A short Course for

zTree

Slides and Concept by Dennis Kubitza

briq Institute on Behavior & Inequality

2. Juli 2018

Contents

Content Overview

zTree and Programming

We already used some basic programming in our zTree Experiments.

- Variable Definitions
- Basic Calculation
- If Clauses
- Random Numbers

But zTree offers much more.

zTree and Programming

In zTree it is possible to:

- Write complex Programs / Calculations
- Easily program interaction between users

To understand programming in zTree we will...

- ...start with the fundamentals of Programming in the Single Player Scenario.
- ... afterwards we will switch to Multiplayer Games

Content Overview

Adding a Program

Programms can be added in zTree with

$$Treatment > NewProgram$$

but only if you have selected

- logfile
- a Stage
- a Button

Program Behaviour

Depending on where you define them, they have different behaviors.

logfile The program will be executed **before** the Treatment

Stage The program will be executed **before** the Stage

Button The programm will be executed **after** clicking the Button

General Syntax

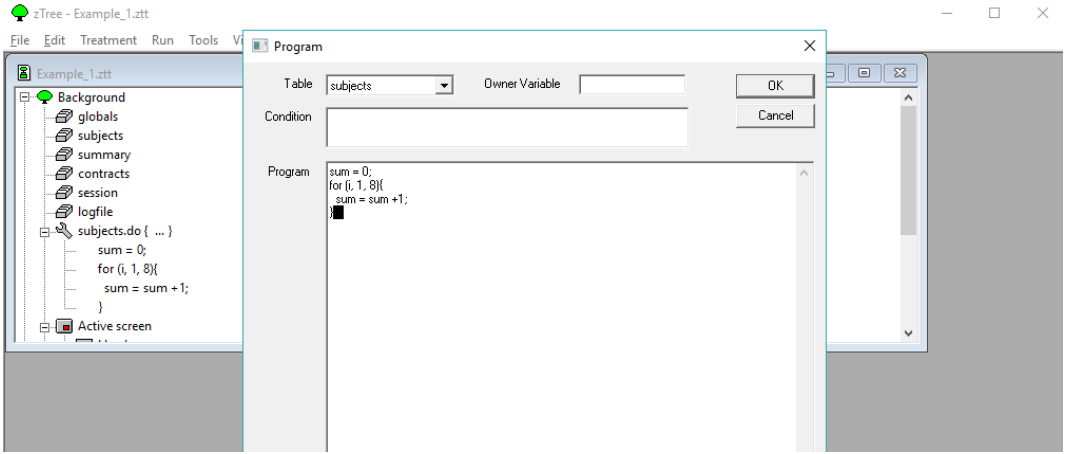
A Programm is a collection of different commands, like

- Variable Definitions
- Calculations
- Control Structures

After each command a line has to be Terminated with

;

Example


zTree - Example_1.ztt
File Edit Treatment Run Tools
Example_1.ztt
Background
globals
subjects
summary
contracts
session
logfile
subjects.do { ... }
sum = 0;
for (i, 1, 8){
sum = sum +1;
}
Active screen
Program
Table
subjects
Owner Variable
OK
Condition
Cancel
Program
sum = 0;
for (i, 1, 8){
sum = sum +1;
}

Defining Variables in zTree

Variables can be definded/reasigned by giving a Variable Name and a Value

$$name = value$$

In General a Variable has two different properties in zTree:

1. Visibility
2. Type

Visiblity

When creating a Program we have to choose a Table a Programm belongs to. This determines who can see this variable

When a variable is defined in

- **subjects** it is defined for each subject seperatly. And only this particular subject has access to it by default.
- **globals** a single Variable is defined for all subjects. They all have access to it.

Depending on for whom this Variable is defined, the will be saved in different tables.

Variable Types

Despite the zTree Internal Attributes of Variables Visibility each Variable has also a Type.

- A type tells what is stored in a variable. There a 3 general types
  - Numbers (1, 2, 3, 0.01, -0.8 ...)
  - Strings ("How are you", "Hello", "Dennis")
  - Booleans (true, false)
  - Arrays (A indexed Set of Numbers, Strings, Booleans, or Arrays)

A variable of a certain type can never be overwritten with a variable from an other type

The Type of the first definition determines what Type a Variable has to contain.

Example

Program

Table: globals

Owner Variable:

Condition:

Program:

```
int_global_sum = round( random()* 10, 1);
str_welcome_text = "Willkommen";
bool_show_text = TRUE;
array arr_somenumbers[3];
arr_somenumbers[0]=1;
arr_somenumbers[1]=2;
arr_somenumbers[2]=3;
```

Note on Arrays

Creating an array is different from creating the other types:

1. First, we need to tell zTree how big the Array shall be by

$$array \quad < name\_of\_array > [< size\_of\_array >]$$

2. After that we can assign different values to the indexes by

$$< name\_of\_array > [< index >] =< value >$$

Arrays indexes start always with 0. The hightest index is therefore $< size\_of\_array > -1$

Internal Variables

Despite user defined variables, zTree offers some internal Variables that are always accessible:

Session	Each zTree instance you Start has a unqiue Session
Subject	Each Subject has a unique Name definded by the Client
Group	Each Subject belongs to a Group
Profit	Each Subject has a unique Profit for each Treatment
TotalProfit	Each Subject has a unique Total Profit for each Session

Acessing Variables

It is only possible to Access Variables that are already included in one of the tables. For that reason:

Always define the variables below logfile and initialize them with 0, "" or False.

Number Commands

Despite basic Opterations like $*/+-$ zTree offers different Commands for altering Numbers:

round(Number1,Number2)	Round Number1 to Number2 digits
roundownd(Number1,Number2)	Round Number1 to Number2 digits
random()	Generates a uniform random Number between 0 and 1
min(number1, number2)	gives the minimum of number1 and number2
mod(number1, number2)	number1 mod number2
power(number1, number 2)	$number1^{number2}$
...	

Like in Mathematics, the evaluation follows a certaing order. Everything in Parantheses is evaluated first.

$$int\_random\_20 = rounddown(random() * 21, 1);$$

On random Variables

zTree does not offer any generation of random Numbers despite a uniform draw in [0,1], random standard gaussian and random standard poisson. It is possible to generate a draw from mostly all other continuous distributions F(x) by calculating

$$x = F^{-1}(random())$$

Example: $x = ln(1 - random())/(-lambda)$

String Commands

There are also special commands to alter strings

mid(string, number1, number2)	Copy everything from strings, starting at Pos number1 for number2 letters
pos(string, string2, number)	Position of String2 in String1, starting at number.
len(string)	Returns the length of a string
stringtonumber(s)	Converts string s to a number
string1 + string2	appends string2 to string1

In contrary to arrays, the first letter of a string is at position 1. The 0 as a value is saved for special purposes, for example

$$pos(string, string2, number) == 0 \Leftrightarrow string2 \ not \ found$$

Boolean commands

Booleans containg the values True or False and implement logic. You can also calculate them with

x == y	Checks whether x equals y
x != y	Checks whether x does not equal y
x >= y, x > y, x<= y, x < y	Checks for bigger/smaller
x & y	logical and
x | y	logical or

Control

Control Structures are important for programming, as they implement dynamics. There are two important Constructs

Case Destinctions It is necessary to do different things in different Situtations. We can implement this with the commands **if** , **elsif** and **else**

Repititions If we want to repeat something certain times or until something happens we can implement loops with **while** and **for**.

if Clauses

The if clause is structured as follows

```
if (BOOLEAN1)
{
    Code that shall be executed if BOOLEAN1 = True
}
elseif (BOOLEAN2)
{
    Code that shall be executed if BOOLEAN1 = False and BOOLEAN2 = True
}
else
{
    Code that shall be executed if BOOLEAN1 = False and BOOLEAN2 = False
}
```

while

The while loop executes a program as long as a condition holds

```
while (CONDITION)
{
    Code that shall be executed if CONDITION = True
    Change CONDITION
}
```

Normally the Condition is implemented as a Boolean Comparison

{ condition_variable > 5}

for

The for loop executes a program a certain number of times

```
for (variable, starting_number, end_number)
{
    Code that shall be executed.
}
```

The variable is incremented in each step, and its value can be accessed in the inner code.

zTree Features

Despite the general Programming, zTree offers some internal commands and features:

LeaveStage = 1;	Forces a Subject to Leave a Stage
Display Condition	Each Frame has a Field Display Condition. If the comparision results to FALSE the Element is not shown.
Checkers	A checker is a small Programm for Buttons. Only if it evaluates to TRUE the Button will be exectued.

Task

The Iban is structured as follows:

XXCC bbbb bbbb kkkk kkkk kk

Implement a Stage asking for the IBAN as a string. Do following calculations.

- create a boolean variable "bool_germanänd set it to 1 if the nation-code xx mathes DE
- extract CC and save it as an own number variable ïnt_checksum"
- Try to delete all Whitespaces in the Rest of the IBAN
  - Using a while loop
  - Copying 4 letters after each occurence of a Space.
  - Append these 4 letters to an other string

Any Questions???