In [37]:

```python
import sys
import numpy as np

from gamspy import (
    Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
    Domain,Number,Sum,Product,Smax,Smin,Ord,Card,SpecialValues
)

options = Options(variable_listing_limit=0)
m = Container(options=options)
```

In [38]:

```python
locations = Set(m, "locations", records = ["Burrow", "LeakyCauldron", "Godri
"LittleWhinging", "Cokeworth", "Egypt"])

i = m.addAlias('i', locations)
j = m.addAlias('j', locations)
k = m.addAlias('k', locations)

arcs = Set(m, 'arcs', domain=[i, j], records=[
    ("Burrow", "LeakyCauldron"), ("LeakyCauldron", "Burrow"), ("Burrow", "Go
    ("LeakyCauldron", "GodricHollow"), ("GodricHollow", "LeakyCauldron"), ("
    ("GodricHollow", "LittleHangleton"), ("LittleHangleton", "GodricHollow")
    ("LittleHangleton", "Cokeworth"), ("Cokeworth", "LittleHangleton"), ("Li
    ("Cokeworth", "Egypt"), ("Egypt", "Cokeworth")
])


distances = Parameter(m, 'distances', domain=[i,j], records=[["Burrow", "Lea
            ["Burrow", "GodricHollow", 60],
            ["Burrow", "LittleHangleton", 50],
            ["LeakyCauldron", "GodricHollow", 10],
            ["LeakyCauldron", "LittleWhinging", 70],
            ["GodricHollow", "LittleHangleton", 20],
            ["GodricHollow", "LittleWhinging", 55],
            ["GodricHollow", "Cokeworth", 40],
            ["LittleHangleton", "Cokeworth", 50],
            ["LittleWhinging", "Cokeworth", 10],
            ["LittleWhinging", "Egypt", 60],
            ["Cokeworth", "Egypt", 80]
])

distances[i, j].where[distances[j, i] > 0] = distances[j, i]

x = Variable(m, "x", domain=[i,j], type="positive")

supply = Parameter(m, "supply", domain=[i], records=[("Burrow", 1), ("Egypt"

balance = Equation(m, "balance", domain=i)
balance[i].where[~i.last] = Sum(j.where[arcs[i, j]], x[i, j]) - Sum(j.where[



floo = Model(m, 'short', equations=m.getEquations(), problem="MIP",
sense=Sense.MIN,
```

```
objective = Sum(arcs, distances[arcs] * x[arcs])
)
```

In [39]:

```
floo.solve()
```

Out[39]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	160	7	25	MIP	CPLEX	0.00

In [40]:

```
pi = Variable(m, 'pi', type='positive', domain=i)

dualcons = Equation(m, 'dcons', domain=[i,j])
dualcons[i, j].where[arcs[i,j]] = pi[i] - pi[j] <= distances[i, j]

d = Model(m, name="d", equations=[dualcons], problem="LP", sense=Sense.MAX,

d.solve(solver='cplex',solver_options={'lpmethod': 3, 'netfind': 2, 'preind'
```

Out[40]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	160	25	8	LP	CPLEX	0.00

In [43]:

```
# Marginal value from the primal is the shortest distance from egypt
display(balance.records)
display(pi.records)
print("As you can see the marginal values from the primal solution are equal
```

	i	level	marginal	lower	upper	scale
0	Burrow	1.0	160.0	1.0	1.0	1.0
1	LeakyCauldron	0.0	120.0	0.0	0.0	1.0
2	GodricHollow	0.0	110.0	0.0	0.0	1.0
3	LittleHangleton	0.0	110.0	0.0	0.0	1.0
4	LittleWhinging	0.0	60.0	0.0	0.0	1.0
5	Cokeworth	0.0	70.0	0.0	0.0	1.0

	i	level	marginal	lower	upper	scale
0	Burrow	160.0	0.0	0.0	inf	1.0
1	LeakyCauldron	120.0	0.0	0.0	inf	1.0
2	GodricHollow	110.0	0.0	0.0	inf	1.0
3	LittleHangleton	110.0	0.0	0.0	inf	1.0
4	LittleWhinging	60.0	0.0	0.0	inf	1.0
5	Cokeworth	70.0	0.0	0.0	inf	1.0
6	Egypt	0.0	-0.0	0.0	inf	1.0

As you can see the marginal values from the primal solution are equal to the variable values from the dual solution!

In [44]:

```
DistEgypt = Parameter(m,'DistEgypt')
DistEgypt[:] = balance.records.loc[balance.records['i'] == 'LeakyCauldron',

print(f"The distance (length) from the Leaky Cauldron to Egypt is {DistEgypt
```

```
The distance (length) from the Leaky Cauldron to Egypt is 120.0 miles
```