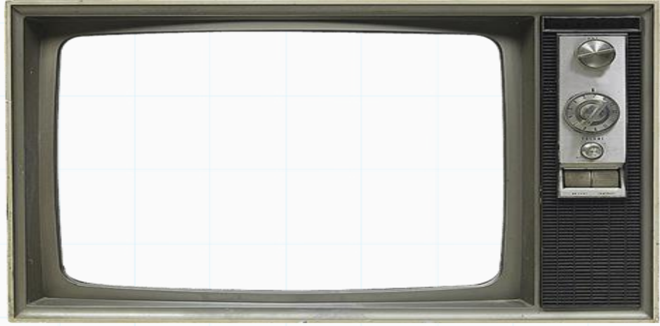
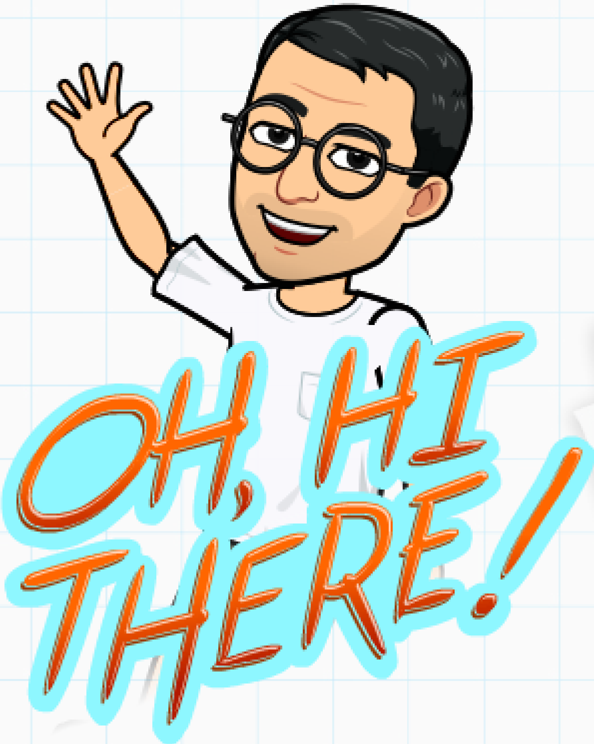
Pesquisa Operacional

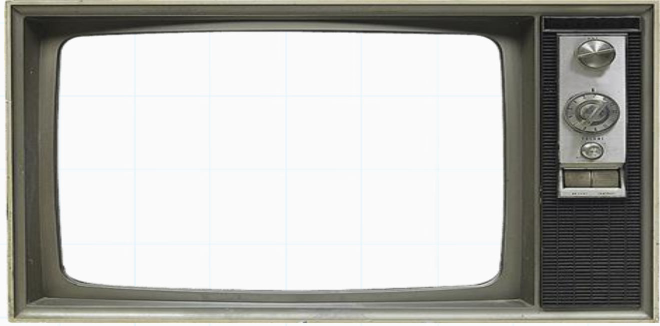


Professor : Yuri Frota

yuri@ic.uff.br



Solvers

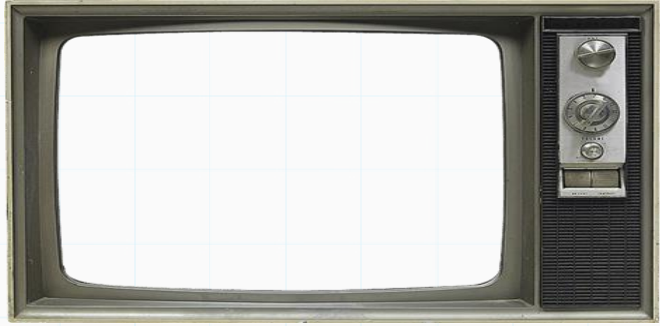


GUROBI OPTIMIZATION

X

IBM ILOG CPLEX

Solvers



GUROBI OPTIMIZATION

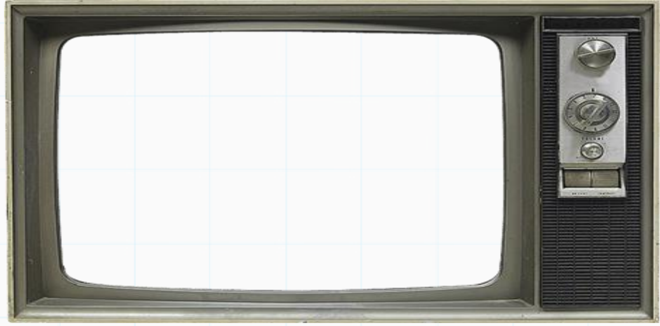
MARVEL

X

IBM ILOG CPLEX

DC

Solvers



GUROBI OPTIMIZATION

x

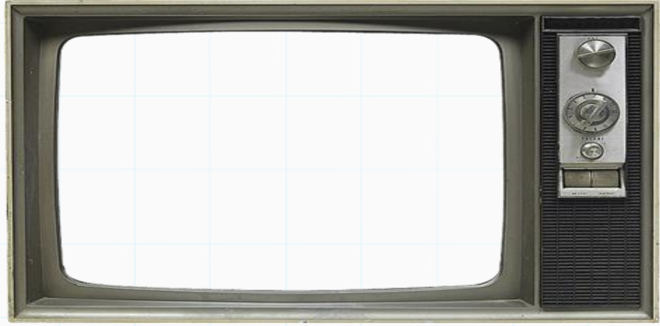
IBM ILOG CPLEX

MARVEL

DC

FICO Xpress

Solvers



GUROBI OPTIMIZATION

X

IBM ILOG CPLEX

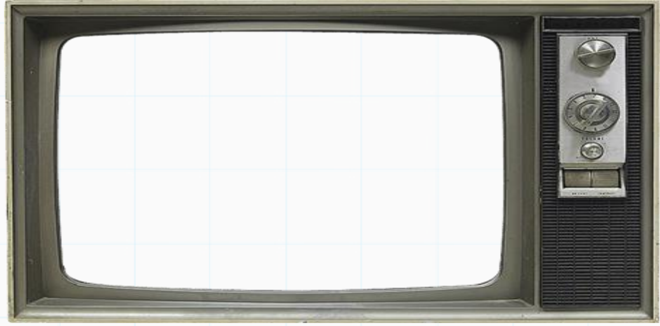
MARVEL

DC

FICO Xpress

DARK HORSE COMICS

Solvers



GUROBI OPTIMIZATION

x

IBM ILOG CPLEX

MARVEL

DC

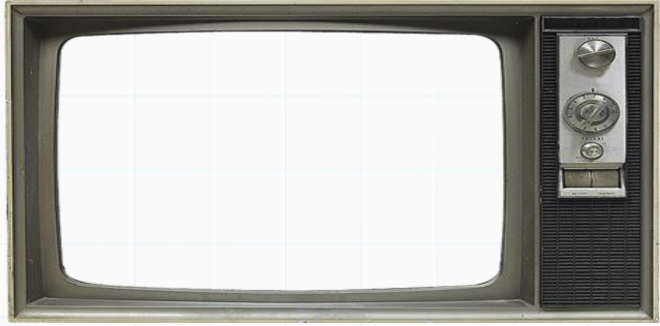
FICO Xpress

DARK HORSE COMICS

mosek

image

Solvers



GUROBI OPTIMIZATION

x

IBM ILOG CPLEX

MARVEL

DC

GLPK

NLOPT

COIN|OR

SCIP

FICO Xpress

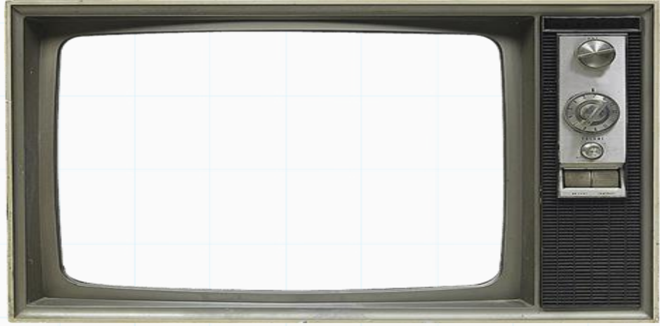
DARK HORSE COMICS

mosek

lpsolve

image

Solvers

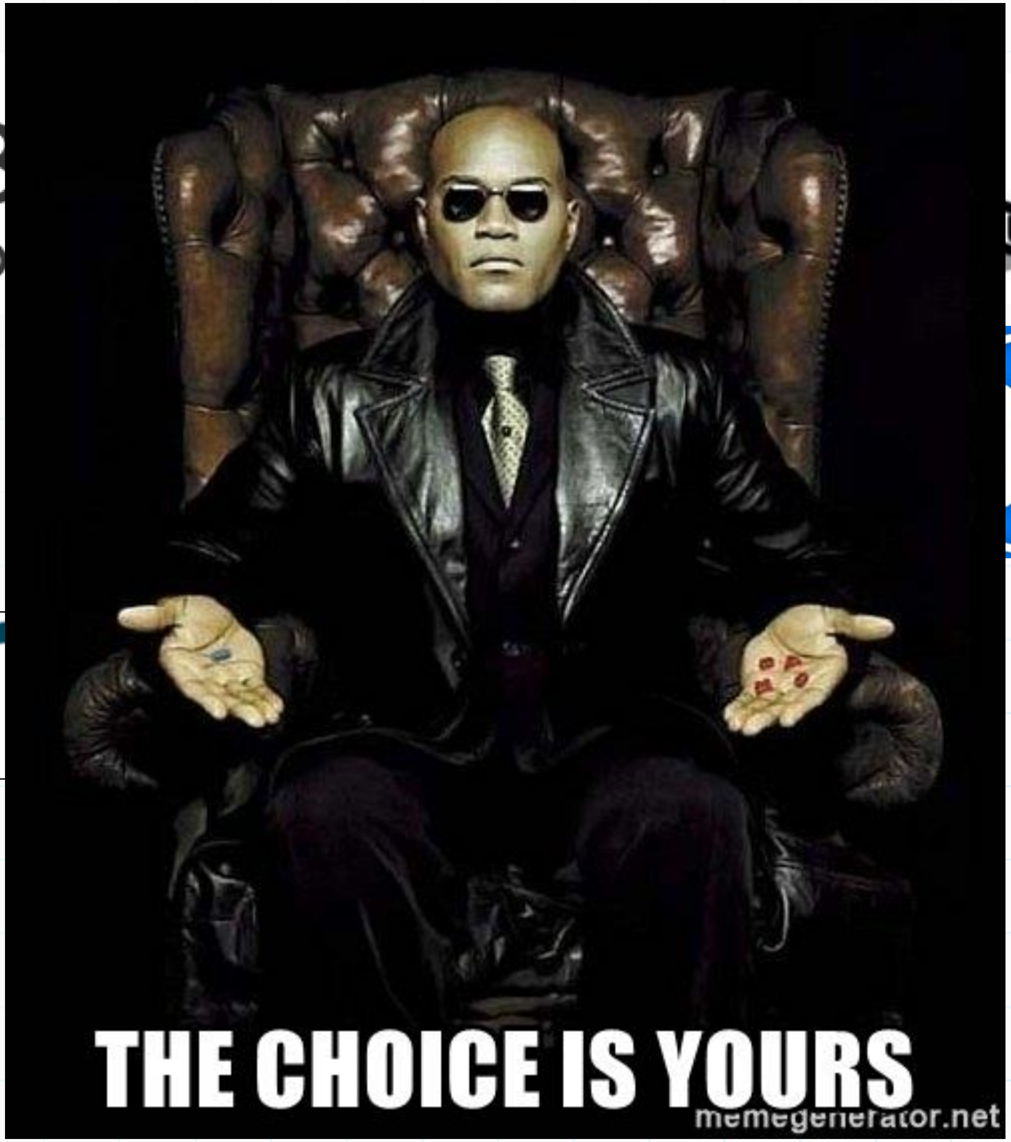


GUROB
OPTIMIZATIO

MARVEL

FICO Xpress

DARK HORSE COMICS



LOG CPLEX

COIN|OR

GLPK

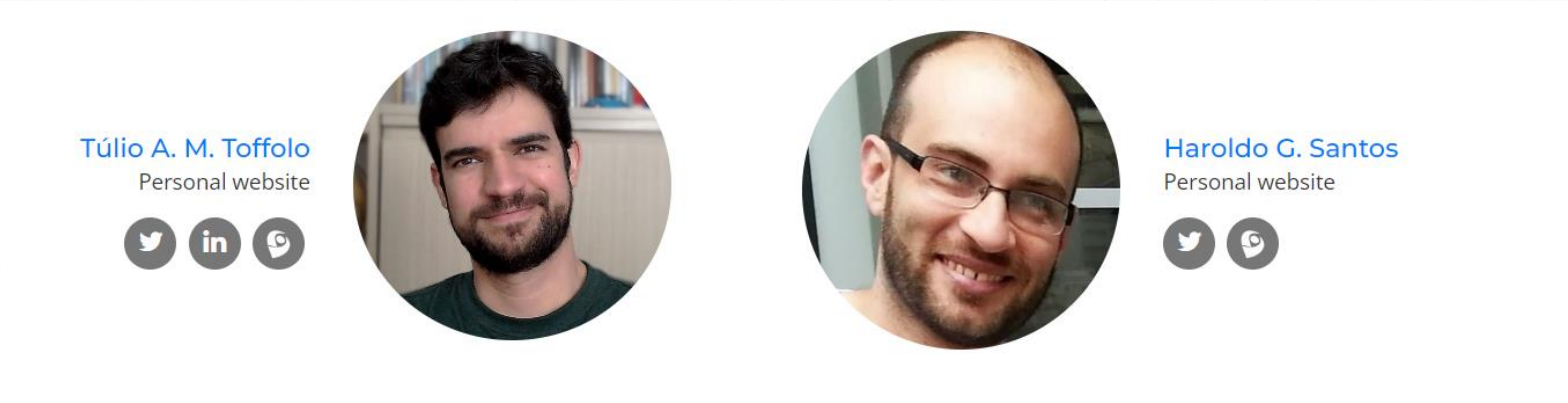
NLOPT

SCIP

lpsolve

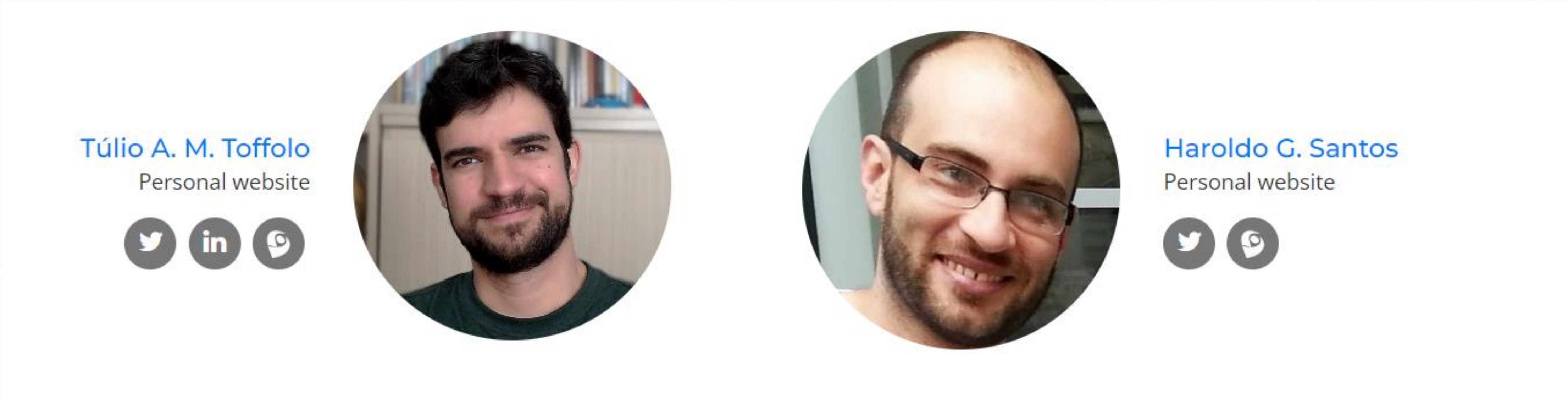
Python-MIP

- No curso vamos ver o Python-MIP:
  - Python-MIP: biblioteca do python para problemas PPL e PPI



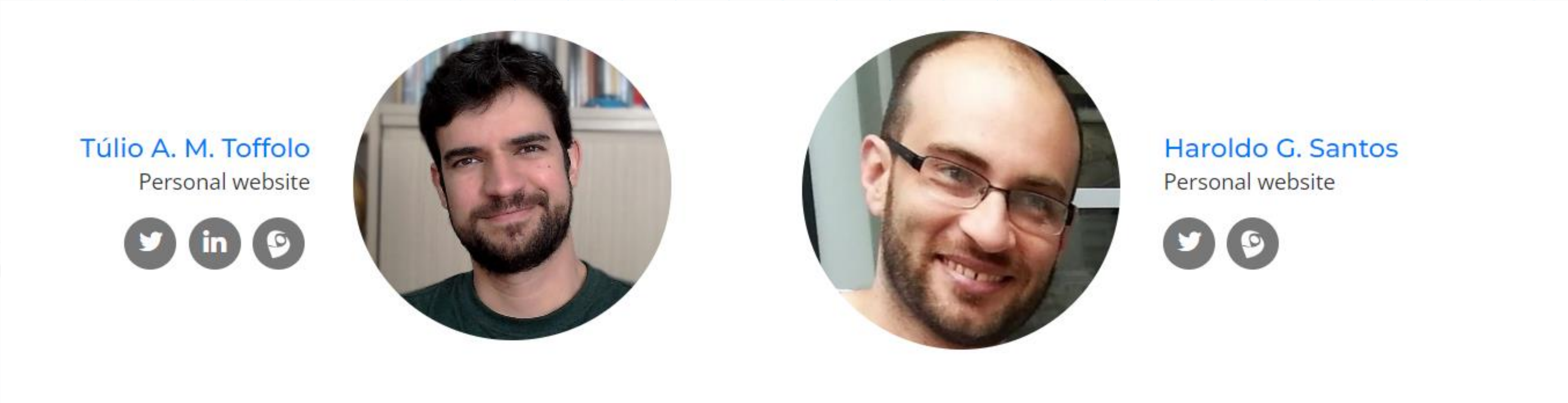
Python-MIP

- <u>No curso vamos ver o Python-MIP:</u>
    - <u>Python-MIP</u>: biblioteca do python para problemas PPL e PPI
        - Muito fácil de instalar e usar, já vem com o solver gratuito (COIN-OR)
        - Desempenho lento e manipulação limitada (COIN-OR)
        - Ideal para problemas pequenos, testes e aprendizado (COIN-OR)

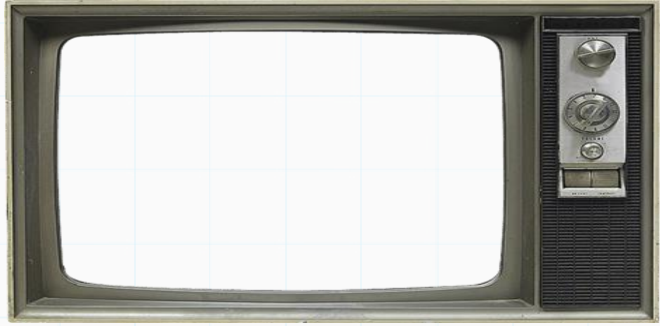


Python-MIP

- <u>No curso vamos ver o Python-MIP:</u>
  - <u>Python-MIP</u>: biblioteca do python para problemas PPL e PPI
    - Muito fácil de instalar e usar, já vem com o solver gratuito (COIN-OR)
    - Desempenho lento e manipulação limitada (COIN-OR)
    - Ideal para problemas pequenos, testes e aprendizado (COIN-OR)
    - Pode ser usado com o GUROBI (se instalado), se tornando uma ferramenta profissional.



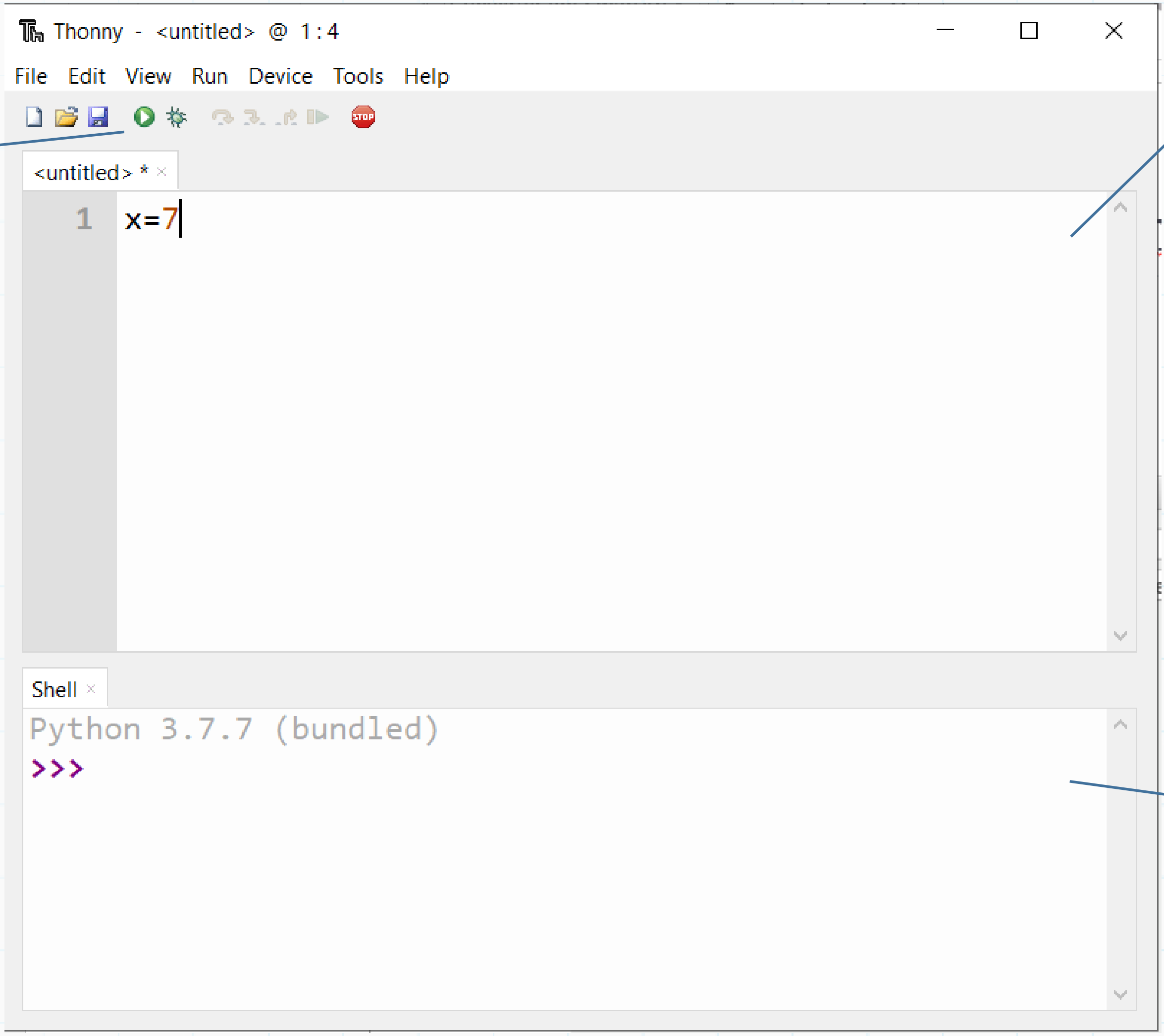
Python-MIP



Instalando o IDE+Compilador

- Usaremos na aula o Thonny (leve e educativo).
- Tem para Windows, MAC e Linux
- https://thonny.org/

para executar o código



onde o código é escrito

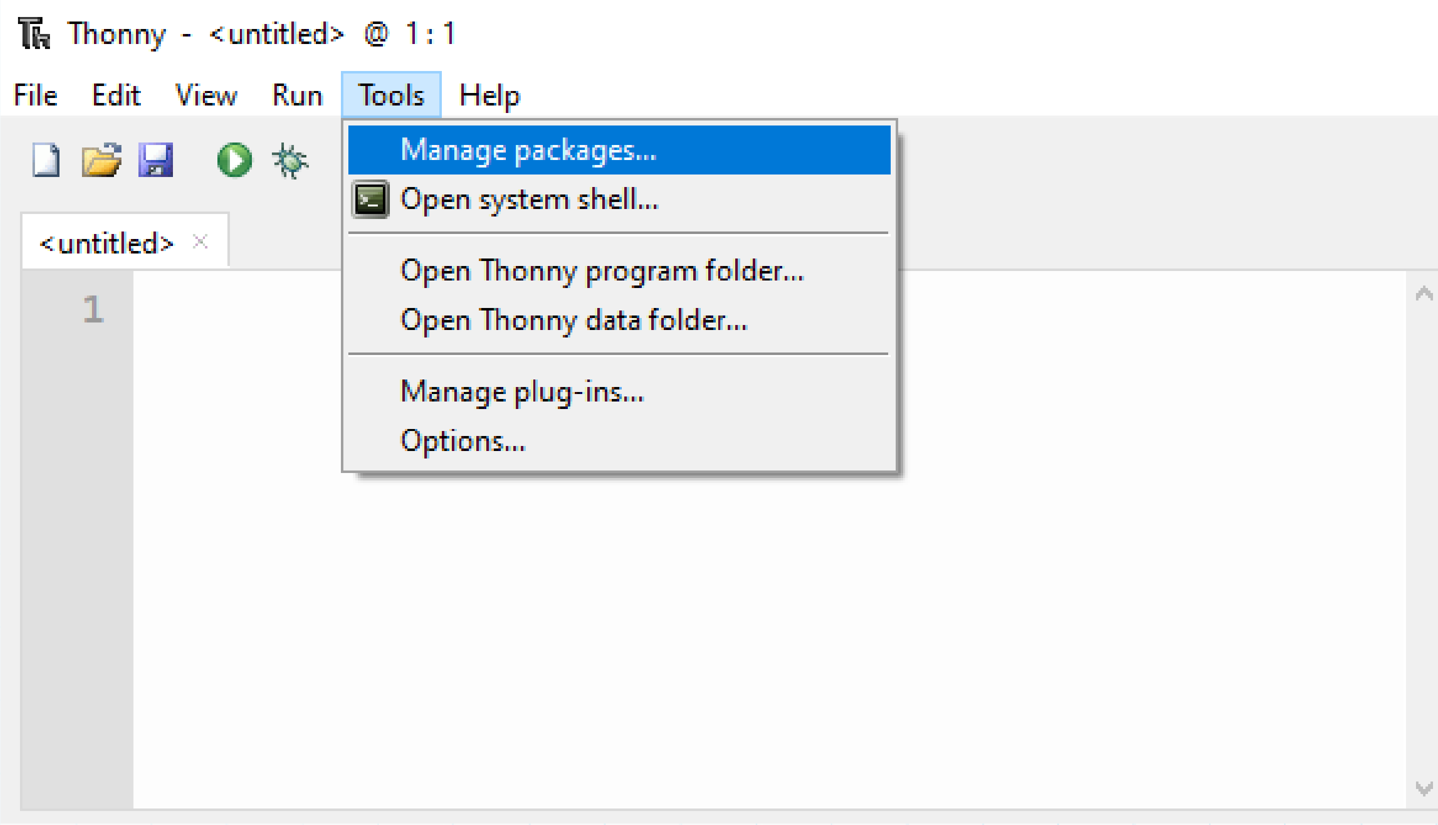
onde a saída é mostrada

Thonny

Python IDE for beginners

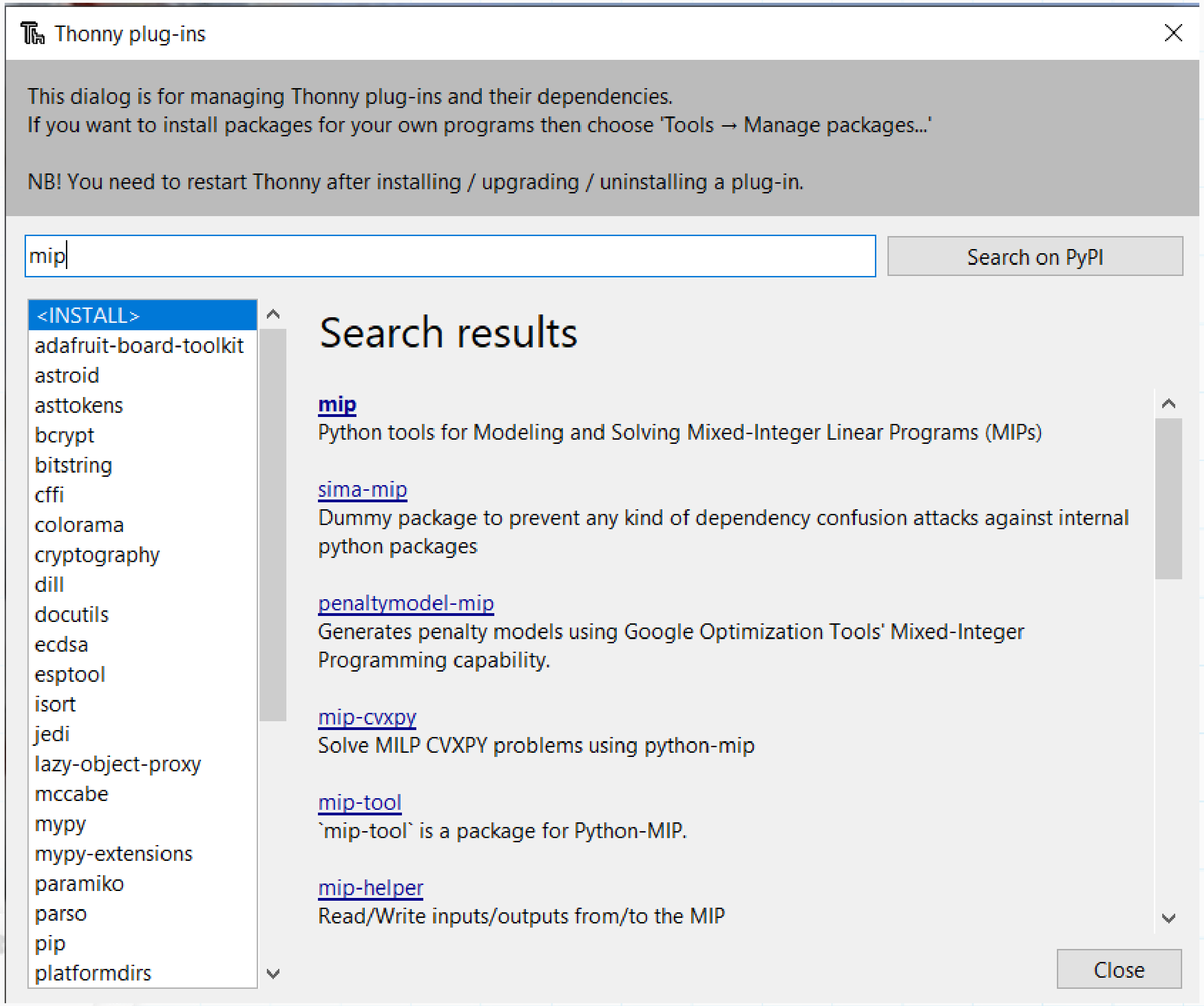
Python-MIP

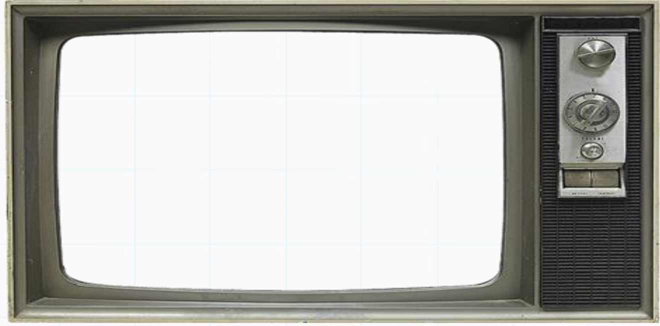
Instalando a biblioteca Python-MIP



Python-MIP

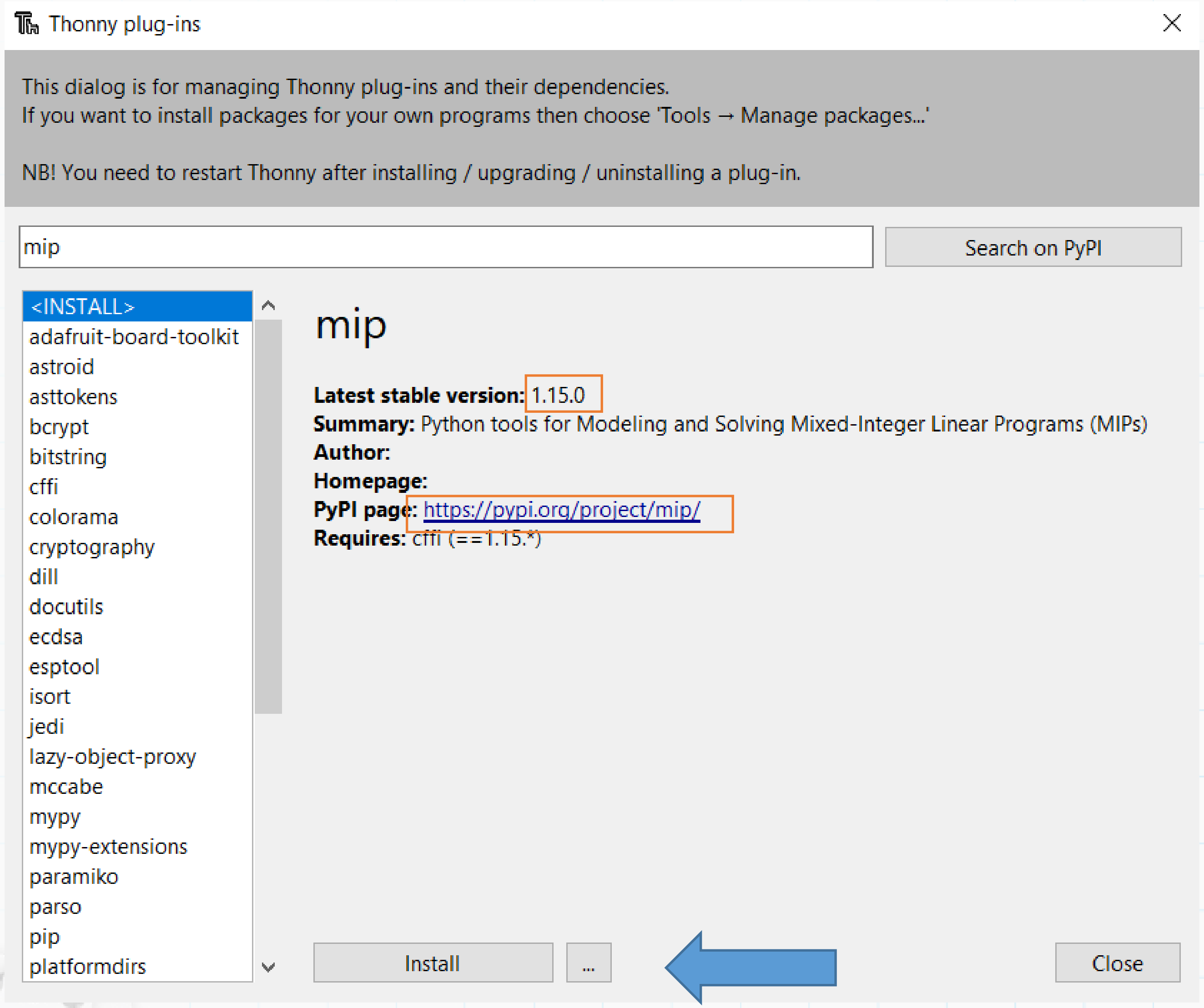
Instalando a biblioteca Python-MIP

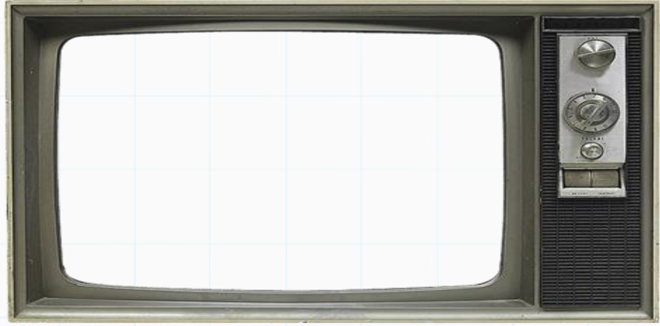




Python-MIP

Instalando a biblioteca Python-MIP





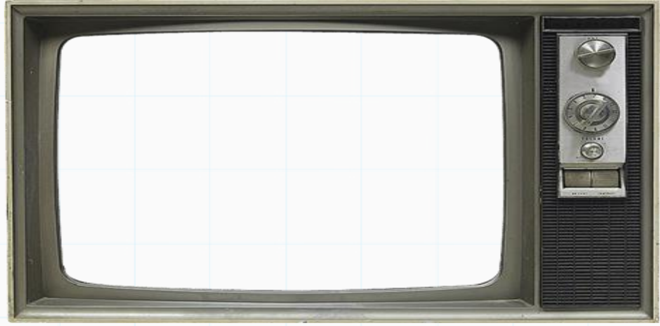
Pronto, acabou toda a instalação e está pronto para usar

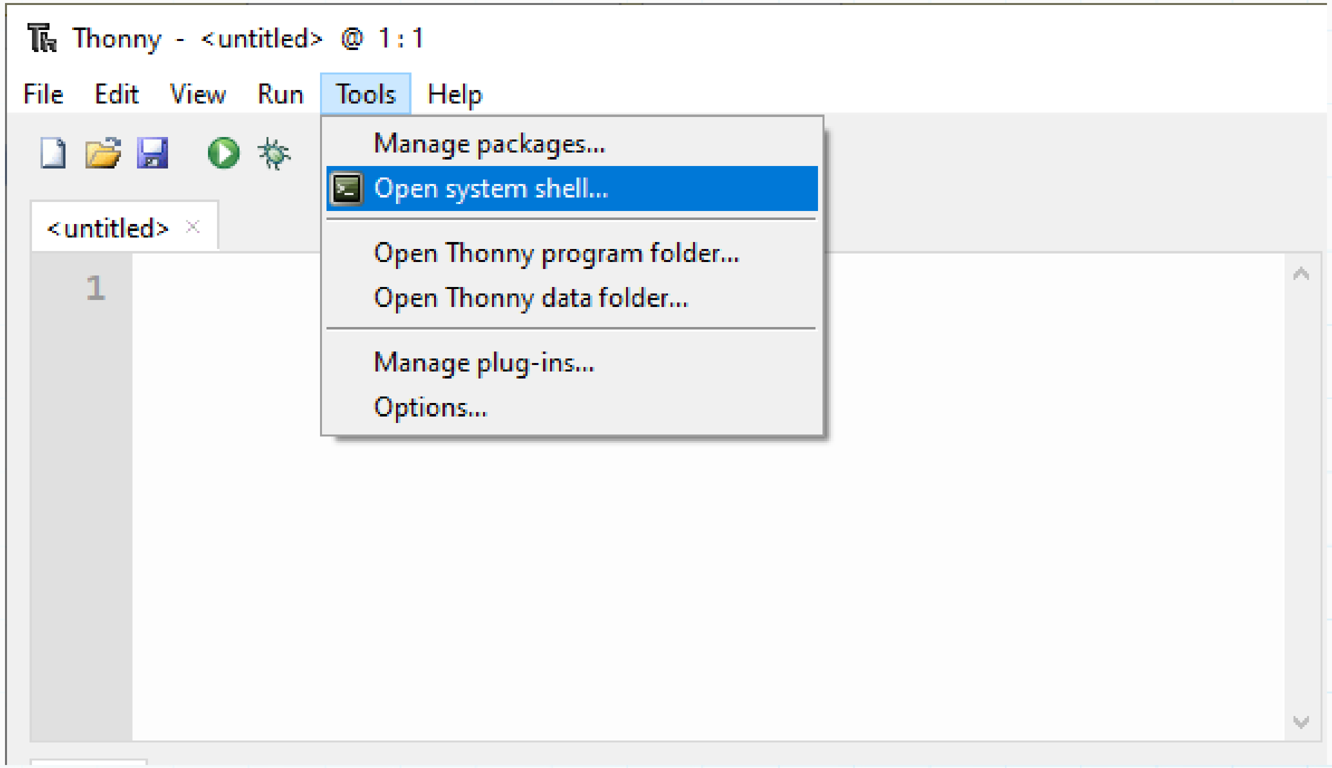
NÃO ESQUEÇA DE REINICIAR O THONNY !

Python-MIP

Instalando a biblioteca Python-MIP

Alguns computadores podem ter problema e não conseguir usar o gerenciador de pacotes



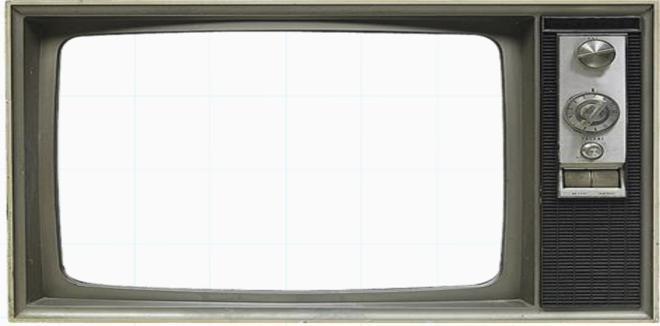


Nesse caso podemos instalar por linha de comando

Python-MIP

Instalando a biblioteca Python-MIP

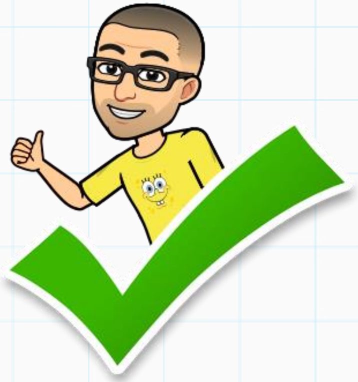
Alguns computadores podem ter problema e não conseguir usar o gerenciador de pacotes



C:\Windows\system32\cmd.exe

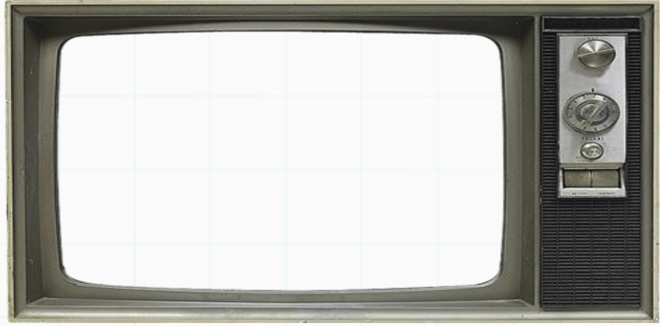
```
****************************************************************************************
Some Python commands in the PATH of this session:
 - python    == C:\Users\yuri\AppData\Local\Programs\Thonny\python.exe
 - pip       == C:\Users\yuri\AppData\Local\Programs\Thonny\Scripts\pip.bat
 - pip3      == C:\Users\yuri\AppData\Local\Programs\Thonny\Scripts\pip3.bat
 - pip3.7    == C:\Users\yuri\AppData\Local\Programs\Thonny\Scripts\pip3.7.bat

****************************************************************************************

G:\Meu Drive\Cursos\PO\Yuri\13 - LAB Python-MIP\Coloração>pip install mip_
```

Nesse caso podemos instalar por linha de comando:
`pip install mip`



NÃO ESQUEÇA DE REINICIAR O THONNY !

Python-MIP



1-python_exemplo_formulacao: 4 exemplos para vocês testarem

Python-MIP

Ex1 (PPL):

MAX $x_0 + 2x_1 + 3x_2$

s.a.

$-x_0 + x_1 + x_2 \le 20$

$x_0 - 3x_1 + x_2 \le 30$

$x_0 \le 40$

$x_0, x_1, x_2 \ge 0$

Python-MIP

Ex1 (PPL):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX $x0 + 2x1 + 3x2$

s.a.

$-x0 + x1 + x2 <= 20$

$x0 – 3x1 + x2 <= 30$

$x0 <= 40$

$x0,x1,x2 >=0$

- Importa biblioteca mip

Python-MIP

Ex1 (PPL):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX $x0 + 2x1 + 3x2$

s.a.

$-x0 + x1 + x2 <= 20$

$x0 – 3x1 + x2 <= 30$

$x0 <= 40$

$x0,x1,x2 >=0$

- Importa biblioteca mip
- Cria modelo

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2
s.a.
-x0 + x1 + x2 <= 20
x0 – 3x1 + x2 <= 30
x0 <= 40
x0,x1,x2 >=0

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2
s.a.
-x0 + x1 + x2 <= 20
x0 – 3x1 + x2 <= 30
x0 <= 40
x0,x1,x2 >=0

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia
- Cria restrição

Python-MIP

Ex1 (PPL):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

MAX $x0 + 2x1 + 3x2$

s.a.

$-x0 + x1 + x2 <= 20$

$x0 – 3x1 + x2 <= 30$

$x0 <= 40$

$x0,x1,x2 >=0$

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia
- Cria restrição
- Define objetivo

Python-MIP

MAX x0 + 2x1 + 3x2

s.a.

-x0 + x1 + x2 <= 20

x0 – 3x1 + x2 <= 30

x0 <= 40

x0,x1,x2 >=0

Ex1 (PPL):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)

# restrições
model.add_constr(-x0 + x1 + x2 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest1')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

- Importa biblioteca mip
- Cria modelo
- Cria variável e retorna referencia
- Cria restrição
- Define objetivo
- Otimiza e imprime saída

Python-MIP

Ex1 (PPL):

MAX x0 + 2x1 + 3x2

s.a.

-x0 + x1 + x2 <= 20

x0 – 3x1 + x2 <= 30

x0 <= 40

x0,x1,x2 >=0

```
from mip import *

# cria model
model = Mo

# variáve
x0 = mode
x1 = mode
x2 = mode

# restriç
model.add
model.add

# função
model.obje

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

Importa biblioteca mip

rencia

Saída

```
>>> %Run -c $EDITOR_CONTENT

 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming problem using Primal Simplex

 sol =  202.5
```

Python-MIP

MAX   x0 + 2x1 + 3x2
s.a.

Ex1 (PPL):

O CBC não vai funcionar com plataformas 32bits

```
>>> %Run ex1.py

 An error occurred while loading the CBC library:       Win32 platform not supported.

 Traceback (most recent call last):
   File "C:\Users\yuri\OneDrive\Desktop\python exemplo formulacao\ex1.py", line 4, in <module>
     model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)
   File "C:\Users\yuri\AppData\Roaming\Python\Python37\site-packages\mip\model.py", line 87, in __init__
     import mip.cbc
   File "C:\Users\yuri\AppData\Roaming\Python\Python37\site-packages\mip\cbc.py", line 603, in <module>
     Osi_getNumCols = cbclib.Osi_getNumCols
 NameError: name 'cbclib' is not defined
```

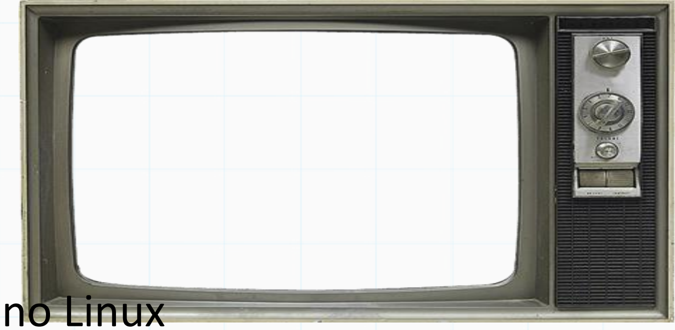
pode tentar resolver em:

https://docs.python-mip.com/en/latest/install.html#using-your-own-cbc-binaries-optional

Ou tentar outro IDE (Pycharm, etc, ...)

```
# saida
print("sol = ", model.objective_value)
```

Python-MIP



- Caso não consiga rodar em Windows, pode tentar instalar Linux e depois instalar o Thonny no Linux

https://www.virtualbox.org/

Welcome to VirtualBox.org!

https://www.vmware.com/in/products/workstation-player/workstation-player-evaluation.html

- Depois instalar linux

https://ubuntu.com/download/desktop

Python-MIP

Ex2 (PPI):

$$\text{MAX} \quad x_0 + 2x_1 + 3x_2 + x_3$$

s.a.

$$-x_0 + x_1 + x_2 + 10x_3 <= 20$$

$$x_0 - 3x_1 + x_2 <= 30$$

$$x_1 - 3.5x_3 = 0$$

$$x_0 <= 40$$

$$x_0, x_1, x_2 >= 0$$

$$2 <= x_3 <= 3 \text{ e inteira}$$

Python-MIP

MAX $x0 + 2x1 + 3x2 + x3$

s.a.

$-x0 + x1 + x2 + 10x3 <= 20$

$x0 - 3x1 + x2 <= 30$

$x1 - 3.5x3 = 0$

$x0 <= 40$

$x0,x1,x2 >= 0$

$2 <= x3 <= 3$ e inteira

Ex2 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo1",sense=MAXIMIZE, solver_name=CBC)

# variáveis
x0 = model.add_var(name='x0', var_type=CONTINUOUS, lb=0, ub=40)
x1 = model.add_var(name='x1', var_type=CONTINUOUS, lb=0)
x2 = model.add_var(name='x2', var_type=CONTINUOUS, lb=0)
x3 = model.add_var(name='x3', var_type=INTEGER, lb=2, ub=3)

# restrições
model.add_constr(-x0 + x1 + x2 + 10*x3 <= 20, name='rest1')
model.add_constr(x0 - 3*x1 + x2 <= 30, name='rest2')
model.add_constr(x1 - 3.5*x3 == 0, name='rest3')

# função objetivo
model.objective = maximize(x0 + 2*x1 + 3*x2 + x3)

# otimiza
model.optimize()

# saida
print("sol = ", model.objective_value)
```

Python-MIP

```
>>> %Run ex2.py
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming relaxation problem using Primal Simplex

 Coin0506I Presolve 2 (-1) rows, 3 (-1) columns and 6 (-3) elements
 Clp1000I sum of infeasibilities 0 - average 0, 1 fixed columns
 Coin0506I Presolve 2 (0) rows, 2 (-1) columns and 4 (-2) elements
 Clp0029I End of values pass after 2 iterations
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0032I Optimal objective 125.2083333 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00

 Starting MIP optimization
 Cgl0004I processed model has 2 rows, 3 columns (1 integer (0 of which binary)) and 6 elements
 Coin3009W Conflict graph built in 0.000 seconds, density: 0.000%
 Cgl0015I Clique Strengthening extended 0 cliques, 0 were dominated
 Cbc0045I Nauty did not find any useful orbits in time 0
 Cbc0038I Initial state - 1 integers unsatisfied sum - 0.0833333
 Cbc0038I Pass   1: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass   2: suminf.    0.08333 (1) obj. -125.208 iterations 0

 Cbc0038I Pass  47: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  48: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  sol =  122.5
```

Prepoc

Python-MIP

```
>>> %Run ex2.py
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming relaxation problem using Primal Simplex

 Coin0506I Presolve 2 (-1) rows, 3 (-1) columns and 6 (-3) elements
 Clp1000I sum of infeasibilities 0 - average 0, 1 fixed columns
 Coin0506I Presolve 2 (0) rows, 2 (-1) columns and 4 (-2) elements
 Clp0029I End of values pass after 2 iterations
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0032I Optimal objective 125.2083333 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00

 Starting MIP optimization
 Cgl0004I processed model has 2 rows, 3 columns (1 integer (0 of which binary)) and 6 elements
 Coin3009W Conflict graph built in 0.000 seconds, density: 0.000%
 Cgl0015I Clique Strengthening extended 0 cliques, 0 were dominated
 Cbc0045I Nauty did not find any useful orbits in time 0
 Cbc0038I Initial state - 1 integers unsatisfied sum - 0.0833333
 Cbc0038I Pass   1: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass   2: suminf.    0.08333 (1) obj. -125.208 iterations 0

 Cbc0038I Pass  47: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  48: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  sol =  122.5
```

Cortes

Python-MIP

```
>>> %Run ex2.py
 Welcome to the CBC MILP Solver
 Version: Trunk
 Build Date: Oct 28 2021

 Starting solution of the Linear programming relaxation problem using Primal Simplex

 Coin0506I Presolve 2 (-1) rows, 3 (-1) columns and 6 (-3) elements
 Clp1000I sum of infeasibilities 0 - average 0, 1 fixed columns
 Coin0506I Presolve 2 (0) rows, 2 (-1) columns and 4 (-2) elements
 Clp0029I End of values pass after 2 iterations
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Clp0000I Optimal - objective value 125.20833
 Coin0511I After Postsolve, objective 125.20833, infeasibilities - dual 0 (0), primal 0 (0)
 Clp0032I Optimal objective 125.2083333 - 0 iterations time 0.002, Presolve 0.00, Idiot 0.00

 Starting MIP optimization
 Cgl0004I processed model has 2 rows, 3 columns (1 integer (0 of which binary)) and 6 elements
 Coin3009W Conflict graph built in 0.000 seconds, density: 0.000%
 Cgl0015I Clique Strengthening extended 0 cliques, 0 were dominated
 Cbc0045I Nauty did not find any useful orbits in time 0
 Cbc0038I Initial state - 1 integers unsatisfied sum - 0.0833333
 Cbc0038I Pass   1: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass   2: suminf.    0.08333 (1) obj. -125.208 iterations 0


 Cbc0038I Pass  47: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  48: suminf.    0.08333 (1) obj. -125.208 iterations 0
 Cbc0038I Pass  sol =  122.5
```

Árvore de enumeração: B&C

Python-MIP

Ex3 (PPI):

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

Existe um formato específico para escrever modelos (.lp)

Variáveis que não aparecem no Bounds tem limites entre 0 e +inf

General = var. inteiras
Binary = var. binárias

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

-lê modelo, podemos ver também número de variáveis e restrições

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

STATUS:

OptimizationStatus.OPTIMAL

OptimizationStatus.FEASIBLE

OptimizationStatus.NO_SOLUTION_FOUND

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

- Retorna referencia para vetor de variáveis
- Cada variável possui um campo ‘x’ com o valor da variável.

Python-MIP

Ex3 (PPI):

```
from mip import *

# cria modelo
model = Model(name="exemplo3",sense=MAXIMIZE, solver_name=CBC)

# le arquivo .lp
model.read('ex3.lp')
print('modelo tem', model.num_cols, 'variaveis')
print('e ', model.num_rows, ' restrições')

# otimiza
status = model.optimize()
print("\n",status)

# valores das variaveis
variaveis = model.vars

for i in range(len(variaveis)):
    print(variaveis[i].x)

# saida
print("sol = ", model.objective_value)
```

ex3.lp - Bloco de Notas

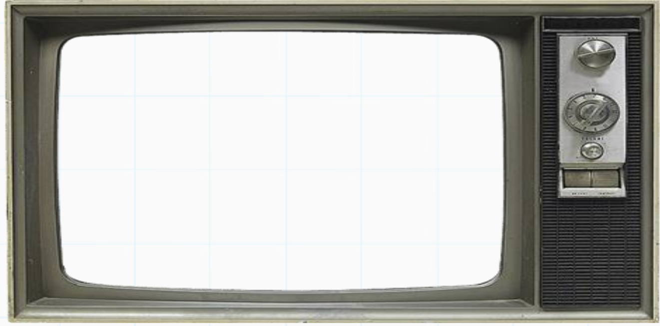
Arquivo Editar Formatar Exibir Ajuda

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
 x4
End
```

```
 OptimizationStatus.OPTIMAL
40.0
10.5
19.5
3.0
sol =  122.5
```

Python-MIP

- Documentação: [link]

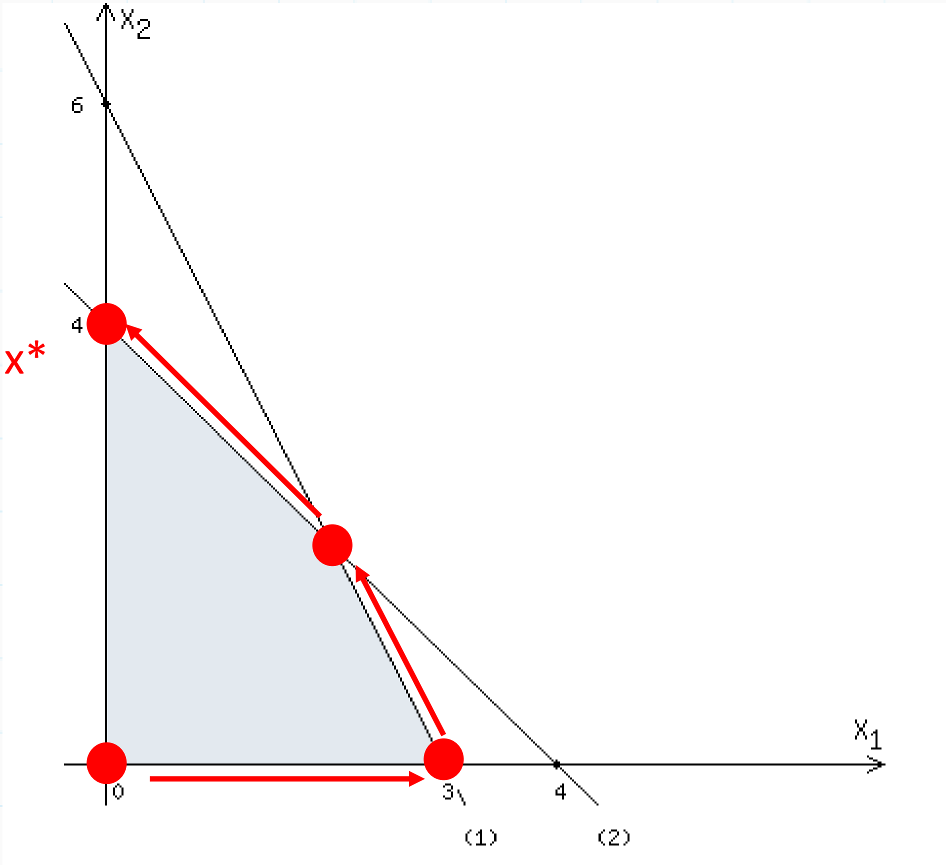


Exercício

PPL1: Vamos ver se conseguimos implementar e resolver esse PPL que já resolvemos com o Simplex anteriormente, para alcançar a mesma solução.

$$\max \quad x_1 + 2x_2$$
$$\text{s.a.} \quad 2x_1 + x_2 \leq 6$$
$$x_1 + x_2 \leq 4$$
$$x_1, \; x_2 \geq 0$$

$x^*_1=0, x^*_2=4$, com $Z^*=8$

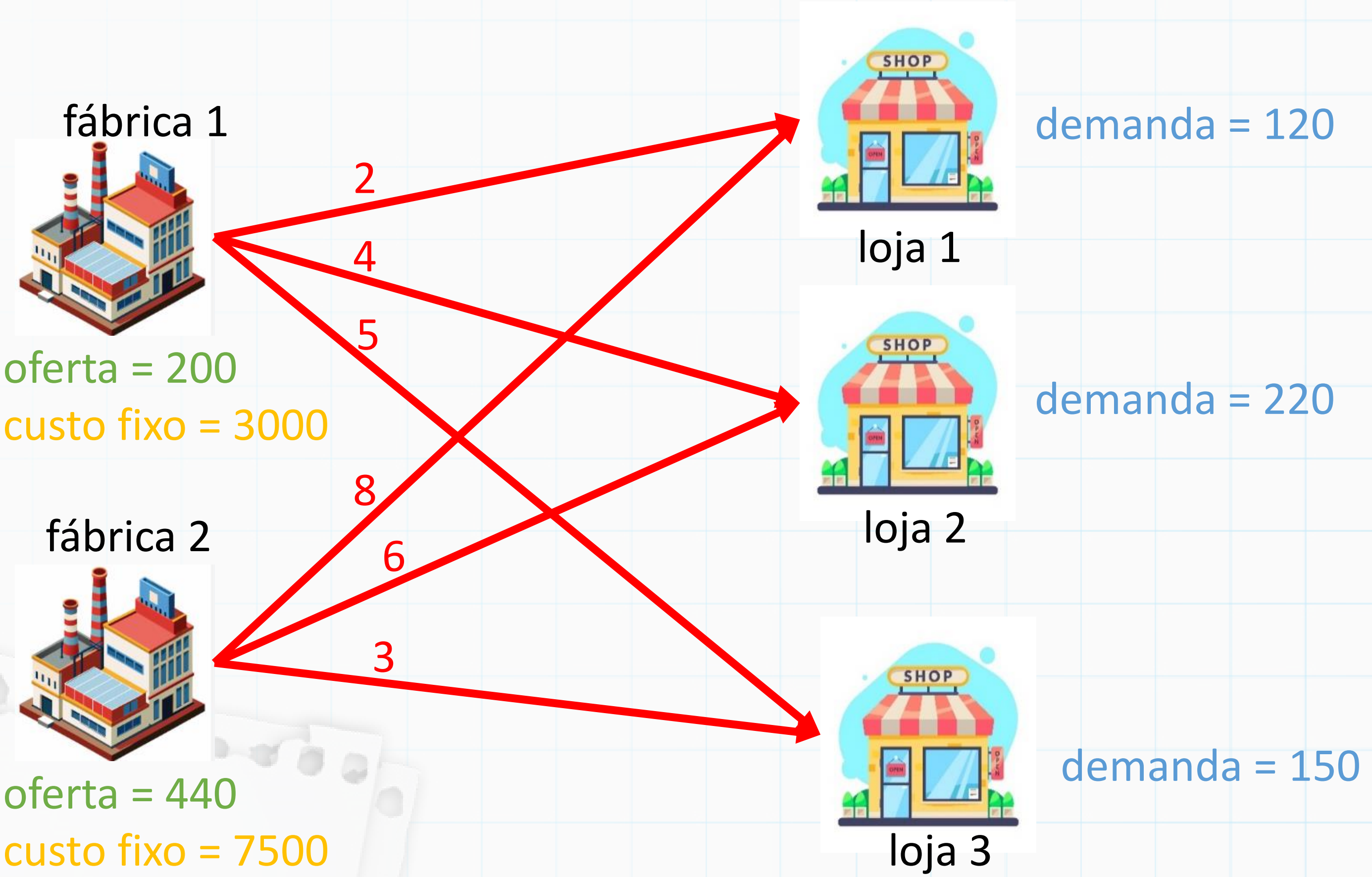


A pergunta é, qual a nova solução se introduzirmos uma nova restrição:

x2 <= 3.5

Exercício

Dolly: A empresa de guaraná Dolly possui 3 lojas para venda de seus produtos. E cada loja possui uma demanda (em azul) especifica (em garrafas). Para atender essa demanda, a empresa possui 2 fábricas, cada uma com uma capacidade de oferta (em verde) de garrafas de Dolly. Além disso, existe um custo unitário de transporte (por garrafa) entre as fábricas e as lojas (em vermelho). Mais ainda, se a fábrica realizar qualquer entrega, existe um custo fixo (em laranja) a ser pago (não importa se entregou 1 ou 1000 garrafas) além do custo por unidade. Queremos ajudar a empresa Dolly a estabelecer as entregas de forma que : (1) toda a demanda das lojas é atendida (2) os limites de oferta das fábricas não são ultrapassados (3) as entregas foram feitas ao menor custo possível.



Modelo a seguir:

xij -> qtd de garrafas enviadas da fabrica i para a loja j (i=1,2 e j=1,2,3)
yi -> se a fabrica i realizou alguma entrega ou não (i=1,2)

Exercício

$$\min \quad 2x_{11} + 4x_{12} + 5x_{13} + 8x_{21} + 6x_{22} + 3x_{23} + 3000y_1 + 7500y_2$$

$$x_{11} + x_{21} = 120$$

$$x_{12} + x_{22} = 220$$

$$x_{13} + x_{23} = 150$$

$$x_{11} + x_{12} + x_{13} \leq 200 \cdot y_1$$

$$x_{21} + x_{22} + x_{23} \leq 440 \cdot y_2$$

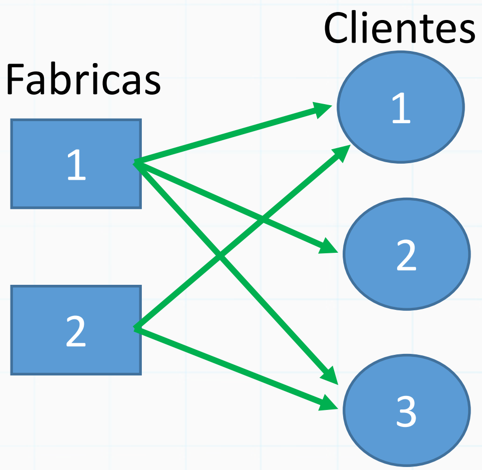
$$x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \in Z^+ \text{ and } y_1, y_2 \in \{0, 1\}$$

Qual deve ser os valores das variáveis no ponto ótimo ? Sabendo que o valor ótimo é 12350.

Exercício

<u>Facilidades:</u> Seja o conjunto de clientes 2 que precisam ser atendidos, e o conjunto de 3 fábricas que podem atender os clientes (<u>veja que a fábrica 2 não atende o cliente 2</u>). Para cada fábrica e para cada cliente temos (em vermelho) o custo de atendimento total do cliente pela fábrica. Porém, se a fábrica atender um ou mais clientes, ela tem um custo fixo de atendimento de 50 (o mesmo para as 2 fábricas), além do custo de atendimento. Além disso, cada fábrica possui uma quantidade de recurso de 5 unidades para atendimento, porém se uma fábrica atender um cliente, este atendimento irá consumir uma demanda de unidades da fábrica (em verde). Precisamos definir que fábricas atenderão que clientes, minimizando os custos de atendimento e os custos fixos. Modele, Implemente e descubra os valores das variáveis na solução ótima.

Solução Ótima para checar se o modelo está correto: 165



Fab/Clientes	1	2	3
1	20/2	15/2	10/2
2	40/2	-	30/2

Variáveis: x_ij -> se fábrica i ∈ I atende cliente j ∈ J, ou não (custo de atendimento)
y_i -> se fábrica i ∈ I atende alguém, ou não (custo fixo)

Restrições: 1) Todo cliente tem que ser atendido 2) Se Fabrica atendeu alguém então tem que construir ela 3) para cada fábrica não ultrapassar o limite de recursos.

Até a próxima