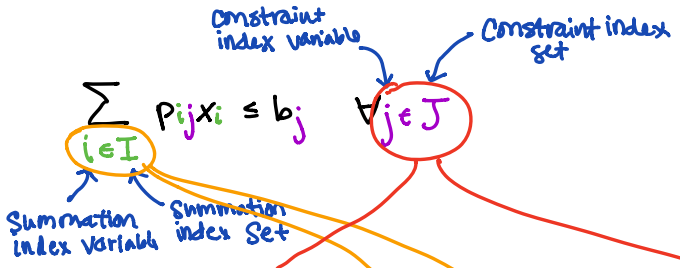
Translating Abstract Constraints to Python

Single index constraint




```
def const_rule(model, j):
    sum(p[i,j]*model.x[i] for i in I) <= b[j]
model.cnst = pyo.Constraint(J, rule=const_rule)
```

Double index constraint

$$\sum_{i \in I} p_{ij} x_{ik} \leq b_{jk} \quad \forall j \in J, k \in K$$

```
def const_rule(model, j, k):
    return sum(p[i,j]*model.x[i,k] for i in I) <= b[j,k]
model.cnst = pyo.Constraint(J, K, rule=const_rule)
```

No constraint indexing

$$\sum_{i \in I} p_i x_i \leq b$$

```
def const_rule(model):
    return sum(p[i]*model.x[i] for i in I) <= b
model.cnst = pyo.Constraint(rule=const_rule)
```

no constraint indexing

No sum

$$x_j \leq b_j \quad \forall j \in J$$

error

```
def const_rule(model, j):
    for j in J:
        return model.x[j] <= b[j]
```

Error

```
for j in J
```

```
model.cnst = pyo.Constraint( J , rule = const_rule)
```

Double sum

$$\sum_{i \in I} \sum_{j \in J} x_{ij} \leq b$$

```
def const_rule(model):
    return sum( model.x[i, j] for i in I for j in J ) <= b
model.cnst = pyo.Constraint( rule = const_rule)
```

Same index set for constraint & summation

$$\sum_{i \in I} x_{ik} \leq \sum_{i \in I} y_{ik}, \text{ for } k \in I$$

Placing parentheses around the return statement allows me to break the line in Python.

```
def const_rule(model, k ):
    return (sum( model.x[i,k] for i in I )
            <= sum( model.y[i,k] for i in I ))
model.cnst = pyo.Constraint( I , rule = const_rule)
```

incorporating an if statement

N = set of nodes

A = set of arcs

$$\sum_{(i,j) \in A : i = n} x_{ij} \leq b_n \quad , \quad \text{for } n \in N$$

This places a restriction on the arcs that appear in this sum, based on the node n

```
def const_rule(model, n):
    return sum(model.x[i,j] for i,j in A if i == n) <= b[n]
model.cnst = pyo.Constraint(N, rule=const_rule)
```