

Révision démo

- Solver utilisé par défaut, solvers disponibles

```
49 print(list_solvers(onlyAvailable=True))
```

- Logging

```
51 status: int = prob.solve(PULP_CBC_CMD(msg=True))
```

- 3 façons de créer une contrainte; avec une expression, un constructeur, sans variable

```
23 c1: LpConstraint = x + y <= 2
```

```
26 # c1: LpConstraint = LpConstraint(e=x + y, sense=LpConstraintLE, name='une contrainte', rhs=2)
```

```
32 prob += x + y <= 2, "La même contrainte"
```

- Attention aux expressions invalides

```
34 # prob += x + y > 2
```

Formulation du PL – Exercice 07

$$\text{maximiser } z = 16000x + 10000y$$

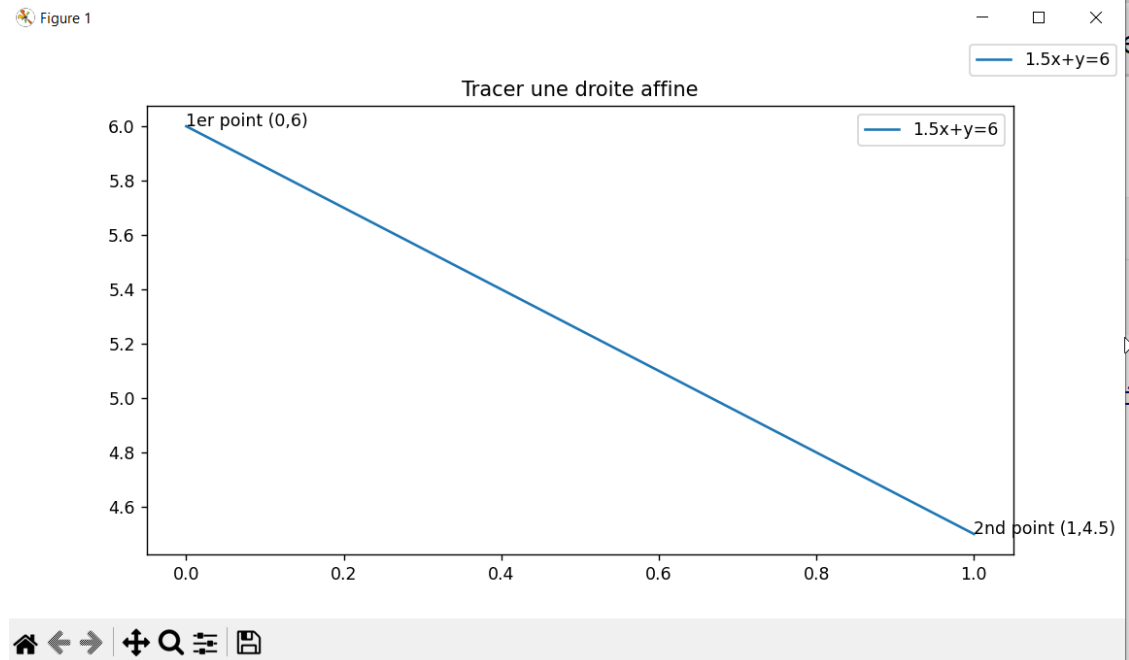
$$\text{s.c. } x + y \leq 400$$

$$2x + y \leq 600$$

$$x \geq 0, y \geq 0$$

Matplotlib

- Tracer une droite affine à partir de 2 points
- Ajouter un titre au graphe
- Ajouter des étiquettes à des coordonnées



Matplotlib

```
4 # Fonction objective (exercice 01);  $1.5x+y=6$ 
5 # Calculez 2 points pour la droite affine représentant la fonction objective
6 # si  $x_1 = 0$ ,  $y_1 = 6-1.5*0$ 
7 # si  $x_2 = 1$ ,  $y_2 = 6-1.5*1$ 
8 x: List[int] = [0, 1] # Les 2 abscisses sur l'axe des X
9 y: List[float] = [6, 4.5] # Les 2 ordonnées correspondantes au 2 X
10
11 # les 2 concepts de bases d'un graphique est la figure (la fenêtre affichée)
12 # et son axe dans lequel on dessine les courbes
13 fig, ax = plt.subplots()
14
15 # Tracer la droite affine sur l'axe (elle n'est pas encore affichée)
16 # ax.plot(x, y)
17 # Création du graphique sur l'axe avec une légende
18 ax.plot(x, y, label="1.5x+y=6")
19 # Place la légende à l'intérieur de la zone rectangulaire d'affichage
20 # soit de la figure
21 fig.legend(loc="upper right")
22 # soit par rapport à l'axe
23 ax.legend(loc="upper right")
```

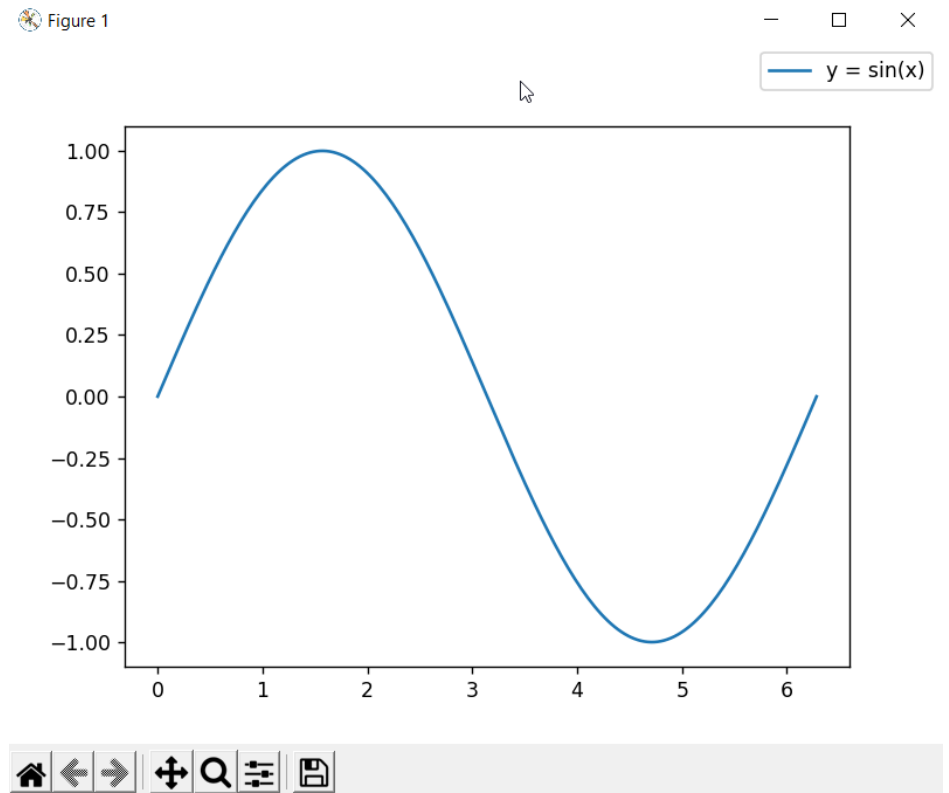
Matplotlib

```
25 # Ajouter un titre au graphique (plt)
26 plt.title("Tracer une droite affine")
27
28 # Ajouter une étiquette à des coordonnées précises
29 plt.text(0, 6, f'1er point (0,6)')
30 plt.text(1, 4.5, f'2nd point (1,4.5)')
31
32 # Afficher le graphique
33 plt.show()
```

Matplotlib

Mon premier graphique

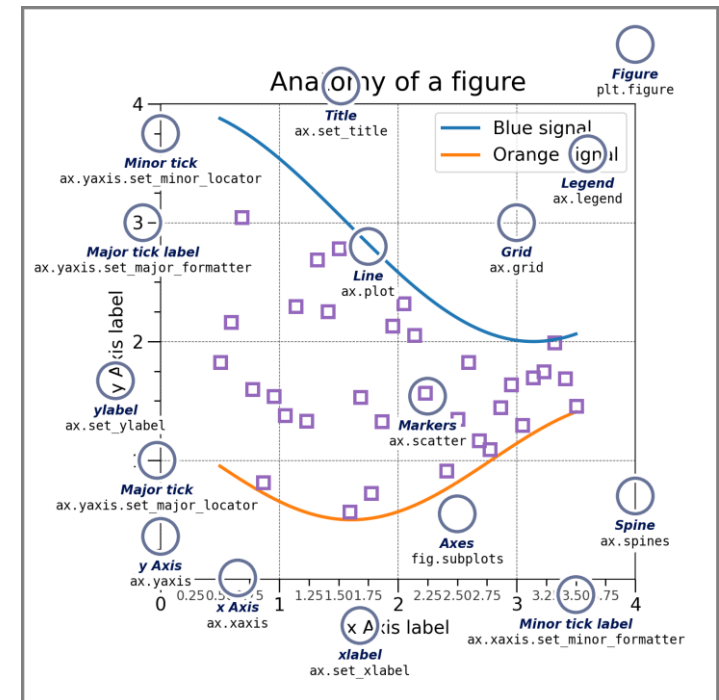
https://matplotlib.org/stable/users/getting_started/



Matplotlib

Les concepts de base

<https://matplotlib.org/stable/tutorials/introductory/usage.html>



Résolution graphique

1. Dessiner la zone de faisabilité
 - Bornes des variables de décision
 - Contraintes
2. Dessiner une fonction objective arbitraire
3. La solution est à l'intersection entre la zone de faisabilité et une droite affine parallèle à la fonction objective arbitraire

Résolution graphique du PL démo1 pulp

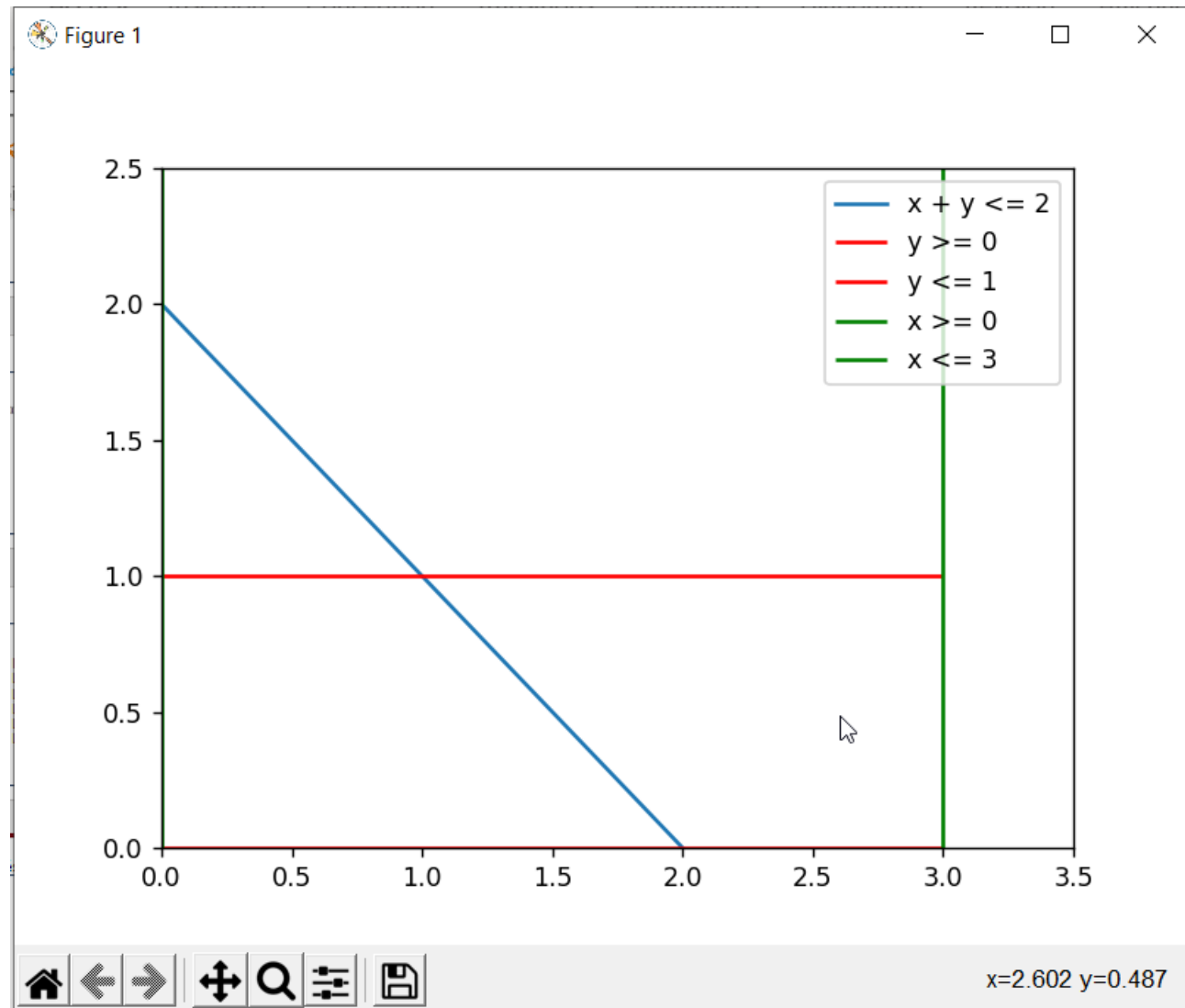
Fonction objective

$$\min_{\substack{0 \leq x \leq 3 \\ 0 \leq y \leq 1}} z = -4x + y$$

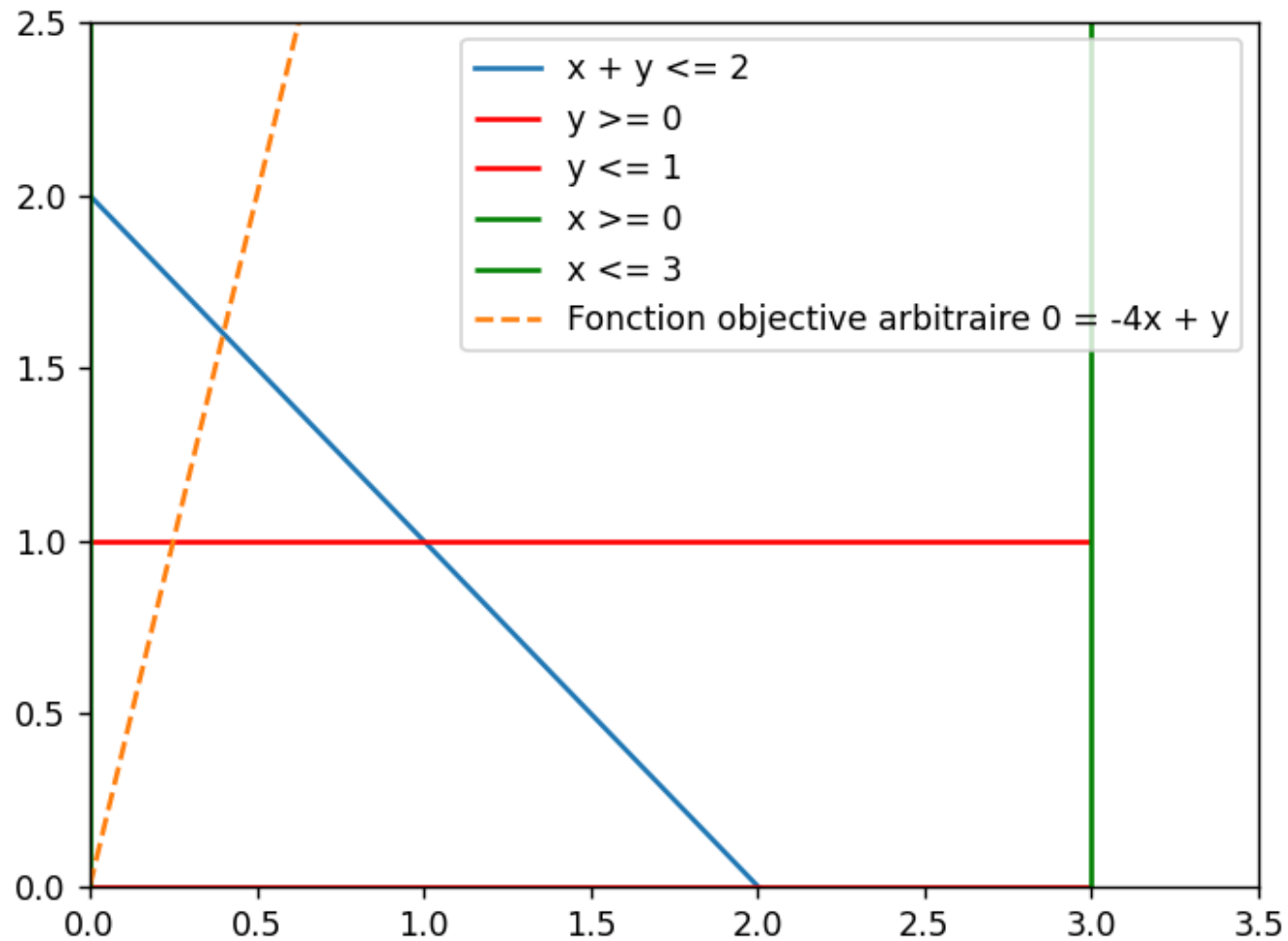
Sous contraintes

$$x + y \leq 2$$

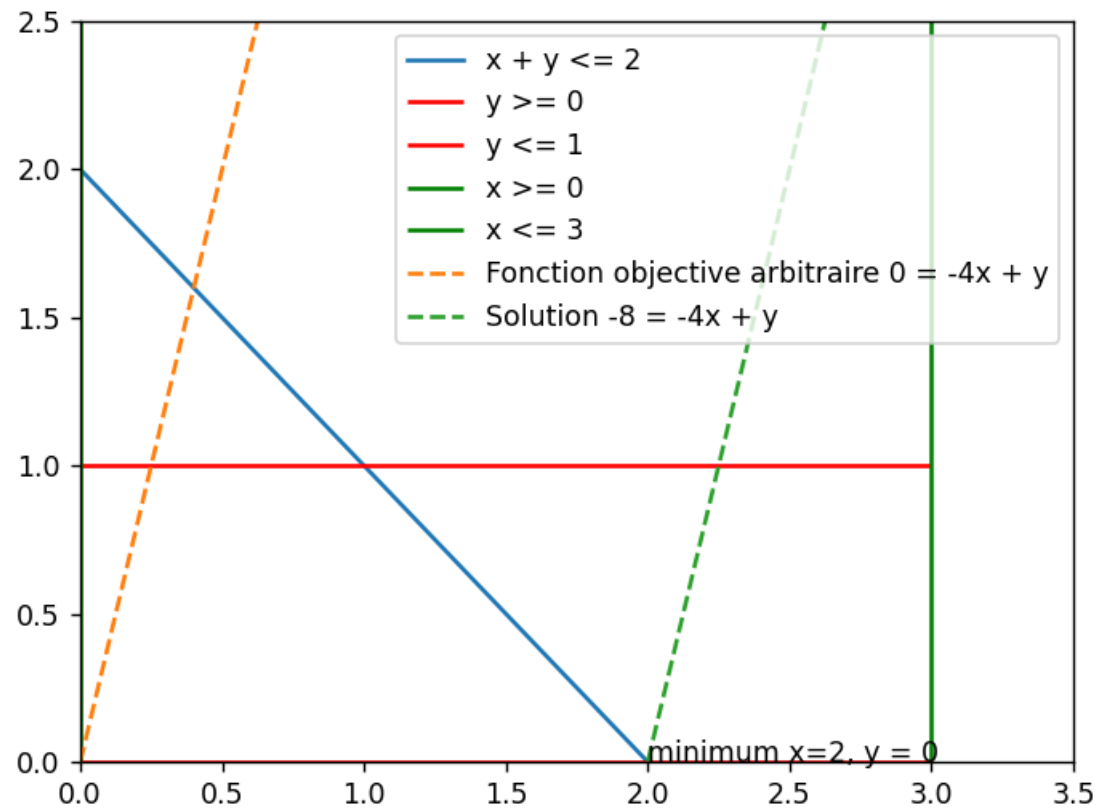
1 - Zone de faisabilité



2 - Fonction objective arbitraire



3 - Optimum



Dictionnaire de données et IpSum()

1. Pourquoi? Séparation IG/Calcul
2. Comment?
 - a) Dictionnaires,
 - b) IpSum()
 - c) « List compréhension »
 - d) `dict(zip(list(« clefs »), list(« valeurs »)))`
3. Démo pulp

Pourquoi?

Les données (constantes des contraintes/fonction objective) peuvent provenir de différentes sources;

- D'une interface graphique, d'un fichier
- Internet, etc...


Séparer (et tester) le code responsable de la collecte de données et du calcul (solver pulp)

Comment?

1. Créer plusieurs structures de données;
 - Clefs du dictionnaire = étiquettes identifiant les variables de décision
`List[str]`
 - 1 dictionnaire de variable de décision, `Dict[str, LpVariable]`
 - Des dictionnaires pour les coefficients de la fonction objective et des contraintes, `Dict[str, float]`
2. Utiliser `pulp.lpSum()`

« List comprehension »

```
1 from typing import List
2
3 XLABEL: str = "X"
4 NVAR: int = 3
5 # List comprehension permet de créer une liste dans une syntaxe abrégée utilisant un for loop
6 DECISION_VAR_KEYS: List[str] = [f"{XLABEL}{i}" for i in range(NVAR)]
7 print(DECISION_VAR_KEYS)
8 |
```



Création d'un dictionnaire avec zip()

```
11 DECISION_VAR_VALUES: List[int] = [i for i in range(NVAR)]
12 print(DECISION_VAR_VALUES)
13
14 DECISION_VAR_VALUES_DICT: Dict[str, int] = dict(zip(DECISION_VAR_KEYS, DECISION_VAR_VALUES))
15 print(DECISION_VAR_VALUES_DICT)
16
```

```
['X0', 'X1', 'X2']
[0, 1, 2]
{'X0': 0, 'X1': 1, 'X2': 2}
```

Démo

```
7 var_decisions_label: List[str] = ["X", "Y"]
8
9 # decision_vars: Dict[str, LpVariable] = LpVariable.dicts("Demo", var_decisions_label, lowBound=0, upBound=2)
10 # ou lorsque les bornes sont différentes
11 x: LpVariable = LpVariable("Demo_X", 0, 3)
12 y: LpVariable = LpVariable("Demo_Y", 0, 1)
13 decision_vars: Dict[str, LpVariable] = dict()
14 decision_vars["X"] = x
15 decision_vars["Y"] = y
16
17 objective_costs: Dict[str, float] = {"X": -4, "Y": 1}
18 objective: LpAffineExpression = lpSum([objective_costs[i] * decision_vars[i] for i in var_decisions_label])
19 prob += objective, "La fonction objective"
20
21 constraint1_constant = 2
22 constraint1_costs: Dict[str, float] = {"X": 1, "Y": 1}
23 c1: LpConstraint = lpSum([constraint1_costs[i] * decision_vars[i] for i in var_decisions_label]) <= constraint1_constant
24 prob += c1, "une contrainte"
```