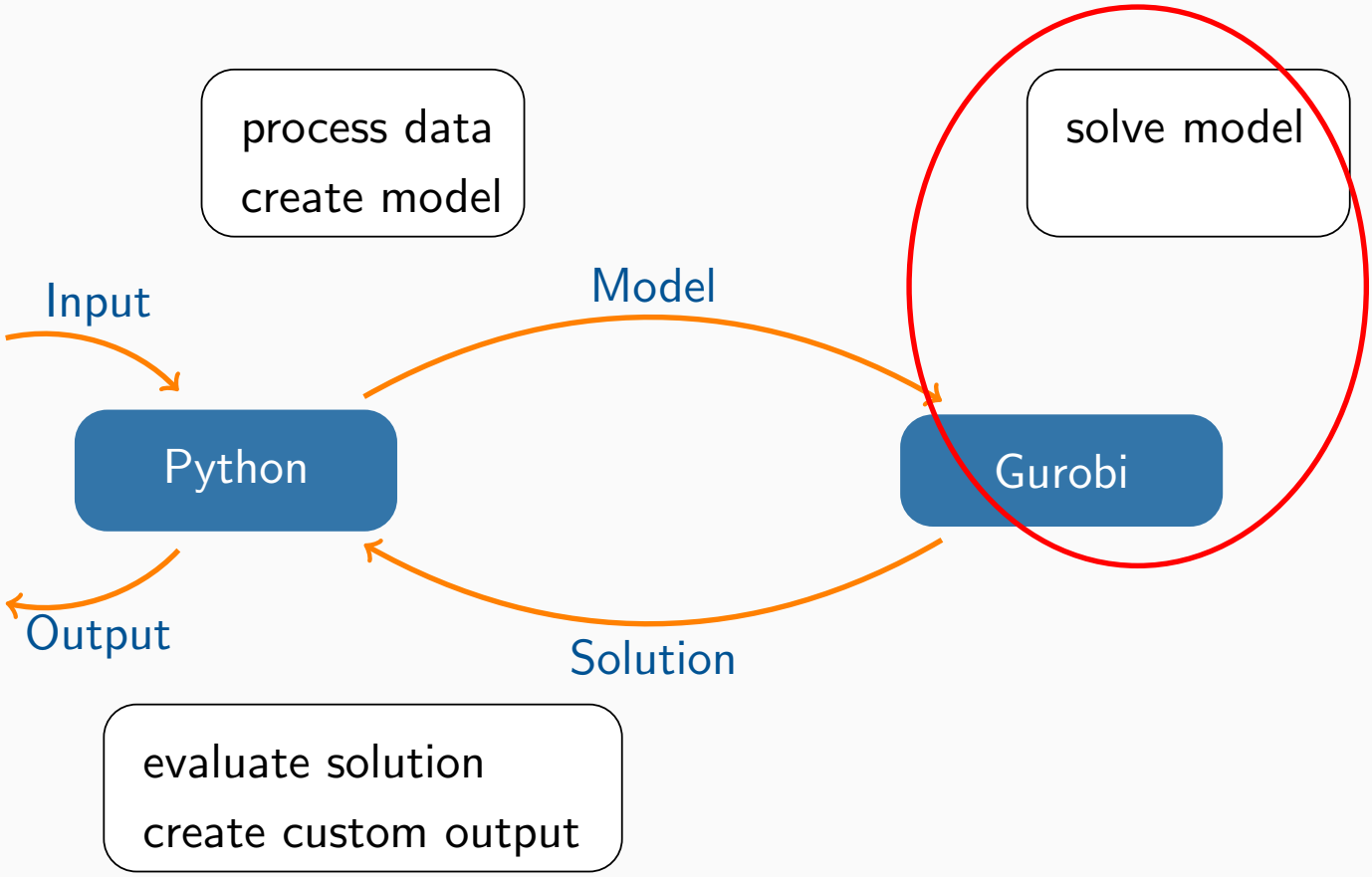
Output Interpretation

Gurobi - LP solver



Presolve

```
Optimize a model with 1500 rows,
   2250000 columns and 4497000 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+01]
Presolve removed 0 rows and 1611 columns
Presolve time: 4.37s
Presolved: 1500 rows, 2248389 columns, 4496778 nonzeros
```

Presolve - Example

$$
\begin{aligned}
x_1 + x_2 + x_3 &\geq 15 \\
x_1 &\leq 7 \\
x_2 &\leq 3 \\
x_3 &\leq 5
\end{aligned}
$$

Presolve - Example

$$x_1 + x_2 + x_3 \geq 15$$
$$x_1 \leq 7$$
$$x_2 \leq 3$$
$$x_3 \leq 5$$

delete constraint and variables in order to reduce the problem

LP methods

- 3 different methods
  - primal/dual simplex
    - robust
    - easy to restart after model modification
  - barrier
    - can be run on multiple cores
- concurrent optimization

Concurrent Optimization

- run simplex *and* barrier at the same time
- first one to finish reports solution
- use multiple cores
- fastest choice for general model

Primal Simplex

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -1.2573140e+02   2.968000e+03   5.593346e+11      7s
    3297    1.2166285e+05   0.000000e+00   6.532640e+07     10s
    4619    8.3232592e+04   0.000000e+00   1.162234e+08     15s
    7000    5.2154173e+04   0.000000e+00   4.234078e+06     25s
    9189    3.7601369e+04   0.000000e+00   1.177763e+07     35s
   10706    3.0986489e+04   0.000000e+00   2.645246e+06     45s
   12019    2.7488411e+04   0.000000e+00   4.758955e+06     56s
   13844    2.3646778e+04   0.000000e+00   3.840044e+05     65s
   15403    2.1713789e+04   0.000000e+00   3.055039e+04     75s
   17347    2.1105977e+04   0.000000e+00   1.777696e+01     85s
   17419    2.1108270e+04   0.000000e+00   0.000000e+00     91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective  2.110826998e+04
```

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -1.2573140e+02   2.968000e+03   5.593346e+11     7s
    3297    1.2166285e+05   0.000000e+00   6.532640e+07    10s
    4619    8.3232592e+04   0.000000e+00   1.162234e+08    15s
    7000    5.2154173e+04   0.000000e+00   4.234078e+06    25s
    9189    3.7601369e+04   0.000000e+00   1.177763e+07    35s
   10706    3.0986489e+04   0.000000e+00   2.645246e+06    45s
   12019    2.7488411e+04   0.000000e+00   4.758955e+06    56s
   13844    2.3646778e+04   0.000000e+00   3.840044e+05    65s
   15403    2.1713789e+04   0.000000e+00   3.055039e+04    75s
   17347    2.1105977e+04   0.000000e+00   1.777696e+01    85s
   17419    2.1108270e+04   0.000000e+00   0.000000e+00    91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective  2.110826998e+04
```

Primal Dual Relations

Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$

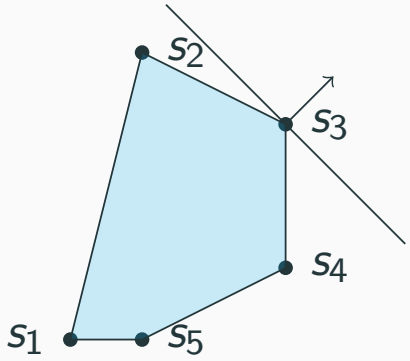
Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$

Primal Dual Relations

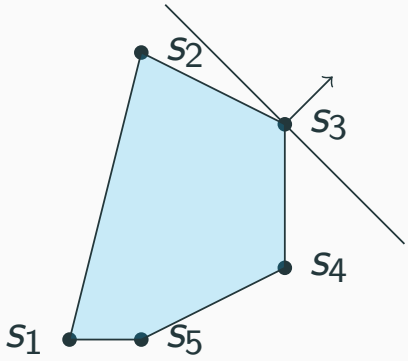
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



Dual:

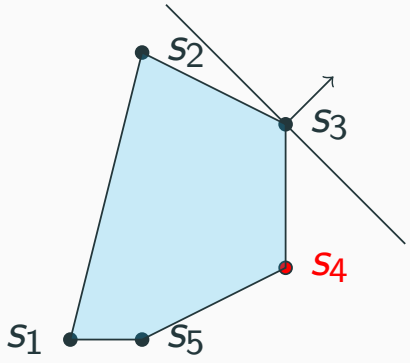
$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$



Primal Dual Relations

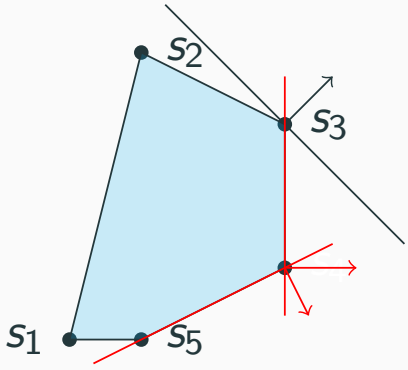
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



Dual:

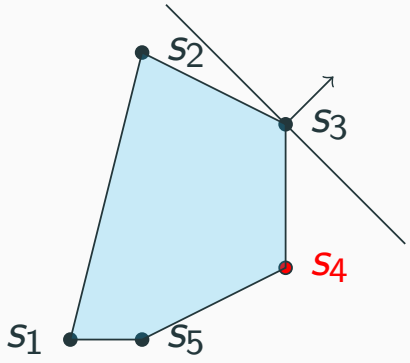
$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$



Primal Dual Relations

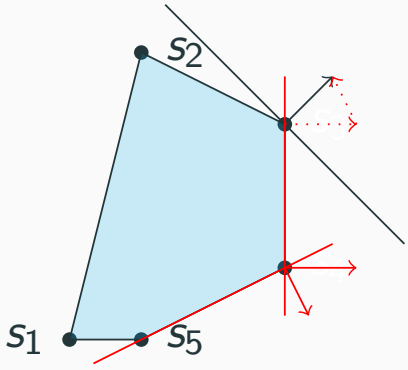
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



Dual:

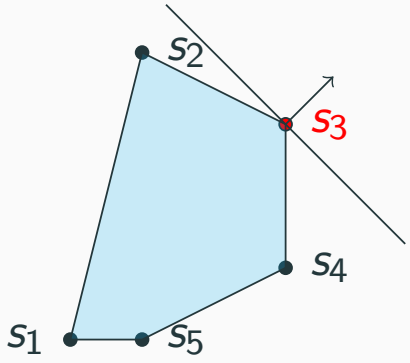
$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$



Primal Dual Relations

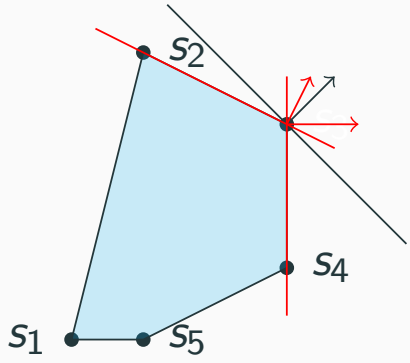
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



Dual:

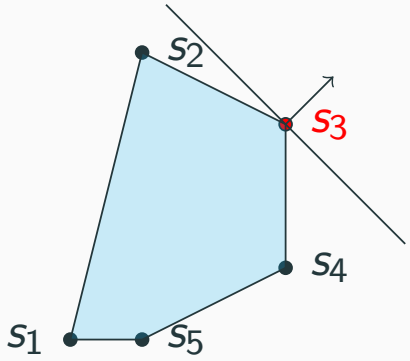
$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$



Primal Dual Relations

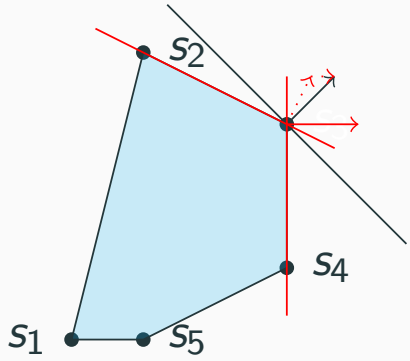
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$



Primal Dual Relations

Primal:

$$\begin{aligned} \min\, & c^\top x \quad \text{s.t.} \\ & Ax \leq b \end{aligned}$$

Dual:

$$\begin{aligned} \max\, & b^\top y \quad \text{s.t.} \\ & A^\top y = c \\ & y \geq 0 \end{aligned}$$

$$c^\top \bar{x} = \bar{y}^\top A \bar{x} =^{(*)} \bar{y}^\top b$$

Dual Simplex

- simplex on dual problem
- obtain primal solution as discussed before
- dominant algorithm used in MIP solving

Barrier Method

Idea:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$

X

Barrier Method

Idea:

$$\min f(x) \quad \text{s.t.}$$
$$c_i(x) \leq 0$$

with f and c_i linear

Barrier Method

Idea:

$$\min f(x) \quad \text{s.t.}$$
$$c_i(x) \leq 0$$

with f and c_i linear

$$\Downarrow$$

$$\min f(x) - \mu \sum_i log(c_i(x))$$

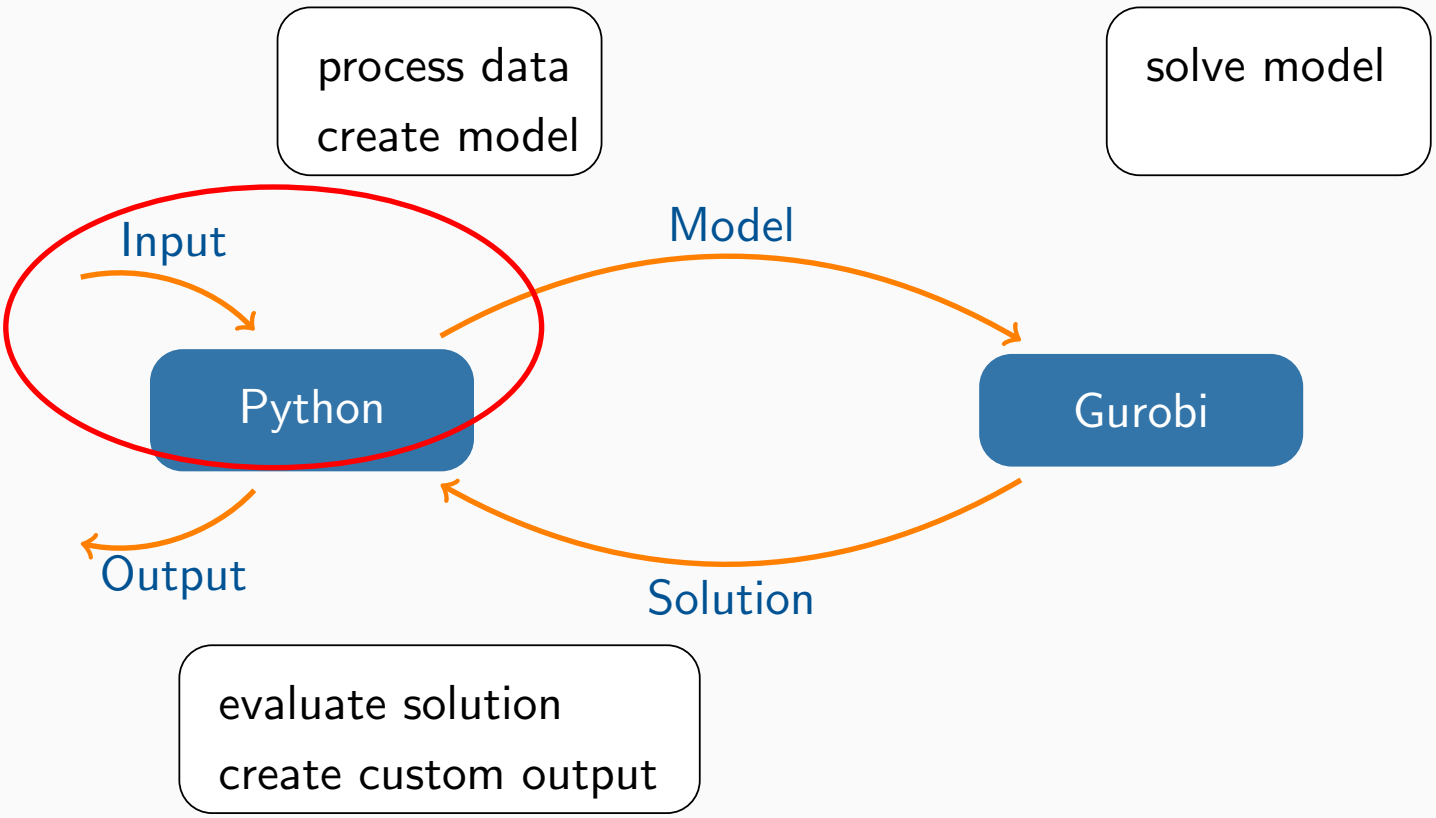
Barrier Method

$$\min f(x) - \mu \sum_i log(c_i(x))$$

- $\mu \to 0$ and solve as nonlinear optimization problem
- also known as Interior Point Method
- few, but expensive calculations
- not clear whether *warm start* is doable

Advanced Input Methods

Advanced Input



Reading Data from Files

Advantages:

- separation between data and code
- faster replacement and sharing of data
- more flexible code
- clearer code

File Formats

- excel (pandas, xlrd)
- csv (csv)
- json (json)
- basically any text-file of some custom format (write parser)

The JSON File Format

```
{
    "oil_types":[
        "heavy","medium","light"
    ],
    "processes":[0,1],
    "production":{
        "heavy":[2,1],"medium":[2,2],"light":[1,4]
    },
    "demand":{
        "heavy":3,"medium":5,"light":4
    },
    "process_cost":[3,5]
}
```

Data Types

- Boolean
- Number (integer or floating point)
- String
- Array [] (ordered list of elements of arbitrary type)
- Object {} (unordered collection of name-value pairs)

Data Types

- Boolean
- Number (integer or floating point)
- String
- Array [] (ordered list of elements of arbitrary type)
- Object {} (unordered collection of name-value pairs)

Translates straight-forward to Python!

Reading JSON files in Python

```
import json

with open("data.json") as json_file:
data = json.load(json_file)
```

Read text files

- generally, input files not given as *json* or *csv* or *xslx*
- any custom format
- solution: read file line by line according to its syntax and create list/dictionary/object

Reading general textfiles in Python

```
filename = 'data.txt'
with open(filename, "r") as file:
    line = file.readline()
    while line:
        print(line.split("separator"))
```

Write text files

- generate different datasets to have stable collection of example data
- straightforward in python (very similar to *print*())

Writing textfiles in Python

```
filename = 'data.txt'
with open(filename, "w") as file:
    file.write("sometext\n")
    file.close()
```