**What to do next?**

**The lessons fort the second session on Pages.py – finish the issues**

**and then JavaScript, Django: tricks likes changing buttons and other information?**

**Preliminary programme:**

**Day 1:**

**09:00-10:30 Welcome: Lesson 0: html**

**11:00-12:30 Lesson 1: Basics: pages.py & JavaScript**

**13:30-15:00 Lesson 2: Basics: python: models.py and troubleshooting**

**15:30-17:00 Lesson 3: Basics: otree: getting started + feedback for the first day**

**Day 2:**

**09:00-10:30 Lesson: Bigger picture**

**11:00-12:30 Lesson: Advanced problems**

**13:30-15:00 Lesson: Interactive designs**

**15:30- Last session: online resources, getting started with own projects**

1. **Welcome:**

* Introductions: Who are you, what are your coding interests?
* Double checking everything works (Python interpreter, oTree, PyCharm)
* How to start up the server (working with the cmd/Terminal window)
* Server information: Data, monitor, links, payments, (admin, rooms?)
* Open the workshop folder/directory in PyCharm
* Overall structure of the course
* Zero is the first number in Python, One in the rest of the languages.

**Log for 90 minutes**

**0-10min Introductions and Goals**

* Hey, I’m…
* Who are you? Name, department, why you are here? How you coded before? Are you thinking of running an experiment

**10-20min Basic Starting**

* How far have you gotten with oTree? Everyone has installed it? Have you tried out to launch it? (2min)
* Download workshop folder, place it in an easy place (Your user folder!)
* Open PyCharm, command window (Terminal in Mac) – start the server, access the programme – collect the commands on a slide:
* **cd workshop** & **otree devserver**

**20-30min What can oTree offer**

* Demonstrate on your computer :
* How to start a session
* How to see the data
* How to monitor what page each participant is on
* How to see the payments

**30-40min What do we do on the course**

* Plan of the course:

First day

1. Web site, html, JavaScript, CSS
2. Python: the basic structure
3. Putting them together

Second day

1. Bigger picture
2. Advanced coding
3. Real time interaction
4. Q&A

**40-70min Exercise on HTML5**

**HTML is made out of elements in <>. Some are long, and require an ending tag </>, some are short, that just appear “within the first tag”. Most elements you can then modify within the brackets, for example, change colour, size or margins.**

**Django is this language made by web site developers that simplifies the HTML. Django code is within {% %} brackets.**

**The exercises will demonstrate with some examples how each of these languages work.**

* done, welcome/templates/welcome/MyPage.html
* w3 schools is the resource for html
* otree web page is a good place to start with Django commands rather than the Django site
* Google!

**70-90min Recap**

* Go through the answers
* Any problems that occurred?
* Puzzles?
* What did we learn this lesson – short recap
* Learn more from w3 schools / Google

1. **Lesson : Basic structure of oTree 1: web pages**

* Basic Django commands
* Basic HTML5, CSS, JavaScript
* Buttons, different content by Django ifs, JavaScript on clicks, inputs, forms
* pages.py: Manage (Add, Remove, Control) pages in the sequence
* “Inspect” html code on your Chrome Browser
* (Have a ready made models.py, tell the names in the html file.)

**Log for 90 minutes**

**0-10min To start**

* Recap, set goals for this session: we learned how to construct web pages last session, now we learn how to manage the sequence of pages + JavaScript
* Varying the static content, varying dynamic content with JavaScript
* What are the basic good practice coding principles with these coding languages? No repetition, if it can be avoided
* A lot of things have already been made, copy paste and modify.

**10-20min Topic**

* Pages.py file controls the sequence of pages.
* Defines each page as Python class and then defines the order of these pages
* The page classes inherit properties from generic oTree pages, set up already – easy to demonstrate for example with wait pages (you do not html code them but a lot of information appears anyways)
* We do not do a Python tutorial per se: mathematical expressions, lists, dictionaries, functions (return), classes and their objects, if and for, modules (f.ex. random number generators) (imported)
* First number is 0.
* Indentations matter a lot! (also good practice for the other languages, not necessary for example with html or Django)

**20-40min Exercises**

* Do modifications to the page.py file
* Formfields (keep the models.py ready)
* Parameters for templates
* Ask friend for help, compare code

**40-60min Recap + Topic extensions**

* **Go through the correct answers (10 mins) – crucial for the next exercises**
* What is self? What is a class?
* Useful functions for the wait page (maybe just mention as possibility):
* wait\_for\_all\_groups = True
* There are a few problems with the app: the amount of reciprocity is not limited to the maximum, like is the case with the trust decision. Code coding practice: do not allow for mistakes.
* One solution: inspect code, copy paste element, add a max by hand
* Another useful language that depends on indentations: JavaScript, for the dynamic content of the pages. Maybe we should flip these around… but pages.py is more important.

**60-80min JavaScript**

* Change the color of buttons

**80-90min Recap**

* Difference between dynamic and static code: when to use JS?
* Recognizing different types of python code: helps with syntax and googling for help

1. **Lesson: Basic structure of oTree 2: python**

* Look at the ready made aspects in the models.py file
* Adjust models
* Add a form/data point that you then collect on the page, fx. change answer scales
* Players, Groups, Subsession
* Set payoffs-fuctions

**2. Log for Lesson 2, 90 minutes, (After lunch lesson)**

**0-10min**

* What have we done so far: pages, html, what the users see and how that is constructed
* The experimenter’s side: what data to collect: models.py
* Constants, session parameters (settings.py), subsession, group, player, participant
* Where to store stuff: depends? Variance (constant, treatment, random), access (individual, group, everybody), relevance (ready made, payment related etc.)
* pk: primary number of the dataset, “row number”
* Python, Classes… some basic language understanding of python, ready built functions to use heritage (existed already with the pages and waiting pages)

**10-20min Basic models file**

* Go through the model of the trust game, explain some features
* Defines the basic structure of the experiment: how many players, types, groups, rounds
* Defines what variables are for groups and players
* Defines payoff functions
* Can do a lot of other things, constrain variables etc.

**20-40min Exercises**

* Create a basic model (html s half ready)
* Constants, Groups, Players, exceptions, requirements such as max and min.
* Payoff function
* Probably a lot of troubleshoots!

**40-60 min Recap**

* Go through the exercises

**60-80 min Exercise TROUBLESHOOTING?**

**80-90 min Recap**

* Models: defines your data structures
* Next we are going to modify models, pages, html templates all together: make an app from scratch

1. Lesson: Basic structure of oTree 3: putting web pages and python core together

* Modify models.py, pages, and html templates all together
* Add forms, pages
* Build from scratch by highlight the potential to copypaste from the earlier examples!!!
* Make a feedback questionnaire – send the code – and what did you learn today?

**2. Log for Lesson 3, 90 minutes or less, (Last session of the day)**

**0-10min**

* What have we done so far: modify web pages, modify the sequence of pages, modify the models and data
* Next we look even deeper into the otree: how to set up a new app, what are the necessary functions, why their names need to be the same etc
* Constructing from scratch a feedback survey – I want you to make the structure, a lot of the material is ready but has to just be put into the right place
* I want to do it, and then send me your answers by email.

**10-20min: settings.py**

* You modify things here that are common to all your apps and settings, such as, language, currency, security settings such as passwords etc.
* I used the room feature and labels when running experiments in a lab as the seats were set
* You also select the apps to display and how many participants there are per app and their sequence

**20-30min: Create a new app**

* Notice the amount of files created: this is the basic structure of oTree, but by now they should look familiar
* Some of the empty files are very important, do not get rid of any of them. They define the files as objects and thus make it possible for the python interpreter to understand their relations

**30-50min: Make a survey**

* You find instructions for creating the survey in an extra file called lesson3

1. Generate a new app:

* Write on the command window:

otree startapp survey

1. Add the new app as an entry to the settings.py

* Copy one of the existing ones and modify the information such that survey is the app used. Feel free to adjust the names. At least one demo participant needed.

1. Modify the models.py to create the following variables:

* Html is for creating

Modify MyPage.html to include the following feedback questionnaire:

Modify pages.py under MyPage to include the following form fields

Modify models.py to include the following models

Answer the feedback questionnaire:

Then go to the otree page, download your data and send it by email to Essi.

What feedback do I want?

Did you succeed with the tasks? Did you succeed with some of the learning goals? Have some scale!

Appropriate goals:

* Tell the different elements of coding languages apart (HTML5, JavaScript, JSON, Python, Django, oTree) and know when to use which;
* Understand the overall structure, the purpose of different files and how they come together;
* Learn the basic syntax of the different elements and languages and to be able to search for solutions online;
* Construct basic if-sentences, for-loops and static and dynamic web pages
* Decode error messages and find potential solutions: to *inspect* code on the browser, to be able to interpret terminal output, and oTree feedback;

Were the tasks: too long, too difficult, too complicated, unclear, too buggy, with too many mistakes?

If you can remember a task, which one was it, where did you get stuck?

What did you find most useful in the day:

Learning basic languages: html, Django, JavaScript, Python

Learning coding debugging: console log (not yet) / inspection of code / oTree feedback

Learning oTree, it’s structure, how you can modify it

Learning about the online resources/how to use them

Goals left for the second day:

* Break down an experimental design into small, tangible coding problems;
* Decode error messages and find potential solutions: to *inspect* code on the browser, to be able to interpret terminal output, and oTree feedback;
* Follow general guidelines for ‘good coding practices’.

How do the files talk with each other

\_\_init\_\_(self,…)

* Import commands –later, radiogrid and adding packages, later!

Day 2:

1. Lesson: Bigger picture

* refresher for half the lesson!
* settings.py
* session variables
* html page, pages, models – need to correspond
* adding apps, adding projects
* Sequencing apps
* Revise previous topics

1. Lesson: Advanced problems

* Groups, ready made functions, etc.
* Waiting pages
* Interpreting the error codes

1. Lesson: Interactive applications

* Simple extensions through Channels: consumers.py and routing.py
* Make a chat box
* Inspect code -console, print on the command window, error feedback
* Modify an auction
* Django custom models

1. Last lesson:

* Online resources
* To get started with your project: what do you need to do?
* Split the projects in small goals, spread across groups, send codes in advance and then, we will investigate them in the follow-up session
* Feedback

Quiz, pictures, people give preferences and then they are allocated to roles by those preferences, strict quotas. Subgroup structure for, vote for the politician.

Twitter actions in the programme