Course on mathematical modelling: AMPL and CPLEX

teacher: Giacomo Lanza

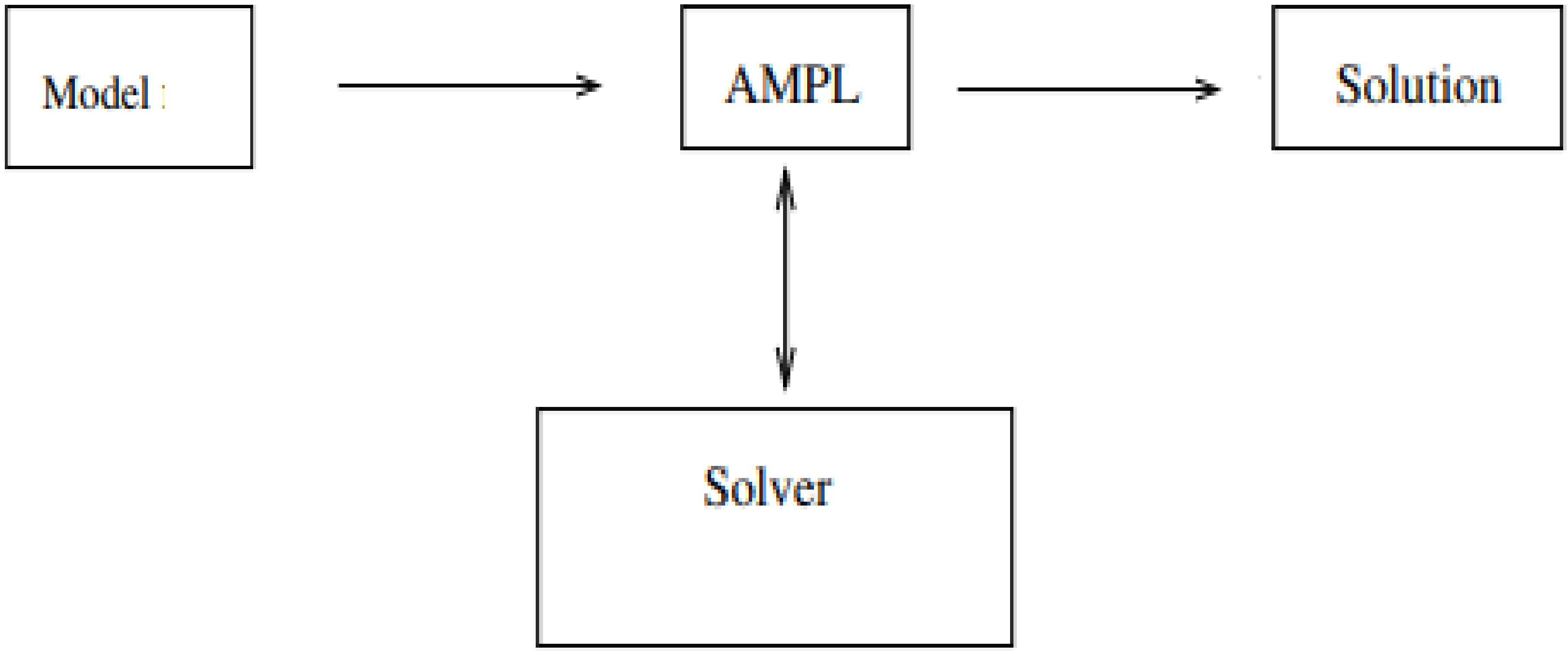
Dipartimento di Informatica, Università di Pisa

a.a. 2019-2020

Short Introduction to AMPL

- **AMPL: A M**athematical **P**rogramming **L**anguage (1985)
- AMPL is a **modeling language** for describing a wide range of high-complexity large-scale optimization problems (LP, MIP, QP, NLP)
- AMPL's (algebraic) syntax and interactive command environment are designed to help formulate models, communicate with a wide variety of solvers and examine solutions
- Supports both open source (CBC) and commercial (**MINOS**, **CPLEX**, Gurobi, Xpress) solvers

Short Introduction to AMPL



Short Introduction to AMPL

- Its syntax is similar to the usual mathematical notation for optimization problems (summation, mathematical functions, …)
- Flexible: two separated files for model and data, a unique script to run
- Full Edition to **pay** or free for 30 days
- It exists a free size-limited AMPL Demo Version (500 variables and 500 constraints plus objectives for linear problems) which also includes demo packages of solvers

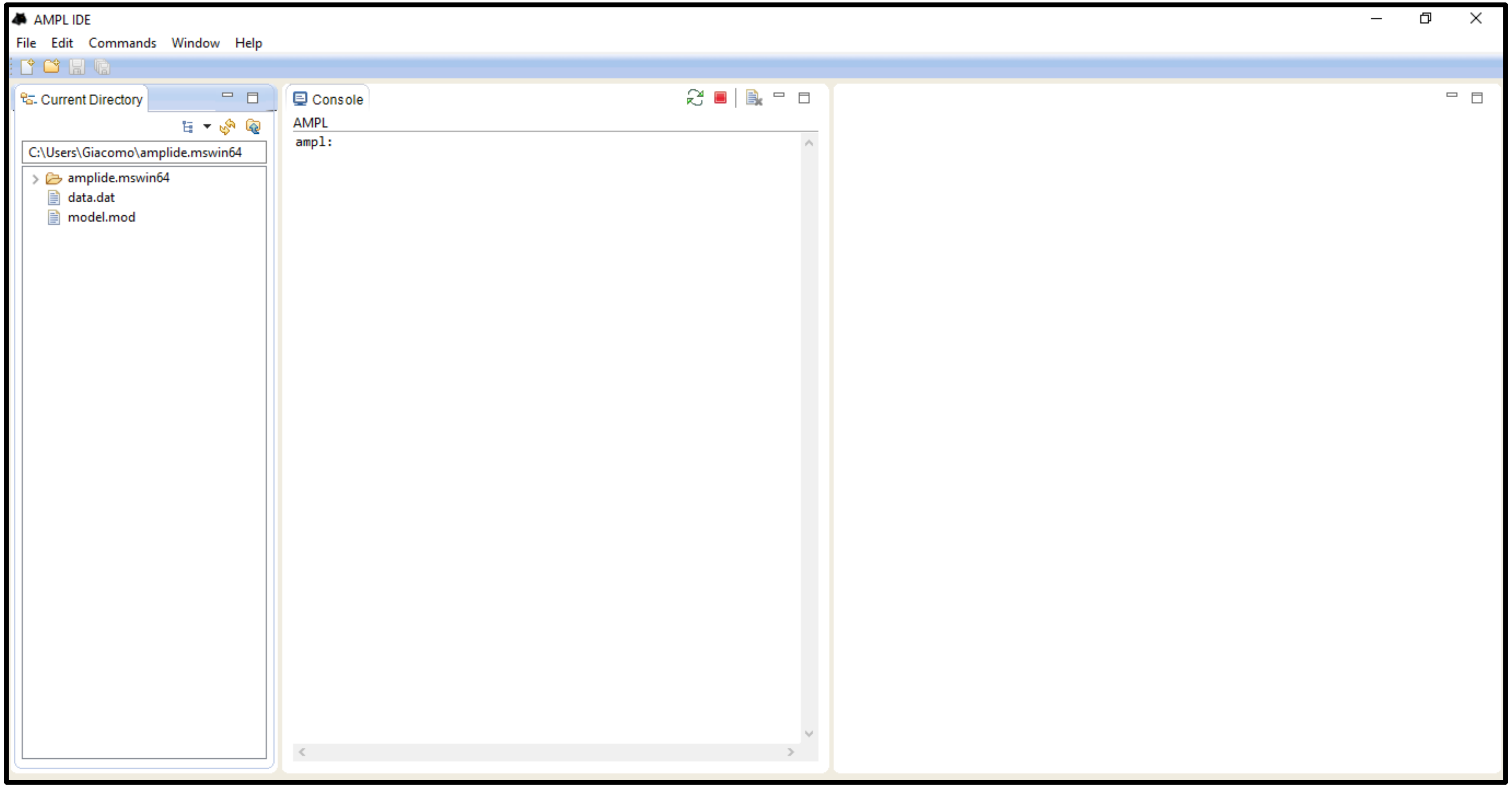
AMPL Installation

- https://ampl.com/try-ampl/download-a-free-demo/
- Using the Command Prompt on your pc OR using an **Integrated Development Environment (IDE)** which provides a simple and straightforward enhanced modeling interface for AMPL users (AMPL IDE)

AMPL Installation

- **AMPL IDE download for Windows** or **AMPL IDE download for Linux**
- **To install**: double-click the zipfile, extract the folder named *amplide.mswin32* or *amplide.mswin64*; this will be your *AMPL folder*
- **To run**: Inside your *AMPL folder*, double-click the *amplide folder icon* and then double-click the ***amplide.exe*** (under Windows and Mac OS the program will have a black cat's-head icon)

AMPL Installation



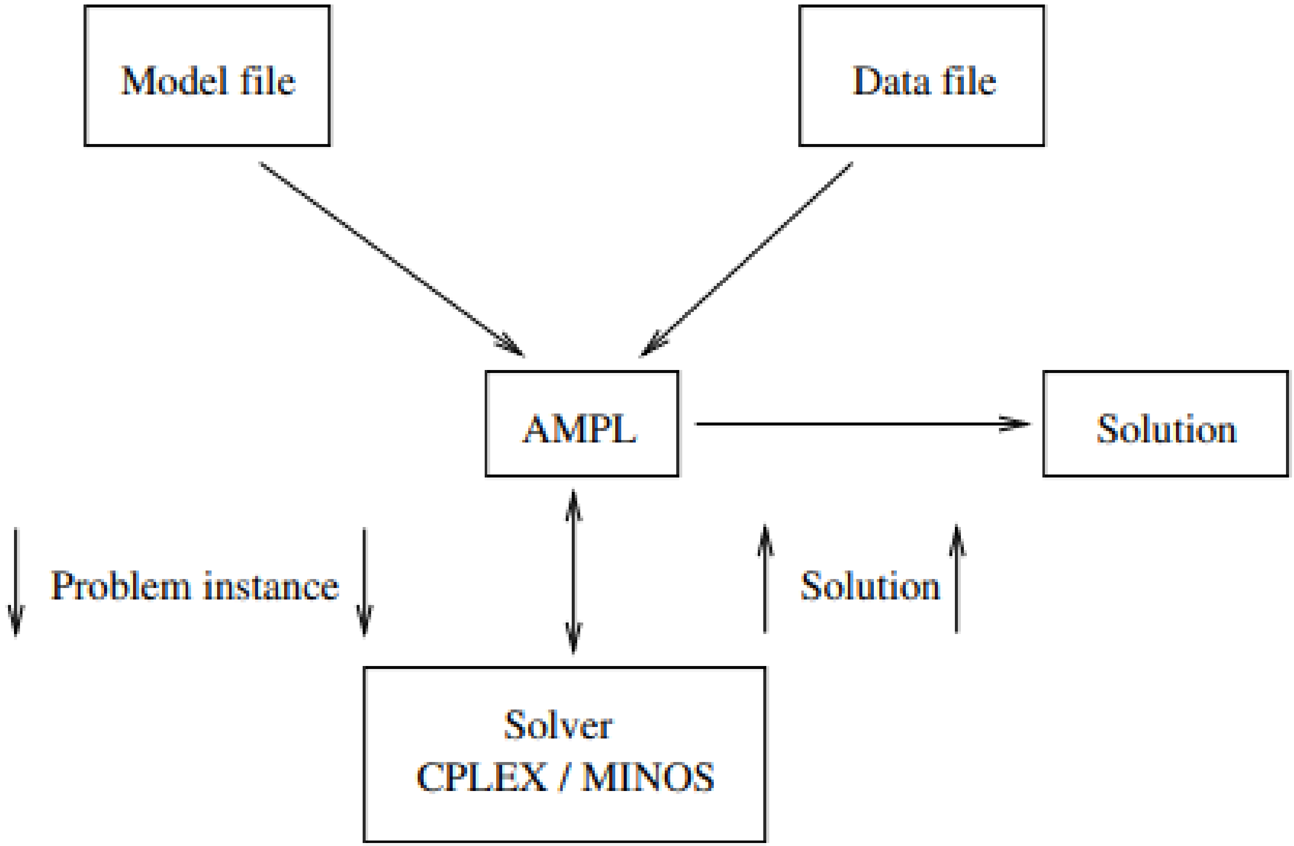
How does AMPL work?

Two text files (the filenames you select must be at most eight characters followed by a three character extension):

- a model file *model.mod*
- a data file *data.dat*

Both files need to be saved in your AMPL folder!

How does AMPL work?



AMPL gives us the chance to express the algebraic representation of a model and the values of parameters in files: a **model file** and a **data file**. AMPL reads the model from the .mod file, data from the .dat file and puts them together into a format that the solver understands. Then, AMPL hands over this problem instance to the solver, which in turn, solves the instance, and hands back the solution to AMPL.

AMPL General Syntax: *model.mod*

Elements:

- Parameters
- Variables
- Obj Function
- Constraints

AMPL General Syntax: *model.mod*

Parameters:

- Each parameter declaration starts with the keyword ***param***
- Single parameters are declared using the syntax "***param*** *name_parameter;*"
- Indexed parameters (vector or matrix) are declared using the syntax "***param*** *name_parameter {range};*"

```
param name_parameter;
param name_parameter{i in 1..n};
param name_parameter{i in 1..n} {j in 1..m};
```

AMPL General Syntax: *model.mod*

Variables:

- Each variable declaration starts with the keyword ***var***
- Single variables are declared using the syntax "***var*** *name_variable;*"
- Indexed variables (vector or matrix) are declared using the syntax "***var*** *name_variable* {range};"

```
var name_variable1;
var name_variable 2{i in 1..n};
var name_variable 3{i in 1..n} {j in 1..m};
```

AMPL General Syntax: *model.mod*

The objective function:

- Is declared using the syntax "***maximize*** *name_obj*:" or "***minimize*** *name_obj*:"
- The objective statement
- A semi-colon

```
maximize name_obj: t*x + sum {i in 1..n} p[i]*y[i];
```

AMPL General Syntax: *model.mod*

Constraints:

- Are declared using the syntax "**subject to** *name_constr*:"
- The equation or inequality
- A semi-colon

```
subject to name_constr1: sum{i in 1..n} y[i] <= t;
subject to name_ constr2{i in 1..n}: y[i] >=0;
subject to name_ constr3: x >= 0;
```

AMPL General Syntax: *data.dat*

Parameters:

- Each parameter declaration starts with the keyword ***param***, a colon and equal sign and the value
- If a parameter has **more than one component** (vector or matrix), list the parameter index followed by the value

```
param name_parameter1:= 2;
param name_parameter2 := 1 10
                         2 15;
```

Notes about AMPL Language

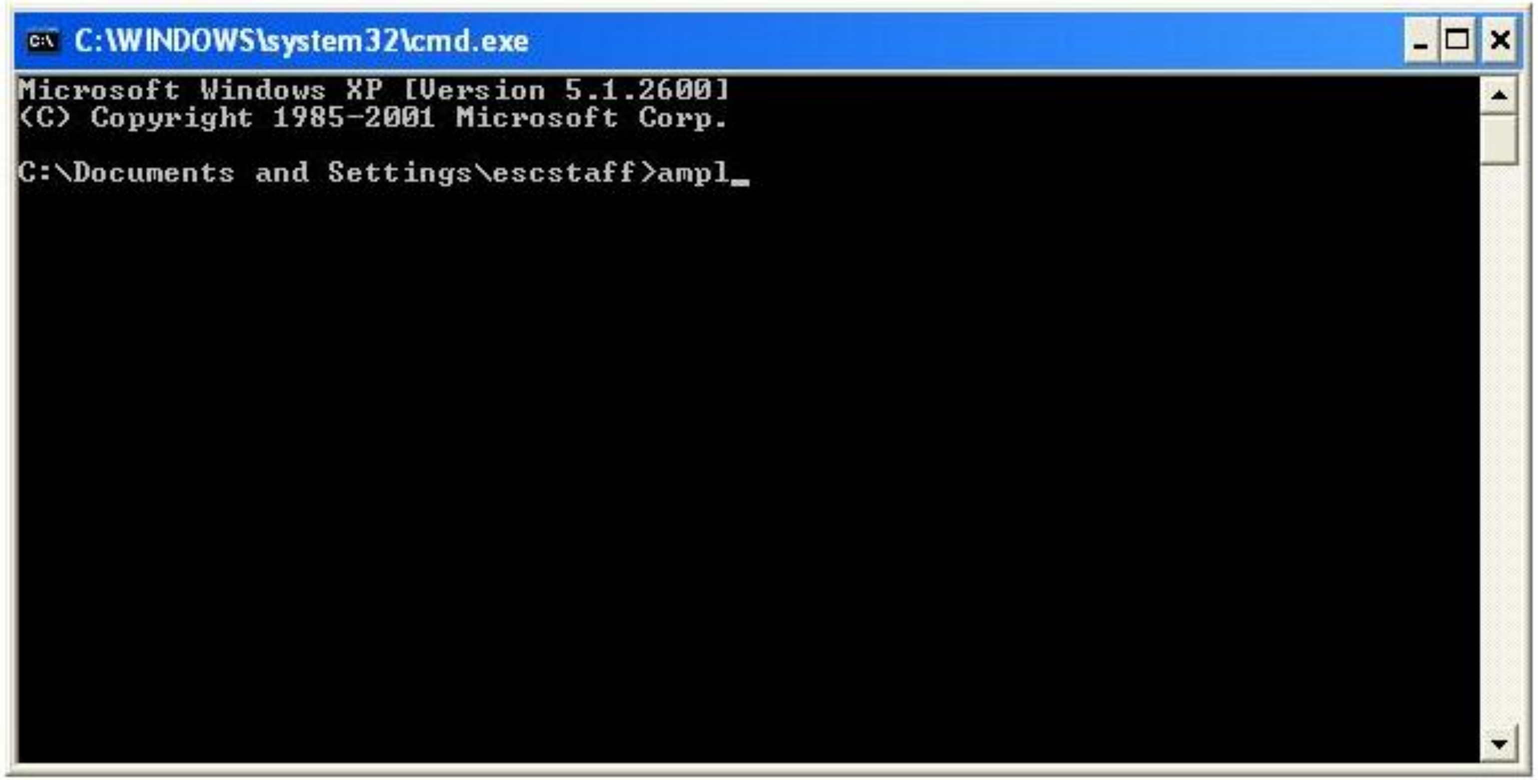
- The # symbol indicates the start of a comment (everything after that symbol is ignored)
- All lines of code must end with a **semi-colon**
- Variables, parameters, obj and constraints **names** can be anything meaningful, made up of upper and lower case letters, digits and underscores, but must be **unique** (variables, parameters, obj and constraints cannot have the same name)
- AMPL is **case sensitive**

How to solve a Model

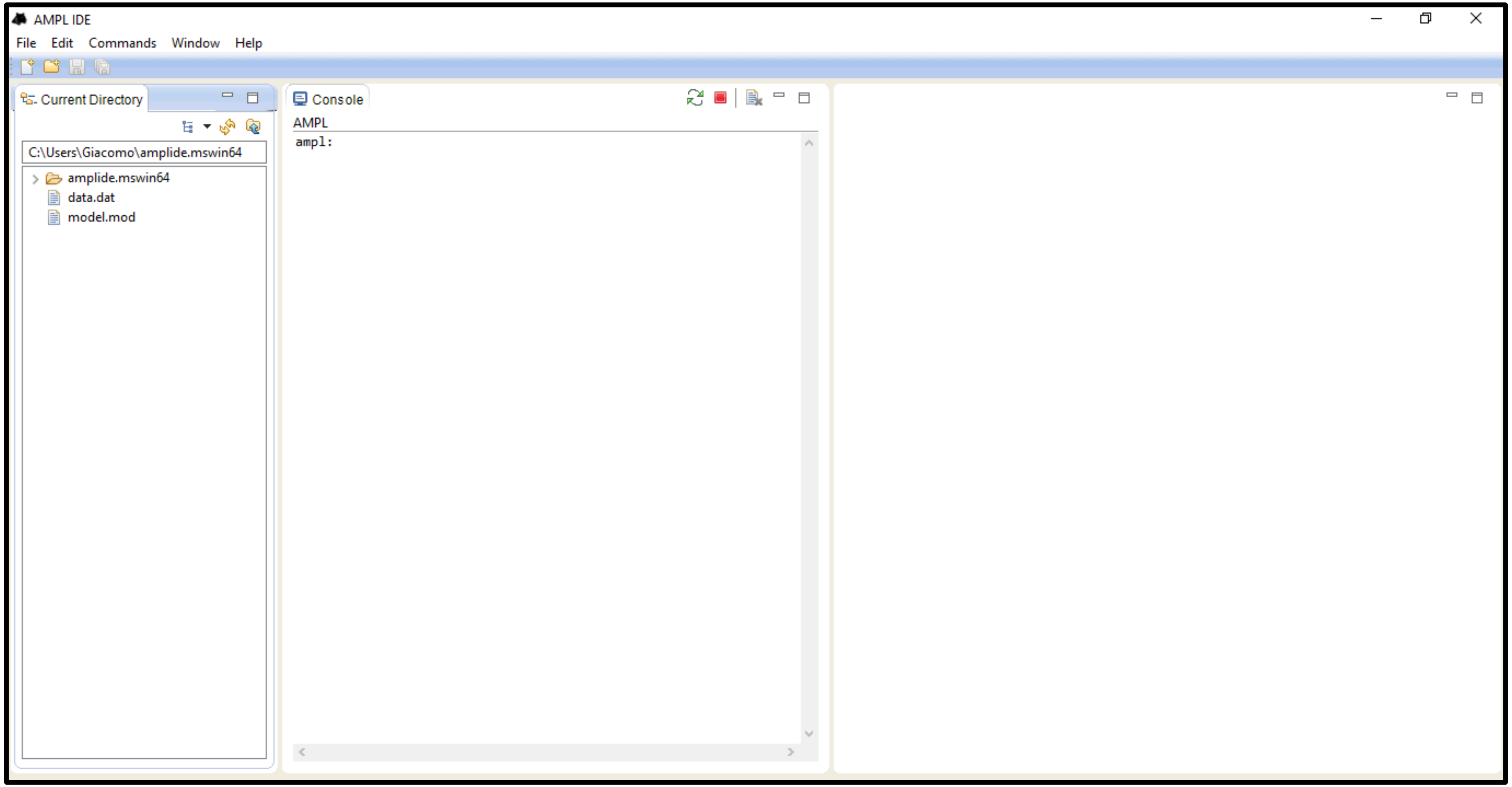
Steps:

- Upload the .mod file (*model model.mod*)
- Upload the .dat file (*data data.dat*)
- [Specify a solver (*option solver cplexamp*)] <- optional
- Solve the problem (*solve*)
- *display* the solution
- Before reloading the model, you must first reset AMPL by typing *reset*

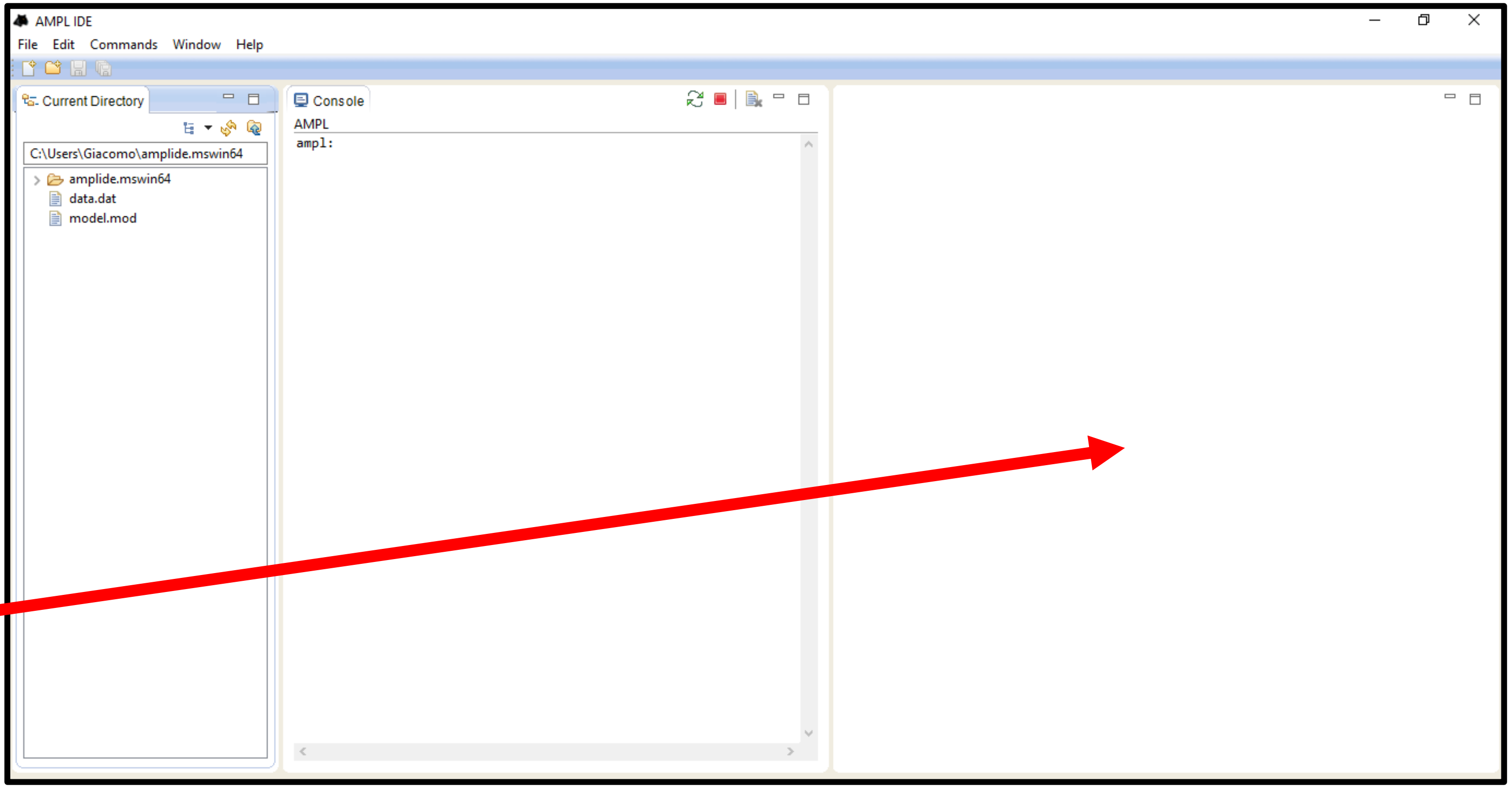
AMPL IDE Description



AMPL IDE Description



AMPL IDE Description

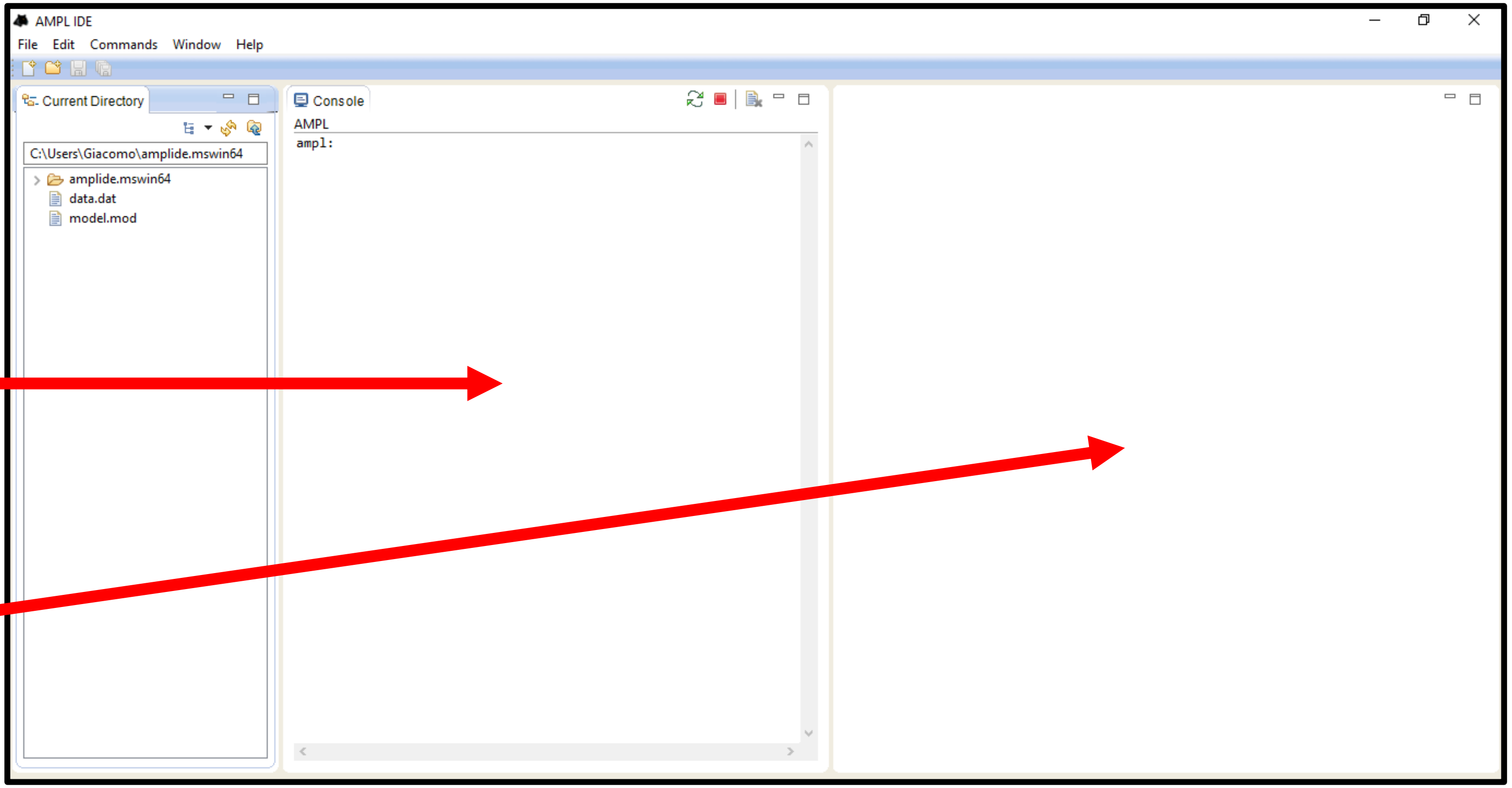


Text Editor (where your .mod and .dat are displayed)

AMPL IDE Description

AMPL Console (where you write AMPL instructions to solve a problem)

Text Editor (where your .mod and .dat are displayed)

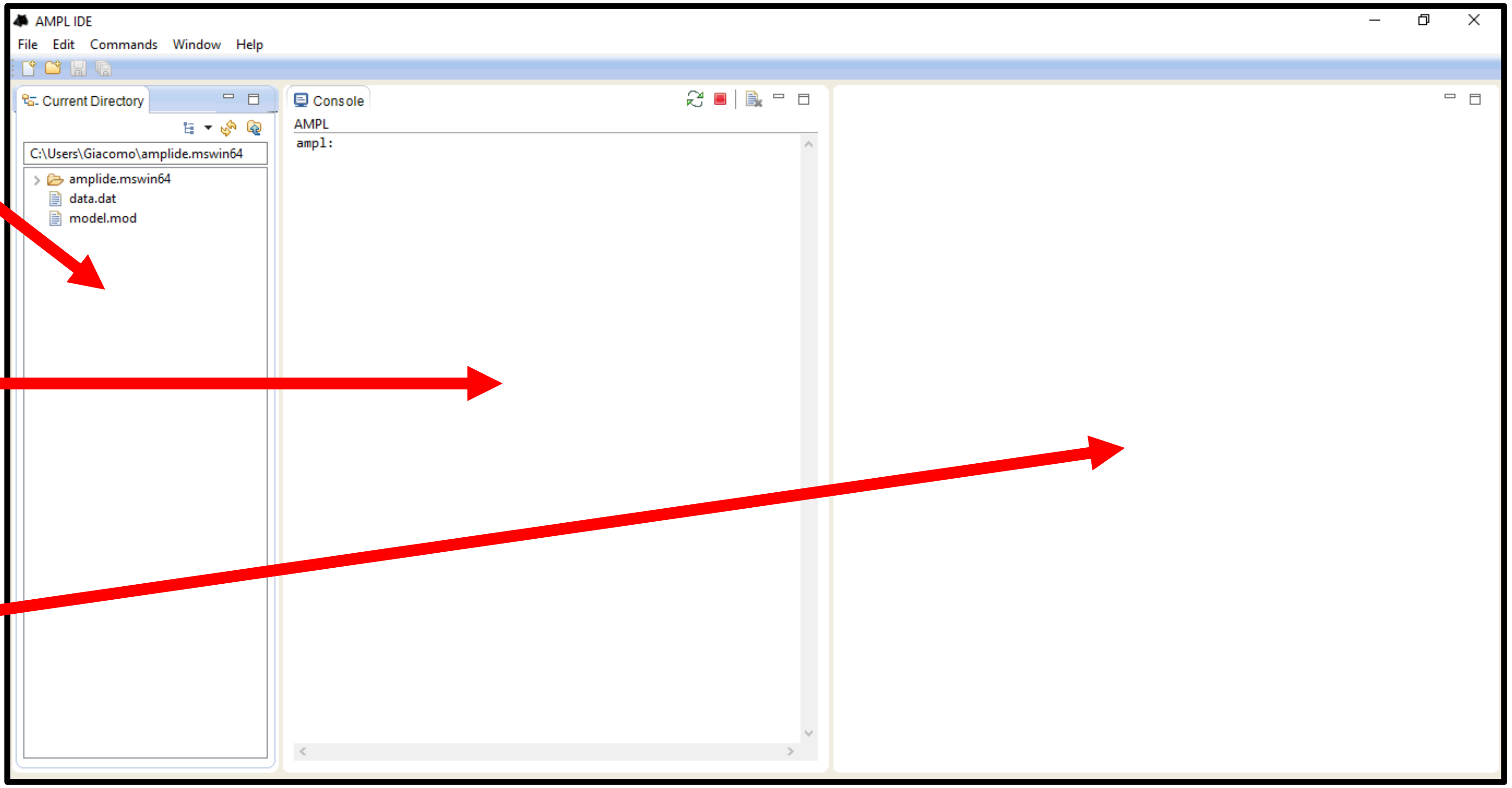


AMPL IDE Description

AMPL folder (where you should save your model and data files)

AMPL Console (where you write AMPL instructions to solve a problem)

Text Editor (where your .mod and .dat are displayed)



A Simple LP Example (Chapter 2.2.1, Page 15)

"Blue Ridge Hot Tubs" produces and sells two types of hot tubs: "Aqua-Spa" and "Hydro-Lux". The manager buys hot tub shells from a local supplier, and then adds pumps and tubing to the shells to create hot tubs.

The resources available in the next production cycle are:

- 200 pumps;
- 2,880 feet of tubing;
- 1,566 production labor hours.

The operating requisites are:

- each Aqua-Spa requires 12 feet of tubing and 9 hours of labor;
- each Hydro-Lux requires 16 feet of tubing and 6 hours of labor.

The profit for the manager is 350 on each Aqua-Spa he sells and 300 on each Hydro-Lux he sells (the manager is confident to sell each hot tub he/she produces).

The decision problem of Blue Ridge Hot Tubs can be stated in the following way: how many Aqua-Spa and Hydro-Lux are to be produced, taking into account the limited resources and the operating requisites, so as to maximize the profit during the next production cycle?

A Simple LP Example (Chapter 2.2.1, Page 15)

The overall LP model is therefore:

$$
\begin{aligned}
\max\ 350x_1 + 300x_2 & \\
x_1 + \quad x_2 &\le 200 \\
9x_1 + \quad 6x_2 &\le 1{,}566 \\
12x_1 + \quad 16x_2 &\le 2{,}880 \\
x_1 \qquad\qquad &\ge 0 \\
x_2 &\ge 0
\end{aligned}
$$

A Simple LP Example: Set Definition

"Blue Ridge Hot Tubs" produces and sells two types of hot tubs: "Aqua-Spa" and "Hydro-Lux". The manager buys hot tub shells from a local supplier, and then adds pumps and tubing to the shells to create hot tubs.

The resources available in the next production cycle are:

- 200 pumps;
- 2,880 feet of tubing;
- 1,566 production labor hours.

The operating requisites are:

- each Aqua-Spa requires 12 feet of tubing and 9 hours of labor;
- each Hydro-Lux requires 16 feet of tubing and 6 hours of labor.

AMPL General Syntax: Sets

AMPL lets us define **sets** composed of elements with specific names and use these names directly to index parameters and variables.

- Each set is declared with the keyword "***set*** *name_set;*"

A Simple LP Example: Set Definition

"Blue Ridge Hot Tubs" produces and sells two types of hot tubs: "Aqua-Spa" and "Hydro-Lux". The manager buys hot tub shells from a local supplier, and then adds pumps and tubing to the shells to create hot tubs.

The resources available in the next production cycle are:

- 200 pumps;
- 2,880 feet of tubing;
- 1,566 production labor hours.

The operating requisites are:

- each Aqua-Spa requires 12 feet of tubing and 9 hours of labor;
- each Hydro-Lux requires 16 feet of tubing and 6 hours of labor.

```
# model
set Resources;
set Tubs;
```

A Simple LP Example: Set Definition

"Blue Ridge Hot Tubs" produces and sells two types of hot tubs: "Aqua-Spa" and "Hydro-Lux". The manager buys hot tub shells from a local supplier, and then adds pumps and tubing to the shells to create hot tubs.

The resources available in the next production cycle are:

- 200 pumps;
- 2,880 feet of tubing;
- 1,566 production labor hours.

The operating requisites are:

- each Aqua-Spa requires 12 feet of tubing and 9 hours of labor;
- each Hydro-Lux requires 16 feet of tubing and 6 hours of labor.

```
# model
set Resources;
set Tubs;
```

```
# data
set Resources := pumps tubing laborhours;
set Tubs := aquaspa hydrolux;
```

A Simple LP Example: Parameters Definition

$$
\begin{aligned}
\max\ 350x_1 + 300x_2 & \\
x_1 + x_2 &\leq 200 \\
9x_1 + 6x_2 &\leq 1{,}566 \\
12x_1 + 16x_2 &\leq 2{,}880 \\
x_1 &\geq 0 \\
x_2 &\geq 0
\end{aligned}
$$

```
# model
set Resources;
set Tubs;
```

```
# data
set Resources := pumps tubing laborhours;
set Tubs := aquaspa hydrolux;
```

A Simple LP Example: Parameters Definition

$$
\begin{aligned}
\max \quad & \boxed{350x_1 + 300x_2} \\
& x_1 + x_2 \leq 200 \\
& 9x_1 + 6x_2 \leq 1{,}566 \\
& 12x_1 + 16x_2 \leq 2{,}880 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

```
# data
set Resources := pumps tubing laborhours;
set Tubs := aquaspa hydrolux;

param Profit:=
aquaspa 350
hydrolux 300;
```

```
# model
set Resources;
set Tubs;

param Profit{Tubs};
```

A Simple LP Example: Parameters Definition

$$
\begin{aligned}
\max\ 350x_1 + 300x_2 & \\
x_1 + x_2 &\leq \boxed{200} \\
9x_1 + 6x_2 &\leq \boxed{1{,}566} \\
12x_1 + 16x_2 &\leq \boxed{2{,}880} \\
x_1 &\geq 0 \\
x_2 &\geq 0
\end{aligned}
$$

```
# model
set Resources;
set Tubs;

param Profit{Tubs};
param Availabilities {Resources};
```

```
# data
set Resources := pumps tubing laborhours;
set Tubs := aquaspa hydrolux;

param Profit:=
aquaspa 350
hydrolux 300;

param Availabilities:=
  pumps   200
  tubing    1566
  laborhours   2880;
```

A Simple LP Example: Parameters Definition

$$\max\ 350x_1 + 300x_2$$
$$\begin{aligned} x_1 + x_2 &\le 200 \\ 9x_1 + 6x_2 &\le 1{,}566 \\ 12x_1 + 16x_2 &\le 2{,}880 \\ x_1 &\ge 0 \\ x_2 &\ge 0 \end{aligned}$$

```
# model
set Resources;
set Tubs;

param Profit{Tubs};
param Availabilities {Resources};
param Requirement {Resources, Tubs};
```

```
# data
set Resources := pumps tubing laborhours;
set Tubs := aquaspa hydrolux;

param Profit:=
aquaspa 350
hydrolux 300;

param Availabilities:=
  pumps   200
  tubing   1566
  laborhours   2880;

param Requirement:
   aquaspa hydrolux :=
   pumps   1   1
   tubing    9   6
   laborhours   12  16;
```

A Simple LP Example: Variables Definition

- identify the decision variables: what are the fundamental decisions that must be made to solve the problem?
  - x_1: number of Aqua-Spa hot tubs to produce;
  - x_2: number of Hydro-Lux hot tubs to produce;

```
# model
set Resources;
set Tubs;

param Availabilities {Resources};
param Requirement {Resources, Tubs};
param Profit{Tubs};

var Quantity {Tubs};
```

A Simple LP Example: Objective Definition

- state the objective function as a linear combination of the decision variables: the manager has a profit of 350 on each Acqua-Spa he/she sells, and of 300 on each Hydro-Lux he/she sells; therefore the total profit, to be maximized, is

$$\max\ 350x_1 + 300x_2;$$

```
# model
maximize Total_Profit:  sum {j in Tubs} Profit[j] * Quantity[j];
```

A Simple LP Example: Contstraints Definition

- state the constraints as linear combinations of the decision variables:
  - only 200 pumps are available, and each hot tub requires one pump:

$$x_1 + x_2 \leq 200;$$

  - only 1,566 labor hours are available, and each Aqua-Spa requires 9 labor hours while each Hydro-Lux requires 6 labor hours:

$$9x_1 + 6x_2 \leq 1{,}566;$$

  - only 288 feet of tubing is available, and each Aqua-Spa requires 12 feet while each Hydro-Lux requires 16 feet:

$$12x_1 + 16x_2 \leq 2{,}880;$$

```
# model
subject to ConstrAvail {i in Resources}: sum {j in Tubs} Requirement[i,j] * Quantity[j] <= Availabilities[i];
```

A Simple LP Example: Non-Negativity Constraints Definition (I/II)

- nonnegativity constraints: we cannot produce a negative number of hot tubs:

$$x_1 \geq 0, \;\; x_2 \geq 0.$$

```
# model
subject to non-neg {j in Tubs}: Quantity[j] >= 0;
```

A Simple LP Example: Non-Negativity Constraints Definition (I/II)

- nonnegativity constraints: we cannot produce a negative number of hot tubs:

$$x_1 \geq 0, \ x_2 \geq 0.$$

```
# model
subject to non-neg {j in Tubs}: Quantity[j] >= 0;
```

```
# model
set Resources;
set Tubs;

param Availabilities {Resources};
param Requirement {Resources, Tubs};
param Profit{Tubs};

var Quantity {Tubs} >= 0;
```

A Simple LP Example: Model and Data Files

```
set Resources;
set Tubs;

param Availabilities {Resources};
param Requirement {Resources, Tubs};
param Profit{Tubs};

var Quantity {Tubs} >= 0;

maximize Total_Profit:  sum {j in Tubs} Profit[j] * Quantity[j];

subject to ConstrAvail {i in Resources}:
  sum {j in Tubs} Requirement[i,j] * Quantity[j] <= Availabilities[i];
```

```
data;
set Tubs := aquaspa hydrolux;
set Resources := pumps tubing laborhours;
param Availabilities:=
  pumps   200
  tubing    1566
  laborhours   2880;
param Requirement: aquaspa hydrolux :=
   pumps   1   1
   tubing     9   6
   laborhours   12  16;
param Profit:=
aquaspa 350
hydrolux 300;
```

A Simple LP Example: Launching with Specific Solver (Cplex)

```
# Console
ampl: model BlueRidge.mod;
ampl: data BlueRidge.dat;
ampl: option solver cplexamp;
ampl: solve;
```

A Simple LP Example: Launching with Specific Solver (Cplex)

Solver message

```
# Console
ampl: model BlueRidge.mod;
ampl: data BlueRidge.dat;
ampl: option solver cplexamp;
ampl: solve;

CPLEX 12.6.1.0: optimal solution found.
2 iterations, objective 66100
2 MIP simplex iterations
0 branch-and-bound nodes
```

A Simple LP Example: Launching with Specific Solver (Cplex)

Solver message (**algorithm used**, **type of soluiton**, **obj value**, iterations of the algorithm)

```
# Console
ampl: model BlueRidge.mod;
ampl: data BlueRidge.dat;
ampl: option solver cplexamp;
ampl: solve;

CPLEX 12.6.1.0: optimal solution found.
2 iterations, objective 66100
2 MIP simplex iterations
0 branch-and-bound nodes
```

A Simple LP Example: Launching with Default Solver (Minos)

```
# Console
ampl: model BlueRidge.mod;
ampl: data BlueRidge.dat;
ampl: solve;

MINOS 5.51: optimal solution found.
2 iterations, objective 66100
```

A Simple LP Example: Display Solution

```
# Console
ampl: model BlueRidge.mod;
ampl: data BlueRidge.dat;
ampl: option solver cplexamp;
ampl: solve;

CPLEX 12.6.1.0: optimal solution found.
2 iterations, objective 66100
2 MIP simplex iterations
0 branch-and-bound nodes

ampl: display Quantity;
```

A Simple LP Example: Display Solution

```
# Console
ampl: model BlueRidge.mod;
ampl: data BlueRidge.dat;
ampl: option solver cplexamp;
ampl: solve;

CPLEX 12.6.1.0: optimal solution found.
2 iterations, objective 66100
2 MIP simplex iterations
0 branch-and-bound nodes

ampl: display Quantity;
Quantity [*] :=
aquaspa  118
hydrolux   76;
```

Values of Variables →

A Simple LP Example: changing Parameters

$$\begin{aligned} \max\ 350x_1 + 300x_2 & \\ x_1 + \quad x_2 &\leq 200 \\ 9x_1 + \quad 6x_2 &\leq 1{,}566\ (1{,}520) \\ 12x_1 + \quad 16x_2 &\leq 2{,}880\ (2{,}650) \\ x_1\ , \quad x_2 &\geq 0 \end{aligned}$$

A Simple LP Example: changing Parameters

```
set Resources;
set Tubs;

param Availabilities {Resources};
param Requirement {Resources, Tubs};
param Profit{Tubs};

var Quantity {Tubs} >= 0;

maximize Total_Profit:  sum {j in Tubs} Profit[j] * Quantity[j];

subject to ConstrAvail {i in Resources}:
  sum {j in Tubs} Requirement[i,j] * Quantity[j] <= Availabilities[i];
```

```
data;
set Tubs := aquaspa hydrolux;
set Resources := pumps tubing laborhours;
param Availabilities:=
 pumps   200
 tubing    1520       #1556
 laborhours   2650; #2650
param Requirement: aquaspa hydrolux :=
  pumps   1   1
  tubing    9   6
  laborhours   12  16;
param Profit:=
aquaspa 350
hydrolux 300;
```

A Simple LP Example: Display Solution

```
# Console
ampl: reset;
ampl: model BlueRidge.mod
ampl: data BlueRidge.dat
ampl: option solver cplexamp;
ampl: solve;
CPLEX 12.6.1.0: optimal solution; objective
64305.55556
3 dual simplex iterations (1 in phase I)
ampl: display Quantity;
Quantity [*] :=
 aquaspa  116.944
hydrolux   77.9167;
```

A Simple LP Example: Integer Version

$$
\begin{aligned}
\max\ 350x_1 + 300x_2 & \\
x_1 + \quad x_2 &\le 200 \\
9x_1 + \quad 6x_2 &\le 1{,}566 \ (1{,}520) \\
12x_1 + \ 16x_2 &\le 2{,}880 \ (2{,}650) \\
x_1 \ , \quad x_2 &\ge 0 \\
x_1 \ , \quad x_2 &\ \text{integer}
\end{aligned}
$$

A Simple LP Example: Model and Data Files

```
set Resources;
set Tubs;

param Availabilities {Resources};
param Requirement {Resources, Tubs};
param Profit{Tubs};

var Quantity {Tubs} integer;

maximize Total_Profit:  sum {j in Tubs} Profit[j] * Quantity[j];

subject to ConstrAvail {i in Resources}:
  sum {j in Tubs} Requirement[i,j] * Quantity[j] <= Availabilities[i];
```

```
data;
set Tubs := aquaspa hydrolux;
set Resources := pumps tubing laborhours;
param Availabilities:=
  pumps   200
  tubing   1520
  laborhours   2650;
param Requirement: aquaspa hydrolux :=
   pumps   1   1
   tubing    9   6
   laborhours   12  16;
param Profit:=
aquaspa 350
hydrolux 300;
```

A Simple LP Example: Display Solution

```
# Console
ampl: reset;
ampl: model BlueRidge.mod
ampl: data BlueRidge.dat
ampl: option solver cplexamp;
ampl: solve;
CPLEX 12.6.1.0: optimal integer solution;
objective 64100
2 MIP simplex iterations
0 branch-and-bound nodes
ampl: display Quantity;
Quantity [*] :=
 aquaspa  118
hydrolux   76;
```

AMPL Main Commands:

• reset;	# reset the environment
• model *modelfilename*.mod;	# model upload
• data *datafilename*.dat;	# data upload
• option solver *nameofsolver*;	# optimizer selection
• solve;	# solve
• display *nameofvariables*;	# displays variables