

oTree: A Crash Course

罗干松

github:<https://github.com/MarvinLuoGS>

This Version:20230701

oTree: A Crash Course

一、python及oTree基础概念与安装

- (一) python安装及相关的编程概念
- (二) oTree的安装及相关材料

二、oTree基础

- (一) 编写oTree用到的语言
- (二) oTree文件与程序架构
- (三) oTree运行逻辑

三、程序编写与调试

- (一) 简单问卷
- (二) 公共品博弈
- (三) 信任博弈
- (四) 程序测试
- (五) 对程序的扩展说明
 - 1. 处理点数和收益
 - 2. Timeout的设置
 - 3. round和app之间的数据传递
 - 4. group和role的设置

四、正式实验操作

- (一) 服务器架设
- (二) 程序准备
- (三) 启动实验
- (四) 问题处理

五、数据保存与基础清洗

- (一) 数据管理
- (二) 数据下载与文件类型
- (三) 基础的数据清洗: otree2stata.py和otree2data.py
 - 1.otree2stata的使用
 - 2.otree2data的使用

目标

1. 了解python和oTree以及它们背后的一些编程概念，便于更好读懂官方文档
2. 学会基础的oTree编程，包括后端的python以及前端可能涉及的HTML+CSS+JavaScript语言
3. 基础的oTree实验运行及管理

文件说明

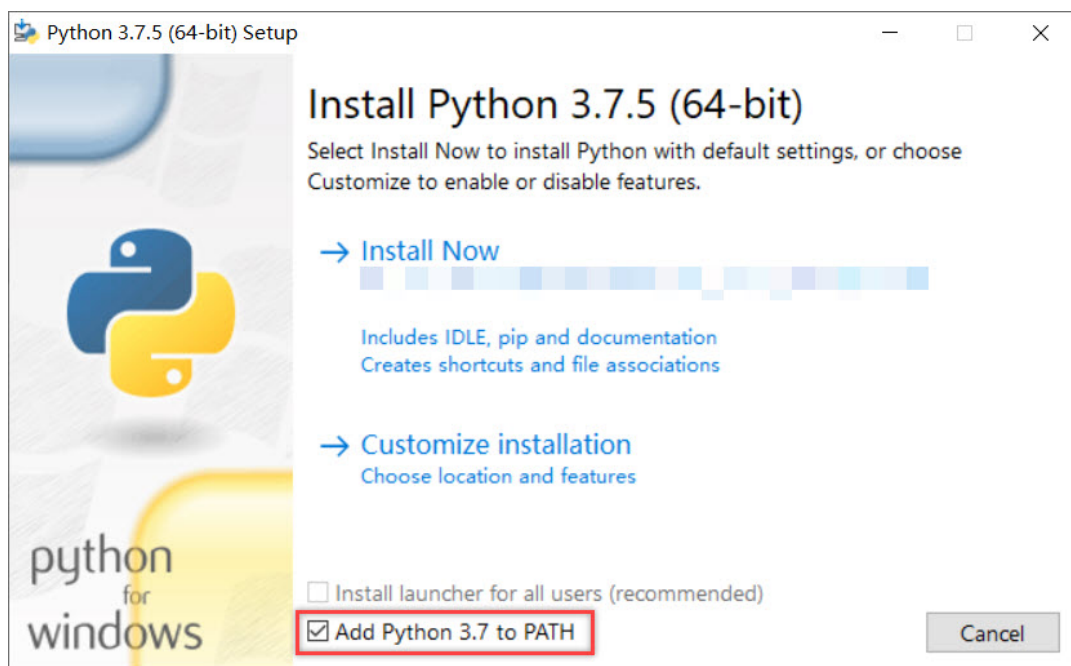
- doc: Linux系统的服务器架设方式和中文翻译的旧版本的官方文档 (md格式)
- firstexp和sample_exp: oTree程序, firstexp和下面的讲解对应, sample_exp是官方的简单示例程序
- pypro: otree2stata和otree2data以及数据CSV

一、python及oTree基础概念与安装

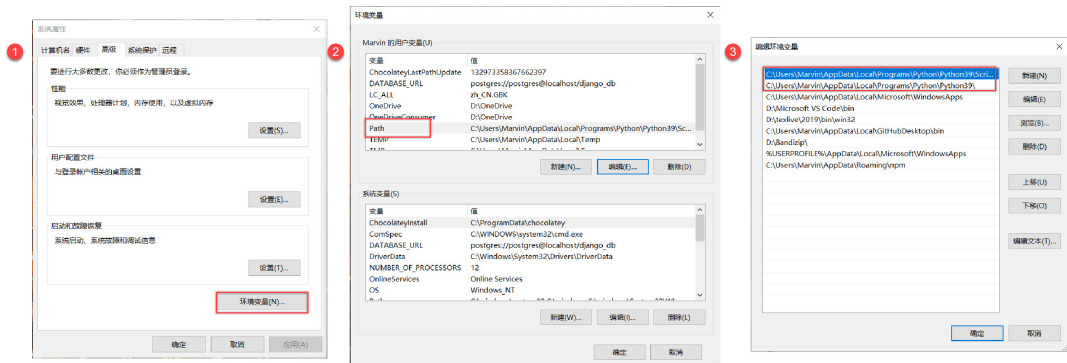
oTree在更新到5版本之后，作者推荐使用的编写方式是由oTree官方自己提供的oTree studio平台，通过点击拖拽等方式写代码。因为个人没怎么用过oTree studio，所以不介绍oTree studio的使用。个人感觉虽然oTree studio比较简单但可能损失了很多灵活性，而在自己电脑上一步步写出来虽然有一点难度但能有更好的灵活性，从学习oTree的角度讲也能更有体会更深入。**本教程基于Win10系统进行，Mac OS在涉及安装、服务器架设或操作方面有差异，程序编写本身无明显差异。**

(一) python安装及相关的编程概念

1. python解释器 (Interpreter)：python是一种高级语言，计算机要理解高级语言必须要将其转换为0-1的机器语言，python解释器就是做这么一个事情。所谓安装python就是安装python解释器（python3.10/3.9/3.8是不同版本的解释器），解释器是将python代码解析/翻译为二进制机器语言的工具
 - 安装：python官网即可下载，建议下载stable release的版本，且是windows installer格式的（目前最新的版本是3.10，保证版本在3.7以上），安装的时候记得勾选Add python to PATH（如下图例子），直接选择install now即可（选择customize installation修改安装路径的话记得所有安装内容都要勾选）



注意：如果之前已经安装过python，需要确定一下系统的环境变量里面是否已经有python的路径了，可以在系统设置里面搜索“环境变量”打开（如下图），并参考如下步骤，正确添加进环境变量应该有两条是python的路径，比较建议只保留一个版本的python的路径。环境变量下的用户变量只对当前用户有效，系统变量对所有用户有效，一般自己用的电脑上只有一个用户，所以只要用户变量下面的Path有python路径即可，安装时给所有用户都安装的选项不必要



2. 编辑器 (Text Editor) : 写文档要用word、做幻灯片要用PPT, 写代码也要用编辑器, 极端来说文本文档 (txt) 也可以用作编辑器。但一个好的编辑器无疑能提高效率。用于下面要介绍的IDE/类IDE工具里面同样集成了编辑器, 因此这里不单独下载编辑器。

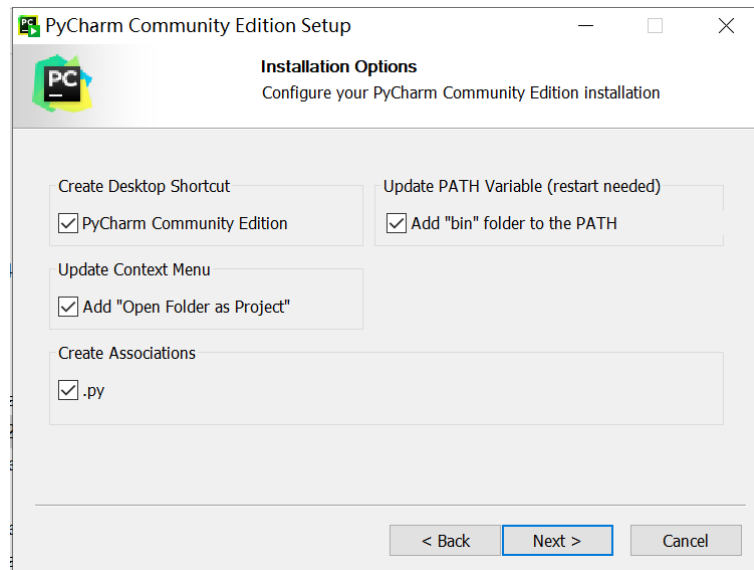
3. 集成开发环境 (IDE, Integrated Development Environment) : 集成集成代码编辑器、编译器、调试器和图形用户界面等功能, 可以提高编写代码的效率, python自带IDLE, 另外有Jupyter Notebook、Pycharm、Spyder (主要针对python)、VSCode (针对多种语言), 这里比较推荐VSCode或Pycharm

- 安装: VSCode和Pycharm都可以直接在各自的官网下载, 建议下载VSCode, 下面的讲解也是基于VSCode:

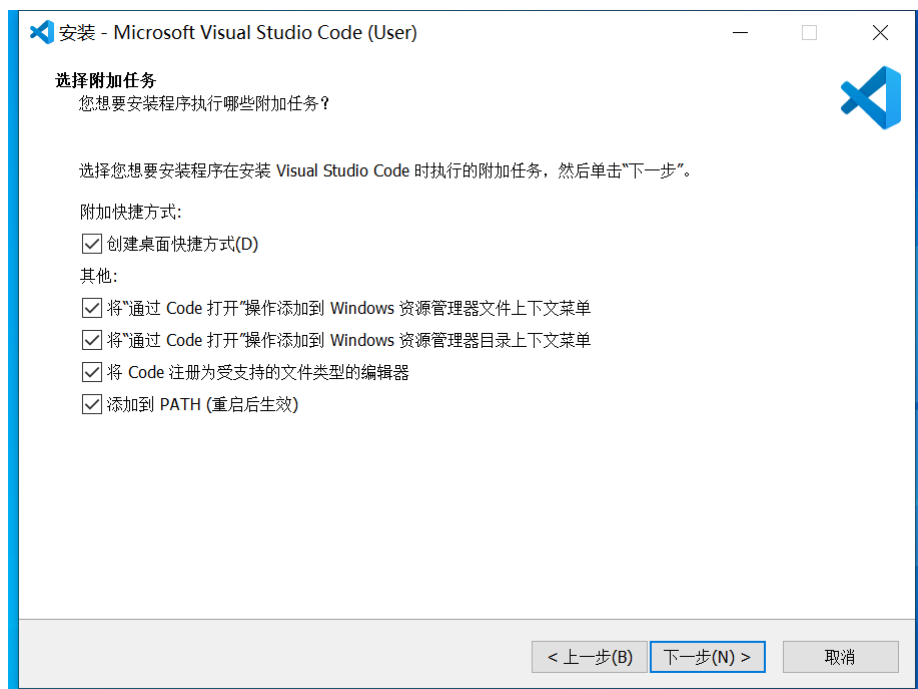
- VSCode: <https://code.visualstudio.com/>

- Pycharm: <https://www.jetbrains.com/pycharm/>, 下载免费的社区版 (community) 即可

- Pycharm是早期的oTree文档中作为例子介绍的，这是因为早期的oTree实际上依赖于一个非常流行的开源web应用框架Django，而Pycharm本身对使用Django进行开发比较好（对HTML+CSS+JavaScript等语言的支持），所以早期文档会以Pycharm为例子介绍，但是oTree更新到5版本之后已经不再依赖Django，而是转向使用自己的框架oTree Lite，一种和Django有点像但又不是Django的框架。安装pycharm时勾选下面这些内容：



- 个人习惯使用VSCode，这也是本教程中使用的工具，可以安装很多针对不同语言的插件，但其实对使用oTree来说这些插件都不是必须的。安装VSCode时勾选下面这些内容：



- 安装插件的方法是在VSCode的界面左边菜单上选择“Extensions”（扩展），然后搜索相关的扩展进行安装，下面推荐几个个人使用比较好的插件：
 - **Python**：必装，微软官方提供的VSCode的python支持插件，安装的时候认准发布者是微软官方，安装这个插件也会自动安装Pylance、Jupyter等一系列python相关插件
 - **Django、Django Template**：虽然最新的oTree不是基于Django，但仍有类似之处，可以安装用于辅助
 - **Chinese (Simplified) Language Pack**：微软官方提供的汉化包，一般打开Code自动就会提示要不要安装

- VSCode本身的设置：点击左下角齿轮→setting→新出现的页面上搜索“BracketPairColorization”并勾选（这个功能可以使成对的括号有相同的颜色）、搜索“Linked Editing”并勾选（这个功能可以使得在修改成对的标签时只要修改一个另一个也自动替换）
 - 关于VSCode的详细的介绍请参考官方文档或B站上的一些教程：<https://code.visualstudio.com/docs>
4. 包管理工具：python安装时自带包安装和管理器pip，这是python的一个可以在python的包仓库Python Package Index(PyPI)里面下载各种python的包的工具，oTree安装将使用到这个工具。这个工具的作用有点类似于stata下载包时用的ssc install命令。pip在下载包的时候会递归地找到正在下载的包所依赖的其他包并一起下载
5. 关于python及相关安装的内容，还可以进一步参考的一些教程：
- 最权威的资料：python官方文档：<https://docs.python.org/3/>
 - [Python+Anaconda+PyCharm的安装和基本使用【适合完全零基础】不只是教你如何安装，还告诉你为什么这么安装](#)
 - [四十分钟Python快速入门 | 无废话且清晰流畅 | 手敲键盘 | 停止东奔西走赶快入坑吧~](#)
 - github上7.1k stars的python语言基础50课：<https://github.com/jackfrued/Python-Core-50-Courses>
 - 密歇根大学教授Charles Severance的入门课程：[【中英字幕】Python for everybody-不可错过的五星推荐python入门课程！](#)
 - 教授的个人网站：<https://online.dr-chuck.com/>
 - oTree的官方文档提供了对python的很简单的介绍
 - 本次只会介绍最基础的python内容，其他的python应用（爬虫、数据分析以及Anaconda等等）在本次培训中不会涉及，想进一步学习的可以去找python的其他书籍或课程，入门可参考：<https://www.runoob.com/python3/python3-tutorial.html>

(二) oTree的安装及相关材料

oTree是一个开源的Python框架，用于在实验经济学中设计、开发和进行在线实验。它提供了一个简单的方式来创建各种类型的实验，包括博弈论、行为经济学和社会科学实验。

注：直接采用下面的安装方法是全局安装了oTree，如果想要避免python不同的包之间相互干扰，可以先建立虚拟环境再安装，关于什么是虚拟环境以及虚拟环境如何使用，参考以下两个说明：

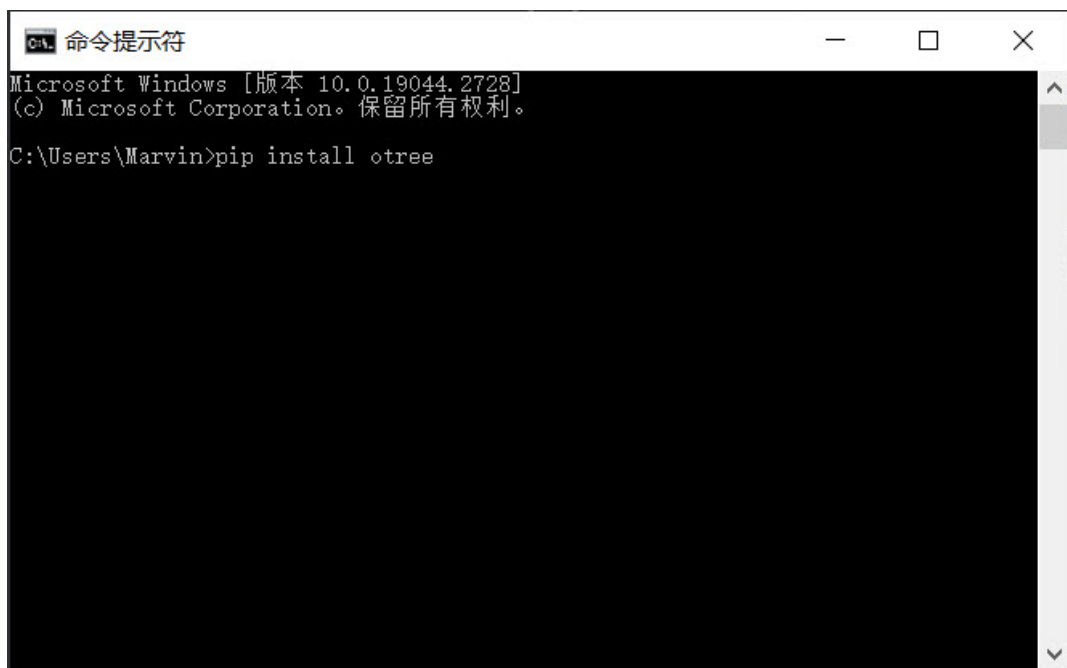
- [官方文档对虚拟环境的说明](#)
- [freeCodeCamp对虚拟环境的说明](#)

1. oTree的安装：根据oTree官方文档的说明，在命令行/命令提示符/cmd/终端中输入命令进行安装，首先win+R，输入cmd打开命令提示符，在打开的命令提示符窗口中输入以下命令：

```
1 | pip install otree
```

等待安装完成即可，另外一些常用命令如下：

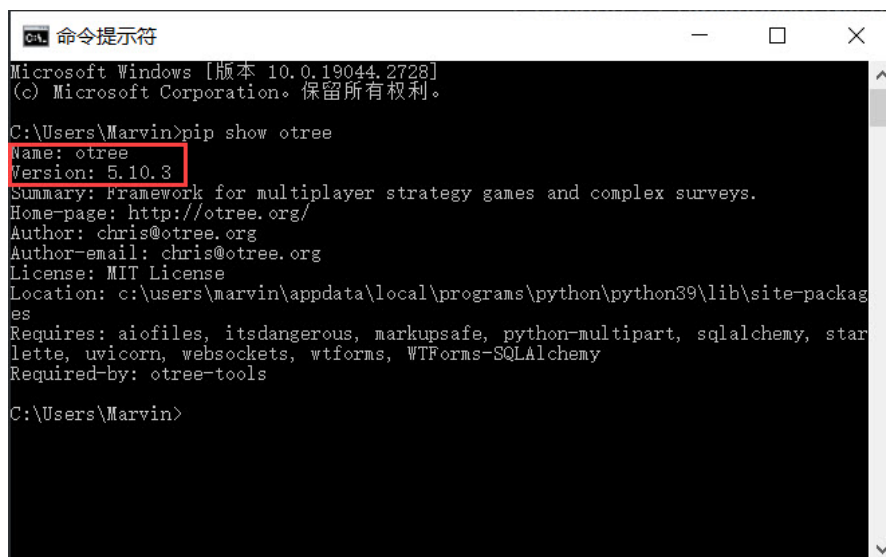
```
1 | pip install -u otree #升级otree
2 | pip uninstall #卸载某个包
3 | pip help #列出pip相关命令
```



```
命令提示符
Microsoft Windows [版本 10.0.19044.2728]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Marvin>pip install otree
```

- Path环境变量里面设定python路径的原因在此，这样将直接给该python安装上oTree包
- 安装完成后，同样在命令提示符窗口里面输入命令“pip show otree”可以列出otree的信息（最新的版本是5.10.3），命令“pip list”可以列出所有已安装的python包及相应版本



```
命令提示符
Microsoft Windows [版本 10.0.19044.2728]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Marvin>pip show otree
Name: otree
Version: 5.10.3
Summary: Framework for multiplayer strategy games and complex surveys.
Home-page: http://otree.org/
Author: chris@otree.org
Author-email: chris@otree.org
License: MIT License
Location: c:\users\marvin\appdata\local\programs\python\python39\lib\site-packages
Requires: aiofiles, itsdangerous, markupsafe, python-multipart, sqlalchemy, starlette, uvicorn, websockets, wtforms, WTForms-SQLAlchemy
Required-by: otree-tools

C:\Users\Marvin>
```

2. 学习和使用oTree的相关资料网站：

- **oTree官方文档：** <https://otree.readthedocs.io/en/latest/index.html>
 - oTree 的官方文档，提供了对oTree的详细介绍和说明，在实际写程序的时候很多情况下需要开着官方文档搜索相关内容
 - 有中文版的文档，还是建议看英文的
 - 官方文档中对入门来说较为重要的章节：**Conceptual Overview、Models、Pages、Templates、Forms**
- **oTree Hub：** <https://www.otreehub.com/>
 - 官方提供支持的网站，包含有论坛、用户公开的程序代码、Chris整理的一些重要的示例代码等内容，注册使用
 - **Forum：** 官方论坛，可以post自己遇到的问题寻求帮助，推荐在这个官方论坛上问问题，也可以订阅邮件更新，Forum页面拉到最底下勾选通过邮件接收更新即可。（原来的讨论组是Google Groups上建立的：<https://groups.google.com/g/otree?pli=1>，也可以在上面搜索问题）

- **Public projects/Featured**: 用户分享的自己的程序代码，提供了可以运行测试的 Demo、直接在网页上打开浏览的源码 (Browse source code) 以及代码下载 (Download)，Featured 里面则是由 Chris 本人整理过的一些非常有用的代码片段或小功能，其中 otree-snippets 这个内容是 Chris 本人亲自写的，提供了很多使用例子。了解了 oTree 的基础内容之后，后续的学习和应用很多都是通过看和理解别人的代码完成的
- oTree collection: <https://ckgk.de/otree.htm>
 - Christian 整理的一些重要的 oTree 程序、实现某个功能的代码片段和工具。和 hub 里面的相比较主要差异是这些程序很多都是完整的高度集成的实验程序，一个程序里包含了许多不同的实验任务，程序大部分放在 Git Hub 上，这也说明其实可以去 Git Hub 上搜 oTree，可以找到别人分享的代码。
 - 优点是提供了完整的实验程序，缺点是很多程序的写法是旧版本的 otree 代码，有一些功能在新版本中可能不支持，另外就是不那么容易看懂这些代码，需要比较多的 python 语言和前端语言的积累
- oTree 官方的 Github 仓库: <https://github.com/oTree-org?tab=repositories>
 - 有官方的示例程序和文档
- oTree 的论文，需要引用: Chen, D. L., Schonger, M., & Wickens, C. (2016). oTree—An open-source platform for laboratory, online, and field experiments. Journal of Behavioral and Experimental Finance, 9, 88-97.
- (参考了解) oTree 在 oTree 5 之前用的框架是 Django，Django 是一个由 python 编写的开源的 web 应用框架，由于 Django 框架本身已经提供或者说带有一些成熟的已经编写好的类或组件 (比如表单、上传文件、滑块、评分等功能要用到的组件)，因此在这个框架基础之上进行开发无需重复造轮子，可以直接使用这些类或组件。简单来说，Django 这个框架是用于服务器后端的，承担了数据库管理、数据交互等功能，完成这些功能使用的编程语言是 python，在前端 (html 文件) 部分 Django 也有自己的 html 模板写法。现在 oTree 在构建前端页面的时候使用的框架是流行的 Bootstrap。

二、oTree基础

这一节将介绍oTree编程涉及的基础概念，官方文档中对这些概念都进行了全面的介绍，这里只针对部分基础的和常用的进行介绍，深入了解这些概念还需要阅读官方文档。

(一) 编写oTree用到的语言

虽然前面介绍了python及oTree的安装，但编写otree程序并不只是涉及python一种语言，还涉及到三种语言，分别是HTML（超文本标记语言）、CSS（层叠样式表）、JavaScript。其中python用于后端，后三种语言用于前端。下面对这些语言及他们在oTree程序中承担的任务进行简单的介绍

- python语言：用于完成后端变量定义、数据交互、计算等功能的语言。与z-Tree不同的是，oTree有明确的编程思想，即面向对象编程思想。
 - 关于什么是面向对象编程，可以参考这个简短的介绍：[【8分钟搞懂面向对象编程 | 面向过程 vs 面向对象 | OOP | 封装 继承 多态】](#)（注：这个UP主也有一个python入门课程）
 - 简单来说，面向对象是将某个任务涉及到的不同事物、属性、方法进行抽象分类，与之相对的是面向过程编程，即将完成某个任务划分成不同步骤，编写完成每一步骤的代码。
 - 就写oTree而言，**对python语言的深入了解不是必须的**，但熟悉python肯定有好处，有必要了解的基础内容有以下一些：**数值类型和数学计算、逻辑运算（与、或、非）、数据结构（元组、列表、字典、集合）、控制语句（循环和条件）、函数和模块、面向对象和类、字符串**等。python的相关教程的基础部分一般都会包含以上这些内容。
- HTML：一种标记语言而非编程语言，通过一套标记标签（markup tag）用于描述网页内容，即哪个地方是标题、哪个地方是文本内容、哪个地方插入图片等
 - 常见常用的一些标签

```
1 <p>
2     段落标签
3 </p>
4 <ul>
5     <li>无序列表</li>
6 </ul>
7 <ol>
8     <li>有序列表</li>
9 </ol>
10 <div>
11     块级元素，可用于组合其他元素的容器，也常用于文档布局。<span>这个则是内联元
    素，用来组合行内元素比如文本</span>
12 </div>
13 <h3>下面是一个带表头的表格</h3>
14 <table>
15     <tr> <!--定义表格的行-->
16         <th>月份</th> <!--定义表格的表头-->
17         <th>销售</th>
18     </tr>
19     <tr>
20         <td>一月</td> <!--定义表格的单元格-->
21         <td>1000</td>
22     </tr>
23     <tr>
24         <td>二月</td> <!--定义表格的单元格-->
25         <td>2000</td>
```



```
26     </tr>
27 </table>
```

- CSS：描述HTML文档样式的语言，说明每个HTML文档中每个元素应该如何显示，比如字体是否加粗、标红，按键的位置、大小等等
- JavaScript：web编程语言，用于说明HTML上每个元素的动态变化、进行输入验证和数据传输等，比如当点击某个按键的时候某个文字变红、在前端验证输入的答案是否正确等
- 前端页面是直接和被试交互的，因此页面的设计很可能会影响到被试的决策，这说明编写过程中必须认真考虑前端页面的呈现。oTree本身提供的前端框架能很大程度上减少对这些语言的使用，简化了前端页面的编写过程，但是这也损失了很多自由度。当需要对前端页面进行一些调整的时候，不可避免地需要使用三门语言。这三种语言深究起来也是一个大坑，所幸上手比较容易，不一定需要精通，只要了解基础的内容，能读懂上面提到的网站中的程序例子即可。
- 实际上，oTree在生成前端页面的时候使用了一个非常流行的前端框架Bootstrap，这个框架提供了许多有用的组件和JS插件，oTree一些自带的按键、输入框等都是基于Bootstrap而来的
- 进一步了解或尝试三门语言，可以参考和查阅如下网站
 - 关于三门语言的速成入门介绍：
 - HTML：[【为初学者准备的：HTML 速成】](#)或[【二十分钟HTML快速入门 | 无废话且清晰流畅 | 手敲键盘 | WEB前端必备语言~】](#)
 - CSS：[【为初学者准备的：CSS 速成】](#)
 - JavaScript：[【为初学者准备的：JavaScript 速成】](#)或[【四十分钟JavaScript快速入门 | 无废话且清晰流畅 | 手敲键盘 | WEB前端必备程序语言~】](#)
 - 专门介绍web编程的教程网站，涵盖各种语言，提供了直接可运行可修改的示例：<https://www.w3school.com.cn/index.html>（中文）<https://www.w3schools.com/default.asp>（英文）
 - 菜鸟教程，同样提供了直接可运行修改的示例：<https://www.runoob.com/>
 - Bootstrap框架的官方文档：<https://getbootstrap.com/docs/5.0/getting-started/introduction/>

(二) oTree文件与程序架构

1. 文件

- 为了了解oTree的文件类型和程序架构，我们先创建一个新的oTree程序文件夹
- 选定储存的位置（假设储存路径为E盘下的EXP文件夹），点击VSCode左上角File菜单→Open Folder打开EXP文件夹，在VSCode中打开Terminal（终端，可用快捷键Ctrl+~或点击界面左下角的图标打开，其他快捷键可参考VSCode的官方文档），此时终端中显示的路径应为"E:\EXP"
- 输入命令创建程序文件夹：

```
1 otree startproject firstexp
```

- 在创建文件夹的时候，oTree会问是否“Included sample games?”，输入y并按回车将尝试联网下载样例程序在文件夹中，这些样例程序也可以作为学习的参考，但是由于样例程序可能存放在github或者别的国外平台上，所以由于某些网络原因可能无法成功下载，需要多尝试几次
- 一般创建程序文件夹不必包括样例程序
- 一个完整的实验的程序（比如包括不同实验任务和实验后问卷在内）叫做project

- 按照提示，输入命令将工作路径改至程序文件夹内：

```
1 | cd firstexp
```

- 新建的程序文件夹没有具体的实验程序，新建实验程序需要使用命令：

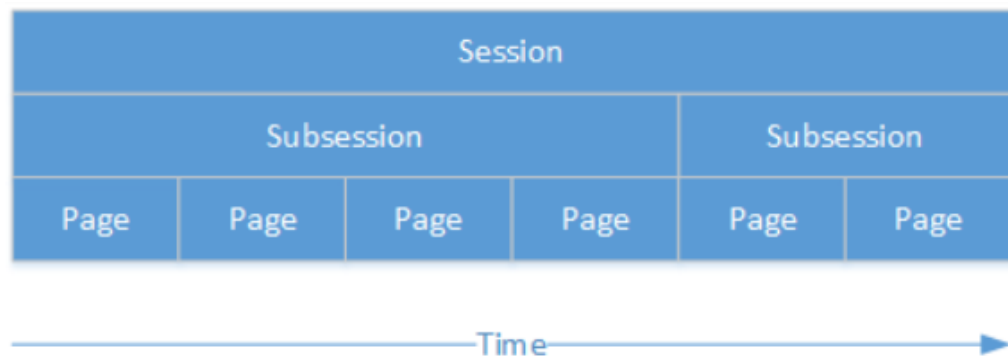
```
1 | otree startapp firstapp
```

创建新的实验程序文件夹，在oTree中具体某个实验任务（比如偏好测试、问卷）的程序叫做app（有点类似于ztt）

- 在app文件夹里面，有一个名为__init__.py的文件和MyPage.html、Results.html两个文件，其中py文件用于实验的后端部分，html文件用于实验的前端部分

2. 程序架构

- 重要概念：Session、Subsession、Page、Group、Player、Participant
 - Session在oTree中指的是一场活动（event），这个活动中有许多参加者参加进来完成一系列的任务或游戏，官方文档提供的例子：A number of participants will come to the lab and play a public goods game, followed by a questionnaire. Participants get paid EUR 10.00 for showing up, plus their earnings from the games.
 - Subsession是Session下的一个概念，多个Subsession组成一个Session，一个Subsession可以理解为一个实验任务、实验环节等，比如在上面的例子中，一个公共品博弈是一个Subsession，后续的问卷调查也是一个Subsession，这两个共同构成了一个Session
 - 每个Session下又由许多个Page组成，比如公共品博弈中有规则介绍、决策输入、结果报告等page，图示例子参考下图：



- 如果一个任务重复多次，那么每一轮都看作一个Subsession，比如公共品博弈任务重复10次就是10个subsession
- Subsession又可以进一步划分为不同的Group，每个Group里面包含Player。比如在公共品博弈中，20个人平分4个Group，每个Group5个Player，每个Player都会看到一系列的Page并作出决策。不分组的单人决策情景可以理解为一人一组
- Participant意味参加者、被试，与之相对的Player可以理解为角色、身份，在不同的Session和Subsession之间，同一个Participant可以是不同的Player，比如在第一轮中作为委托人，在第二轮中作为代理人
- 层级排序如下：
 - Session（一个Session包含多个Subsession，Participant参加Session）
 - Subsession（一个Subsession包含多个Group）
 - Group（一个Group包含多个Player）
 - Player（一个Player可以看到多个Page）

- Page

- 从低层级的对象向上获取高层级的对象的字段前需要使用如下写法：
 - `player.participant/player.group/player.subsession/player.session/group.subsession/group.session/subsession.session`
- `__init__.py`文件的内容可以分为`models`和`pages`两部分（旧版oTree中`models`和`pages`是两个分开的py文件，新版中合并为一个），`models`部分有C类（常数类）、Subsession类、Group类、Player类4个类，`pages`部分中都是Page类

- C类下的字段包括关于实验程序的名字、分组人数、轮次数等实验设置相关的字段，也可以添加一些整场实验中需要用到的不变的参数（常见的例子有投资回报率、惩罚成本等等），规范来说，C类下的变量名字全部字母大写
- 从对象的角度来说，Subsession、Group、Player是不同的类，每个类有不同的字段，从数据的角度来说Subsession、Group、Player可以理解为不同的数据表，每个表的一列就叫做Field，数据就是储存在Field里面，体现为对象的字段

- 例子：

```
1 class Player(BasePlayer):
2     name = models.StringField() #文本
3     age = models.IntegerField() #整数
4     is_student = models.BooleanField() #布尔
```

- 每个类下面都有一些内置的字段和方法，表示这个类可以执行的一些任务或功能，关于这些内置的字段和方法以及设定方式请参考官方文档Models一节下面的介绍
- 从层级来说，Subsession中的数据由同一场次的所有player共享，而Group中的数据则由该组内的Player共享
 - 假如Subsession下有一个字段是`cost`，那么该场实验中所有player调用`player.subsession.cost`都是得到同样的数据，无论这个player属于哪一组
 - 假如Group下有一个字段是`total`加总了所有人的年龄，那么第一组的人调用的`player.group.total`得到的是本组年龄和，第二组的人调用的`player.group.total`得到的是本组的年龄和，两个组调用得到的数据是不一样的
- `pages`部分的Page类定义了参加者将看到的页面，`page_sequence`则定义了这些页面出现的顺序（即定义的先后不影响出现的先后，`page_sequence`才是控制页面先后顺序的），每一个Page都需要有对应的html文件，否则将会报错
 - 比如定义了一个Page名字叫MyPage并加入了`page_sequence`中，则必须存在MyPage.html文件，否则会报错
 - Page类同样有很多的内置方法，比较重要的有`is_displayed()`、`vars_for_template()`、`before_next_page()`等，同样具体的设定方式请参考官方文档Pages一节下面的介绍
 - 有一类特殊的page是WaitPage，是参加者在等待别人决策时可以看到的等待界面，由oTree内置，在需要使用地方加入`page_sequence`即可，不用另外建立html文件
- html文件是实验的前端部分，在页面上展示什么信息、如何展示都需要在html文件中写入
 - oTree基于oTree Lite框架提供了简单的Template语法，可以快速排布所需要的信息和元素，写法：

```
1  {{ block title }}
2      收益情况
3  {{ endblock }}
4
5  {{ block content }}
6      你的收益是{{ player.payoff }}
7      {{ next_button }}
8  {{ endblock }}
```

- 这个Template语言只是用于快速地展示数值，不能完成任何的计算、赋值、修改等任务，这些任务要么交由后端python完成再传向前端，要么使用JavaScript
- 使用Template标记的html文件会先交由oTree系统（服务器端）扫描并转化成相应的HTML标记，再交由浏览器（被试端）根据HTML标记展示相应的网页
- 可以使用HTML、CSS、JavaScript等语言灵活修改，更具体的方法可参考官方文档Templates一节的介绍

(三) oTree运行逻辑

- __init__.py文件是管理后端的文件，定义了整个实验的逻辑、计算任务、数据交互等重要内容，也管理着后端与前端的交互和数据传输以及前端的页面顺序等
- html文件是前端的文件，给参加者展示了实验任务的具体信息、决策任务、实验结果等内容，一般不进行计算任务、变量赋值和数据修改等
- oTree为后端、前端文件的编写以及前后端的交互提供了方便，也为实验数据的储存、参加者的连接等提供了简单的方法，便于快速编写和开展实验
- 参加者通过浏览器获取经oTree处理的页面，在页面上进行决策，相关数据传回服务器并储存在数据库中，这个交互的过程同样由oTree帮助完成

三、程序编写与调试

下面开始具体的实验编写介绍，包括一个简单的调查问卷以及两个常见实验的快速实现，并介绍如何测试程序以及进阶的程序修改。这三个例子可以看作分别对应三种最基础的实验形式：单人决策、多人同时决策、多人序贯决策

(一) 简单问卷

- 我们首先实现一个简单的问卷，了解oTree的数据类型、前端的输入方式与显示方式等基础内容
- 首先将工作路径转移至上文中创建的程序文件夹（firstexp），然后输入以下命令：

```
1 otree startapp questionnaire
```

- 在问卷调查中，每个人填写的都是自己的情况，因此只需要在Player类下添加字段即可。首先，如果想要参加者填写就读的学校信息，可以按如下方式添加字段：

```
1 class Player(BasePlayer):
2     school = models.StringField(
3         label = '您就读的学校是',
4     )
```

这里的school是字段名，StringField表明这个字段记录的是字符型数据，label属性则是参加者在填写问卷时可以看到题目引导标签

- 第二个问题我们想知道参加者的年龄，可以这样添加字段：

```
1 age = models.IntegerField(
2     min = 0,
3     max = 99,
4     label = '您的年龄是',
5 )
```

IntegerField表明age记录的是整数型的数据，这里进一步设定了该字段的最小值和最大值，当参加者填写的数据不在允许范围中时将无法提交并出现错误提示。这是第一种对输入进行验证给出错误提示的方法，这种验证方法不经过服务器。

- 接下来我们还想知道参加者的性别、年级以及是否主修经济类专业，与前面的问题不同的是，这些问题的答案可以从有限的选项中选择，因此我们可以事先给定一些选项，让参加者从选项中选择，这里使用了三种不同的常见选项展示方式：

```
1     #dropdown menu
2     gender = models.IntegerField(
3         choices = [
4             [0, '女'],
5             [1, '男'],
6         ],
7         label = '您的性别是',
8     )
9     #horizontal radio buttons
10    grade = models.IntegerField(
11        choices = [
12            [1, '大一'],
```

```

13         [2, '大二'],
14         [3, '大三'],
15         [4, '大四'],
16         [5, '硕士生'],
17         [6, '博士生'],
18     ],
19     widget = widgets.RadioSelectHorizontal,
20     label = '您的年级'
21 )
22 #vertical radio buttons
23 major = models.BooleanField(
24     choice = [
25         [False, '否'],
26         [True, '是'],
27     ],
28     widget = widgets.RadioSelect,
29     label = '您是否主修经济类专业?',
30 )

```

这三个字段中，都事先设定了choice参数，choice参数是一个列表，列表的元素还是列表，每个列表里面的第一个元素是数据的取值，比如整数型的1、2、3，布尔型的True和False，第二个元素是与这个数据取值绑定的选项文本，也就是参加者可以在前端看到的选项字样。widget参数控制了选项的展示形式，gender字段中没有指定widget参数，默认会以下拉菜单的形式展示；grade字段指定widget为widgets.RadioSelectHorizontal，则六个选项水平排列，而major字段指定widgets.RadioSelect则选项垂直排列

- 在问卷中有一些问题不要求所有人都回答，这个时候可以指定参数blank：

```

1 suggest = models.LongStringField(
2     blank = True,
3     label = '如果您有意见或建议，请填写在下框中',
4 )

```

blank为True表明在填写时该字段可以留空，需要被试填入数据的字段默认是不允许留空的，另外这里使用的LongStringField在输入时会显示更大的文本框

- 另外，有的时候我们也想要对参加者输入的数据进行验证（比如测试性问题检验答案是否正确），上面提到的设定最大最小值参数的方法不适用，我们在这里添加一个attention字段用于举例说明如何进行输入的验证，注意这个验证需要在PAGES部分指定，将在下面讲解

```

1 attention = models.IntegerField(
2     label = '请在下框内填入1234以提交本页'
3 )

```

- 上面完成了后端MODELS部分的设定，下面是有关问卷页面的PAGES部分设定，这一部分内容的主要功能是管理前端页面的逻辑、数据输入以及运算等等。在问卷调查这个例子中，我们需要两个页面，第一个页面是参加者填写问卷，第二个页面是展示参加者刚刚填写的信息。
- 第一个页面完成输入任务，如果参加者需要在页面上填入数据，那么这个页面需要设定form_model和form_fields，以说明当前页面需要填写哪些字段以及这些字段来自哪一类，这里参加者需要填写的是player类下的一些字段：

```

1 class Questionnaire(Page):
2     form_model = 'player'
3     form_fields =
4     ['school', 'age', 'gender', 'grade', 'major', 'suggest', 'attention']
5
6     @staticmethod
7     def error_message(player, values):
8         if values['attention'] != 1234:
9             return '输入有误!'

```

注意这一页面进行了上面提到的输入验证，这个验证方法定义了一个staticmethod（静态方法，这是一个python面向对象编程里面比较专业的概念，只需要知道要这样写就可以，下面的例子中出现的静态方法同理），这个方法的名字叫error_message，定义了这个方法后oTree会在运行的时候自动调用这个方法，因此可以理解为一个内置的方法叫error_message，我们只需要填写需要验证的内容，oTree会根据填写的内容进行验证。

这里需要判断的是attention的输入值是否是1234，values参数（这里出现的values只是这个参数的一个记号，统一改成别的名字也可以）的数据类型是字典，表示执行的时候传入的是一个字典，这个字典包含了在页面上输入的所有字段。根据attention这个字典的key（键）可以获取相应的value（值），并判断是否等于1234，如果不等于，则返回return后面的错误提示，这个错误提示会显示在页面上方。

第一个页面的html文件内容非常简单，由于我们已经设定好了字段以及相应的题目标签作为引导，在html文件中只需要写入如下内容即可：

```

1 {{ block title }}
2     问卷调查
3 {{ endblock }}
4 {{ block content }}
5
6     {{ formfields }}
7
8     {{ next_button }}
9 {{ endblock }}

```

oTree在生成网页时会自行解释花括号中的内容，转换为标准的html文件供浏览器显示。其中{{ formfields }}会自动生成form_fields中设定的待填写字段，{{ next_button }}会生成一个提交按钮。

- 这里的内置的错误提示形式需要通过服务器进行验证，可以理解为学生交卷，老师改卷。另一种可行的方法是在前端借助JavaScript进行验证，即学生自己对答案并订正后才交回给老师。由于涉及到不少JavaScript的内容，不进行展开，具体的代码可以参考附带的代码示例。
- 为什么还需要这样一种前端验证方法？一个直接的原因是减轻服务器的运算负担，这个时候传回的数据服务器不需要再进行验证了；第二个原因是，如果在生成待输入字段的时候没有使用oTree自带的花括号形式（除{{ formfields }}外的其他形式参考官方文档Forms一节的内容），而是用原生HTML语言的<input>标签等生成输入框，这个时候生成的输入框在样式和逻辑上都会有很大不一样。
 - 使用{{ formfields }}生成的输入框在页面字段经验证存在错误后不会清空先前已填写的内容。

- 假如一个页面上有这么一个输入框使用了原生HTML标签，并且参加者已经填入了某个值。这个页面经后端验证后存在错误需要重新填写，这个时候参加者那边重新显示的页面上不会保留先前已填写的值，需要重新填写。为了避免这种情况出现，所以采用前端验证的方式而不经服务器。
- 这也可以看出oTree自带的元素或功能其实包含很多细微的特点和方便之处

显示的页面如下：

问卷调查

您就读的学校是

您的年龄是

您的性别是

您的年级

☐ 大一 ☐ 大二 ☐ 大三 ☐ 大四 ☐ 硕士生 ☐ 博士生

您是否主修经济类专业？

☐ 是
☐ 否

如果您有意见或建议，请填写在下框中

请在下框内填入1234以提交本页

下一页

- 第二个页面中我们想要展示被试刚刚输入的内容，一般来说可以在前端中直接调用相应的字段，但有的时候我们想要展示的是一些简单计算过得到的数据或者是依据某个条件进行展示。参考下面的例子：
- py文件：

```

1 class Results(Page):
2
3     @staticmethod
4     def vars_for_template(player):
5         if player.gender == 0:
6             gender_text = '女'
7         else:
8             gender_text = '男'
9         if player.grade <= 4:
10             grade_text = '本科生'
11         else:
12             grade_text = '研究生'
13         return dict(
14             gender_text = gender_text,
15             grade_text = grade_text
16         )

```

这里使用的静态方法是vars_for_template，这个方法可以在后端进行一些计算或赋值得到需要在前端进行展示的变量，这个方法返回的值是一个字典，前端页面可以直接调用字典的key（键）显示相应的值。在上面的例子中，dict是生成字典的一个方法，等号左边是字典的键，右边是字典的值，前端调用的是字典的键。例子中是根据先前输入的数据得到不同的文本，并用于展示。

- 相应的html文件：

```
1  {{ block title }}
2      问卷内容展示
3  {{ endblock }}
4
5  {{ block content }}
6      <p> 您就读的学校是 {{ player.school }}</p>
7      <p> 您的年龄是 {{ player.age }}</p>
8      <p> 您的性别是 {{ gender_text }}</p>
9      <p> 您现在是 {{ grade_text }}</p>
10     {{ if player.major == True }}
11     <p> 您<b>主修</b>经济类专业</p>
12     {{ else }}
13     <p> 您<b>不主修</b>经济类专业</p>
14     {{ endif }}
15
16 {{ endblock }}
```

这个例子中，学校和年龄是直接调用显示输入的数据，性别和年级则是调用vars_for_template中返回的字典中键，主修专业这里使用了花括号的条件语句。

显示的页面如下：注意下面出现的Debug info是用于程序调试的，每一页都会有，正式实验的时候需要关闭（参考正式实验操作下面的程序准备）

问卷内容展示

您就读的学校是 浙江大学

您的年龄是 18

您的性别是 男

您现在是 本科生

您**主修**经济类专业

Debug info	
vars_for_template	
gender_text	'男'
grade_text	'本科生'
Basic info	
ID in group	1
Group	33

- 在设定好页面之后，记得在py文件中的page_sequence中设定好页面顺序

```
1 | page_sequence = [Questionnaire, Results]
```

至此完成简单问卷的例子编写。

(二) 公共品博弈

- 下面介绍的是公共品博弈的编写例子，一些已在前面的例子中介绍的内容不再赘述
- 新建一个公共品博弈的app文件夹，在公共品博弈中，有两个全局层面的常数，一个是参加者的禀赋，另一个是投入禀赋之后的放大倍数，这种常数可以在C类中定义：

```
1 | class C(BaseConstants):
2 |     NAME_IN_URL = 'publicgood'
3 |     PLAYERS_PER_GROUP = 3
4 |     NUM_ROUNDS = 5
5 |
6 |     ENDOWMENT = cu(200)
7 |     MULTIPLIER = 2
```

- 这里的禀赋的写法表示这个数值是货币型的数值，如果设置了USE_POINTS为True，则表示200点（关于点数和收益的问题，参考（五）中的处理点数和收益以及官方文档Currency一节的内容），MULTIPLIER设置为2
- NAME_IN_URL的字符会出现在进入实验后浏览器显示的网址中，PLAYERS_PER_GROUP设定了小组人数为3（若无小组，则设定为None），NUM_ROUNDS设定了轮次为5
- 注意python里面所有的常数的名字应该全部用大写字母
- 公共品博弈的决策是个体决定自己的贡献额，而计算公共品投资回报则需要在小组层面进行计算，因此需要在Group类下定义小组总贡献额和回报，在Player类下定义贡献额

```
1 | class Group(BaseGroup):
2 |     total_contribution = models.CurrencyField()
3 |     individual_share = models.CurrencyField()
4 |
5 |
6 | class Player(BasePlayer):
7 |     contribution = models.CurrencyField(
8 |         min = 0,
9 |         max = C.ENDOWMENT,
10 |         label = "请输入您的贡献额"
11 |     )
```

- 如果在Player类下定义总贡献额或回报也是可以的，这个时候就需要想办法调用同组内其他人的贡献额数据进行计算（可以自行尝试这种写法，参考官方文档Multiplayers games一节下面Group小节的内容，需要用到get_others_in_group()这个内置方法），写在Group类中调用更方便
- 到这里就定义完成所需要的字段了，在这一部分还剩余两个问题：①如何分组？②如何计算收益？
 - 关于分组，这里设定5轮都是随机匹配，需要用到内置的随机分组方法：

```
1 | def creating_session(subsession):
2 |     subsession.group_randomly()
```

- 这里定义的creating_session是一个函数，不需要定义在某个类下面。def creating_session(subsession)是一个内置的函数，用于在开始实验前设定分组、角色等内容，名字写法固定。在启动实验时，名为creating_session的函数会被自动调用执行
- subsession.group_randomly()是每轮随机分组的内置方法（关于分组和角色分配，参考（五）的“group和role的设定”相关内容）
- 关于计算收益，需要自行设定收益计算函数并调用（单独写收益计算函数可以提高代码可读性，也便于修改和查找问题）

```

1 def set_payoffs(group:Group):
2     player_list = group.get_players() #获得包含小组内所有玩家的列表
3     contribution_list = [p.contribution for p in player_list] #获得小组内所有贡献额列表
4     group.total_contribution = sum(contribution_list) #求得贡献额总和
5     group.individual_share = group.total_contribution * C.MULTIPLIER / C.PLAYERS_PER_GROUP
6     for p in player_list: #对小组内的每一个玩家，减去自己的贡献额后加上分得的收益
7         p.payoff = C.ENDOWMENT - p.contribution + group.individual_share

```

- 由于小组内的三人收益是共同决定的，因此从小组层面进行计算
- (group:Group)表示的意思是：group是函数需要的参数的标记，Group是这个参数建议取的值为Group类，这样写是提醒作用，防止在后面调用的时候出错
- 下面是PAGES部分的内容，这一任务分为两个页面，第一个页面是输入贡献额，第二个页面是报告收益，由于输入贡献额的速度有快慢，因此需要添加等待页面等待小组内或所有人完成决策后再进入收益报告
- 贡献决策页面如下：

```

1 class Contribution(Page):
2     form_model = 'player'
3     form_fields = ['contribution']

```

对应的html文件如下：

```

1 {{ block title }}
2     第{{ player.round_number }}轮决策
3 {{ endblock }}
4
5 {{ block content }}
6
7     <p>
8         <u>
9             <li>在这个实验中你和另外两人随机配对，组内共有{{ C.PLAYERS_PER_GROUP }}人。</li>
10            <li>每人都有实验点{{ C.ENDOWMENT }}。</li>
11            <li>投入公共池中的实验点会乘以{{ C.MULTIPLIER }}。</li>
12        </u>
13    </p>
14
15    {{ formfields }}
16

```

```

17
18     {{ next_button }}
19
20 {{ endblock }}

```

显示如下：

第1轮决策

- 在这个实验中你和另外两人随机配对，组内共有3人。
- 每人都有实验点200点。
- 投入公共池中的实验点会乘以2。

请输入您的贡献额

点

下一页

- 完成决策后，需要一个WaitPage让参加者等待。WaitPage不需要在文件夹下写对应的html文件，只需要在py文件里设定即可。这里的等待有两种方式，第一种是等待同组三人都完成进入等待界面后，等待结束，进入下一页收益报告，第二种写法是等待在场所有人都完成后才一起进入下一页，下面给出两种写法：

```

1 class ResultswaitPage(WaitPage):
2     title_text = '请耐心等待！'
3     body_text = '请保持安静，如果有问题，请咨询实验员。'
4
5     @staticmethod
6     def after_all_players_arrive(group: Group):
7         set_payoffs(group)
8
9     #wait_for_all_groups = True
10    #@staticmethod
11    #def after_all_players_arrive(subsession: Subsession):
12    #    group_list = subsession.get_groups()
13    #    for g in group_list:
14    #        set_payoffs(g)

```

- title_text和body_text设定了等待页面显示的标题和文本内容
- after_all_players_arrive是一个内置的方法，表示等待所有人（group或subsession）到达这一页面后才执行，第一种方法是等待同组所有人都到达之后才执行，因此传入的参数是group，直接调用前面定义的收益计算函数即可
- 第二种方法（注释掉的方法）中首先设定了wait_for_all_groups = True，表示要等所有小组都到达了才执行，这个时候after_all_players_arrive必须是一个subsession的方法，传入subsession参数，然后再对subsession里面的小组逐一执行收益计算函数
- 等待页面显示如下：

请耐心等待!

请保持安静, 如果有问题, 请问实验员。

- 最后的收益报告不需要在py文件里写代码, html如下:

```
1  {{ block title }}
2      第{{ player.round_number }}轮收益报告
3  {{ endblock }}
4
5  {{ block content }}
6
7      <p>
8          <ul>
9              <li>本轮你投入了{{ player.contribution }}。</li>
10             <li>小组总的贡献额是{{ group.total_contribution }}。</li>
11             <li>组内每人可分得{{ group.individual_share }}。</li>
12             <li>你的本轮收益是{{ player.payoff }}。</li>
13         </ul>
14     </p>
15     {{ if player.round_number == C.NUM_ROUNDS }}
16     <p>
17         你的{{ C.NUM_ROUNDS }}轮总收益是{{ participant.payoff }}。
18     </p>
19     {{ endif }}
20
21     {{ next_button }}
22
23  {{ endblock }}
```

- 唯一需要说明的是, 这里调用了payoff和participant下的payoff, payoff是内置的用于统计收益的字段, payoff和participant类下都有这个字段, 直接调用即可(关于这个字段的说明, 参考(五)中的处理点数和收益)
- 显示如下:

第1轮收益报告

- 本轮你投入了100点。
- 小组总的贡献额是300点。
- 组内每人可分得200点。
- 你的本轮收益是300点。

下一页

- 最后的安排页面顺序: 在重复多轮的实验中, 这些页面会重复出现

```
1  page_sequence = [Contribution, ResultswaitPage, Results]
```

至此完成一个简单的公共品博弈的编写。

(三) 信任博弈

- 下面介绍的是信任博弈的编写例子，一些已在前面的例子中介绍的内容不再赘述
- 新建一个信任博弈的app文件夹，与前面的博弈不同的是，信任博弈不仅有分组，还是一个序贯博弈，需要区分角色和决策的先后
- 信任博弈的两个参数是初始禀赋和投资的放大倍数，同样这两个参数写在C类下

```
1 class C(BaseConstants):
2     NAME_IN_URL = 'trust_game'
3     PLAYERS_PER_GROUP = 2
4     NUM_ROUNDS = 5
5
6     ENDOWMENT = cu(100)
7     MULTIPLIER = 3
```

- 信任博弈的决策是角色A决定发送的点数，角色B决定的是返还的点数，这两个决策都是在两人配对的小组Group层面做出的（同样，另一种写法是将这个决策写在Player类下面）。在Group类下添加两个字段：

```
1 class Group(BaseGroup):
2     sent_amount = models.CurrencyField(
3         min=cu(0),
4         max=C.ENDOWMENT,
5         label="请输入您要发送的点数",
6     )
7     sent_back_amount = models.CurrencyField(
8         min=cu(0),
9         label="请输入您要返还的点数："
10    )
```

- 对sent_amount的范围限定是显然的，但是sent_back_amount的上限则是不确定的，需要根据发送额确定。这种属于动态的验证。这里只需要确定返还额的上限，因此使用的是{{field_name}}_max()的函数：

```
1 def sent_back_amount_max(group: Group):
2     return group.sent_amount * C.MULTIPLIER
```

{{field_name}}_max()这种格式定义的函数会被自动执行，用于实验中动态确定上限，类似的函数还有{{field_name}}_min()等（更多验证方法参考官方文档Forms一节下面的Dynamic form field validation的内容）。

- 下面要定义的是收益计算函数：

```
1 def set_payoffs(group: Group):
2     p1 = group.get_player_by_id(1)
3     p2 = group.get_player_by_id(2)
4     p1.payoff = C.ENDOWMENT - group.sent_amount + group.sent_back_amount
5     p2.payoff = group.sent_amount * C.MULTIPLIER - group.sent_back_amount
```


这里使用了group内置的get_player_by_id()的方法，这个方法用于在组内寻找某个有id_in_group的角色，这里的player1是先行动的角色A，player2是后行动的角色B，角色的分配由oTree根据分组情况随机设定，因此可以不手写角色分配代码（关于角色分配，参考（五）的group和role的设定这部分）

- 最后确定分组方式，采用随机分组方式的同时，又需要保证参加者的角色固定，因此设定如下参数：

```
1 def creating_session(subsession):
2     subsession.group_randomly(fixed_id_in_group=True)
```

fixed_id_in_group取值True即可保证在随机分组时id_in_group即角色不变

- 下面是PAGES部分的内容，考虑博弈时序，需要决策的页面有发送和返还两个，而小组内不同角色还需要等待对方决策，因此还要有两个WaitPage，最后还要加上结果报告的页面。除此以外，我们还想在实验开始前提供简短的实验说明并且告知参加者他们的角色。
- 我们首先考虑作为实验介绍的开始页，在这一页上提供简短的实验说明以及告知角色，另外，这一页仅需要在第一轮出现，在第二轮重复的时候，这一页不再出现，因此还需要设定显示条件：

```
1 class Introduction(Page):
2
3     @staticmethod
4     def vars_for_template(player: Player):
5         if player.id_in_group == 1:
6             role_text = '角色A'
7         else:
8             role_text = '角色B'
9         return dict(
10             role_text = role_text
11         )
12
13     @staticmethod
14     def is_displayed(player: Player):
15         return player.round_number == 1
```

- 设定显示条件用到的方法是is_displayed，return后的判断条件满足才会显示页面，这里设定仅在第一轮显示
- Introduction页面需要着重介绍的在html页面中，首先这里给出示例代码：

```
1 {{ block title }}
2     实验介绍
3 {{ endblock }}
4 {{ block content }}
5
6     {{ include_sibling 'instructions.html' }} <!-- 外部插入实验说明 -->
7
8     <p><b>
9         您的角色是:  {{ role_text }}
10     </b></p>
11
12     <button class="otree-btn-next btn btn-primary" id="btn"
13 style='float:right'>请阅读实验说明（20）</button>
14     <!-- 使用了oTree内置的button的class -->
```

```

15     <script>
16         var btn = document.getElementById('btn'); //获取元素
17         var secs = 20; //设定了倒计时为20秒
18         btn.disabled=true; //禁止点击
19         for (var i=1; i<=secs; i++) {
20             window.setTimeout("update(" + i + ")", i * 1000);
21         } //每1秒执行一次update函数
22         function update(num) {
23             if (num==secs) { //倒计时结束，按键的文本改变，并且可以点击
24                 btn.textContent = "开始实验";
25                 btn.disabled = false;
26             }
27             else { //倒计时中，实时更新文本里面的数字
28                 var printnr = secs - num;
29                 btn.textContent = "请阅读任务说明 (" + printnr + ")";
30             }
31         }
32     </script>
33     {{ endblock }}

```

- 可以看到，这里并没有出现实验说明，而是使用了插入html的方式在这个页面上插入了实验说明，格式是{{ include_sibling 'instructions.html' }}，这是oTree自带的一个写法，可以方便地在页面上插入其他内容，特别是那些需要重复出现的内容
 - 一个好处是，如果需要修改实验说明等需要重复出现的内容，只需要修改一个文件即可，不用修改多个文件，大大降低了代码重复度
 - 这种方式插入的html内容和主页面的html文件放在同一个文件夹即可
 - 有的时候CSS和JS的代码会作为单独的文件保存在_static文件夹中，这些文件的读取方式参考官方文档Miscellaneous→Advanced features→Static files的内容，例子：

```

1 <link rel="stylesheet" href="{{ static 'mystyle.css' }}">
2 <script src="{{ static 'myscript.js' }}"></script>

```

- 第二个需要说明的是这里使用了一个有点复杂的自定义按键，这个按键在20秒之内是无法点击的，只有20秒之后才能点击进入下一页，这强制要求参加者停留一段事件阅读说明
- **注意：**在static文件夹中保存的css和js文件很可能并不会随着修改而实时更新，这是因为启动了一次server之后，浏览器将js文件和css文件缓存下来了，因此即使本地修改了css和js文件，浏览器可能还是会使用修改前缓存的文件。为了让新修改的文件起效，可以进行一次全页面的刷新（**full page reload，一般的快捷键是Ctrl+F5**）
 - 参考代码解释理解，JavaScript的作用就是获取页面上的任一个元素并进行动态实时的更新，包括这些元素的内容、属性、样式等等
 - 倒计时未结束：

实验介绍

实验说明

本实验将分为两个角色：角色A与角色B。每一轮开始的时候，角色A会获得实验点100点。角色A可以决定将实验点的一部分或全部发送给角色B，发送出去的实验点将乘以3。角色B可以决定将放大后的实验点中的部分或全部返还给角色A。

实验开始前将随机决定您的角色。本实验进行5轮，每一轮您将和另一位角色的参加者随机匿名地匹配在一起。您的角色在所有轮次中固定不变。

您的角色是：角色A

请阅读任务说明 (18)

- 倒计时结束：

实验介绍

实验说明

本实验将分为两个角色：角色A与角色B。每一轮开始的时候，角色A会获得实验点100点。角色A可以决定将实验点的一部分或全部发送给角色B，发送出去的实验点将乘以3。角色B可以决定将放大后的实验点中的部分或全部返还给角色A。

实验开始前将随机决定您的角色。本实验进行5轮，每一轮您将和另一位角色的参加者随机匿名地匹配在一起。您的角色在所有轮次中固定不变。

您的角色是：角色A

开始实验

- 第一个决策页面是发送决策，需要限定只有角色A (id_in_group==1) 才会显示，而角色B则等待：

```
1 class Send(Page):
2     form_model = 'group'
3     form_fields = ['sent_amount']
4
5     @staticmethod
6     def is_displayed(player: Player):
7         return player.id_in_group == 1
8
9
10 class SendBackWaitPage(WaitPage):
11     title_text = '请耐心等待!'
12     body_text = '请保持安静，如果有问题，请询问实验员。'
```

对应的发送决策页面，这里在页面下方插入了实验说明：

```
1 {{ block title }}
2     角色A决策-第{{ player.round_number }}轮
3 {{ endblock }}
4 {{ block content }}
5
6     <p>
7         您是角色A。您有实验点{{ C.ENDOWMENT }}。您会发送多少给角色B？
8     </p>
9
10    {{ formfields }}
11    <p>
12    {{ next_button }}
13    </p>
```

```

14
15     {{ include_sibling 'instructions.html' }}
16
17 {{ endblock }}

```

显示页面如下：

角色A决策-第1轮

您是角色A。您有实验点100点。您会发送多少给角色B？

请输入您要发送的点数

点

下一页

实验说明

本实验将分为两个角色：角色A与角色B。每一轮开始的时候，角色A会获得实验点100点。角色A可以决定将实验点的一部分或全部发送给角色B，发送出去的实验点将乘以3。角色B可以决定将放大后的实验点中的部分或全部返还给角色A。

实验开始前将随机决定您的角色。本实验进行5轮，每一轮您将和另一位角色的参加者随机匿名地匹配在一起。您的角色在所有轮次中固定不变。

- 第二个决策是返还决策，这个页面只显示给角色B（id_in_group==2），角色A进入等待界面，在角色B完成决策后，计算最后的收益：

```

1 class SendBack(Page):
2     form_model = 'group'
3     form_fields = ['sent_back_amount']
4
5     @staticmethod
6     def is_displayed(player: Player):
7         return player.id_in_group == 2
8
9     @staticmethod
10    def vars_for_template(player: Player):
11        group = player.group
12        tripled_amount = group.sent_amount * C.MULTIPLIER
13        return dict(tripled_amount=tripled_amount)
14
15
16 class ResultswaitPage(waitPage):
17     title_text = '请耐心等待！'
18     body_text = '请保持安静，如果有问题，请询问实验员。'
19
20     after_all_players_arrive = set_payoffs

```

这里的after_all_players_arrive = set_payoffs是一种简写形式，和上一例子中的写法是一样的作用，注意这里没有设定wait_for_all_groups，默认为False，只需要当前轮次的小组成员到达即可对应的返还决策页面如下：

```

1 {{ block title }}
2     角色B决策-第{{ player.round_number }}轮
3 {{ endblock }}
4 {{ block content }}
5

```

```

6      <p>
7          您是角色B。
8          角色A发送了实验点{{ group.sent_amount }}，因此你获得实验点{{
tripled_amount }}。
9          您将会返还多少实验点给角色A?
10     </p>
11
12     {{ formfields }}
13     <p>
14     {{ next_button }}
15     </p>
16
17     {{ include_sibling 'instructions.html' }}
18
19 {{ endblock }}

```

显示如下：

角色B决策-第1轮

您是角色B。角色A发送了实验点50点，因此你获得实验点150点。您将会返还多少实验点给角色A？

请输入您要返还的点数：

点

下一页

实验说明

本实验将分为两个角色：角色A与角色B。每一轮开始的时候，角色A会获得实验点100点。角色A可以决定将实验点的一部分或全部发送给角色B，发送出去的实验点将乘以3。角色B可以决定将放大后的实验点中的部分或全部返还给角色A。

实验开始前将随机决定您的角色。本实验进行5轮，每一轮您将和另一位角色的参加者随机匿名地匹配在一起。您的角色在所有轮次中固定不变。

- 最后是收益报告的页面：

```

1 class Results(Page):
2
3     @staticmethod
4     def vars_for_template(player: Player):
5         group = player.group
6
7         return dict(tripled_amount=group.sent_amount * C.MULTIPLIER)

```

对应的html：

```

1     {{ block title }}
2     收益报告-第{{ player.round_number }}轮
3     {{ endblock }}
4     {{ block content }}
5
6         {{ if player.id_in_group == 1 }}
7             <p>
8                 本轮您有实验点{{ C.ENDOWMENT }}，
9                 您发送给角色B {{ group.sent_amount }}，
10                角色B返还 {{ group.sent_back_amount }}。
11            </p>

```

```

12     <p>
13         本轮您的收益点数:
14         {{ C.ENDOWMENT }}-{{ group.sent_amount }}+{{
group.sent_back_amount }}=<strong>{{ player.payoff }}</strong>
15     </p>
16     {{ else }}
17     <p>
18         角色A发送给您 {{ group.sent_amount }},
19         放大后您的点数是 {{ tripled_amount }},
20         您返还了 {{ group.sent_back_amount }}。
21     </p>
22     <p>
23         本轮您的收益点数:
24         ({{ tripled_amount }})-({{ group.sent_back_amount }})=
<strong>{{ player.payoff }}</strong>
25     </p>
26     {{ endif }}
27
28     {{ if player.round_number == C.NUM_ROUNDS }}
29     <p>
30         在{{ C.NUM_ROUNDS }}轮中, 您一共获得实验点{{ participant.payoff
}}
31     </p>
32     {{ endif }}
33
34     <p>{{ next_button }}</p>
35
36
37     {{ endblock }}

```

- 两个角色的收益报告:

收益报告-第1轮

本轮您有实验点100点, 您发送给角色B 50点, 角色B返还 50点。

本轮您的收益点数: 100点-50点+50点=**100点**

下一页

收益报告-第1轮

角色A发送给您 50点, 放大后您的点数是 150点, 您返还了 50点。

本轮您的收益点数: (150点)-(50点)=**100点**

下一页

- 页面顺序如下: Introduction仅在第一轮显示

```

1 | page_sequence = [
2 |     Introduction,
3 |     Send,
4 |     SendBackWaitPage,
5 |     SendBack,
6 |     ResultswaitPage,
7 |     Results,
8 | ]

```

至此完成简单的信任博弈的编写

(四) 程序测试

- 在编写好程序后，首先需要在自己的电脑上进行调试。在调试前需要在settings.py中进行一些设定：以上面编写的三个程序为例，在SESSION_CONFIGS中填入内容：

```

1 | SESSION_CONFIGS = [
2 |     dict(
3 |         name = 'questionnaire',
4 |         display_name = 'questionnaire',
5 |         app_sequence = ['questionnaire'],
6 |         num_demo_participants = 2,
7 |     ),
8 |     dict(
9 |         name = 'publicgood',
10 |         display_name = 'publicgood',
11 |         app_sequence = ['publicgood'],
12 |         num_demo_participants = 6,
13 |     ),
14 |     dict(
15 |         name = 'trustgame',
16 |         display_name = 'trustgame',
17 |         app_sequence = ['trustgame'],
18 |         num_demo_participants = 2,
19 |     ),
20 | ]

```

- name和display_name保持一致即可，display_name是后台看到的名字
 - app_sequence可以添加多个app在列表中，app将按顺序执行
 - num_demo_participants是测试的时候生成的参加者链接数量，注意要是程序中设定的每组最小人数的倍数
- 在终端中将路径转移至程序文件夹，使用如下命令启动测试服务器：

```
1 | otree devserver
```

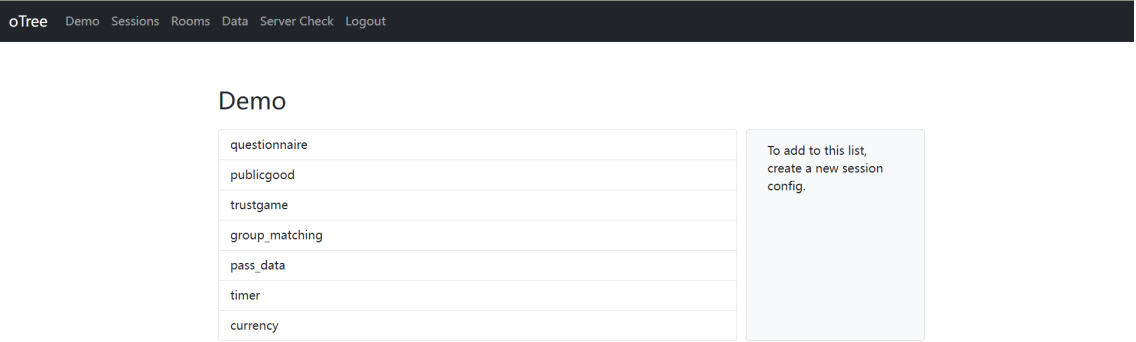
- 输入命令后，正常启动的服务器会显示下图的内容：本地服务器的地址是localhost:8000，关闭服务器的快捷键是Ctrl+C

```

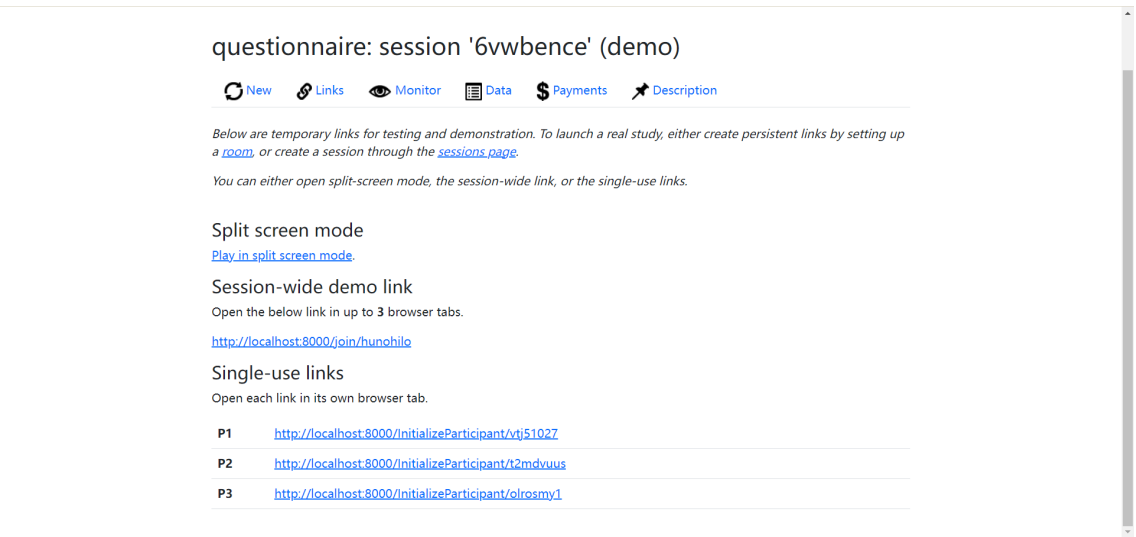
PS E:\EXP\firstexp> otree devserver
Open your browser to http://localhost:8000/
To quit the server, press Control+C.

```


- 按住Alt（或Ctrl，取决于设置）点击链接可以快速打开服务器，进入服务器后台，可以看到Demo列表，点进需要测试的程序即启动相应的实验demo



- 在实验的后台可以看到有不同的标签页，默认展示的是Links标签页，下面的链接可以点击打开用于测试



- Monitor页是观察实验进展的，发生变动的参加者该行会变绿然后逐渐变淡

questionnaire: session '6vwbence' (demo)

	Code	Label	Progress	App	Round	Page name	Waiting for	Time
P1	vtj51027		1/2	questionnaire	1	Questionnaire		4m
P2	t2mdvuus		1/2	questionnaire	1	Questionnaire		4m
P3	olrosmy1		1/2	questionnaire	1	Questionnaire		4m

3/3 participants started.

- Data页可以看到数据，同样发生变动的单元格会变绿然后逐渐变淡

questionnaire: session '6vwbence' (demo)

[New](#) [Links](#) [Monitor](#) [Data](#) [Payments](#) [Description](#)

	group	id_in_group	role	payoff	school	age	gender	grade	major	suggest	attention	group. id_in_subsession
P1	1	1		0.0								1
P2	1	2		0.0								1
P3	1	3		0.0								1



questionnaire



Round 1

- Payment记录了支付信息

questionnaire: session '6vwbence' (demo)

[New](#) [Links](#) [Monitor](#) [Data](#) [Payments](#) [Description](#)

Session

Session config	questionnaire
Session code	6vwbence
Participation fee	10.00元

Participants

Code	Label	Progress	Payoff (bonus)	Total
vtj51027		1/2	0.00元	10.00元
t2mdvuus		1/2	0.00元	10.00元
olrosmyl		1/2	0.00元	10.00元

Summary

Total payments	30.00元
Mean payment	10.00元

- oTree在测试过程中可以直接对代码进行修改，保存后通过刷新或者点击New就可以使修改生效，不用关闭测试服务器

(五) 对程序的扩展说明

1. 处理点数和收益

- 在上面的例子中，涉及点数和收益的转化时都使用了currencyfield，通过settings.py中修改相关设置直接将点数转换为实际货币收益
- settings.py中与currencyfield和收益相关的设置：
 - real_world_currency_per_point: 每1点实验点等于多少现实货币，在计算收益payoff时会自动作用于其中的currencyfield
 - participation_fee: 出场费
 - REAL_WORLD_CURRENCY_CODE: 指定使用的真实货币，人民币是CNY
 - USE_POINTS: 设定为True会将实验中出现的货币字段的单位转换为点数；设定为False表明不在实验中使用实验点数作为单位，这也将使得real_world_currency_per_point失效

- 假设设定`real_world_currency_per_point=0.50`, `participation_fee=10.00`, `REAL_WORLD_CURRENCY_CODE = 'CNY'`, 参加者在实验中获得了20, 数据保存在`earned`这个`currencyfield`中, 并直接作为收益`payoff` (可修改附带的程序进一步测试理解)
 - 如果设定`USE_POINTS`为`True`, 则`earned`、`player`和`participant`两个类下面的`payoff`都使用点数作为单位, 参加者获得的点数是20, 最终收益是:
 $20 \times 0.5 + 10 = 20\text{元}$
 - 如果设定`USE_POINTS`为`False`, 则`earned`、`player`和`participant`两个类下面的`payoff`都直接使用货币“元”作为单位, `real_world_currency_per_point`不再起作用, 最终的收益是 $20 + 10 = 30\text{元}$
 - 一般使用`USE_POINTS=True`
- `player`和`participant`中的`payoff`: `player`和`participant`中都有内置的字段`payoff`, 得到的收益应该写入`payoff`字段中, 如前所述, 一个`participant`可能有很多`player` (比如很多轮次/`subsession`), 因此`participant`中的`payoff`就是自动加总所有`subsession`中`payoff`的一个总`payoff`
 - 在计算`player.payoff`的时候需要自己定义收益计算的函数, 并在后续的某个页面或某个环节中进行调用, 比如在多轮博弈中设定每一轮都会计算一次`player.payoff`
 - `oTree`并没有一个内置的类似`TotalPayoff`的概念, 起加总作用的是`participant`中的`payoff`, 在只有一轮的时候, 两个`payoff`相等, 在多轮的时候, `participant.payoff`起加总的作用
 - `participant.payoff`也是可以进行修改的, 比如四舍五入、取整等等
 - `participant`自带`payoff`字段, 因此跨app调用的时候不需要写在`PARTICIPANT_FIELDS`中 (调用数据的内容见下面第3点)
 - `participant`的`payoff`还没有加上出场费, 最终参加者可以拿到的钱 (后台`Payment`标签下显示的数字) 是经过换算的`payoff`加上`participation_fee`得到的
 - `participant_payoff_plus_partipation_fee()`是内置的计算最终现金收益的方法, 可以用于前端的输出展示等
- 在处理收益上, 如果需要对不同实验任务采用不同的收益率, 那么可以在不同实验中计算`payoff`的时候写入不同的比例, 而不使用统一的`real_world_currency_per_point` (设置为1并把`USE_POINTS`设定为`True`即可, 所有比例换算都手动完成), 有的时候甚至不一定使用`CurrencyField`而使用`IntegerField`或`FloatField`, 也即所有的运算都通过自定的函数进行, 最终只需记得将结果存入`payoff`

2. Timeout的设置

- 默认中页面没有结束的限制时间, 只有在提交或达成其他设定条件时才会结束当前页面进入下一页面
- 需要设定页面的限制时间时, 涉及两个内置的字段: `time_out_seconds`和`timer_text`
- 内置倒计时方法: 设定`time_out_seconds`设定了页面总时间, 设定`timer_text`设定倒计时的文本提示语, 注意, 内置的倒计时会在时间耗尽后自动跳过当前页面, 即使当前页面信息或决策没有填完也会提交
 - 在需要设定倒计时的`Page`中设定即可, 不需要在`html`文件中写倒计时内容, `oTree`会自动在页面上调用内置的`otree-timer`

```

1 class Timer1(Page):
2     timeout_seconds = 20
3     timer_text = '当前页面的剩余时间'
```

- 一个拓展: 在倒计时仅剩10秒时才显示时间, 在`html`上添加如下的`css`和`js`代码:

- ```

1 <style>
2 .otree-timer {
3 display: none;
4 }
5 </style>
6
7 <script>
8 let customTimerEle = document.getElementById('time-left');
9 document.addEventListener("DOMContentLoaded", function (event) {
10 $(''.otree-timer__time-left').on('update.countdown', function
(event) {
11 if (event.offset.totalSeconds === 10) {
12 $(''.otree-timer').show();
13 };
14 });
15 });
16 </script>

```

- 内置倒计时也可以对样式进行自定义，参考下面的例子：

- ```

1 class Timer2(Page):
2     timeout_seconds = 20

```

- ```

1 <style>
2 .otree-timer {
3 display: none;
4 }
5 </style>
6
7 <p>
8 剩余 秒
9 </p>
10
11 <script>
12 let customTimerEle = document.getElementById('time-left');
13 document.addEventListener("DOMContentLoaded", function (event) {
14 $(''.otree-timer__time-left').on('update.countdown', function
(event) {
15 customTimerEle.innerText = event.offset.totalSeconds;
16 });
17 });
18 </script>

```

- 有的时候我们并不需要一个自动提交的页面，倒计时仅起到催促、提醒的软约束作用，这个时候的倒计时就不能使用oTree自带的倒计时功能，而应该使用JS实现
  - 最简单的方式：使用浏览器弹窗进行提醒，倒计时结束后浏览器弹窗，弹窗文本内容为alert里面的内容

- ```

1 <script>
2     setTimeout(
3         function () {
4             alert("倒计时结束，请做出决策！");
5         },
6         10*1000 // 60 seconds
7     );
8 </script>

```

- 这种方法在页面上没有显示时间，要实现页面上显示倒计时比较复杂，参考如下的代码：

- ```

1 <p id="timer_posi">剩余时间：</p>
2 <script>
3 //自己设定的倒计时，倒计时结束不会提交当前页面
4 $(document).ready(function(){
5 window.onload = function(){
6 var time;
7 //获取待修改元素
8 var pTime = document.getElementById('timer_posi');
9 //设定倒计时的分和秒
10 var m = 0;
11 var s = 30;
12 var total_time = m*60+s;
13 //防止页面刷新重新开始倒计时的代码
14 //如果在能找到某个值countDown，则说明已刷新过
15 if(sessionStorage.getItem('countDown')){
16 time = sessionStorage.getItem('countDown');
17 var time1 = new Date().getTime(); //获取当前时间
18 var remain_time = total_time - (time1-time)/1000 //获取剩
19 余时间（s）
20 m = Math.floor(remain_time/60);
21 if (remain_time < 60){
22 s = Math.floor(remain_time);
23 };
24 if (remain_time >= 60){
25 s = Math.floor(remain_time - m*60)
26 };
27 //m = m - Math.floor((time1-time)/1000/60); //倒计时时长-
28 (现在时间-开始时间)=剩余时长
29 //s = s - Math.floor((time1-time)/1000%60);
30 }else{
31 time = new Date().getTime(); //没有刷新过，则获取当前时间作为
32 页面开始时间，并保存为countDown
33 sessionStorage.setItem('countDown',time);
34 };
35 //设定计时器，利用setInterval(function(),1000)，表示每过1秒（1000
36 毫秒）执行一次function()
37 var timer = setInterval(function(){
38 //如果上一次秒是00，则改变取值：比如4：00后显示3：59
39 if(s == '00' && m > 0){
40 s = 59;
41 m--;
42 } else{ //否则只是秒值递减
43 s--;

```

```

41 };
42 //如果秒值是0-9, 则拼接一个0, 显示为“01”形式
43 if(s >= 0 && s <10){
44 s = '0' + Number(s)
45 };
46 //如果分值是0-9, 则拼接一个0, 显示为“01”形式
47 if(m >= 0 && m <10){
48 m = '0' + Number(m)
49 }
50 //拼接字符串, 用于显示
51 time_text = "剩余时间: "+m+": "+s;
52 pTime.innerHTML = time_text;
53 //在倒计时结束后, 额外添加加红加粗的提醒语
54 if (s <= 0 && m <= 0) {
55 clearInterval(timer); //使用这个代码将在倒计时结束后清除倒
 计时, 最终只会显示00: 00
56 //如果不使用这个代码, 秒上的取值会为负, 比如“00:-4”
57 time_text = "剩余时间: 00:00";
58 pTime.innerHTML = time_text + '<p>请尽快决策! </p>';
59 }
60 },1000)
61 };
62 })
63 </script>

```

### 3. round和app之间的数据传递

- 在oTree中每一轮的subsession、group和player对象都是独立的, 在第一轮输入的数据并不会传递至第二轮
- 为了获得前一轮或者前面某些轮次中的数据, 需要借助subsession、group和player对象内置的方法:
  - in\_previous\_rounds(): 返回一个列表, 列表元素为前面所有轮次中的对象, 比如10轮时 player.in\_previous\_round()返回的列表元素是前面9轮的player对象
  - in\_all\_rounds(): 和上一个的区别是返回的列表包括当前轮次对象
  - in\_rounds(m,n): 返回m至n轮的对象组成的列表
  - in\_round(m): 返回m轮的对象, 比如返回前一轮的player对象是 player.in\_round(player.round\_number - 1)
- 在app之间进行传递, 则需要使用到Participant fields, 将需要的数据存到participant对象中, 首先需要在setting.py中设定PARTICIPANT\_FIELDS
- 一个简单的在round和app之间传递数据的例子如下 (注意: 下面的例子仅展示部分代码, subsession、group等没有修改保持默认的类不展示, 完整代码参考附带的程序文件夹):
  - 在第一个app中, 设定NUM\_ROUNDS = 2:

```

1 class C(BaseConstants):
2 NAME_IN_URL = 'pass_data_part1'
3 PLAYERS_PER_GROUP = None
4 NUM_ROUNDS = 2

```

- 在Player类中设定一个字段用于储存数据:

- ```

1 class Player(BasePlayer):
2     pass_number = models.IntegerField(label='输入一个数字用于在round
    和app之间传递')

```

- MyPage仅在第一轮出现，用于输入数据，并向Participant传递数据

- ```

1 class MyPage(Page):
2 form_model = 'player'
3 form_fields = ['pass_number']
4
5 @staticmethod
6 def before_next_page(player: Player, timeout_happened):
7 participant = player.participant #获取player的participant
8 participant.pass_number = player.pass_number #将player的
 pass_number传给participant
9 #participant.var这样的写法就是使用participant类中的字段
10
11 @staticmethod
12 def is_displayed(player: Player):
13 return player.round_number == 1

```

- 相应的MyPage.html写法:

- ```

1 {{ block title }}
2     输入数字
3 {{ endblock }}
4 {{ block content }}
5     <p>这是第{{ player.round_number }}轮</p>
6     {{ formfields }}
7     {{ next_button }}
8
9 {{ endblock }}
10

```

- Results仅在第二轮出现，获取第一轮输入的数据并展示在页面上

- ```

1 class Results(Page):
2
3 @staticmethod
4 def vars_for_template(player: Player):
5 prev_player = player.in_round(1) #获取第一轮的玩家对象
6 number_round_1 = prev_player.pass_number
7 return dict(
8 number_round_1 = number_round_1 #为页面展示准备的变量
9 #前端不允许player.in_round(1).pass_number这样的写法，所
 以需要转换以下
10)
11
12 @staticmethod
13 def is_displayed(player: Player):
14 return player.round_number == 2

```

- 相应的Results.html写法:



- ```

1  {{ block title }}
2      展示数字
3  {{ endblock }}
4
5  {{ block content }}
6      <p>这是同一app的第{{ player.round_number }}轮</p>
7      <p>你在上一轮中输入的数字是{{ number_round_1 }}</p>
8      {{ next_button }}
9  {{ endblock }}
```

- 第二个app的Resultses直接从Participant中获取数据并展示

- ```

1 {{ block title }}
2 第二个app-展示数字
3 {{ endblock }}
4
5 {{ block content }}
6
7 <p>
8 在前一个app中你输入的数字是{{ participant.pass_number }}
9
10 </p>
11 {{ endblock }}
```

- 在setttins.py中设定PARTICIPANT\_FIELDS:

```
1 PARTICIPANT_FIELDS = ['pass_number']
```

## 4. group和role的设置

- 在实验中不可避免地会遇到分组匹配和角色分配的问题，比如独裁者博弈、信任博弈等，如何在oTree中进行随机的分组和角色分配？
- 内置的重要变量或字段：
  - group: 每个参加者都有一个group的字段表明参加者所属的组别
  - id\_in\_group: 参加者在自己小组内的组内id，取值为1、2、3.....，取值与在C类中设定的PLAYERS\_PER\_GROUP有关，这个变量实际上是区分了不同的角色，因为oTree的角色是按Player 1/Player 2这样定义的，id\_in\_group = i即为Player i。无论是否对不同角色设定了不同任务，这个id序号分配都会自动进行
  - role: 这个字段也用于指示角色，取值取决于id\_in\_group，与id\_in\_group不同的是这个字段可以取字符型的值，在C类中定义的以“\_ROLE”结尾的变量的值将会按顺序赋值给id\_in\_group=1, 2, 3的被试（参见下面的例子）
  - 在下面说明的分组与角色分配方式的例子中，我们以2人一组的委托代理博弈为例，首先在C类中添加如下的两个角色变量

- ```

1 PRINCIPAL_ROLE = 'principal'
2 AGENT_ROLE = 'agent'
```

- 内置分组与角色分配方式**

- oTree自带了随机分组和分角色的功能，即对被试的group和id_in_group进行随机分配

- 如果什么分组方法都不设定，默认的分组方法是固定匹配和固定角色（即固定的id_in_group），并且匹配方式必定是P1和P2匹配，P3和P4匹配（以2人一组为例），而id_in_group是按1、2、1、2.....这样的顺序依次安排，即P1和P3是1，P2和P4是2，因此采用内置函数分配时“随机”是在给被试分配编号（比如抽实验说明）时完成的
- 设定分组方法（无论是用内置的函数还是另外的分配方法）都要定义一个creating_session函数并在其中进行（注意这个函数不是在某一个类下定义的）
- 如果每轮都进行随机匹配，且不存在角色，使用内置的分组函数如下：

```
1 def creating_session(subsession):
2     subsession.group_randomly() #随机匹配的内置方法
3     print(subsession.get_group_matrix()) #输出分组结果的矩阵，所有轮次的匹配结果矩阵将会依次打印在终端上
```

- 如果每轮都进行随机匹配，且需要固定角色，使用内置的分组函数如下：

```
1 def creating_session(subsession):
2     subsession.group_randomly(fixed_id_in_group=True) #设定fixed_id_in_group可以在重新分组时保持角色不变
3     print(subsession.get_group_matrix()) #输出分组结果的矩阵，所有轮次的匹配结果矩阵将会依次打印在终端上
```

- 内置的分组和角色分配函数有一个可能的缺点是它只能按P1、P2、P3的顺序进行分配，如果在实验室中对固定的电脑设定固定的编号的话（设定方式参考下面正式实验操作一节中的程序准备内容），这样造成的结果是：1号电脑总是同一个角色，2号电脑总是另一个角色，如果固定匹配的话，1号电脑和2号电脑总是分在一组。另一方面，如果想要实现另外的一些匹配方式也需要进行自行设定随机和分配方式

• 自定分组与角色分配方式

- 要进行自定义的分组方式，首先要理解oTree以什么作为分组的依据，或者说分组函数以什么作为输入值
- oTree进行分组时依赖于方法set_group_matrix()，这个方法以matrix作为输入值，但是python里面并没有“矩阵”这样的数据结果，所以这里的matrix实际上是一个嵌套的列表，比如：[[1,2],[4,3]]
- 在角色分配上，id_in_group是在分组后自行按顺序分配的，以[[1,2],[4,3]]这个分组矩阵为例，第一个子列表中的第一个元素是1，所以P1的id_in_group是1，第二个子列表中第一个元素是4，所以P4的id_in_group是1。所以只要确定好分组矩阵，就决定好了分组和角色分配方式
- 因此所有的分组和角色分配的问题归结为，如何获得一个分组矩阵？用python的语言来说，如何获得一个嵌套列表？我们以上面提到的经典分配方式为例，手动实现。以下的代码例子仅作参考，熟悉python有更优更简洁的写法
- 利用内置的group、id_in_group和role字段的好处是有一些内置的方法可以就是以这些字段作为输入的（比如找到某个组的人、找到某个角色的人，具体可参考官方文档Multiplayer games一节下面的Group小节）
- 固定匹配和固定角色的情况下，只需要在第一轮做好随机分配即可，后续轮次复制第一轮的结果即可，与内置方法的差异是，P1不一定和P2匹配：

```
1 def creating_session(subsession):
2     import random #python中用于生成随机数的的一个包，一般python自带
3     #固定角色和固定搭配的情况下，只需要在第一轮做好随机分配即可，后续轮次复制第1轮的结果即可，与内置方法的差异是P1不一定和P2匹配了
```

```

4     if subsession.round_number == 1:
5         num_participant = len(subsession.get_players()) #获得实际的参
        加人数, get_players返回的是一个列表
6         participant_list = [ x for x in range(1,num_participant+1)]
        #生成一个[1,2,3,4,5,6]这样的列表, 这个列表将用于后续的矩阵生成
7         random.shuffle(participant_list) #随机步骤, 将列表打乱
8         group_matrix = [] #生成一个空的列表, 往其中添加元素生成矩阵
9         for i in range(1,int(num_participant/C.PLAYERS_PER_GROUP) +
        1): #要分几个组, 就往其中填入几个空列表作为元素
10            group_matrix.append([])
11            i += 1
12            i = 0
13            for li in group_matrix: #对group_matrix里面的每个空列表进行循环,
        往其中依次填入数字
14                while len(li) < C.PLAYERS_PER_GROUP: #在空列表中的元素个数没
        达到小组人数时依次往里面填入数字, 达到小组人数时停止填入, 继续往下一个列表填入
15                    li.append(participant_list[i])
16                    i += 1
17            subsession.set_group_matrix(group_matrix) #所有的空列表都填充完
        成, 分组矩阵已生成, 使用set_group_matrix进行分组
18        else:
19            subsession.group_like_round(1) #通过group_like_round()复制第1
        轮的分组结果
20        print(subsession.get_group_matrix()) #输出所有轮次分组匹配的结果
        matrix, 可以看到所有轮次中的matrix都是一样的

```

- 每轮随机匹配, 且不存在角色的情况下, 实际上是固定匹配中第一轮的随机分组代码每轮执行一次即可, 与内置方法没有特别大的差异:

```

1  def creating_session(subsession):
2      import random #python中用于生成随机数的的一个包, 一般python自带
3      #无固定角色+每轮随机匹配, 实际上就是所有被试都是相同角色然后进行随机匹配和上
        面的区别就是后续的轮次不用复制第一轮的结果, 每轮都是重新随机
4      num_participant = len(subsession.get_players())
5      participant_list = [ x for x in range(1,num_participant+1)]
6      random.shuffle(participant_list)
7      group_matrix = []
8      for i in range(1,int(num_participant/C.PLAYERS_PER_GROUP) + 1):
9          group_matrix.append([])
10         i += 1
11         i = 0
12         for li in group_matrix:
13             while len(li) < C.PLAYERS_PER_GROUP:
14                 li.append(participant_list[i])
15                 i += 1
16         subsession.set_group_matrix(group_matrix)
17         print(subsession.get_group_matrix())

```

- 每轮随机匹配, 但角色固定, 这个时候的重点在于如何实现随机分配角色, 并将不同角色进行匹配, 下面的方法与内置方法相比差异在于不再按顺序分配id_in_group:
- 在分配之前, 首先需要在C类中额外增加一个变量以指示角色数量:

```

1  R_NUM = 2

```

- 在Subsession类中增加一个字段用于储存角色列表:

- ```

1 class Subsession(BaseSubsession):
2 role_list = models.StringField()

```

- 具体的写法如下:

- ```

1 def creating_session(subsession):
2     import random
3     #有固定角色+每轮随机匹配，分配的重点在于随机分配角色，并将不同角色之间进行匹
    配，和内置方法相比不再按顺序分配id_in_group
4     num_participant = len(subsession.get_players()) #获得实际的参加人
    数，get_players返回的是一个列表
5     if subsession.round_number == 1: #这里的第一轮要随机生成的一个列表只是
    角色编号的列表，比如[2,1,1,2,1,2]
6         role_list = [x for x in range(1,C.R_NUM+1)] *
    int(num_participant/C.PLAYERS_PER_GROUP) #将[1,2]这样的列表按组数扩展
7         random.shuffle(role_list) #将角色列表随机排列
8         subsession.role_list = str(role_list) #将随机后的角色列表以字符
    型格式保存（为什么这样保存？）
9     else:
10        prev_subsession = subsession.in_round(1) #后续的轮次找到第1轮的
    subsession对象（为什么是对象？）
11        subsession.role_list = prev_subsession.role_list #将第1轮的
    subsession对象的role_list字段复制过来，就获得了角色分配列表
12        #####由于拿来的是字符串，带有括号和空格等字符，所以需要去除不必要的字符、拼合
    为列表、将列表的字符型的数字转换为数值型
13        role_list = subsession.role_list.strip("[")
14        role_list = role_list.strip("]")
15        role_list = role_list.replace(" ", "")
16        role_list = role_list.split(",")
17        role_list = list(map(eval,role_list))
18        #####
19        ##已经获得角色分配列表后，下一步是按角色进行匹配，思路是将被试按角色分为两个
    pool，每次匹配分别从两个pool拿出一个进行匹配
20        role_list_temp = [x for x in range(1,C.R_NUM+1)] #作为分组对照的列
    表[1,2]
21        role_matrix = [] #将角色分为两个pool，包含两个pool的矩阵
22        for i in range(1,C.R_NUM+1): #按角色数量先生成对应数量的空列表
23            role_matrix.append([])
24            i += 1
25        ##先分配好第一个角色pool，再分配第二个角色pool，另一种实现方法是将这个时候
    获得的role_matrix储存起来，同样需要考虑如何将字符串转化为列表
26        for j in range(0,C.R_NUM):
27            for i in range(0,num_participant):
28                if role_list[i] == role_list_temp[j]:
29                    role_matrix[j].append(i+1)
30            j += 1
31        ##每个pool里面进行随机
32        for j in range(0,C.R_NUM):
33            random.shuffle(role_matrix[j])
34            j += 1
35        ##用于分组的matrix
36        group_matrix = []
37        for i in range(1,int(num_participant/C.PLAYERS_PER_GROUP) + 1):
38            group_matrix.append([])
39            i += 1

```

```

40     ##按角色往分组matrix里面放入编号
41     i = 0
42     while i < int(num_participant/C.PLAYERS_PER_GROUP): #i表示组别
43         for j in range(0,C.R_NUM): #j表示角色编号
44             group_matrix[i].append(role_matrix[j][i]) #先往i组里放入角
色j, 再放入角色j+1
45             j += 1
46             i += 1 #i组中已放入j、j+1角色, i组的分配已完成, 分配i+1组, 直到分配完
所有组
47     subsession.set_group_matrix(group_matrix) #当轮分组矩阵已生成, 使用
set_group_matrix进行分组
48     print(subsession.get_group_matrix())

```

- 了解这些自定分组方式的目的在于了解oTree如何通过group_matrix设定分组, 另一方面通过这个例子也说明**如何将oTree编程的问题转化为一个python的问题**, 从python的角度思考如何解决问题, 更一般地说, 遇到了别的编程问题, 最重要的也是将问题转化为用编程语言表达 (python怎么实现某某功能/JavaScript如何实现某个功能等等), 弄清楚问题的本质才能问百度谷歌必应乃至ChatGPT
- 在creating_session里面定义的分组和角色分配会在session创建时 (即开始实验时) 就执行, 此时参加者还没有进行任何的决策。有的时候, 分组和角色分配是在session创建之后依据一定规则确定的 (比如参加者自行决定, 或依据某个决策进行分组), 这个时候就不应该在creating_session里进行, 而应该借助WaitPage

```

1 class ShuffleGroup(waitPage):
2     wait_for_all_groups = True
3
4     @staticmethod
5     def after_all_players_arrive(subsession):
6         #设定wait_for_all_groups之后, after_all_players_arrive需要
以subsession作为参数
7         subsession.group_randomly()
8         #各种分组代码等

```

- 从这里也可以看到, 要实现更灵活的分组和角色分配需要手动写分组代码设定所需要的group_matrix, 而这归根到底就是生成所需要的矩阵 (以列表为元素的嵌套列表)

四、正式实验操作

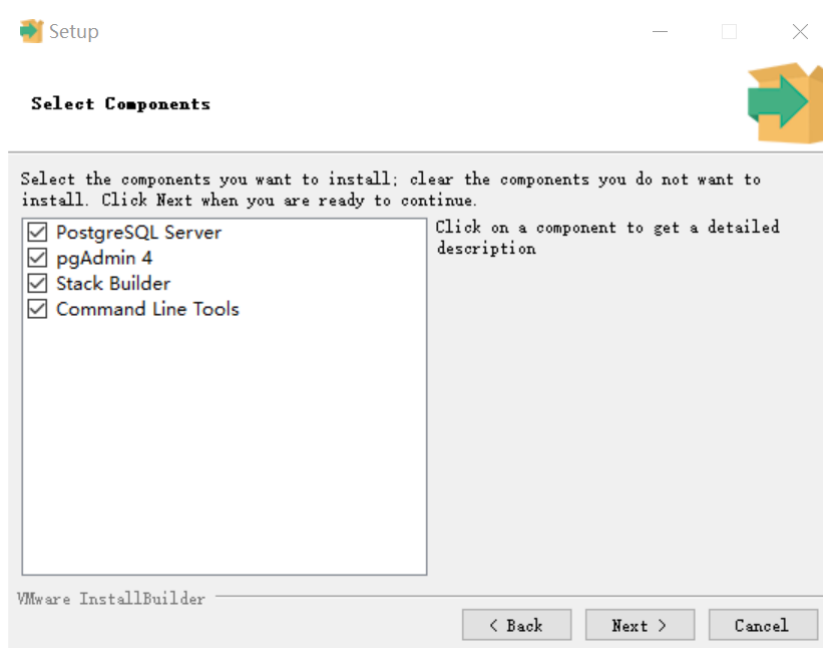
(一) 服务器架设

1. 安装数据库PostgreSQL和psycopg2包

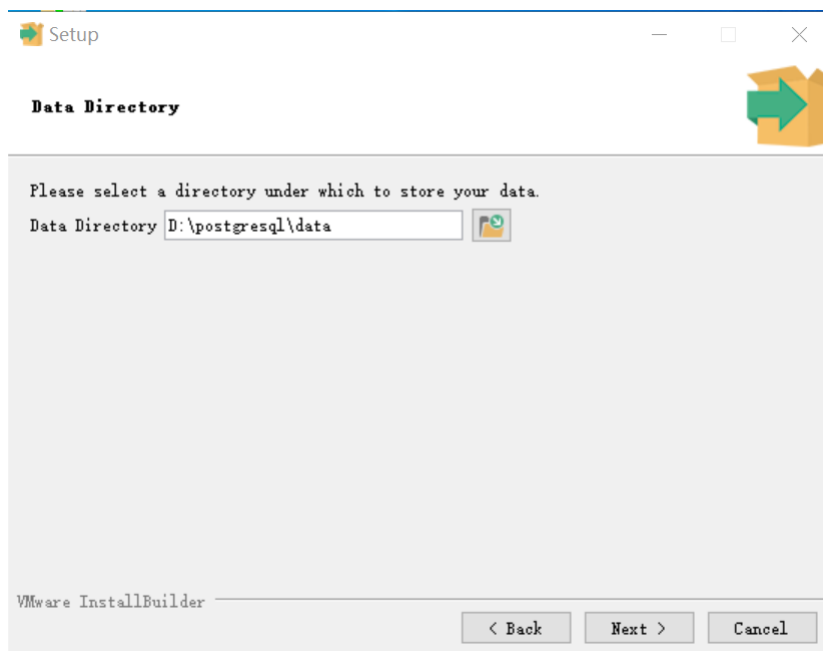
oTree自带的数据库是SQLite，据官方文档所说这个数据库在面对短时大量的访问时表现不好，因此建议换用更正式性能更好的PostgreSQL数据库。下面的例子中使用**14版本**的PostgreSQL。

- PostgreSQL的官方网站：<https://www.postgresql.org/>
- 下载地址：<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
- 安装要点：

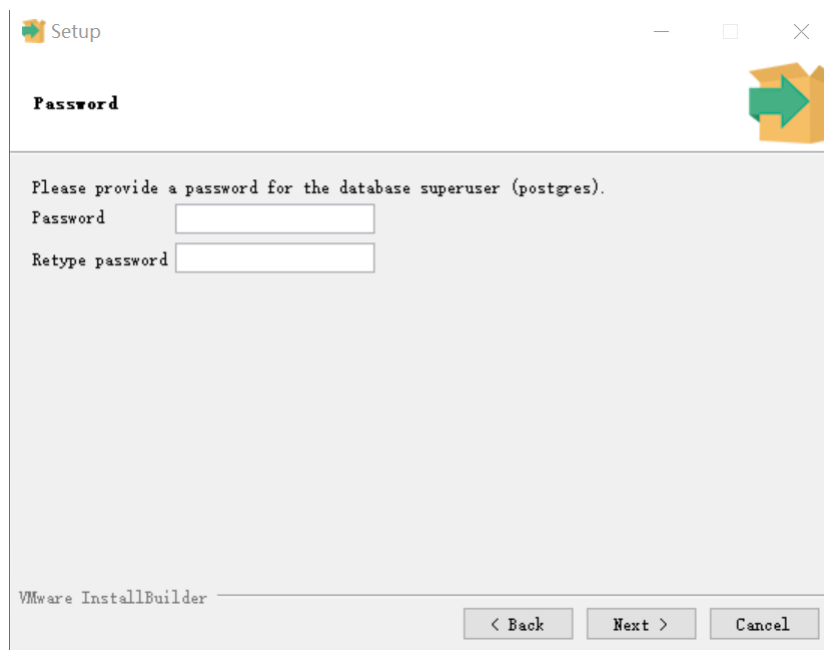
1. 所有可选的安装项都选上



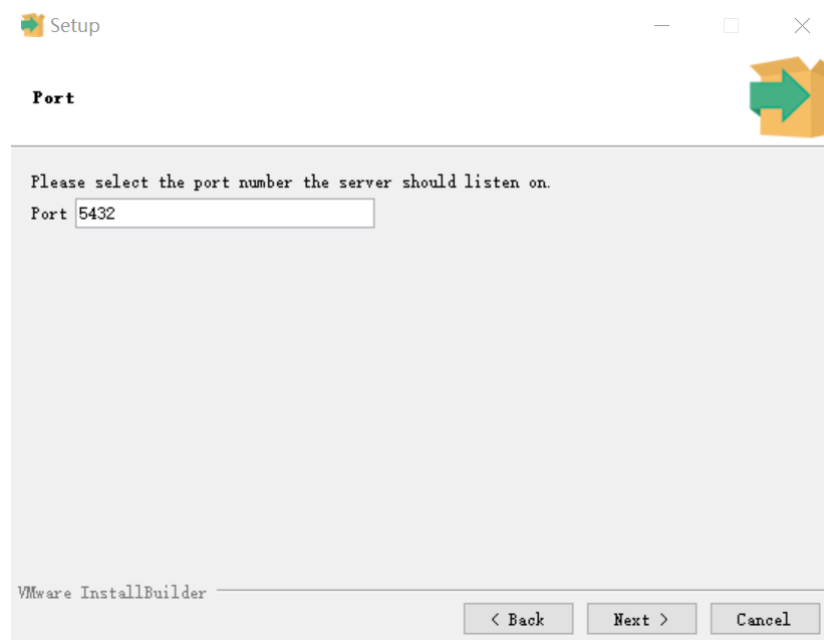
2. data文件夹目录和程序文件夹一致（默认）



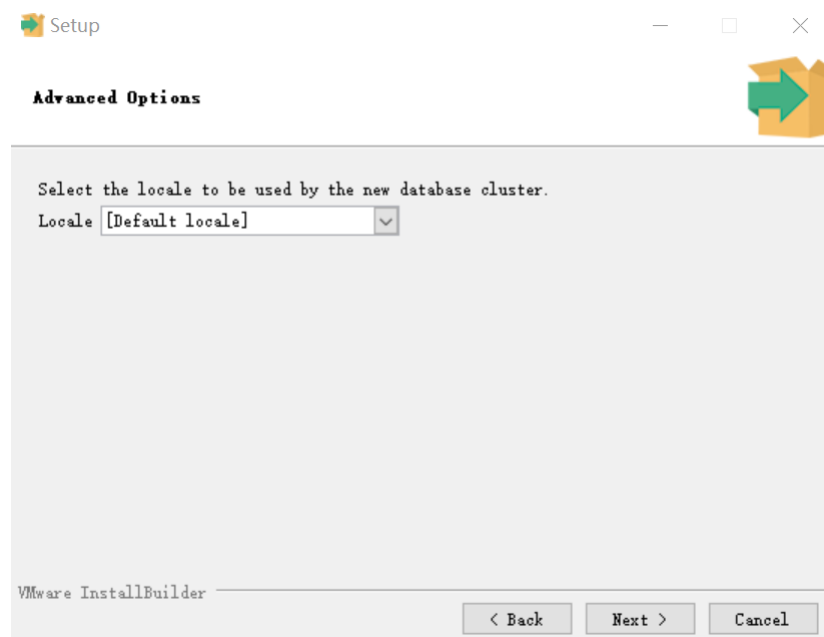
3. 设定superuser权限用户postgres的密码（重要）



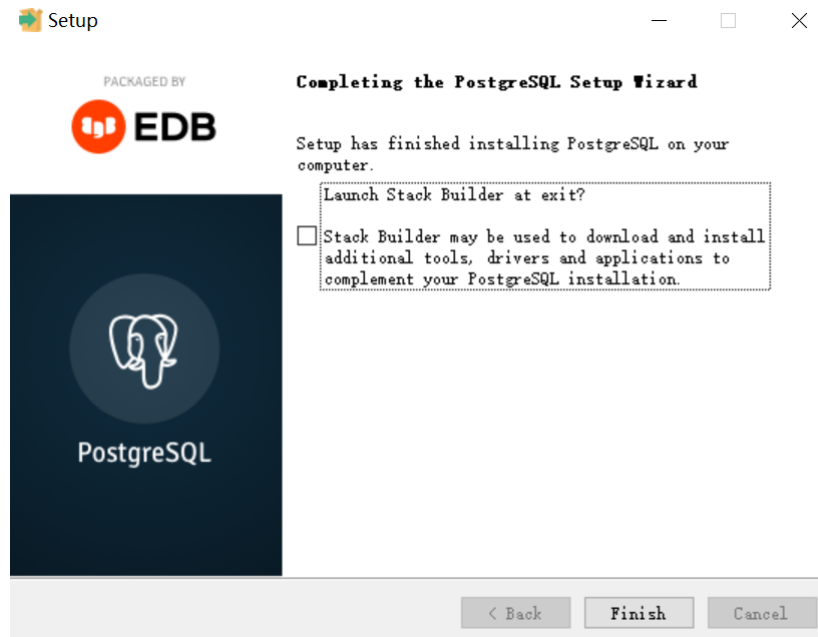
4. 端口保持默认5432即可



5. locale保持默认即可



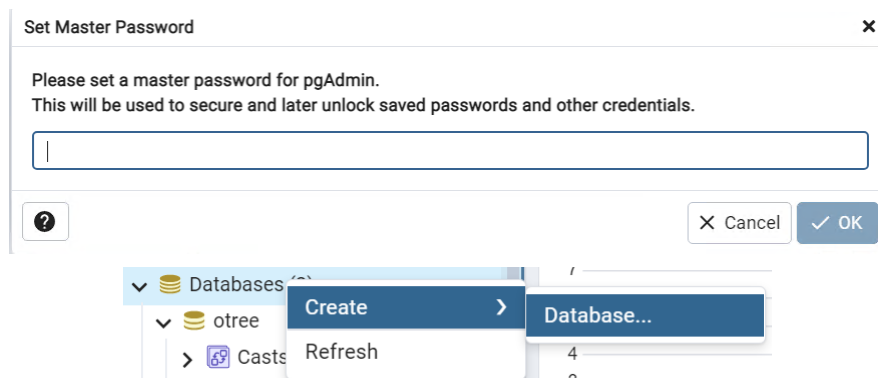
6. stack builder不用launch



2. psycopg2是python的一个与数据库进行交互的包，可直接使用pip命令安装： `pip install psycopg2`

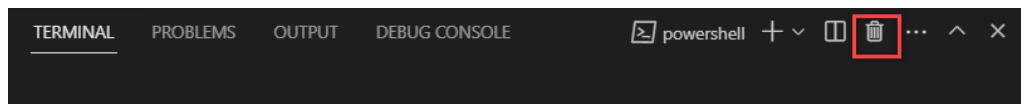
3. 新建数据库并修改环境变量

- 打开新安装的postgresql的管理界面（名字应该是pgAdmin4）后还会要求设置一个master密码，防止搞混设置为和superuser一样的密码即可。点开左边Servers，输入postgres用户的密码连接，在下面展开的菜单下找到DataBase，右键打开菜单选择Create→Database，输入新的Database的名字（例子中为otree），保存即可。



- 在修改数据库环境变量前，记得关闭所有的VSCode窗口，然后再去修改环境变量，如果在开着VSCode的时候修改环境变量，新修改的环境变量不会起作用，必须关闭后重新打开VSCode才能生效

- (可选) 以前似乎在关闭VSCode之前还需要关闭终端再重启VSCode才能生效，现在应该不用，如果需要关闭终端，点下图红框中的垃圾桶图标：



- 打开修改环境变量的窗口（参考上面安装python时的步骤），在用户变量（系统变量也可）里面添加新的变量DATABASE_URL，变量值结构为：postgres://postgres:<password>@localhost/<database name>，括号里面的内容填入自己设置的密码和数据库名称即可

- 修改完成后，打开VSCode中的终端或cmd，输入python启动python，依次输入如下命令：

```
1 import os
2 os.environ["DATABASE_URL"]
```

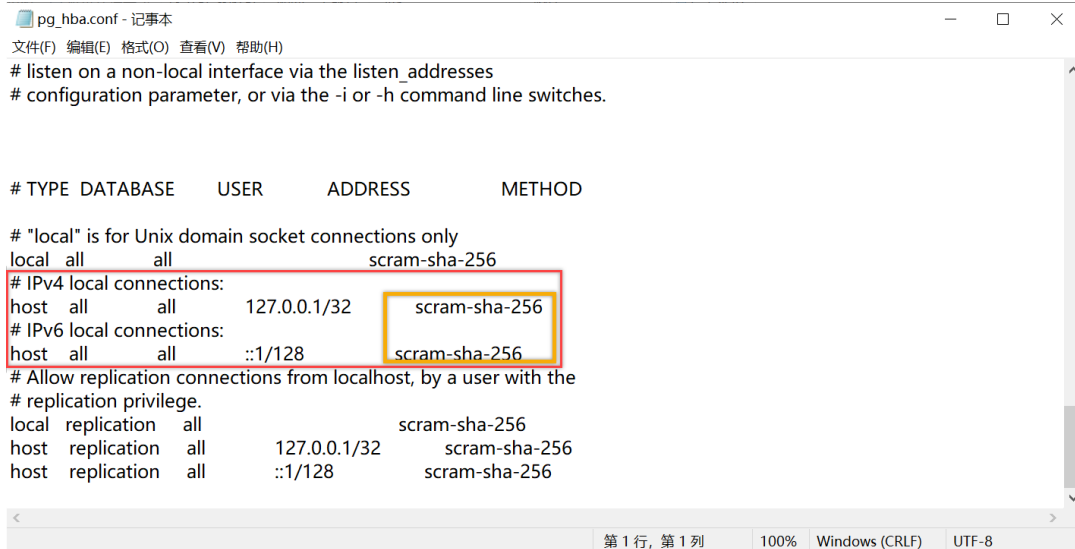
如果看到正确输出了刚刚设置的数据库环境变量（如下图），则正确设置和读取环境变量，然后输入quit()退出python即可

```
PS D:\sample_exp> python
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.environ["DATABASE_URL"]
'postgres://postgres:@localhost/otree'
>>> quit()
```

- 在VSCode终端或cmd中打开任意一个otree程序文件夹或将路径更改至某个otree程序文件夹，输入命令“otree resetdb”后按提示操作，Database engine显示“postgresql”即表示成功设置数据库

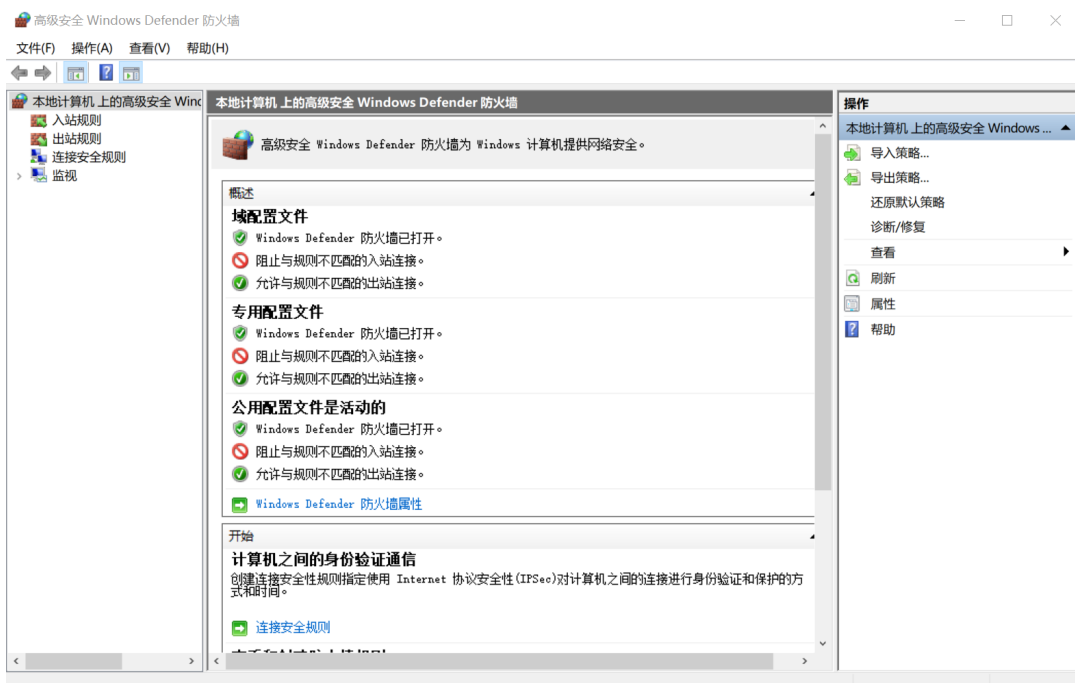
```
PS D:\newtest> otree resetdb
This will delete and recreate your database.
Proceed? (y or n): y
Database engine: postgresql
Created new tables and columns.
```

- (可选) 按上述步骤做下来可以正常运行，如果不能，之前的官方文档里面还建议尝试如下修改：在postgresql文件夹里面找到data文件夹下面的pg_hba.conf文件，文本文档格式打开，拉到最后IPv4和IPv6里面最后的METHOD的值都改成trust

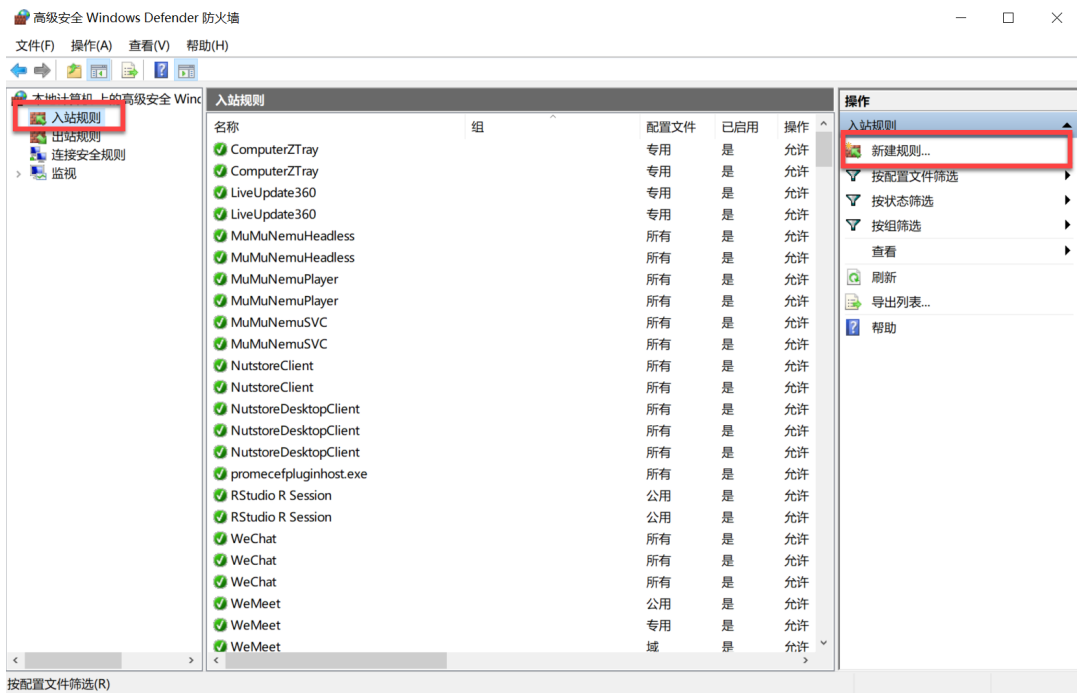


4. 端口入站规则修改

- 打开“高级安全Windows Defender防火墙”



- 点击左边“入站规则”，然后点击右边“新建规则”



- 规则类型选择“端口”

要创建的规则类型

☐ 程序(P)
控制程序连接的规则。

☒ 端口(O)
控制 TCP 或 UDP 端口连接的规则。

☐ 预定义(I):
@FirewallAPI.dll, -80200
控制 Windows 体验功能连接的规则。

☐ 自定义(C)
自定义规则。

- 规则应用于TCP，特定端口输入"80,8000"

此规则应用于 TCP 还是 UDP?

☒ TCP

☐ UDP

此规则应用于所有本地端口还是特定的本地端口?

☐ 所有本地端口(A)

☒ 特定本地端口(S): 80, 8000
示例: 80, 443, 5000-5010

- 下一页选择“允许连接”

连接符合指定条件时应该进行什么操作?

☒ 允许连接(A)
包括使用 IPsec 保护的连接，以及未使用 IPsec 保护的连接。

☐ 只允许安全连接(C)
只包括使用 IPsec 进行身份验证的连接。连接的安全性将依照 IPsec 属性中的设置以及“连接安全规则”节点中的规则受到保障。

☐ 阻止连接(K)

- 何时应用该规则都选上

何时应用该规则?

☒ **域(D)**
计算机连接到其企业域时应用。

☒ **专用(P)**
计算机连接到专用网络位置(例如, 家或工作单位)时应用。

☒ **公用(U)**
计算机连接到公用网络位置时应用。

- 最后一页的名称自定义 (例子为otree)

名称(N):
otree

描述(可选)(D):

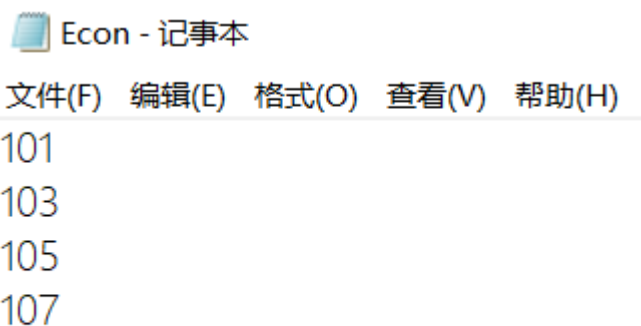
- 测试连接: 找到自己电脑当前的IPv4地址 (一般在当前网络连接的属性里面可以查看), VSCode终端或cmd打开任意一个otree程序文件夹或将路径更改至某个otree程序文件夹, 输入命令“otree prodserver [ip adress]: 8000”, 括号内填入自己的ip地址, 然后在浏览器输入地址“ [ip adress]: 8000”连接, 正常进入后台即连接成功, 后续可以用别的电脑或手机进一步测试

5. 以上是在windows系统的电脑或服务上的设置步骤, 如果是Linux系统的服务器, 参考另一个文档: [Linux Server old version-edited](#)

(二) 程序准备

1. 准备Rooms和Participant label

- 使用otree启动正式实验需要通过otree自带的Room功能, 可以理解为开启一个 (虚拟的) “实验室”, 每场实验都需要开启一次实验室, label就是这个“实验室”中的座位编号。label的作用就是给每个被试编号, 方便在后台观察被试进展, 也用于在实验网页被关闭的时候方便被试进入原来的页面接着完成实验 (相当于方便被试找到座位)。
- 创建Rooms首先在程序文件夹里面新建一个文件夹名为“_rooms”, 在里面新建文本文档, 写入设定的参加者的label作为label file, 如下图例子:



- 在settings.py里面加入ROOMS的设定, 如下图例子, 一个room设定是一个字典dict:

```

1 ROOMS = [
2     dict(
3         name = 'Econ',
4         display_name = 'Econ',
5         participant_label_file = '_rooms/Econ.txt',
6         #use_secure_urls = True,
7     )
8 ]

```

- name是显示在浏览器网址栏的名字，可修改
- display_name是显示在oTree后台的Rooms列表的名字，可修改
- participant_label_file是label file的路径
- use_secure_urls可以给显示的网址进一步加密，可选用，参考下面的例子

```

1 #未加密
2 http://localhost:8000/room/econ101/?participant_label=Student1
3 http://localhost:8000/room/econ101/?participant_label=Student2
4 #加密
5 http://localhost:8000/room/econ101/?
6 participant_label=Student1&hash=29cd655f
7 http://localhost:8000/room/econ101/?
8 participant_label=Student2&hash=46d9f31d

```

- 以上完成了rooms和label的设置。下一个问题是，在启动实验后，oTree后台不是按照label的顺序排列被试，而是按照正式启动实验后，连接的先后顺序排列被试。以101、103、105、107被试为例，我们希望在后台他们按照这个顺序连接，但有可能107号被试先连接上服务器进而出现在后台Monitor的第一位。因此需要额外加一段代码，使得被试在后台是按照label的顺序排列的，代码如下：

```

1 def creating_session(subsession):
2     labels = [101,103,105,107]
3     for p,label in zip(subsession.get_players(),labels):
4         p.participant.label = label

```

这一段代码需要添加在实验程序运行的**第一个app的__init__.py**文件中，不用放在某一个类里面。

- **(可选)** 在进行多场次实验的时候，还可以在creating_session中添加的内容是session.label。这是因为，oTree启动每一个session时会随机生成一串code作为session.code，这个值可以区分不同场次，但是不够直观，而session自带了label字段。因此可以在creating_session中指定session.label，这样下载的数据中就带有直观的label作为区分。这里获取不同场次开始实验的时间并格式化，作为session的label

```

1 def creating_session(subsession):
2     import datetime as dt
3     session = subsession.session
4     now = dt.datetime.now()
5     format_date = now.strftime("%Y-%m-%d %H:%M:%S")
6     session.label = format_date

```

2. 关闭debug mode

- debug mode是在测试的时候使用的，会在每个页面下显示当前页面的相关变量信息，以及出问题的时候提示具体信息，在正式实验的时候需要关闭
- 关闭方式是在实验程序的settings.py文件中赋值：

```
1 | DEBUG = False #改为True即开启debug mode
```

3. (可选) 实验后台登录密码

- otree启动后在同一网络环境中的设备都可以输入相应的服务器ip连接后台，添加后登录密码后在连接后台后必须输入密码才能使用后台的功能，但这不是必须要设置的
- 在settings.py中有如下两个变量：

```
1 | ADMIN_USERNAME = 'admin'
2 | ADMIN_PASSWORD = environ.get('OTREE_ADMIN_PASSWORD')
```

第一个变量设定了后台管理的用户名，第二个设定了登录密码

- 登录密码可以在环境变量中添加OTREE_ADMIN_PASSWORD设置，变量的值即为登录密码
- 还可以设置程序的权限等级：在环境变量中添加OTREE_AUTH_LEVEL，正式实验可设置为STUDY，网络公开供其他人参考的程序样例一般会设置值为DEMO

(三) 启动实验

1. 启动实验的命令

启动实验前，重置一下数据库（关于数据库和数据的保存的具体内容，参考第五节）：

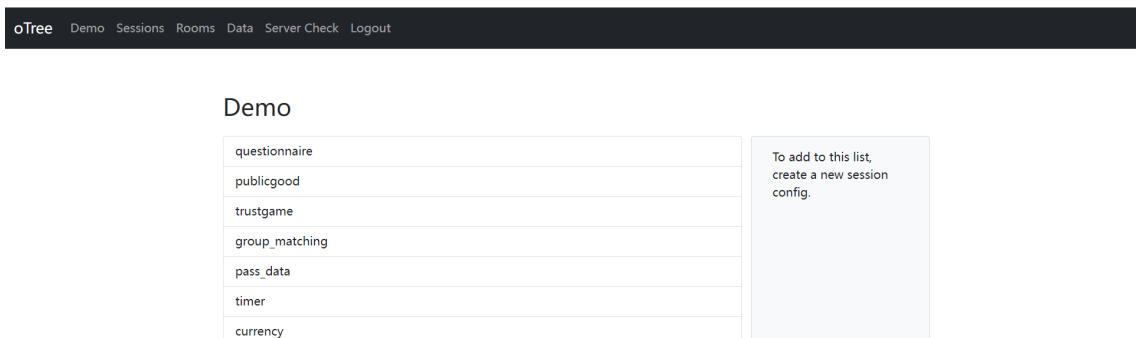
```
1 | otree resetdb
```

在完成所有准备工作后，在VSCode或cmd终端里面，将工作路径改为程序文件夹，输入启动正式实验的命令：otree prodserver [ip address]:8000，参考下面的例子，注意修改ip地址

```
1 | otree prodserver 192.168.1.1:8000
```

正常启动后，在电脑浏览器上输入网址即可连接后台，如：192.168.1.1:8000，如果先前设置了后台登录密码，输入密码登录进入后台，可以看到如下所示的界面：

- 在实验所使用的网络环境中，别的电脑、手机、平板等网络设备同样可以输入后台网址连接后台，不一定需要在作为主机的电脑上连接后台



2. 检查服务器

检查服务器，点击上方菜单的Server Check，尤其注意DEBUG mode和数据库的情况，保证DEBUG mode已关闭，数据库为Postgres

Server Readiness Checks

You have the latest version of oTree (5.10.3).

DEBUG mode is off

Password protection is on. Your app's AUTH_LEVEL is STUDY.

Postgres is configured.

3. 通过Rooms启动实验

- 记住不能点击DEMO列表启动实验，这样不是正式的实验session，点击DEMO列表会马上启动demo session，这样数据文件里会有来自demo session的数据行
- 点击上方菜单的Rooms，可以看到已设定的Rooms列表，点击

oTree Demo Sessions Rooms Data Server Check Logout

Rooms

Current rooms:

Econ

- 进入Rooms的设定界面，上面可设置人数，人数和_inti_.py文件中设定的PLAYERS_PER_GROUP有关，需要是该值的倍数，先不要点击Create，等待连接

Room: Econ

Create a new session

Session Config

contest ▾

Number of participants

|

Must be a multiple of 1

Create

- 设定界面的下方是被试连接情况以及用于连接的网址，有专用网址和房间网址，使用专用网址不用手动输入label，使用房间网址则需要手动输入编号，为方便分发，可以使用房间网址
 - 从下图例子中可以方便地看到101和105已连接，103和107未连接
 - 如果使用手机、平板等设备方便扫描二维码的话，可以使用插件将房间网址转化为二维码进行分发，扫描二维码后即可进行连接（参考谷歌浏览器插件Quick QR，安装后在房间网址上右键菜单中可生成二维码）

2 participants present

▼ Show/Hide

101 105

2 participants not present

▼ Show/Hide

103 107

Persistent URLs

These URLs will stay constant for new sessions, even if the database is recreated.

Participant-specific URLs

These URLs contain the labels, so participants don't have to enter their label manually.

▼ Show/Hide

http://10.189.214.208:8000/room/Econ?participant_label=101

http://10.189.214.208:8000/room/Econ?participant_label=103

http://10.189.214.208:8000/room/Econ?participant_label=105

http://10.189.214.208:8000/room/Econ?participant_label=107

Room-wide URL

Here is the room-wide URL anyone can use. Users will be prompted to enter their participant label, which will be validated against your participant_label_file.

<http://10.189.214.208:8000/room/Econ>

4. 被试端操作

- 如果使用房间网址连接，首先进入如下输入页面，需要手动输入label，这两个内置页面需要修改才能显示为中文：在settings.py中的LANGUAGE_CODE变量赋值“zh-hans”

欢迎！

请输入你的参与标签。

下一页

- 输入label后或直接使用专用链接，此时实验员没有点击Create的话则进入如下的等待界面

请等待

等待会话开始

5. 正式启动

- 连接完成后点击Create，启动实验session
- 如果已按前面的步骤设定了label，则在Monitor中可以看到所有被试按label排列，可以方便地通过Progress、App、Round、Page name等列观察被试进展

contest: session 'ijag9yzo'

 Edit  Links  Monitor  Data  Payments  Description

	Code	Label	Progress	App	Round	Page name	Waiting for	Time
P1	6fgk1blb	101	1/4	contest	1	DEMOPage		1m
P2	oxnhkv7	103	1/4	contest	1	DEMOPage		1m
P3	dtp9uott	105	1/4	contest	1	DEMOPage		1m
P4	co5zdh3c	107	1/4	contest	1	DEMOPage		1m

4/4 participants started.

Advance slowest user(s)

- Advance slowest user(s)在正式实验中不要点，会让被试强制跳过当前页面，即使有未输入的决策，这会造成数据缺失
- 实验完成后，重新点进Rooms中刚刚启动的room可以看到下图，点close this room即表示房间关闭实验结束，后续同一实验的其他场次可以使用同样的room设置启动新的场次，每一个room设定同时只能有一个active session，可以理解为实验室被占用

Room: Econ

[Go to active session.](#)

Close this room

- 当然也可以一个场次给一个room的设置，在settings.py的ROOMS加入更多room设定即可

(四) 问题处理

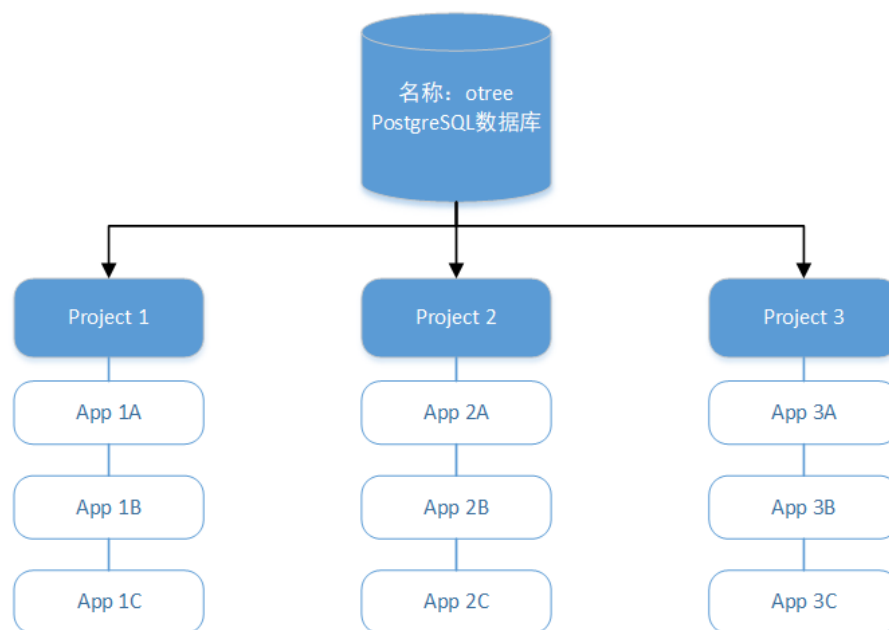
- oTree的后台运行非常稳定，只要服务器运行正常网络不中断即可。即使主机中断（Ctrl+C）了，及时重启也能继续实验，重新启动后通过菜单Rooms→Econ仍可以看到在进行的实验（active session），点击“Go to active session”进入即可
 - 这也说明，如果出现的bug不涉及数据库的改变（比如增加或减少字段），一些网页显示上的bug可以及时处理
 - 由于正式实验用到了数据库，因此中途换主机非常麻烦，所以一定要保证主机正常
 - 如果真的发生了问题主机需要转移，则需要参考如下的步骤
 - 将原主机上的数据库备份导出（导出步骤参考下面的数据管理部分），和实验程序一起转移到一个新电脑上，这个新电脑同样应该配置好python、oTree、PostgreSQL、环境变量等服务器相关配置
 - 打开PostgreSQL的管理界面pgAdmin，新建数据库，将原数据重载至新数据库中（重载操作同样参考下面数据管理部分）
 - 修改新电脑的环境变量，确保数据库名字是刚刚新建的已重载数据的数据库
 - 重新启动实验程序，通过Rooms菜单进入active session，后台的数据应已正常导入，可以继续实验
 - 如果使用的是局域网，尽可能地把新电脑的ip地址改为和原来的ip地址一样
 - 如果无法修改新电脑的ip地址，则被试端应该重新输入网址和label进入
- 如果被试中途出现断网或掉线的情况，重新联网后刷新页面或在浏览记录里点开原链接即可
- 如果被试关掉了实验网页，就重新进入room链接，输入label，即可继续原来中断的实验

- 实验链接不识别机器，被试设备中途出故障换用其他机器同样可以通过输入链接重新连接继续实验，这也是room和label的重要作用

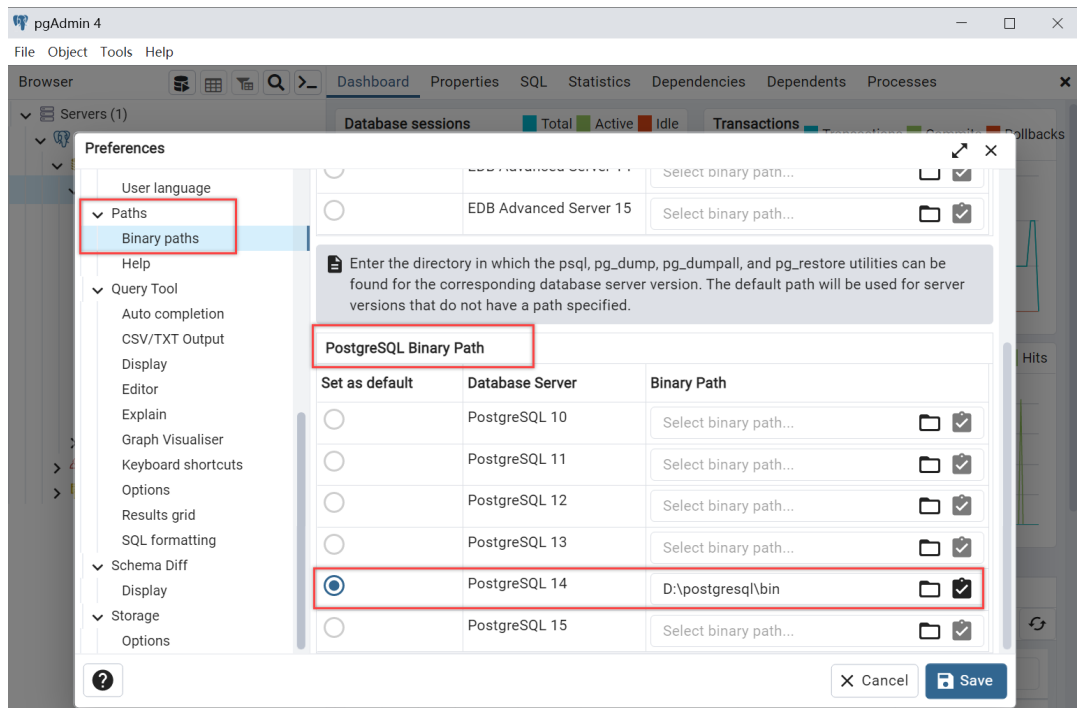
五、数据保存与基础清洗

(一) 数据管理

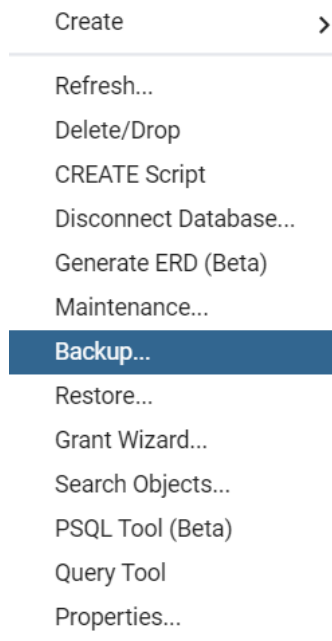
- oTree使用了两种数据库：PostgreSQL和SQLite，分别用于正式实验和测试debug
 - 测试数据库是基于SQLite的，测试程序时生成的数据库文件是程序文件夹下面的db.sqlite3文件，删除这个文件后再启动测试服务器又会新建这个文件，即完成测试数据库的重建
 - 启动实验前重置的是用于正式实验的基于PostgreSQL的数据库，下面讲的数据管理也是针对这个正式实验的数据库
- 考虑如下图所示的情况，我们有一个名字叫otree的正式实验数据库，然后我们三个实验Project，每个实验下又有若干个app



- 在第一天我们首先进行了Project1的正式实验，通过Room启动了正式的Session。这个实验由三个App（实验任务）组成，在运行完上午的场次后，数据库otree里面已经保存了上午场次的数据，而下午将进行第二个场次。
- 对同一个实验的不同场次，中途不需要重置数据库，下一场（session）的数据会作为新数据增加在数据文件末尾，等所有场次做完后，所有场次的数据都在一个csv文件里。**（当然保险起见可以做完一场下载一次数据）
- 完成Project1的实验后，所有的正式数据都存在数据库中。
- 在第二天我们要进行Project2的正式实验，在进行新的project之前，可以**进行备份数据库然后重建数据库的操作**，这样Project2会将数据存放在一个新的空数据库中
- 同理在第三天进行Project3的正式实验时也进行这样的操作
- 总结一个**建议就是，在进行同一个实验的不同场次时不用重置数据库，在进行不同实验之前确保下载好和备份好数据并重置数据库**
- (可选)** 备份 (Backup) 和重载 (Restore)数据库的操作
 - 这部分操作涉及到PostgreSQL的操作，**不是必须的**，其中有一些SQL的细节也没有完全清楚
 - 要进行备份和重载操作，首先需要做如下设定：左上角File菜单→Preference，在弹出的窗口中下拉找到Paths选项，在Binary paths子选项中找到PostgreSQL Binary Path，我们使用了14版本，所以在该行的Binary Path中选择安装文件夹下面的bin文件夹，并保存。



- 备份操作：打开pgAdmin4，在原有的实验数据库（上面的例子中这个数据库叫otree）上点右键，在右键菜单中选择Backup



- 在弹出的框中选择保存的路径和文件名，格式Format选择Custom或Tar均可，点击右下角Backup即完成备份

Backup (Database: otree)

General Dump options

Filename: C:\Users\Marvin\Documents\data_sample_custom

Format: Custom

Compression ratio:

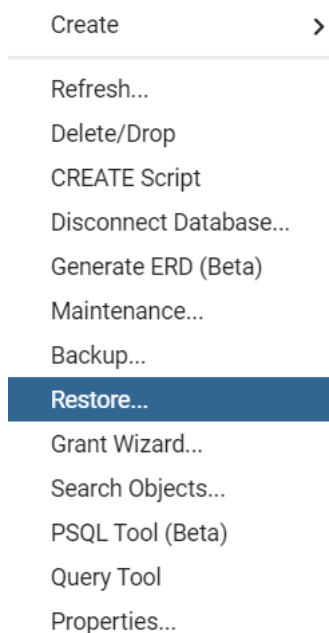
Encoding: Select an item...

Number of jobs:

Role name: Select an item...

Buttons: [i] [?] [X Cancel] [Backup]

- 重载操作：打开pgAdmin4，新建一个数据库（上面的例子中数据库名字叫otree，假设新建的数据库名字叫otree_backup），**注意这个数据库必须是新建的空白的，不能被用于启动任何的oTree实验**。右键数据库，在右键菜单中选择Restore



- 在新弹出的窗口中Format选择Custom or tar，Filename选择数据库备份文件，点击右下角Restore即完成重载

Restore (Database: otree_backup)

General Restore options

Format

Custom or tar

Filename

C:\Users\Marvin\Documents\data_sample_custom

Number of jobs

Role name

Select an item...

i

?

Cancel

Restore

- 修改环境变量中的DATABASE_URL（修改方法和注意事项见前文），将原来的数据库名字改成刚刚重载的数据库（在例子中是将otree改成otree_backup），然后重新打开VSCode并启动（prodserver）原来的实验Project，即可在后台的Data菜单重新下载数据

(二) 数据下载与文件类型

- 点击最上方的菜单中的Data标签可以进入数据下载页面
- 在页面上可以看到三个不同的数据文件，分别是实验project中所有app中的数据都整合为一个文件的所有app-wide数据文件、project中不同app数据分开储存的per-app数据文件、以及参加者在不同页面上所耗时间的数据，点击Excel或Download就可以下载对应的数据文件，数据文件都是CSV格式，可以用Excel打开，文件名后带有下载日期

All apps

[Excel](#) | [Plain](#)

Data for all apps in one file. There is one row per participant; different apps and rounds are stacked horizontally. This format is useful if you want to correlate participants' behavior in one app with their behavior in another app.

Per-app

These files contain a row for each player in the given app. If there are multiple rounds, there will be multiple rows for the same participant. This format is useful if you are mainly interested in one app, or if you want to correlate data between rounds of the same app.

questionnaire	Excel	Plain
trustgame	Excel	Plain

Page times

If this data is important to you, make sure to download it before restarting the server. Otherwise, you might lose the latest observations.

[Download](#)

- All apps的文件中，每个参加者只占一行，所有app中的数据都水平地拼接在一起，同一个app有多个轮次的也是水平地拼接在一起

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	session.co	session.lat	session.ms	session.ms
2	1	k5lp75zg	103	0	32	32	questionn	Results	00:20.6	1			640	k9iugz2q				
3	2	x4bg3yf1	101	0	32	32	questionn	Results	00:20.6	1			380	k9iugz2q				
4																		
5																		
6																		
7																		
8																		

- Per-app文件中，每个app的数据单独存放，每一行是一个参加者在该轮的数据，如果同一个app有多轮，则一个参加者有多行的数据，每一行来自一个轮次

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	participant	player.id_i	player.role	player.pay	group.id_i
2	1	k5lp75zg	103	0	32	32	questionn	Results	00.20.6	1			640	1		120	1
3	2	x4bg3yf1	101	0	32	32	questionn	Results	00.20.6	1			380	2		80	1
4	1	k5lp75zg	103	0	32	32	questionn	Results	00.20.6	1			640	1		120	1
5	2	x4bg3yf1	101	0	32	32	questionn	Results	00.20.6	1			380	2		100	1
6	1	k5lp75zg	103	0	32	32	questionn	Results	00.20.6	1			640	1		130	1
7	2	x4bg3yf1	101	0	32	32	questionn	Results	00.20.6	1			380	2		110	1
8	1	k5lp75zg	103	0	32	32	questionn	Results	00.20.6	1			640	1		170	1
9	2	x4bg3yf1	101	0	32	32	questionn	Results	00.20.6	1			380	2		90	1
10	1	k5lp75zg	103	0	32	32	questionn	Results	00.20.6	1			640	1		100	1
11	2	x4bg3yf1	101	0	32	32	questionn	Results	00.20.6	1			380	2		0	1
12																	
13																	

- Page times文件中记录了每个参加者离开某一页面进入下一页面的时刻（时间戳），这个时刻是相对1970年1月1日这个计算机时间原点经过的秒数，相邻的两个时刻相减就得到在该页面上花费的时间

	A	B	C	D	E	F	G	H	I	J	K	L
1	session_cd	participant	participant	page_inde	app_name	page_name	epoch_time_completed	round_nu	timeout_h	is_wait	page	
2	k9iugz2q	1	k5lp75zg	0		InitializePe	1679976020	1	0	0		
3	k9iugz2q	2	x4bg3yf1	0		InitializePe	1679976020	1	0	0		
4	k9iugz2q	1	k5lp75zg	1	trustgame	Introducti	1679976064	1	0	0		
5	k9iugz2q	2	x4bg3yf1	1	trustgame	Introducti	1679976066	1	0	0		
6	k9iugz2q	1	k5lp75zg	2	trustgame	Send	1679976072	1	0	0		
7	k9iugz2q	1	k5lp75zg	3	trustgame	SendBackl	1679976072	1	0	1		
8	k9iugz2q	2	x4bg3yf1	3	trustgame	SendBackl	1679976072	1	0	1		
9	k9iugz2q	2	x4bg3yf1	4	trustgame	SendBackl	1679976077	1	0	0		
10	k9iugz2q	1	k5lp75zg	5	trustgame	ResultsWa	1679976077	1	0	1		
11	k9iugz2q	2	x4bg3yf1	5	trustgame	ResultsWa	1679976077	1	0	1		
12	k9iugz2q	1	k5lp75zg	6	trustgame	Results	1679976080	1	0	0		
13	k9iugz2q	2	x4bg3yf1	6	trustgame	Results	1679976081	1	0	0		
14	k9iugz2q	1	k5lp75zg	8	trustgame	Send	1679976086	2	0	0		

- 比如在A时刻离开Introduction页进入第一次决策页，B时刻完成第一次决策进入等待页，B-A就得到第一次决策花费的时间
- 如果在页面上花费的时间很重要，不建议使用这个Page times文件。建议是在实验中使用python的time包等处理时间的包记录时间在某个字段中，以保存在数据CSV文件里方便分析，处理时间的方式也是获取前后两个时间戳并相减。
- 如果在页面上停留的时间并不重要，这个文件意义不大。所以无论怎样，这个文件都不是很重要
- Chris提供了一个可以用PageTimes文件计算每页停留时间的python程序（pypro文件夹里的pagetimes.py），参考里面的注释即可使用

(三) 基础的数据清洗：otree2stata.py和otree2data.py

- oTree记录的数据的格式基本都是player.var这种格式，即变量名前面有类名或者有别的前缀，为了更好地在stata中处理，需要对变量名进行基础的修改清洗，这里介绍两个程序：otree2stata和otree2data
 - 注意：这两个程序本质上就是使用python的一些包（csv、argparse、pandas等）读取CSV文件并进行修改，这是python的重要应用之一，参考前面提到的python教程，熟悉之后可以自行修改
- 下面这样清洗后的数据导入Stata14可以正常使用无乱码，如果是Stata13或更早版本，可能存在中文编码的问题需要进一步修改

1.otree2stata的使用

- otree2stata是整理了一大堆oTree程序的Christian König gen. Kersting编写的，github：<https://github.com/chkgk/otree2stata>
- 这个程序会去除类的前缀，像"player.payoff"和"participant.payoff"这种去除前缀会混淆的变量则改为下划线，程序运行后原来的数据文件不做修改，生成新的有_stata后缀的csv文件，可以在stata中导入

- 使用方法：使用前将otree2stata.py和需要转换的csv文件放在一个文件夹里，并将工作路径修改至该文件夹，然后输入如下命令，格式为python+otree2stata.py+数据文件.csv：

```
1 python otree2stata.py questionnaire_2023-03-28.csv
```

- 程序正常运行后不会有其他提示，在同一文件夹内会生成有_stata后缀的csv文件，可在stata中导入数据：

```
1 import delimited E:\EXP\questionnaire_2023-03-28_stata.csv, encoding(UTF-8)
```

- 注意点①：作者提供的源程序在处理中文字符时会报错，需要在open()函数中指定encoding='UTF-8'才能正常读取有中文字符的数据，pypro文件夹里面的是已修改的程序：

```
1 with open(input_path, 'r', encoding='UTF-8') as f:
2     #.....
3 with open(output_filename, 'w', newline='', encoding='UTF-8') as f:
4     #.....
```

这样生成的_stata.csv文件直接用excel打开的话会看到中文是乱码，但是在导入stata时指定encoding参数（见上）就可以得到正确显示的dta文件

- 注意点②：otree2stata.py只能对单独的app数据文件使用，不能对整合所有app数据的所有_apps_wide文件使用，并且这个程序只改名字不删变量

2.otree2data的使用

- 这个程序的作者已经不可考
- 这个程序使用了pandas包里面的函数来读取和修改文件，在使用前需要确定是否已安装pandas的包
- 这个程序对整合的所有_apps_wide文件使用，使用后原来的数据文件同样不发生变动
- 使用方法：
 - 首先打开程序，修改程序里面的data和app_name两个变量，data中修改文件名为待读取修改的文件名，app_name中写app的名字，出现在app_name列表中的app的数据会被提取：

```
1 data = pd.read_csv('all_apps_wide_2023-03-28.csv', sep=',')
2 app_name = ['trustgame', 'questionnaire']
```

- 同样将程序和csv文件放在同一个文件夹中，输入命令即可：

```
1 python otree2data.py
```

- 运行完成后没有提示，会在同一文件夹内生成单独的app数据文件
- stata导入生成的csv文件即可，同样记得命令参数encoding(UTF-8)
- 注意：这个命令在重命名变量的同时也会删掉一些oTree内置的一些系统变量，仅保留与实验有关的变量