuPPy)

**def generate\_separation\_problem(self):**

**self.sepProb = LpProblem (name='separation '+self.prob.name, sense=LpMinimize)**

**self.pi = dict(list((v,LpVariable('pi'+v, None, None, LpContinuous,**

**None))**

**for v in self.var))**

**self.sepProb += lpSum(self.pi[v]\*self.x0[v] for v in self.var)**

**Exemple #29**

**0**

**Fichier :** [**BB.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fgarretvo19%2FGiMPy%2Fblob%2Fmaster%2FGiMPy-master%2Fsrc%2Fgimpy%2FBB.py) **Projet :** [**garretvo19/GiMPy**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fgarretvo19%2FGiMPy)

**cur\_index, parent, branch\_var, sense, rhs = Q.pop()**

**if cur\_index is not 0:**

**cur\_depth = T.get\_node\_attr(parent, 'level') + 1**

**else:**

**cur\_depth = 0**

**print ""**

**print "----------------------------------------------------"**

**print ""**

**print "Node: %s, Depth: %s, LB: %s" %(cur\_index,cur\_depth,LB)**

**#====================================**

**# LP Relaxation**

**#====================================**

**#Compute lower bound by LP relaxation**

**prob = LpProblem("relax",LpMaximize)**

**prob += lpSum([OBJ[i]\*var[i] for i in VARIABLES]), "Objective"**

**for j in range(numCons):**

**prob += lpSum([MAT[i][j]\*var[i] for i in VARIABLES]) <= RHS[j], \**

**CONSTRAINTS[j]**

**#Fix all prescribed variables**

**branch\_vars = []**

**if cur\_index is not 0:**

**print "Branching variables: "**

**branch\_vars.append(branch\_var)**

**if sense == '>=':**

**prob += LpConstraint(lpSum(var[branch\_var]) >= rhs)**

**else:**

**prob += LpConstraint(lpSum(var[branch\_var]) <= rhs)**

**print branch\_var,**

**Exemple #30**

**0**

**Fichier :** [**hitchcock.py**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fwlxiong%2FPyMarkovActv%2Fblob%2Fmaster%2FPyMarkovActv-master%2Fhitchcock.py) **Projet :** [**wlxiong/PyMarkovActv**](https://python.hotexamples.com/fr/site/redirect?url=https%3A%2F%2Fgithub.com%2Fwlxiong%2FPyMarkovActv)

**class TransProblem(object):**

**def \_\_init\_\_(self, home\_list, work\_list, util\_matrix):**

**""" Input a list of utils**

**utils = [ #Works**

**#1 2 3 4 5**

**[2,4,5,2,1],#A Homes**

**[3,1,3,2,3] #B**

**]**

**"""**

**self.util\_matrix = util\_matrix**

**self.homes = dict((home, home.houses) for home in home\_list)**

**self.works = dict((work, work.jobs) for work in work\_list)**

**self.utils = makeDict([home\_list, work\_list], util\_matrix, 0)**

**# Creates the 'prob' variable to contain the problem data**

**self.prob = LpProblem("Residential Location Choice Problem", LpMinimize)**

**# Creates a list of tuples containing all the possible location choices**

**self.choices = [(h, w) for h in self.homes for w in self.works.keys()]**

**# A dictionary called 'volumes' is created to contain the referenced variables(the choices)**

**self.volumes = LpVariable.dicts("choice", (self.homes, self.works), 0, None, LpContinuous)**

**# The objective function is added to 'prob' first**

**self.prob += (**

**lpSum([self.volumes[h][w] \* self.utils[h][w] for (h, w) in self.choices]),**

**"Sum\_of\_Transporting\_Costs",**

**)**

**# The supply maximum constraints are added to prob for each supply node (home)**

**for h in self.homes:**

**self.prob += (**

**lpSum([self.volumes[h][w] for w in self.works]) <= self.homes[h],**

**"Sum\_of\_Products\_out\_of\_Home\_%s" % h,**

**)**

**# The demand minimum constraints are added to prob for each demand node (work)**

**for w in self.works:**

**self.prob += (**

**lpSum([self.volumes[h][w] for h in self.homes]) >= self.works[w],**

**"Sum\_of\_Products\_into\_Work%s" % w,**

**)**

**def solve(self):**

**# The problem data is written to an .lp file**

**self.prob.writeLP("ResidentialLocationChoiceProblem.lp")**

**# The problem is solved using PuLP's choice of Solver**

**self.prob.solve(solvers.GLPK())**

**# print the utility matrix**

**print "Utility Matrix", self.util\_matrix**

**# The status of the solution is printed to the screen**

**print "Status:", LpStatus[self.prob.status]**

**# The optimised objective function value is printed to the screen**

**print "Total Utility = ", value(self.prob.objective)**

**def get\_solution(self):**

**# put the solution variables into a dict**

**sol\_var = {}**

**for vol in self.prob.variables():**

**sol\_var[vol.name] = vol.varValue**

**# construct the solution dict**

**opt\_sol = {}**

**for home in self.homes:**

**for work in self.works:**

**key = "choice" + "\_" + str(home) + "\_" + str(work)**

**opt\_sol[(work, home)] = sol\_var[key]**

**print (work, home), opt\_sol[(work, home)]**

**return opt\_sol**