One shot games

Exercise 2

Part III

One-shot Two-player Games

One-shot Two-player Games

One shot games

- Matching
- Table functions
- Programming tools
  - The scope operator
  - `if` as function/as statement

Exercise 2

Interactive experiments

The subject's payoff also depends on the decisions of other subjects.
Example: **Prisoner's dilemma**

	cooperate	**defect**
cooperate	3,3	0,5
defect	5,0	1,1

One shot games
Matching
Table functions
Programming tools
The scope operator
`if` as function/as statement
Exercise 2

Interactive experiments

The subject's payoff also depends on the decisions of other subjects.
Example: **Prisoner's dilemma**

	cooperate	**defect**
cooperate	3,3	0,5
defect	5,0	1,1

Hint: use **parameters** to make your program more flexible.

	cooperate	**defect**
cooperate	r,r	s,t
defect	t,s	p,p

One shot games
Matching
Table functions
Programming tools
The scope operator
`if` as function/as statement
Exercise 2

PD – steps

1. create pairs
2. set parameters: r, t, s, p
3. initialize variables:
   - partner
   - choice
   - partnerchoice
4. subjects' choice
5. profit calculation
6. results

PD – steps

1. create pairs
2. set parameters: r, t, s, p
3. initialize variables:
   - partner
   - choice
   - partnerchoice
4. subjects' choice
5. profit calculation
6. results

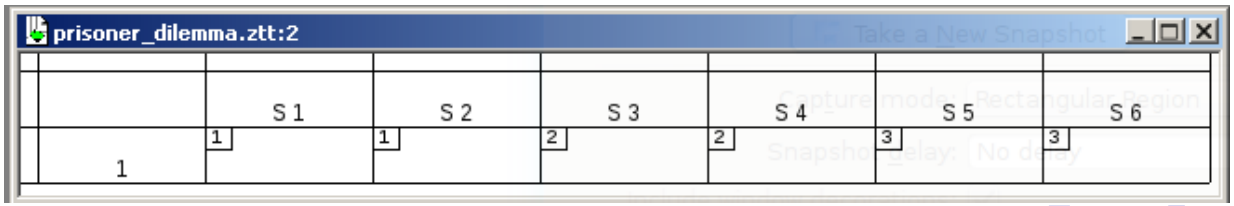
See prisoner_dilemma.ztt

One shot games
Matching
Table functions
Programming tools
The scope operator
`if` as function/as statement
Exercise 2

1. Create pairs - a simple matching procedure

In the `Background` set:

- number of subjects=10
- number of groups=5

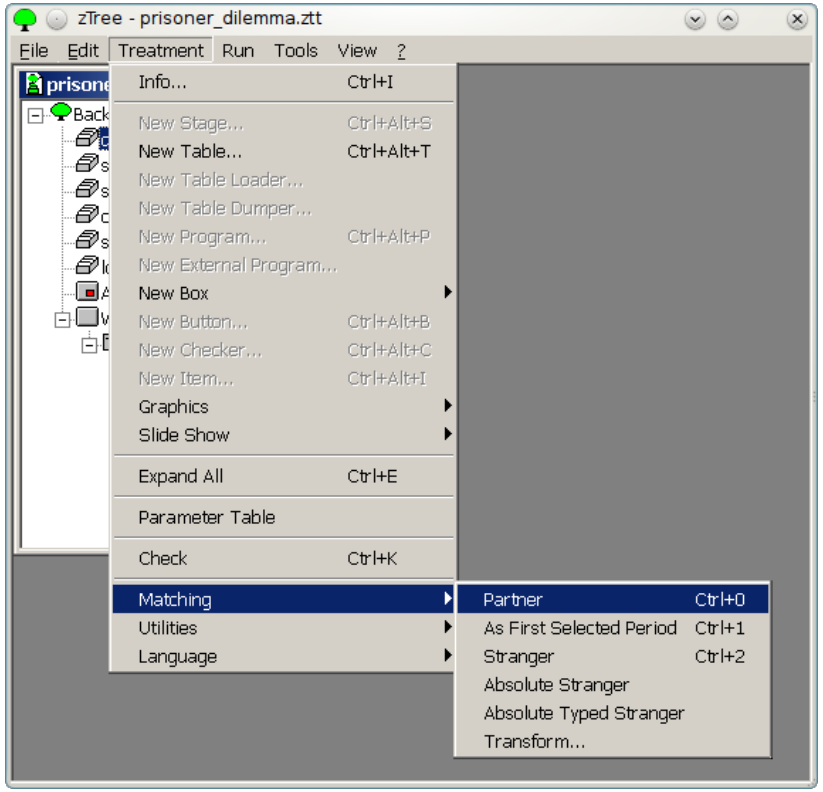


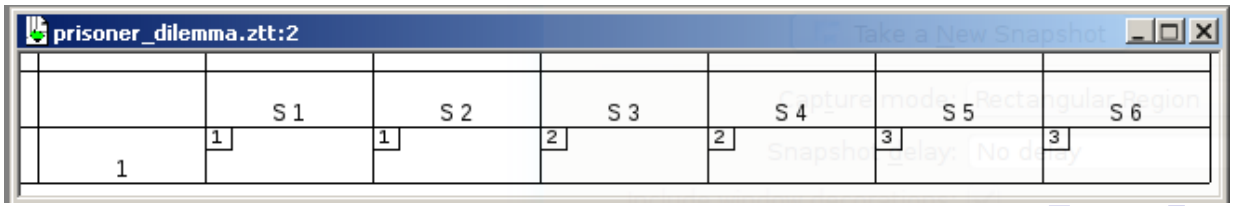
1. Create pairs - a simple matching procedure

In the Background set:

- number of subjects=10
- number of groups=5

Then from the menu select Treatment → Matching → Partner.



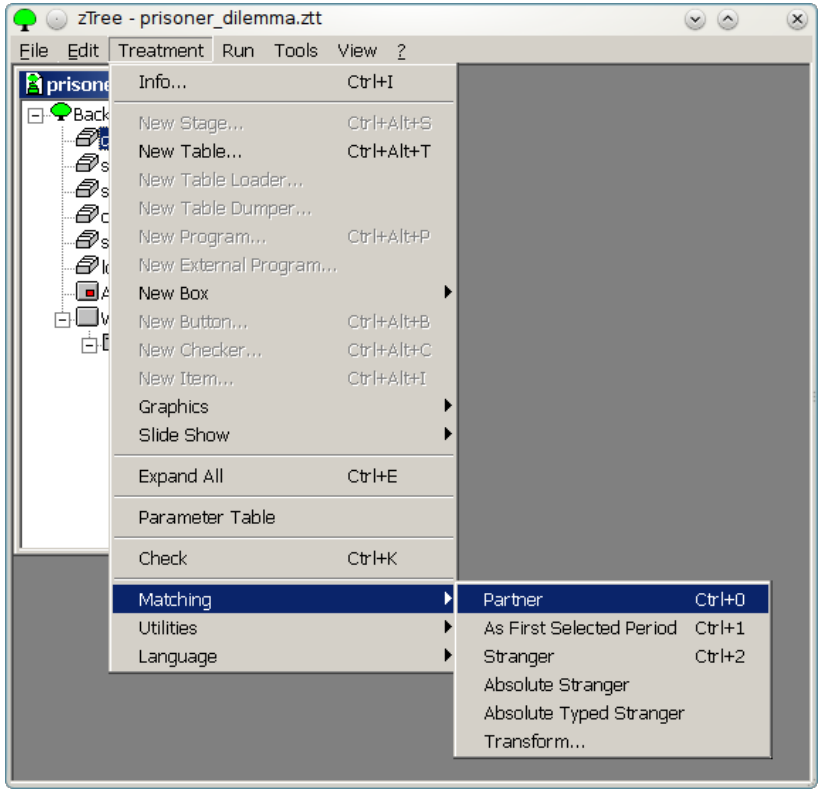


1. Create pairs - a simple matching procedure

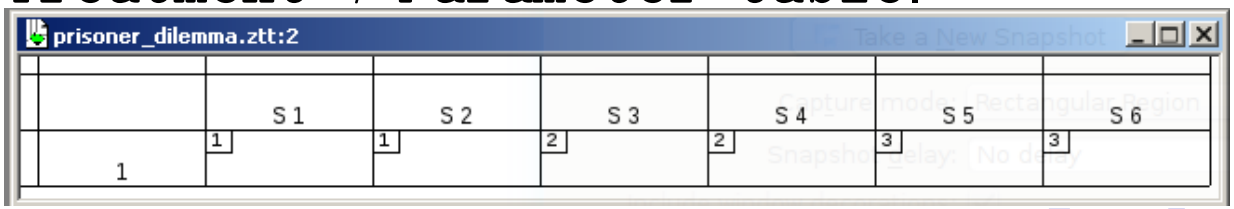
In the `Background` set:

- number of subjects=10
- number of groups=5

Then from the menu select `Treatment` → `Matching` → `Partner`.



To **check the matching**, from the menu select `Treatment`→ `Parameter table`.



2. and 3. Parameters and variables

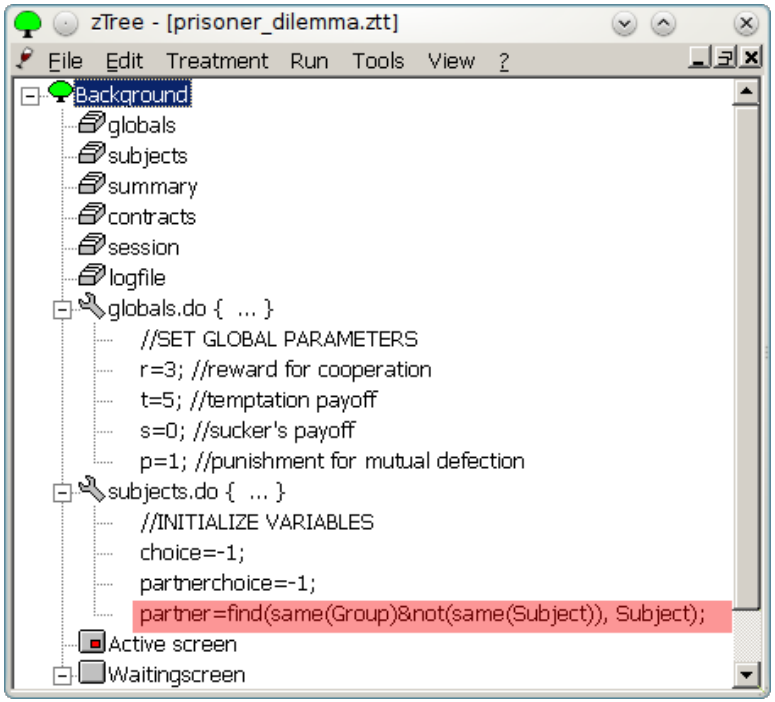


Table functions

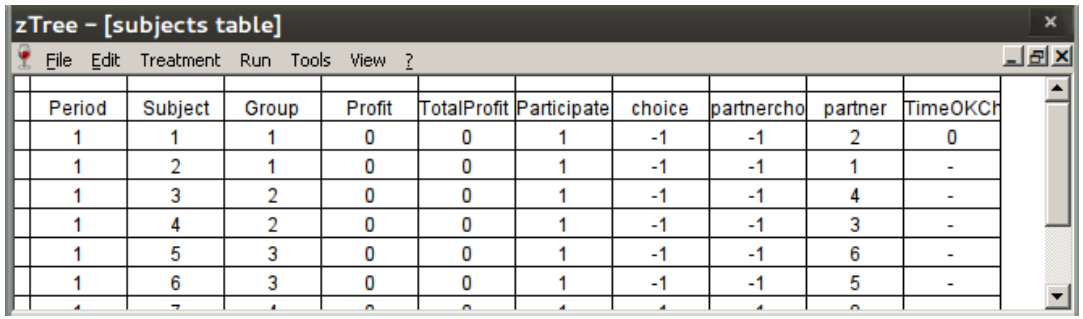
The function `find()` is a table function, i.e. a function which does not only refer to a single record of a table, but runs over the **whole table**.

A complete list of table functions can be found at page 46 of the Reference Manual.

One shot games
Matching
Table functions
Programming tools
The scope operator
`if` as function/as statement
Exercise 2

Find the partner

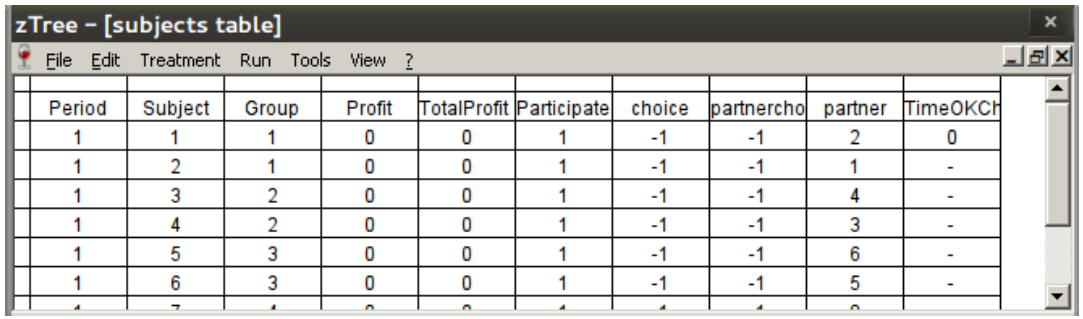
```
partner=find(same(Group)
        &not(same(Subject)),Subject)
```



Period	Subject	Group	Profit	TotalProfit	Participate	choice	partnercho	partner	TimeOKCh
1	1	1	0	0	1	-1	-1	2	0
1	2	1	0	0	1	-1	-1	1	-
1	3	2	0	0	1	-1	-1	4	-
1	4	2	0	0	1	-1	-1	3	-
1	5	3	0	0	1	-1	-1	6	-
1	6	3	0	0	1	-1	-1	5	-

Find the partner

```
partner=find(same(Group)
             &not(same(Subject)),Subject)
```

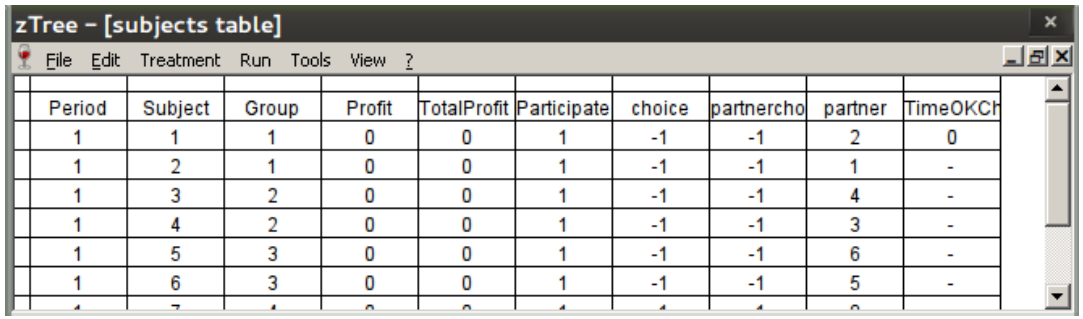


Period	Subject	Group	Profit	TotalProfit	Participate	choice	partnercho	partner	TimeOKCh
1	1	1	0	0	1	-1	-1	2	0
1	2	1	0	0	1	-1	-1	1	-
1	3	2	0	0	1	-1	-1	4	-
1	4	2	0	0	1	-1	-1	3	-
1	5	3	0	0	1	-1	-1	6	-
1	6	3	0	0	1	-1	-1	5	-

- `find(condition, x)` looks at all the records in a table, from top to bottom, and returns the value of variable `x` of the first record in which condition is TRUE.

Find the partner

```
partner=find(same(Group)
             &not(same(Subject)),Subject)
```

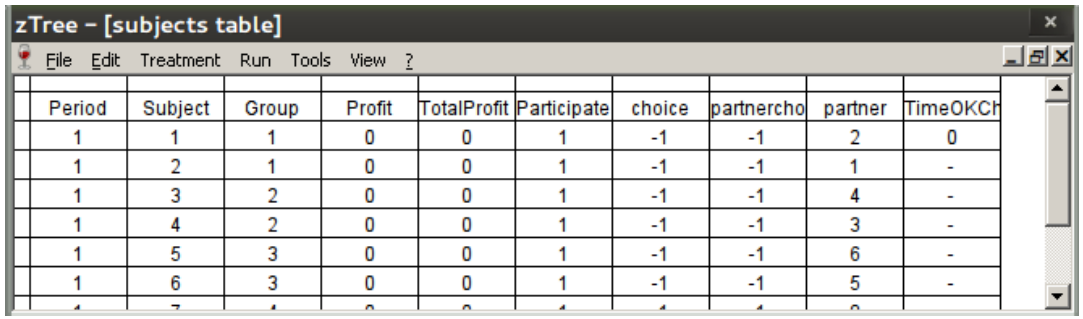


Period	Subject	Group	Profit	TotalProfit	Participate	choice	partnercho	partner	TimeOKCh
1	1	1	0	0	1	-1	-1	2	0
1	2	1	0	0	1	-1	-1	1	-
1	3	2	0	0	1	-1	-1	4	-
1	4	2	0	0	1	-1	-1	3	-
1	5	3	0	0	1	-1	-1	6	-
1	6	3	0	0	1	-1	-1	5	-

- `find(condition, x)` looks at all the records in a table, from top to bottom, and returns the value of variable `x` of the first record in which condition is TRUE.
- `same(x)` is TRUE for all records in the table, where the variable (or expression) `x` takes the same value it has in the "reference record".

Find the partner

```
partner=find(same(Group)
             &not(same(Subject)),Subject)
```

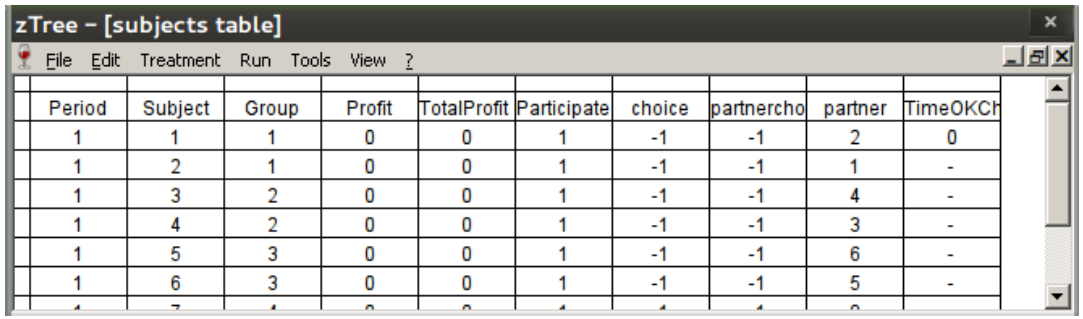


Period	Subject	Group	Profit	TotalProfit	Participate	choice	partnercho	partner	TimeOKCh
1	1	1	0	0	1	-1	-1	2	0
1	2	1	0	0	1	-1	-1	1	-
1	3	2	0	0	1	-1	-1	4	-
1	4	2	0	0	1	-1	-1	3	-
1	5	3	0	0	1	-1	-1	6	-
1	6	3	0	0	1	-1	-1	5	-

- `find(condition, x)` looks at all the records in a table, from top to bottom, and returns the value of variable `x` of the first record in which condition is TRUE.
- `same(x)` is TRUE for all records in the table, where the variable (or expression) `x` takes the same value it has in the "reference record".
- `not(same(x))` is TRUE when `same(x)` is FALSE and vice versa.

Find the partner

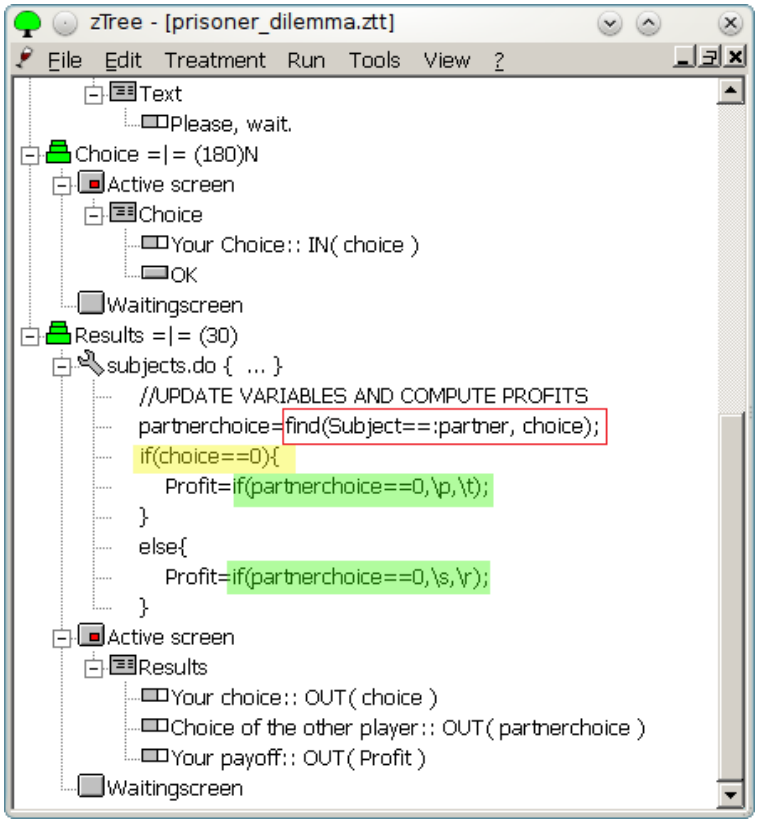
```
partner=find(same(Group)
             &not(same(Subject)),Subject)
```



Period	Subject	Group	Profit	TotalProfit	Participate	choice	partnercho	partner	TimeOKCh
1	1	1	0	0	1	-1	-1	2	0
1	2	1	0	0	1	-1	-1	1	-
1	3	2	0	0	1	-1	-1	4	-
1	4	2	0	0	1	-1	-1	3	-
1	5	3	0	0	1	-1	-1	6	-
1	6	3	0	0	1	-1	-1	5	-

- `find(condition, x)` looks at all the records in a table, from top to bottom, and returns the value of variable `x` of the first record in which condition is TRUE.
- `same(x)` is TRUE for all records in the table, where the variable (or expression) `x` takes the same value it has in the "reference record".
- `not(same(x))` is TRUE when `same(x)` is FALSE and vice versa.

4. 5. and 6. Choices and results



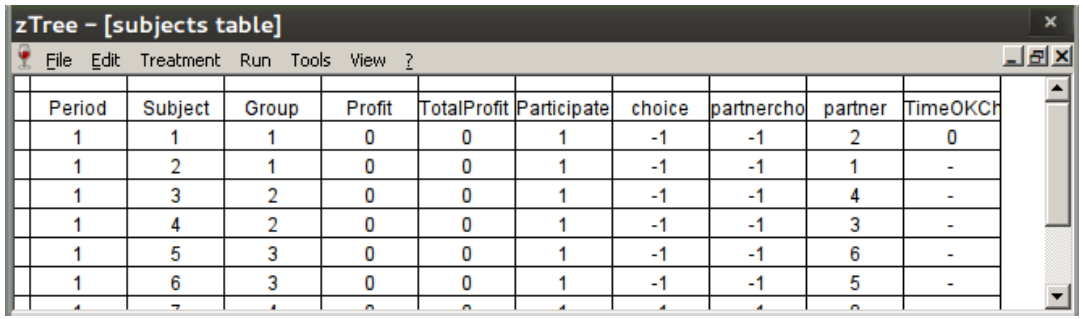
4. **subjects' choice:** with radiobuttons: `!radio:0="D";1="C";`
5. **profit calculation:** scope operator (:) and if as a function and as a statement. See page 44 of the Reference manual.
6. **display results:** layout `!text:0="D";1="C";`

The scope operator

see page 27 of the z-Tree Tutorial

```
partnerchoice=find(Subject==:partner, choice)
```

The scope operator ":" indicates that the variable `partner` belongs to the record in which the cell `partnerchoice` lies, not to the record where `Subject` lies.



Period	Subject	Group	Profit	TotalProfit	Participate	choice	partnercho	partner	TimeOKCh
1	1	1	0	0	1	-1	-1	2	0
1	2	1	0	0	1	-1	-1	1	-
1	3	2	0	0	1	-1	-1	4	-
1	4	2	0	0	1	-1	-1	3	-
1	5	3	0	0	1	-1	-1	6	-
1	6	3	0	0	1	-1	-1	5	-

`if` as function/as statement

```
if(condition a){c1}\\
elsif(condition b1){c2}\\
elsif(condition b2){c3}...\\
else{c4}
```

`if` **as a statement**

- If condition `a` is TRUE, then command(s) `c1` is (are) executed;
- if condition `a` is FALSE and condition `b1` is TRUE, then command(s) `c2` is (are) executed; (etc.)
- if conditions `a`, `b1` and `b2` are FALSE, then command(s) `c4` is (are) executed;

One shot games
Matching
Table functions
Programming tools
The scope operator
`if` as function/as statement
Exercise 2

Exercise 2: the Traveler's dilemma

The Traveler's dilemma is a simple two-players game.

- each player can choose a number **between 2 and 100**
- the choice is simultaneous
- the player's **profit** is:
  - equal to the number he chose, if this number is *equal* to the number chosen by his partner
  - equal to the number he chose $+2$, if this number is *lower* than the number chosen by his partner
  - equal to the number he chose -2, if this number is *higher* than the number chosen by his partner