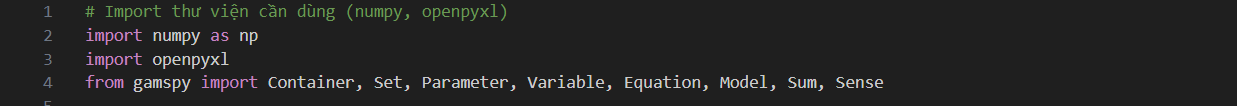
1. **Import thư viện:**

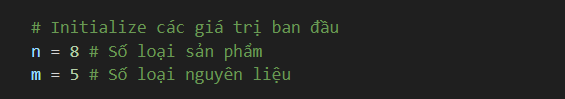


+ numpy được sử dụng để làm việc với mảng và ma trận.

+ openpyxl được sử dụng để đọc dữ liệu từ một tệp Excel.

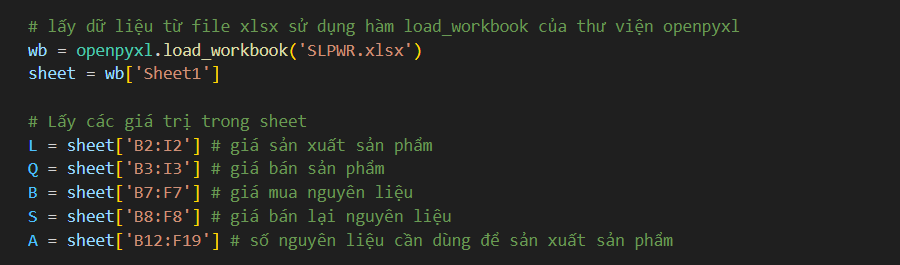
+ gamspy là thư viện được sử dụng để giải bài toán quy hoạch tuyến tính.

1. **Khởi tạo giá trị ban đầu:**



Tạo các biến mô tả số loại sản phẩm (n) và số loại nguyên liệu (m).

1. **Đọc dữ liệu từ file Excel:**



+ wb = openpyxl.load\_workbook('SLPWR.xlsx'): Dòng này tải tệp Excel có tên 'SLPWR.xlsx'.

+ sheet = wb['Sheet1']: Chọn sheet cụ thể có tên là 'Sheet1' từ workbook.

L, Q, B, S, A là các biến đại diện cho các phạm vi dữ liệu khác nhau trong tệp Excel:

L: Đại diện cho giá sản xuất của mỗi sản phẩm. Giá trị nằm trong ô B2:I2.

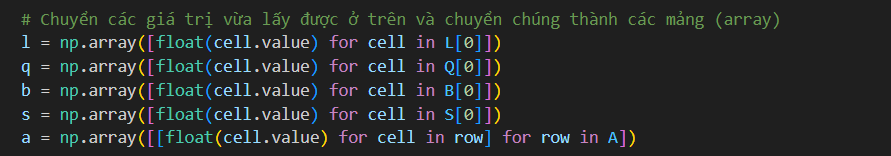
Q: Đại diện cho giá bán của mỗi sản phẩm. Giá trị nằm trong ô B3:I3.

B: Đại diện cho giá mua nguyên liệu. Giá trị nằm trong o B7:F7.

S: Đại diện cho giá bán lại của nguyên liệu. Giá trị nằm trong ô B8:F8.

A: Đại diện cho lượng nguyên liệu cần thiết cho việc sản xuất mỗi sản phẩm. Giá trị nằm trong ô B12:F19.

1. **Chuyển dữ liệu thành mảng**



- Chuyển đổi giá trị từ dòng L[0] thành mảng NumPy l:

+ L[0] là một dòng trong phạm vi L, đại diện cho giá sản xuất của các sản phẩm.

+ [float(cell.value) for cell in L[0]]: Duyệt qua từng ô trong dòng và chuyển giá trị của mỗi ô thành số dấu chấm động.

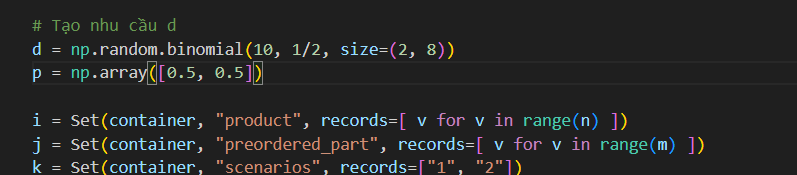
+ np.array(...): Tạo một mảng NumPy từ danh sách các giá trị đã chuyển đổi.

* Làm tương tự đối với các dòng q,b,s.

- Chuyển đổi giá trị từ các dòng A thành mảng NumPy a:

+ [[float(cell.value) for cell in row] for row in A]: Duyệt qua từng dòng trong phạm vi A, chuyển giá trị từng ô trong mỗi dòng thành số dấu chấm động, và sau đó tạo một mảng hai chiều (ma trận) từ các giá trị này.

1. **Tạo nhu cầu ngẫu nhiên và khởi tạo các tập hợp**



* Tạo nhu cầu ngẫu nhiên d:

np.random.binomial(10, 1/2, size=(2, 8)): Tạo một mảng 2x8 với giá trị ngẫu nhiên được sinh theo phân phối nhị thức. Trong trường hợp này, nhu cầu có phân phối nhị thức với n = 10 (số thử nghiệm) và p = 1/2 (xác suất thành công).

* Tạo vector xác suất p:

np.array([0.5, 0.5]): Tạo một mảng NumPy đại diện cho vector xác suất.

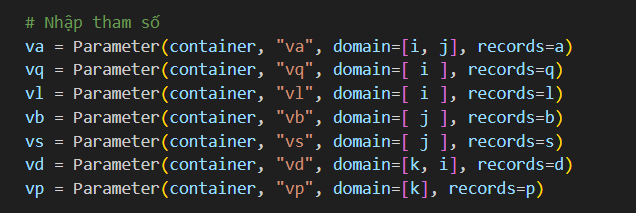
* Khởi tạo tập hợp i, j, và k:

i = Set(container, "product", records=[v for v in range(n)]): Tạo một tập hợp i đại diện cho các sản phẩm. Số sản phẩm được xác định bởi biến n.

j = Set(container, "preordered\_part", records=[v for v in range(m)]): Tạo một tập hợp j đại diện cho các loại nguyên liệu. Số loại nguyên liệu được xác định bởi biến m.

k = Set(container, "scenarios", records=["1", "2"]): Tạo một tập hợp k đại diện cho các scenarios

1. **Nhập tham số và tạo các biến tối ưu hóa**



Nhập vào các tham số

va: số lượng nguyên liệu j cần để sản xuất cho sản phẩm thứ i

vq: giá bán từng sp

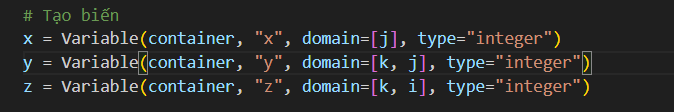
vl: giá sản xuất từng sp

vb: giá mua từng loại nguyên liệu

vs: giá bán lại từng loại nguyên liệu

vd: nhu cầu

vp: xác suất

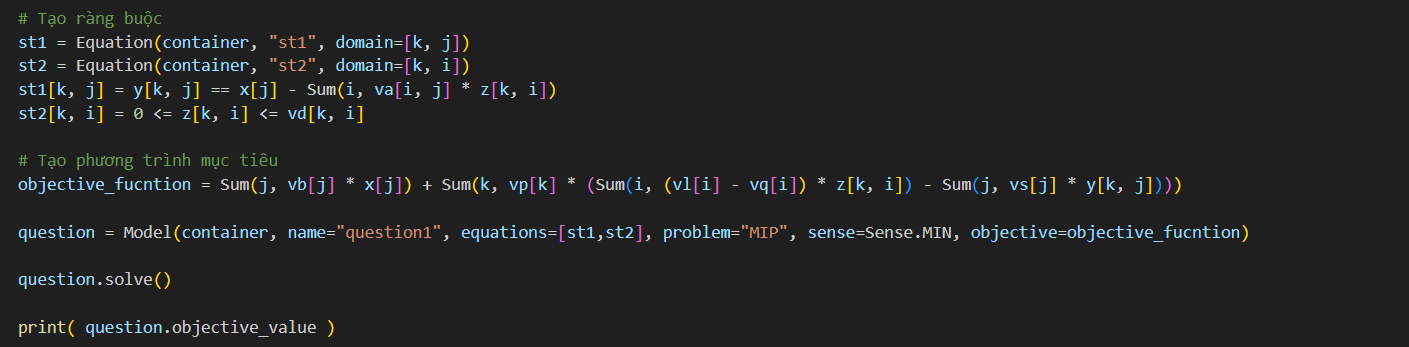


biến x là số lượng của từng loại nguyên liệu.

Biến y đại diện cho một số lượng cần được đặt hàng (preordered) của từng loại nguyên liệu

biến z đại diện cho số lượng nguyên liệu được sử dụng để sản xuất từng sản phẩm

1. **Tạo ràng buộc và phương trình mục tiêu**



* **Tạo ràng buộc st1 và st2:**

st1 = Equation(container, "st1", domain=[k, j]): Tạo một ràng buộc có tên là "st1" với miền giá trị là tập hợp k và j. Ràng buộc này sau đó sẽ được sử dụng để giới hạn giá trị của biến y.

st2 = Equation(container, "st2", domain=[k, i]): Tạo một ràng buộc có tên là "st2" với miền giá trị là tập hợp k và i. Ràng buộc này sau đó sẽ được sử dụng để giới hạn giá trị của biến z.

* **Đặt giá trị cho ràng buộc st1 và st2:**

st1[k, j] = y[k, j] == x[j] - Sum(i, va[i, j] \* z[k, i]): Đặt giá trị cho ràng buộc st1. Điều này giới hạn giá trị của biến y dựa trên giá trị của biến x và z, cũng như các hệ số từ tham số va.

st2[k, i] = 0 <= z[k, i] <= vd[k, i]: Đặt giá trị cho ràng buộc st2 đảm bảo rằng giá trị của biến z nằm trong khoảng được quy định bởi tham số vd.

**Kết quả in ra màn hình :**

****

****

****