Python 最適化第 1 回　Python で学ぶ最適化入門

Python を使って，Excel 内に入力されたデータを読み取り，最適化問題を解く方法を学習する．

0. 準備

0.1 Excel ファイル入出力モジュール OPENPYXL のインストール

今回の演習では，Excel 内のデータを読み込んだり，計算結果を Excel ファイルに書き込んだりする Python プログラムを作成する．Python で Excel ファイルの読み書きをするには，openpyxl というモジュールが便利なので，自分の仮想環境にインストールしよう．

インストール方法は，環境作成したときに matplotlib をインストールした方法と同じ．

0.2 最適化問題作成モジュール PULP のインストール

また今回の演習では，最適化問題を解く Python プログラムを作成する．Python で最適化問題を作成するには，pulp モジュールが便利．

残念ながら pulp モジュールは Anaconda には登録されていないので，手動でインストールする．Jupyter Notebook ファイル内に登録するコマンドを用意したので，一度だけ実行する．

1. 生産計画問題

1.1 対象の問題（Excel ファイル内シート「生産計画」参照）

ある個人経営の手作りアイスクリーム屋では，エスプレッソアイスとラズベリーアイスの 2 品種のみを扱っている．明日に向けて今晩商品を作りたいのだが，主原料である牛乳と作業時間に限りがある（具体的な数値は Excel ファイル参照）．

それぞれの商品が 1 個売れたときの利益を考慮して，総利益をなるべく大きくするには，それぞれ何個ずつ生産すればよいだろうか？ただし人気店のため，生産したアイスは全て売り切れると考えて良いものとする．

1.2 最適化問題を構成する要素（第 5 回演習までずっと使う用語なので覚えよう！）

① **決定変数**

問題において決定したいこと．例）アイスの個数

② **制約条件**

決定変数が守らなければならない条件．等式や不等式で表現．

例）使用する牛乳は上限以内，作業時間も上限以内

③ **目的関数**

どのような決定変数にすれば「最適」なのかを測る基準．式で表現．

例）総利益（最大化）

- **最適解**

  全ての制約条件を満たし，目的関数を最大（最小）にする決定変数の値の集合

- **最適値**

  決定変数が最適解になるときの目的関数の値

- **線形計画問題（Linear Programming：LP）**
  制約条件，目的関数が決定変数の一次式で表現される最適化問題．
  一次式の特徴を利用することで，比較的高速で最適解を得ることができる．
  制約条件は等式（＝）か不等式（≧，≦）を用いる．「≠」や「＞」，「＜」は使用できないことに注意．

1.3 定式化

エスプレッソアイス，ラズベリーアイスの個数をx_1，x_2（決定変数）として，
制約条件と目的関数を表すこと．

最大化	$???x_1 + ???x_2$	利益の最大化
制約条件	$???x_1 + ???x_2 \leq ???$	牛乳の条件
	$???x_1 + ???x_2 \leq ???$	作業時間の条件
	$x_i \geq 0 \quad (i = 1, 2)$	アイスの個数の条件

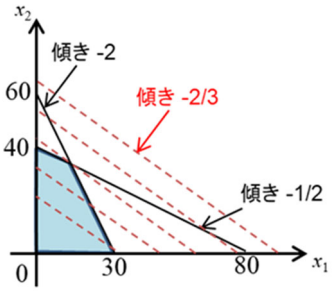
1.4 Jupyter Notebook で数式を書く

Jupyter Notebook のセルを Markup モードにすると，数式を綺麗に記すことができる．数式は$$で囲む範囲に記入する．主な表記方法を以下に示すが，数式が記入済みのセルをダブルクリックして，どのような表記をしているのか確認しよう．もっと詳しく知りたい人は，「Jupyter Notebook 数式」で検索するとよい．

表記	表示	表記	表示	表記	表示
x^2	x^2	¥leq	≤	半角空白	（表示なし）
x_{12}	x_{12}	¥geq	≥	全角文字	そのまま表示
¥¥	改行	¥半角空白	小空白	¥sum_{i=1}^{10} a_ix_i	$\sum_{i=1}^{10} a_i x_i$

1.5 図的解法（参考）

決定変数が 2 つの場合，すべての制約条件を満たす領域を図示して，目的関数を最大化（または最小化）する決定変数の値を求めることができる（本授業では扱わない）．



◎ 課題 1

Excel ファイルを参照して，???の箇所を定数に変更して，定式化を完成しなさい．
上の表を参考にして，正しく綺麗に数式を記すこと．（今後の課題も同様）

1.6 Excel ソルバー・最適化ソルバーによる最適化

Excel の「ソルバー」という機能を用いれば，今回の生産計画問題は解くことが可能（詳細は省略）．しかし，Excel ソルバーで解くことができる問題は，決定変数の個数が 200 個までであるため，現実的な大きさの問題には適用できないことが多い．そのため実用的な問題に対しては，プログラミング言語で作成されたプログラムによって最適解を求める．

最適化問題を解くためのアルゴリズムをプログラミング言語で一から構築することもあるが，線形計画問題のように，決まった形式の最適化問題に対しては，問題を解くためのソフトウェア（最適化ソルバー）を用いることが多い．本実験では無料の最適化ソルバーを利用する．

1.7 Python＋最適化ソルバーのプログラムの流れ

1. 最適化問題を作成するモジュール pulp を取り込む
2. 空の最適化問題を作成
3. 問題の中に，決定変数，制約条件，目的関数を追加する
4. 最適化ソルバーに問題を解かせる
5. 得られた解を表示する

1.8 Python プログラム（コメントは省略）

これ以降，プログラムの各行で何を行っているか，しっかり理解しよう．

```
from pulp import *
model = LpProblem('Production', sense=LpMaximize)
```

- 1 行目：最適化問題を記述する外部モジュール pulp を取り込む
- 2 行目：最適化問題[1]の枠組みを作成して，変数 model に格納．
  第 1 引数の 'Production' は問題の名前．自由に付けてよい．
  第 2 引数の sense=LpMaximize は最大化を表す．
  最小化がデフォルトなので，最小化の場合は指定なし，または sense=LpMinimize

```
x1 = LpVariable('エスプレッソ', lowBound=0)
x2 = LpVariable('ラズベリー', lowBound=0)
```

- 1 行目：決定変数（英語で Decision Variable）を作成．
  第 1 引数の 'エスプレッソ' は決定変数の名前．自由に付けてよいが，異なる決定変数には異なる名前を付けること．
  第 2 引数の lowBound は変数の下限．アイスの個数の条件 $x_i \geqq 0$ を意味する．
  上限は upBound も同様の方法で指定可能．

【余裕のある人向けの解説】

'エスプレッソ'等，引数数で指定する名前は，問題 model 内で扱う決定変数の名前．この名前は，model 内に保存される．一方，左辺の x1 は Python プログラムにおける変数（最適化問題における決定変数ではなく，Python プログラム内で値を格納する箱）の名前．今回 model に追加した決定変数エスプレッソの情報を，Python プログラムでの変数 x1 に格納している．

決定変数の名前（この例では，エスプレッソ）と Python プログラムの変数名（x1）は同じ

[1] 厳密には，線形計画問題（Linear Programming Problem）

でも構わない．

```
model += 100*x1 + 200*x2 <= 8000
model += 10*x1 + 5*x2 <= 300
```

「#制約条件の追加」の下に，上のプログラムを入力しよう．

- 1・2 行目：問題 model に制約条件を追加する．+=の右側が制約条件になる．
  逆向きの不等号は >=，条件が等号の場合は，== になる．
  ここで利用する決定変数は，x1 等の Python の変数．'エスプレッソ'等の名前ではない．

```
model += 100*x1 + 150*x2
```

「目的関数の設定」の下に，上のプログラムを入力しよう．

- 目的関数を設定．+=の右側に目的関数の式を指定する．+= と記されるが，再度実行した場合は，新しく設定した目的関数が有効となり，以前に指定した目的関数は消去される．

```
print(model)
```

- 作成した最適化問題を表示できる．今回のプログラムでは，次のように表示される．確認のために表示しているだけなので，表示しなくても構わない．

```
Production:                                  ←最適化問題の名前
MAXIMIZE                                     ←最大化
100*エスプレッソ + 150*ラズベリー + 0          ←目的関数
SUBJECT TO                                   ←ここから制約条件
_C1: 100 エスプレッソ + 200 ラズベリー <= 8000
_C2: 10 エスプレッソ + 5 ラズベリー <= 300

VARIABLES                                    ←ここから決定変数
エスプレッソ Continuous                       ←Continuous は非負の連続型（実数）
ラズベリー Continuous:
```

- 決定変数名がプログラム内で指定したエスプレッソとラズベリーになっていることが確認できる．model の内容を表示しているので，x1 等の Python の変数ではない．
- 制約条件における_C1，_C2 は制約条件名．PULP モジュールが自動的に付けた名前．

```
model.solve()

if LpStatus[model.status] == 'Optimal':
    print('最適値 =', value(model.objective))
    print('  ・', x1.name, '=', value(x1))
    print('  ・', x2.name, '=', value(x2))
else:
    print('最適解が求まりませんでした．')
```

- 1 行目：最適化問題 model を解く（solve）．大規模で複雑な問題は，この処理に時間がかかる．最適化ソルバーに作成した問題を渡して，解いた結果を受け取っている．
- 3 行目：最適解が求まった場合，model.status に整数 1 が代入される．よって，
  if model.status == 1 で最適解が求まったかどうか判定できるが，プログラムを分かりやすくするために上の表現を用いることが多い．
  【余裕のある人向けの解説】LpStatus は解いた結果の状態を表す文字列リストで，LpStatus[1]には文字列'Optimal'（最適）が格納されている．
- 4 行目：value(model.objective)で目的関数（Objective Function）の値，つまり最適値が得られる．最適化を実行して求められた値を得るには value()が必要，と覚えよう．
- 5 行目：最適解を表示．value(x1)で変数 x1 が表す決定変数の値が得られる．変数 x1 は決定変数エスプレッソに関する情報（例：名前エスプレッソ，下限 0，実数型）を保存していて，決定変数の値のみ得るのには value(x1) を用いる．

プログラムを実行して，解が求まるか確認しよう．（最適値：6333.33…）

1.9 整数条件の追加

元々の問題では，決定するのはアイスの個数なので，x1 と x2 は本来整数である．そこで x1 と x2 が整数である，という条件を追加して最適化を解いてみる．

作成したプログラムを空白のセル内にコピー&ペーストして，決定変数の作成部分を以下のように変更する．

- プログラムを入力するときには Tab キーの補完機能が便利．例えば ”Lp” まで入力して Tab キーを押せば，”Lp”で始まる候補が表示される．矢印キーで選択して，Enter で入力，またはマウスでダブルクリック．補完機能が働かないときは，一度セル内のプログラムを実行すると，次から機能することが多い（実行エラーが出ても気にしない）

```
x1 = LpVariable('エスプレッソ', lowBound=0, cat=LpInteger)
x2 = LpVariable('ラズベリー', lowBound=0, cat=LpInteger)
```

- cat はカテゴリー（category）の略．LpInteger は整数（Integer）型．下限は 0．
  何も指定しない場合は LpContinuous となり，連続型（Continuous．いわゆる実数型）．

実行した結果，整数解が得られたか確認しよう．（最適値：6300）

実数解の場合と比較して，最適値は減少する．なぜ減少するか理由を説明できますか？

◎ 課題 2

二つのアイスの 1 個あたりの利益が現在の設定（100，150）に加え，100，200 の場合，200，150 の場合，合計 3 つの場合における最適値，およびそのときの最適解（x1 と x2 の値）を続けて表示する 1 つプログラムを作成しなさい．整数条件も入れなさい．表示した結果が，どの設定に対する解答なのか分かりやすく出力すること．

余裕がある人は，関数を作成して，コンパクトなプログラムを作成しなさい（先に，関数を使わないバージョンを作成してから挑戦してもよい）．

【補足】一つのプログラムの中で複数回目的関数の設定をすると警告（warning）が表示されるが，今回のように意図的に行っている場合は，気にする必要はない．

（最適値 6300, 8000, 7600. 最適解(12, 34), (0, 40), (14, 32)）

2. ナップサック問題

2.1 対象の問題（Excel ファイル内シート「ナップサック」参照）

K 大学 C 学部では，来年度の図書費の予算が削減されたため，来年度購読する学術雑誌を選択することになった．（具体的な数値は Excel ファイル参照）

所属教員にアンケートを取り，各雑誌に対して購入を希望する度合い（希望度）を設定した．雑誌の価格と希望度を表にまとめた．予算 100 の範囲内で，希望度合計が最大になるように雑誌を選択したい．どの雑誌を選択すればよいだろうか？

2.2 定数

- b_i　　i 番目の雑誌の希望度（価値）．$i = 1, 2, \ldots, 18$.
- c_i　　i 番目の雑誌の価格（コスト）．$i = 1, 2, \ldots, 18$.
- C　　予算．購入する雑誌にかかる費用の上限.

上の記号と数値はこの後のプログラムでも用いるので，Excel ファイルを開いて確認すること．

2.3 最適化問題の構成要素

① 決定変数

x_i　　0-1 型．i 番目の雑誌を購入するなら 1，入れないなら 0.

② 制約条件

購入する雑誌の価格の合計が予算以内

③ 目的関数

購入する雑誌の希望度の合計を最大化

2.4 定式化（2.2・2.3 で定めた定数・決定変数を用いる）

最？化　$\sum_{?=?}^{?} ?$

制約条件　$\sum_{?=?}^{?} ??? $

$x_i = 0 \ or \ 1 \quad (i = 1, 2, \dots, 18)$　　各雑誌を購入する／しない

◎ 課題 3

定式化を完成させなさい.

今回は一般的な表現にするために，定数は 2.2 で示した文字を使うこと.

【注意】

変数と定数の積は，通常定数が左になります．直線の式は $y = ax + b$と書きますが，

$y = xa + b$とは書きませんよね？間違いではないですが，通常は前者のように書きます.

2.5 Python プログラム（途中から）：定数用データの作成

本当は雑誌数 18 であるが，手始めに雑誌数を 5 としてプログラムを作成しよう.

生産計画（整数条件を追加した版）のプログラムを，課題 4 用のセル内のコメントで指示しているように部分的にコピペして，ナップサック問題向けのプログラムのベースを作成し，以降の解説で示すようにプログラムを修正しよう.

```
I = [i+1 for i in range(5)]
b = {1:9, 2:3, 3:3, 4:7, 5:10}
c = {1:16, 2:13, 3:13, 4:14, 5:20}
C = 70
```

- 最適化で使用する定数用のデータを最初に作成しておく.
- I は [1, 2, 3, 4, 5]となるリストで，今回の問題では雑誌 No を表す．range(5) は 1 からではなく 0 から開始されることに注意.
- 希望度 b と価格 c は辞書で表現．例えば，b[2] = 3, b[4] = 7, c[1] = 16, c[5] = 20
- 雑誌数を 5 にしているため，予算上限 C は本当は 100 ですが，仮に 70 としました.

【余裕のある人向け解説】

I = [1, 2, 3, 4, 5]であれば，b や c はリストにすることも可能．リストは 0 番目から始まるので，0 番目にダミー値を入れて，b = [0, 9, 3, 7, 7, 7], c = [0, 16, 13, 13, 14, 20]と設定すれば，辞書と同じ参照方法が実現できる.

辞書にするメリットは，I = [10, 20, 30, 40, 50]のように雑誌 No が連続しない場合や，I = [‘Science’, ‘Nature’, ‘Math’, ‘Computers’, ‘LSI’]のように雑誌名で決定変数を区別する場合にも利用できる点である．そのため，定数を辞書で表現することが多い.

2.6 Python プログラム（つづき）：決定変数の作成

```
model = LpProblem('Knapsack', sense=LpMaximize)
x = {}
for i in I:
    x[i] = LpVariable(f'x{i}', cat=LpBinary)
```

- 扱う問題が変わったので，問題名も'Knapsack'に変更（問題名は自由なので，変更する必然性はない）
- 決定変数 x も辞書型．x[i]が i 番目の雑誌に関する決定変数 x_i に対応する．
- 今回解く問題では雑誌が 5 つなので，決定変数を 5 個作成する．生産計画のプログラムのように決定変数 1 つに対して 1 行ずつ記すなら，

```
x[1] = LpVariable('x1', cat=LpBinary)
x[2] = LpVariable('x2', cat=LpBinary)
x[3] = LpVariable('x3', cat=LpBinary)
x[4] = LpVariable('x4', cat=LpBinary)
x[5] = LpVariable('x5', cat=LpBinary)
```

  となるが，コードが長くなる．for 文を使うと上のようにシンプルに書ける．
- f'x{i}'は print 文で学んだ f 文字列と同じ形式．例えば i=1 なら 'x1' と記しているのと同じ．
- LpBinary はバイナリ型（Binary，0-1 型とも呼ぶ）の決定変数であることを表す．この表現で $x_i = 0 \; or \; 1$ の制約条件を表すことができる．

2.7 Python プログラム（つづき）：制約条件の追加，目的関数の設定

```
model += lpSum(c[i] * x[i] for i in I) <= C
model += lpSum(b[i] * x[i] for i in I)
```

- lpSum 関数は定式化での Σ と同じ働きをする．【注意】先頭の 2 文字 lp は小文字の LP.
  例えば制約条件内の lpSum は，c[1]*x[1] + c[2]*x[2] + c[3]*x[3] + c[4]*x[4] + c[5]*x[5]，すなわち，$\sum_{i=1}^{5} c_i x_i$ を意味する．
  定式化した式とプログラムを比較して，（雑誌数が 5 つになっている点を除いて）プログラムが定式化と同じ内容を表現していることを確認しよう．

2.8 Python プログラム（つづき）：最適化の結果の表示

```
for i in I:
    if value(x[i]) > 0.01:
        print(x[i].name)
```

- ここでは，最適値が 1（購入する）雑誌に対応する変数名を表示している．条件「値が 1」は，「if value(x[i]) == 1:」や「if value(x[i]) > 0:」でもよいと考えるかもしれないが，誤差対策のためにこの表現を採用している．つまり，理論上は X = 0 でもコンピュータ内で

は X = 0.0000000001 かもしれないし，理論上 X = 1 でも X = 0.99999999999 かもしれない．これらの場合でも，正しく表示できるように配慮している．value(x[i]) は 0 か 1 なので，if value(x[i]) > 0.5 とするのが確実であるが，本来の「最適値が 1」，または「最適値が 0 でない」の意味が取りにくい短所がある．右辺の 0.01 の小数桁数には強い意味はなく，誤差エラーが生じなければ他の桁数でも問題ない．

◎ 課題 4

上記の編集を行い，ナップサックナップサック問題を解いて，最適値，および購入する雑誌に対応する変数名を表示するプログラムを作成しなさい．

（最適値 29．最適解 x1, x2, x4, x5）

3. Excel ファイルとの連携：ナップサック問題

これまでのプログラムでは，最適化問題の定数（ナップサック問題での b や c，C）を Python プログラム内に記す必要があった．データが多い場合，プログラムとデータは分けて管理した方が便利．またデータを外部から読み込めれば，別のデータに対する最適化も同じプログラムで実行できる．

これ以降（第 5 回までずっと）は，Excel 内に入力されたデータを Python プログラムが読み込むように変更する．Excel ファイルは Jupyter Notebook ファイルと同じフォルダに入れよう．Google Colaboratory を利用している人は，Excel ファイルををアップロードしよう（環境作成の資料参照）．

3.1 Excel ファイルとの連携処理の流れ

1. Excel 入出力用のモジュール openpyxl を取り込む
2. Excel ファイルを開く
3. 目的のシートを開く（厳密には「シートを指定する」）
4. セルに入力済みのデータを読み込む
5. 主となる処理を行う．ここでは最適化問題を解く．
6. 結果をセルに書き込む（厳密には Excel ファイル内のセルではなく，メモリ内に読み込んだセルに書き込む．そのため，元のファイルには変更がされない．）
7. 開いている Excel ファイルとは別の Excel ファイルとして，書き込んだ内容を保存する（元のデータを消さないように，別のファイルに書き込むようにしている）．

課題 5 用のセル内のコメントで指示されたように，課題 4 のプログラムを課題 5 用のセル内に部分的にコピペしよう．

3.2 Python プログラム：Excel ファイル・シートを開く

```
from pulp import *
from openpyxl import *

book = load_workbook('最適化 1.xlsx')
sheet = book['ナップサック']
```

- 2 行目：Excel ファイル入出力モジュール openpyxl をプログラムに取り込む.
- 4 行目：Excel ファイル「最適化 1.xlsx」を開いて，変数 book に格納.
- 5 行目：開いた Excel ファイル内のシート「ナップサック」を変数 sheet に格納．これでシート「ナップサック」がコンピュータのメモリ上に保存され，データの読み書きができる.

3.3 Python プログラム（つづき）：定数用データの作成

```
I = [i+1 for i in range(18)]
b = {}
c = {}
for i in I:
    b[i] = sheet.cell(row=2, column=1+i).value
    c[i] = sheet.cell(row=3, column=1+i).value
C = sheet.cell(row=3, column=21).value
```

- 1 行目：雑誌の数を 18 に戻したので，I の範囲を変更.
- 2～6 行目：係数 b，c をまず空の辞書として作成して，Excel シートから値を 1 つずつ読み込んでデータを登録する.
- sheet.cell 関数は，シート sheet 内の指定した行（row）と列（column）にあるセルを獲得できる．今回はセル内に入力された値が欲しいので，セルの値 value を指定している.

プログラム内の row や column の数値と Excel シート「ナップサック」を参照しながら，どのセルの値がどの変数に格納されているのか，しっかり理解すること.（のちの課題では，みなさん自信で row や column の数値をプログラム内で指定することになります.）

3.4 Python プログラム（つづき）：最適化の結果出力

```
    for i in I:
        sheet.cell(row=4, column=1+i).value = value(x[i])
    book.save('最適化 1-5.xlsx')
```

- 画面出力する代わりに，最適値を Excel シートに書き込む．どの変数の値が Excel のどのセルに書き込まれているか，row や column の数値を確認してしっかり理解しよう.
- book.save で，現在メモリ上で開いている Excel ファイル book を指定した名前のファイルとして保存できる．今回はファイル名を最適化 1-5.xlsx とした．読み込み元ファイルとは異なるファイル名で保存する方が，元のファイルを破損する心配がない.

◎ 課題 5

上記の編集を行い，ナップサックナップサック問題を解いて，Excel に結果を出力するプログラムを作成しなさい．実行後に最適化 1-5.xlsx を開いて，正しく結果が出力されているか確認しよう. Jupyter Notebook の画面で最適化 1-5.xlsx をクリックしても Excel は起動しません. Jupyter Notebook ファイルが存在するフォルダを表示させて，直接開きましょう　（最適値 64）.

出力されなかったり，最適解が違ったりする場合は，問題 model を出力したり，最適解や最適値を print したりして，原因を追究しよう．

【重要】書き込み先のファイル**最適化 1-5.xlsx** を開いたまま上のプログラムを実行すると PermissionError になる．実行前に，結果の出力先ファイルである**最適化 1-5xlsx** は閉じておくこと．

◎ 課題 6

雑誌を置く棚には重量制限があり，購入する雑誌の合計重量が 80 を超えてはならないことが分かった．予算 100 以内，重量 80 以内で，希望度を最大にするような雑誌の選び方を求めるような最適化問題を定式化しなさい．その後，プログラムを作成し，最適解を求めなさい．

Excel ファイル**最適化 1-5.xlsx** の 9～11 行目にあるデータを読み込み，求めた結果を 12 行目に出力して，ファイル**最適化 1-6.xlsx** として保存すること．

（最適値 61）

◎ 課題 7

所属教員から，希望度合計はせめて 70 は欲しい，という要望があった．また棚の重量制限も簡単な補修で 100 まで上げられることが分かった．財務部に予算増加の交渉をするために，重量制限を満たし，希望度を 70 以上にするために必要な最小の予算額が知りたい．この予算額を求める最適化問題を定式化し，そのプログラムを作成しなさい．

Excel ファイル**最適化 1-6.xlsx** の 17～19 行目にあるデータを読み込み，求めた結果を 20 行目に出力して，ファイル**最適化 1-7.xlsx** として保存すること．

（最適値 111）

【アドバイス】正しい最適解が得られない場合の対応方法

最適化問題とは，

- 制約条件をすべて満たす解（以後「実行可能領解」と記します）のうち，
- 目的関数値が最適（最大 or 最小）

となる解（最適解）を求める問題です．

プログラムによって得られた最適解が正答にならないケースは，以下の 2 つに大別されます．

(1)本当の最適解より悪い解が「最適」として出力されるケース．

その極端なケースでは，最適解が見つからない．

(2)本当の最適解より良い解が「最適」として見つかるケース

まず(1)(2)のどちらにもなり得るのが，目的関数が誤っているときです．目的関数が正しく設定できているかは，print(model)を実行して，最適化モデルを確認するしかありません．

目的関数が正しいのに(1)が起こるケースは，本当の最適解が見つからなかった，ということです．制約条件が実際よりも厳しすぎて，本当の最適解がすべての制約条件を満たすことができなくなったために起こります．

解決するには，（誤って）必要以上に条件を厳しくしている制約条件を見つけることです．問題のサイズが小さいなら，print(model)を使って最適化モデルを表示して，定式化通りにプログラムで最適化モデルが作られているか確認しましょう．

制約条件が多すぎて print で確認するのが大変なときには，プログラム内で制約条件の一部をコメントにして（行の先頭に#を付ける）制約条件を一時的になくしましょう．その結果，最適解が見つかるようなら，コメントにした行の制約条件に問題があるということです．

目的関数が正しいのに(2)が起こるケースは，本当の最適解よりも好ましい解が見つかった，ということです．制約条件が実際よりも緩く，本当なら条件を満たしていない解も実行可能解に含まれてしまっているために起こります．

解決するには，実際より条件が緩い制約条件を見つけることです．(1)と同じ方法でも見つけることは可能ですが，(2)ではより簡単に見つけることができます．

それは，本来の満たす必要のある制約条件のうち，プログラムで得られた（誤った）最適解が満たしていない条件を見つけることです．Excel ファイルに出力された（誤った）最適解を確認すれば比較的容易に見つけられるでしょう．

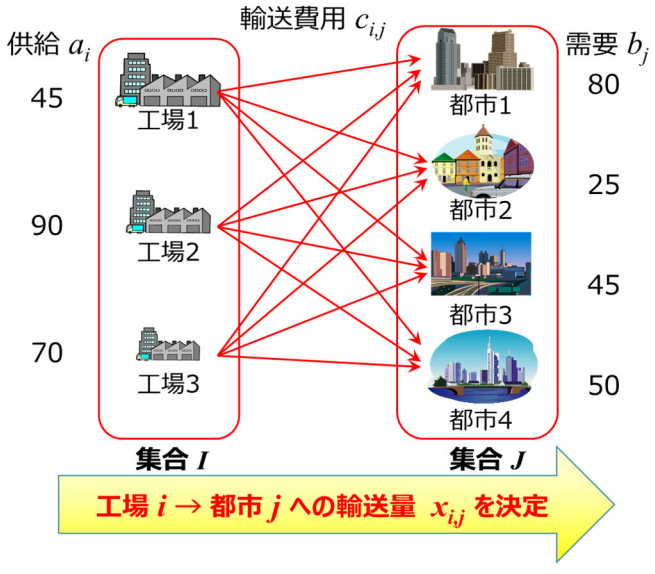
4. 輸送問題

4.1 対象の問題（Excel ファイルシート「輸送」参照）

ある会社は 3 つの工場である商品を生産し，それぞれの工場から 4 つの都市にある倉庫に商品を輸送する．このときの輸送費をなるべく安くしたい．(具体的な数値は Excel ファイル参照)

輸送費は[運ぶ商品の個数]×[輸送単価]で算出され，輸送単価は工場と都市のペアごとに設定されている．また，都市によって商品を必要とする量（需要）が異なり，需要を満たすだけ商品を運びたい．さらに各工場が現在所有する商品の個数（供給）も異なり，供給量までしか出荷できない．

これらの条件を考慮して，総輸送費が最も安くなる運び方，すなわち各工場から各都市への輸送量を決定しよう．



4.2 定数

- a_i　　工場 i の供給量（輸送上限）．$i = 1, 2, 3$.
- b_j　　都市 j での対象商品への需要（必要量）．$j = 1, 2, 3, 4$.
- $c_{i,j}$　　工場 i → 都市 j へ商品を 1 つ運ぶときの輸送単価（コスト）．$i = 1, 2, 3$. $j = 1, 2, 3, 4$.
  これまでと異なり，2 次元（添字が 2 つ）

上の記号はこの後のプログラムでも用いるので，一通り確認すること．

4.3 最適化問題の構成要素

① 決定変数

$x_{i,j}$　　工場 i → 都市 j へ輸送する量．実数型（整数にもできるが，とりあえず）
決定変数の添字が i と j の 2 次元になっている点に注意．

② 制約条件

- 都市には需要だけ商品を届ける．
- 工場からの出荷量は，供給量を超えない

③ 目的関数

総輸送費用の最小化．ある工場からある都市への輸送費用は，輸送単価×輸送量

4.4 定式化

最？化　　$\sum_{?=?}^{?} \sum_{?=?}^{?} ?$

制約条件　　$\sum_{?=?}^{?} ??? \quad (i = 1, 2, 3)$　　工場 i に関する条件

$\sum_{?=?}^{?} ??? \quad (j = 1, 2, 3, 4)$　　都市 j に関する条件

$x_{i,j} \geq 0 \quad (i = 1, 2, 3.\ j = 1, 2, 3, 4)$　　各工場－都市間の輸送量

【重要補足】定式化における $(i = 1, 2, 3)$の意味

例えば，　$x_{i,1} + x_{i,2} = 0 \quad (i = 1, 2, 3)$　という制約条件は，

$$x_{1,1} + x_{1,2} = 0$$
$$x_{2,1} + x_{2,2} = 0$$
$$x_{3,1} + x_{3,2} = 0$$

という 3 つの制約条件を表している．同じ形をした複数の条件を一行で表せて便利．

◎ 課題 8

4.2・4.3 で定めた定数・決定変数を用いて，輸送問題を定式化しなさい．

4.5 Python プログラム（途中から）：定数データの作成

課題 9 用のセル内のコメントで指示されたように，ナップサック問題（課題 5）のプログラムを部分的にコピペしよう．

```
I = [i+1 for i in range(3)]
J = [i+1 for i in range(4)]
a = {}
b = {}
c = {}
for i in I:
    a[i] = sheet.cell(row=11+i, column=8).value
    for j in J:
        c[i, j] = sheet.cell(row=3+i, column=2+j).value
for j in J:
    b[j] = sheet.cell(row=16, column=2+j).value
```

- 定数 a，b，c を Excel シートから読み込む．a はセル範囲 H12:H14，b はセル範囲 C16:F16，c はセル範囲 C4:F6 から読み込む．セル範囲と上のプログラムでの指定が対応していることを確認しよう．
- 定数 c は 2 次元なので，i と j による二重の for 文になる．定数 a と定数 b は 1 次元．

4.6 Python プログラム（つづき）：問題の作成，決定変数の作成

```
model = LpProblem('Transport')
x = {}
for i in I:
    for j in J:
        x[i, j] = LpVariable(f'x{i},{j}', lowBound=0)
```

- 輸送問題は，最小化問題なので LpProblem 内での sense 指定不要．
- 決定変数である辞書 x のキーは，今回は i と j の 2 つ（2 次元）．
- 今回の決定変数は実数型なので，cat は指定不要．$x_{ij} \geq 0$ をここで設定．

4.7 Python プログラム（つづき）：制約条件の追加・目的関数の設定

```
for i in I:
    model += lpSum(??? for ??? in ???) ??? ???
for j in J:
    model += lpSum(??? for ??? in ???) ??? ???
model += lpSum(??? for i in I for j in J)
```

制約条件・目的関数の部分を上のように変更する．**???**の部分は自分で考える．定式化の内容に合うように，プログラムを完成させよう．

- 制約条件の個数は工場数や都市数によって変化するので，for 文を使って工場ごと，都市ごとの制約条件を作成する．
- 目的関数は，定式化では Σ が 2 つ用いるが，プログラムでは lpSum 関数 1 つで書けるのが便利．

4.8 Python プログラム（つづき）：結果の表示

```
for i in I:
    for j in J:
        sheet.cell(row=11+i, column=2+j).value = value(x[i,j])
book.save('最適化 1-9.xlsx')
```

- 決定変数の次元が 2（i と j）なので，for 文も二重になる．

◎ 課題 9

輸送問題のプログラムを完成させなさい．（最適化 1-9.xlsx を開いて結果を確認しよう．）

（最適値 1200）

◎ 課題 10（シート「課題 10」参照）

各工場から各都市への輸送量には制限があることが分かった．工場 i から都市 j へ輸送することができる上限を$d_{ij}(i = 1, 2, 3. j = 1, 2, 3, 4)$とする．シート「課題 10」に入力された$d_{ij}$を用いて，この条件も考慮した輸送問題を解く最適化問題を定式化し，最適な輸送方法を求めるプログラムを作成しなさい．

Excel ファイル最適化 1-9.xlsx のシート「課題 10」から必要なデータを読み込み，求めた結果を同シート内に出力して，ファイル最適化 1-10.xlsx として保存すること．

（最適値 1245）

◎ 課題 11（挑戦問題：シート「課題 11」参照）

輸送は同じ積載量（セル H19）のトラックで運び，輸送料が[トラック 1 台当たりの費用]×[使用するトラックの台数]で算出されることに変更された．トラックには積載量以上の荷物を積むことはできない．同じ工場・都市間に複数のトラックを使用することはできる．

このとき，総輸送費を最小にするような輸送方法，すなわち工場 i から都市 j への輸送量 x_{ij}（実数）と，工場 i から都市 j へ輸送するときに使用するトラックの台数 y_{ij}（非負の整数）を求める最適化問題を定式化し，その答えを出力するプログラムを作成しなさい．（課題 10 で考えた輸送量の上限はここでは考慮しない．）

Excel ファイル最適化 1-10.xlsx のシート「課題 11」から必要なデータを読み込み，求めた結果を同シート内に出力して，ファイル最適化 1-11.xlsx として保存すること．

（最適値 124）

◎ 課題 12（挑戦問題：課題 5 の拡張）

課題 5 の状況において，各出版社によるボリュームディスカウントを考慮する．各出版社による雑誌を 5 冊以上購入すれば，その出版社の雑誌価格が 20%割引になる．

雑誌 i を割引価格で購入するとき 1，割引価格では購入しないとき（通常価格で購入するか，購入しないか）のとき 0 となる決定変数 y_i を導入して，ボリュームディスカウントを含めた課題 5 の問題（予算制約下での合計希望度の最大化）を行うプログラムを作成しなさい．

Excel ファイル**最適化 1-11.xlsx** のシート「課題 12」から必要なデータを読み込み，求めた結果を 5・6 行目に出力して，ファイル**最適化 1-12.xlsx** として保存すること．

（最適値 70）

【追加解説】条件付き合計

「雑誌 i と同じ出版社の雑誌の購入数」は条件付きの Σ で表現できる．publisher[i]を雑誌 i の出版社とすると，lpSum 関数では次のように書ける．

```
lpSum(x[k] for k in I if publisher[k] == publisher[i])
```

print(model)を実行して，正しく定式化できているか確認してみよう．

以上

【ヒント】（課題が解けず，ヒントが欲しい人向け）

◎ 課題 1

Excel のシートを見て，??? に当てはまる数値を読み取りましょう．

数式は，1.4 を見て綺麗に記しましょう．

◎ 課題 2

1.7 節でも記したように，1～5 の順に 3 回実行すればよいです．ただ，今回は 3 つの問題で異なる部分は目的関数のみです．したがって，2 回目・3 回目の最適化を行うときには，1・2 の処理は不要，3 も目的関数の設定のみを行えばよいです．その後，4・5 の処理は 3 回とも実行します．

◎ 課題 3

どの式にも必ず決定変数 x_i が入ります．

制約条件は費用に関する式，目的関数は希望度に関する式になります．

◎ 課題 4

生産計画のプログラムの「問題の作成」から「最適化の実行」までコピペして，資料に従って修正すれば完成です．

◎ 課題 5

課題 4 と同様，資料に従って修正すれば完成です．

Excel ファイルを開く行でエラーが出るときには，ipynb ファイルと同じフォルダに読み込み Excel ファイルが保存されているか確認しましょう．ファイル名の全角半角の違いにも注意．Excel の拡張子（.xlsx）が異なるケースもまれにあります．

◎ 課題 6

うまくできないときには，どこまでは正しく動作しているかを認識することが重要です．処理の順序に合わせて，以下のように確認すると良いでしょう．

- Excel からデータを正しく読み込めているか？読み込んだあとに print を使って，変数に正しい値が格納されたか確認しよう．
- 最適化モデルが正しく作成されているか？モデルを作成したあとに print を使って，定式化と同じようにモデルが作れているか確認しよう．
- 最適解が間違っている，または見つからない場合については，課題 7 の後のアドバイスを参照して下さい．
- Jupyter Notebook で表示される最適値は合っているものの，Excel に正しく表示されない場合は，Excel に出力する部分に間違いがあります．正しいセルの位置に出力されているか確認しましょう．

◎ 課題 7

目的関数は予算の最小化．制約条件は 2 種類で，希望度合計 70 以上，重量 100 以下です．

◎ 課題 8

ある工場 i からある都市 j への輸送費は，輸送単価 $c_{i,j}$ と輸送量 $x_{i,j}$ の積で求まるので，総輸送費は，すべての工場と都市の組合せについて，輸送費の合計を求めればよい．

ある工場 i に関する条件は，「工場 i から出る輸送量の合計は，工場 i の供給量以下」です．

ある都市 j に関する条件は，「都市 j に入る輸送量の合計は，都市 j の需要と同じ（または以上）」です．

◎ 課題 9

資料に従って修正すれば完成です．エラーが出たら，その行を資料と比較しましょう．添え字が i と i の 2 種類あるので，混同しないように気を付けましょう．

◎ 課題 10

課題 9 に制約条件が 1 種類追加されるだけです．通常通り，制約条件を追加する方法のほかに，決定変数を作成するときに変数のとり得る範囲として制約を付ける方法もあります．

◎ 課題 11

- 目的関数に入る決定変数は y_{ij}になる．
- 目的関数に x_{ij} が入っていない点に注意．課題 9 の制約条件だけでは y_{ij} に何の制約もないので，目的関数を最小化するように，$y_{ij} = -\infty$ が最適値になってしまう．$y_{ij} \geq 0$ の整数，という条件を付けても，$y_{ij} = 0$ が最適解になる．
- x_{ij}だけ輸送するには，それを運ぶのに必要なトラックの台数が必要となる．このことを表す 2 つの決定変数 x_{ij} と y_{ij} による制約条件が必要．具体的には以下の通り．
  - トラック 0 台使用するなら，輸送量は必ず 0
  - トラック 1 台使用するなら，輸送量は最大で W
  - トラック 2 台使用するなら，輸送量は最大で $2W$
  - トラック 3 台使用するなら，輸送量は最大で $3W$
  - 同様に考えると，工場 i から都市 j への輸送量x_{ij}の上限が，工場 i から都市 j へ輸送するトラックの台数y_{ij}（と W）で表現できる．

◎ 課題 12

- 目的関数は変更なし
- ボリュームディスカウントによる割引金額は，$\sum_i 0.2c_i y_i$．この分，購入金額が安くなる
- 2 種類の決定変数 x_i と y_i の関係を表す制約条件が必要．例えば，雑誌 1 のボリュームディスカウントに関して，以下の 2 つの制約条件が必要．
  (1) 雑誌 1 を割引価格で購入するとき，雑誌 1 は購入している
  (2) 雑誌 1 を割引価格で購入するとき，雑誌 1 と同じ出版社が発行する雑誌を 5 冊以上購入している

この 2 種類の制約条件を各雑誌に対して作成する．

- 「購入するとき」は if 文を使えない．例えば決定変数を用いて条件(1)を表すと「$y_1 = 1$ のとき，$x_1 = 1$」となる．y_1 の値は最適化問題を解かないと決定しないため，まだ値が分からない Python プログラム内で「if value(y[i]) == 1」とは記せない．しかし，次のように考えると，単なる不等式で条件を表すことができる．
- y_1 も x_1 も 0 か 1 しかとらないことに注目する．「$y_1 = 1$ のとき，$x_1 = 1$」は
  - $y_1 = 1$ のとき，x_1 は 0 と 1 のうち，1 しかとれない．
  - $y_1 = 0$ のとき，x_1 は 0 と 1 のうち，どちらをとっても構わない．

  を意味する．これを別の表現をすると，条件(1)は次のように解釈できる．
  - $y_1 = 1$ のとき，x_1 は最小で 1
  - $y_1 = 0$ のとき，x_1 は最小で 0
- 同様に条件(2)は，
  - $y_1 = 1$ のとき，雑誌 1 と同じ出版社の雑誌の購入数は最小で 5
  - $y_1 = 0$ のとき，雑誌 1 と同じ出版社の雑誌の購入数は最小で 0

  と表せて，課題 11 の積載量と同じ考えが利用できる．
- 定式化の方法は 1 通りではないので，上記以外の方法でももちろん構わない．

【大ヒント】（【ヒント】を見ても，どうしても課題が解けない人向け！）

◎ 課題 1

自信がなければ，とりあえず続きを読んでプログラムを作成してから，定式化を行っても構いません．

◎ 課題 2

流れは次のようになります．

- 問題の作成，決定変数の作成，制約条件の追加
- 目的関数の設定，最適化の実行，最適化の結果表示（その 1）
- 目的関数の設定，最適化の実行，最適化の結果表示（その 2）
- 目的関数の設定，最適化の実行，最適化の結果表示（その 3）

◎ 課題 3

- x_i は，雑誌 i を購入するときに 1，購入しないときに 0 を取ります．
- $c_i x_i$ は，雑誌 i を購入するときにc_i，購入しないときに 0 になります．
- $c_1 x_1 + c_2 x_2$ は，雑誌 1 と雑誌 2 を購入するときには$c_1 + c_2$，雑誌 1 のみ購入するときにはc_1，雑誌 2 のみ購入するときにはc_2，どちらも購入しないときに 0 になります．つまり，雑誌 1 と雑誌 2 のうち，購入した雑誌の合計価格になります．
- 雑誌の数を 2 から 18 にしても考え方は同じ．
- 購入した雑誌の希望度の合計も考え方は同じ

◎ 課題 4

入力ミスがないか確認しましょう．エラーがある場合，どの行でエラーになったか表示されます．もう一度資料と見比べてみましょう！大文字・小文字の違いにも気を付けて下さい．

◎ 課題 5

Excel ファイルに結果が出力されていない場合には，Excel ファイルの更新日時を確認しましょう．プログラムを実行した日時になっていない場合には，どこか異なる場所に保存されている可能性があります．現在実行している ipynb ファイルの更新日時も確認して，確認した ipynb ファイルが本当に現在プログラムを作成している ipynb ファイルなのか確認しましょう．

◎ 課題 6

定式化を正しく記そう．課題 5 と比較して，重量に関する制約条件が一つ追加されるだけです．不等式の向きに注意．

◎ 課題 7

目的関数は最小化になっていますか？制約条件の不等号の向きに注意．

◎ 課題 8

ある工場 i に関する条件には，記号 i が残ります．したがって，Σの添字は j になります．
逆に，ある都市 j に関する条件では，Σの添字は i になります．
Σが苦手な人は，Σを展開した式を書いてみると良いでしょう．

◎ 課題 9

工場に関する制約は，工場 i = 1, 2, 3 それぞれに 1 つずつ必要になるので，for i in I: で 3 回繰り返します．工場 i から出る輸送量は，都市 j = 1, 2, 3, 4 それぞれへの輸送量の合計です．その合計値は工場 i の供給量 a[i] を超えてはなりません．

都市に関する制約は，都市 j = 1, 2, 3, 4 それぞれに 1 つずつ必要になるので，for j in J: で 4 回繰り返します．都市 j へ入る輸送量は，工場 i = 1, 2, 3 それぞれからの輸送量の合計です．その合計値は都市 j の需要 b[j] に等しい（または上回る）必要があります．

◎ 課題 10

追加される条件は，工場と都市間のすべての組合せに対して 1 つずつです．3 つの工場と 4 つの都市との組合せなので，全部で 12 通りあります．プログラムでは，決定変数 x[i, j]を作成するときに，上限 upBound として制約を付けてもよいですし，2 重ループを使って x[i, j]に対する制約条件を追加しても，どちらでも構いません．