The University of Melbourne acknowledges the Traditional Owners of the unceded land on which we work, learn and live: the Wurundjeri Woi-wurrung and Bunurong peoples (Burnley, Fishermans Bend, Parkville, Southbank and Werribee campuses), the Yorta Yorta Nation (Dookie and Shepparton campuses), and the Dja Dja Wurrung people (Creswick campus).

The University also acknowledges and is grateful to the Traditional Owners, Elders and Knowledge Holders of all Indigenous nations and clans who have been instrumental in our reconciliation journey.

We recognise the unique place held by Aboriginal and Torres Strait Islander peoples as the original owners and custodians of the lands and waterways across the Australian continent, with histories of continuous connection dating back more than 60,000 years. We also acknowledge their enduring cultural practices of caring for Country.

We pay respect to Elders past, present and future, and acknowledge the importance of Indigenous knowledge in the Academy. As a community of researchers, teachers, professional staff and students we are privileged to work and learn every day with Indigenous colleagues and partners.

Programming for Experiments with oTree

27 – 28 May 2024

Laura Liu

THE UNIVERSITY OF MELBOURNE

Version date: 29 May 2024

Team introduction

- **Dr. Yang Liu (Laura)**
  - Lab manager/Research fellow at the University of Melbourne
- **Dr. Boon Han Koh (Boon)**
  - Lecturer at the University of Exeter
- **Muhammad Arslan Iqbal (Arslan)**
  - PhD Candidate at the University of Melbourne
- **Lingguo Xu (Lin)**
  - PhD Candidate at the University of Melbourne
- **Dang Song Nguyen Lam (Nguyen)**
  - **Honors graduate**

To know you more

- **A Survey**

- **Briefly introduce yourself**

Outline

MONDAY, 27 MAY 2024

9: 30 – 11:00	**Session 1** ▪ Introduction ▪ Experiment #1Planning
11:00 – 11:30	*Break*
11:30 – 13:00	**Session 2** ▪ Experiment #2 Planning
13:00 – 14:00	*Lunch*
14:00 – 15:30	**Session 3** ▪ O-Tree Structure ▪ GitHub basics
15:30 – 16:00	*Break*
16:00 – 17:00	**Session 4** ▪ Python, PyCharm and oTree installation

TUESDAY, 28 MAY 2024

10: 00 – 12:00	**Session 5** ▪ Converting Experiment #1 to oTree
12:00 – 13:00	*Lunch*
13:00 – 15:00	**Session 6** ▪ Converting Experiment #2 to oTree
15:00 – 15:30	*Break*
15:30 – 17:00	**Session 7:** ▪ Push experiments to Heroku using GitHub and PyCharm Wrapping up - where to find additional resources

THE UNIVERSITY OF MELBOURNE

Plan for experiment

Plan for experiment: pen and paper style

- **How to design an experiment for the following research question?**
  - Behaviour in common pool problem when endowment is ability/effort related

Public Good Game

- You are divided into groups of three
- Every players receive 10 tokens that must be divided between a **Private Account** and a **Common Account**
  - Each token in your **Private Account** is worth $1 to you but $0 to other group members
  - Each token in the **Common Account** is worth $0.6 to all members in your group

$$\textit{Your Payoff} = \$1 \times \textit{Private} + \$0.6 \times \textit{Total Common}$$

- After each round, you learn the total contribution to the Common Account (but not the contribution of each group member)

Plan for experiment: pen and paper style

- **How to design an experiment for the following research question?**
  - Behaviour in common pool problem when endowment is ability/effort related

- **Experiment design**
  - Public good game
  - Determine endowment
  - Rounds? Matching?
  - Payment?
  - Robustness? (reduce noise, control variables)

- **How to implement the design?**

Plan for experiment: pen and paper style

- **Describe the experiment chronologically in groups**

- Group size N = 3
- Multiplier = 1.8
- Rounds = 3
- Exchange rate 1 ECU = 0.5 AUD
- Payment = payment from experiment + show-up fee

Plan for experiment: pen and paper style

Here are the notes from the classroom discussion

- **Preparations by experimenter**
  - Instructions (printed)
  - Physical tokens
  - Envelopes
  - Worksheet for real-effort task
  - Cash
  - Survey (printed)
  - Decision and feedback (game results) sheet for public good game(printed)
  - Testing

Plan for experiment: pen and paper style

Here are the notes from the classroom discussion

- **Time of events during experiment**
  - Participants sign consent form and attention plague
  - Randomly assign settings (usually by participants drawing the seat number in the box)
  - Read the instructions and at the same time provide participants a copy so that they can read at any point of the experiment
  - Ask participants to answer comprehension questions and (or) practice round to check their understanding
  - Draw the random groups of 3 (draw a number from box)
  - Give out the worksheet for real-effort game
  - Collect answers and calculate the score for real-effort game

Plan for experiment: pen and paper style

Here are the notes from the classroom discussion

- **Time of events during experiment (Cont.)**
  - Give their score (and corresponding tokens) back as their endowment
  - Give worksheet for the public good game
  - Collect the worksheet with participants' contribution
  - Calculate group contribution
  - Give feedback (result) sheet back to participants
  - Calculate payment
  - Ask participants to complete survey while calculating the payment.
  - pay the participants in cash that is put in the envelope (double-blind)

Data Collection

What data do we need to collect for this game?

- Here is the summary of the classroom discussion
  - Real-effort questions
  - Real-effort scores (the number of question answered correctly)
  - Individual contribution to public good game
  - Group ID in public good game
  - Participant earnings from public good game
  - Parameters of the public good game (number of people per group, multiplier)
  - Participants' decision time

Pen and Paper

- **Pros and cons?**
  - Pros:
    - Easy to access
    - Everyone can use
    - Cost can be low (depends)
  - Cons:
    - Time-consuming
    - Requires more manpower
    - Mistakes can happen
    - Data is not easy to store and transfer
- **Pen and paper experiments are still often used in field experiments, computerized experiment is more popular in the lab.**

Computerize Experiment #1 and #2

- **What can we computerize?**
  - Experiment #1
  - Experiment #2

Why do I need to learn o-Tree?

- There might be ways to avoid programming by using apps developed by others.
  - Limited functionality, costly…
- It sounds like a lot to learn, but you will only need the basics.
  - Already written packages, the library can be found online
- You have absolute control over your experiments!
- Programming skills are transferable
  - Python, JavaScript are among the top in-demand skills
  - https://www.devjobsscanner.com/blog/top-8-most-demanded-programming-languages/

o-Tree Introduction

o-Tree

- **o-Tree is an open-source platform for web-based interactive tasks (**http://www.otree.org/)
  - Surveys
  - Individual decisions
  - Multiplayer games
  - Markets
- **o-Tree is a Python-based framework, but you need more to run experiments….**
  - Back end: Python
  - User interface: HTML, CSS, JavaScript (dynamic website)
  - Deploy: web server (Heroku)
  - Version control: Git

o-Tree versions

- The current version of oTree is 5.10.4
- Understand semantic versioning: X.Y.Z
  - X: Major versions of packages (quite different from each other)
    - ❖ Program written using oTree 3 will not run on oTree 5! (unless substantially changed)
  - Y: Minor versions (substantive change but incremental changes)
    - ❖ Program written using oTree 5.9.Z might need changes to be run on oTree 5.10.Z
  - Z: Micro versions (patches to fix bugs)
    - ❖ Programs written using oTree 5.10.3 usually do not require any change to be run on oTree 5.10.4
- It is a good idea to stick with the same major and minor versions, and only update the micro version

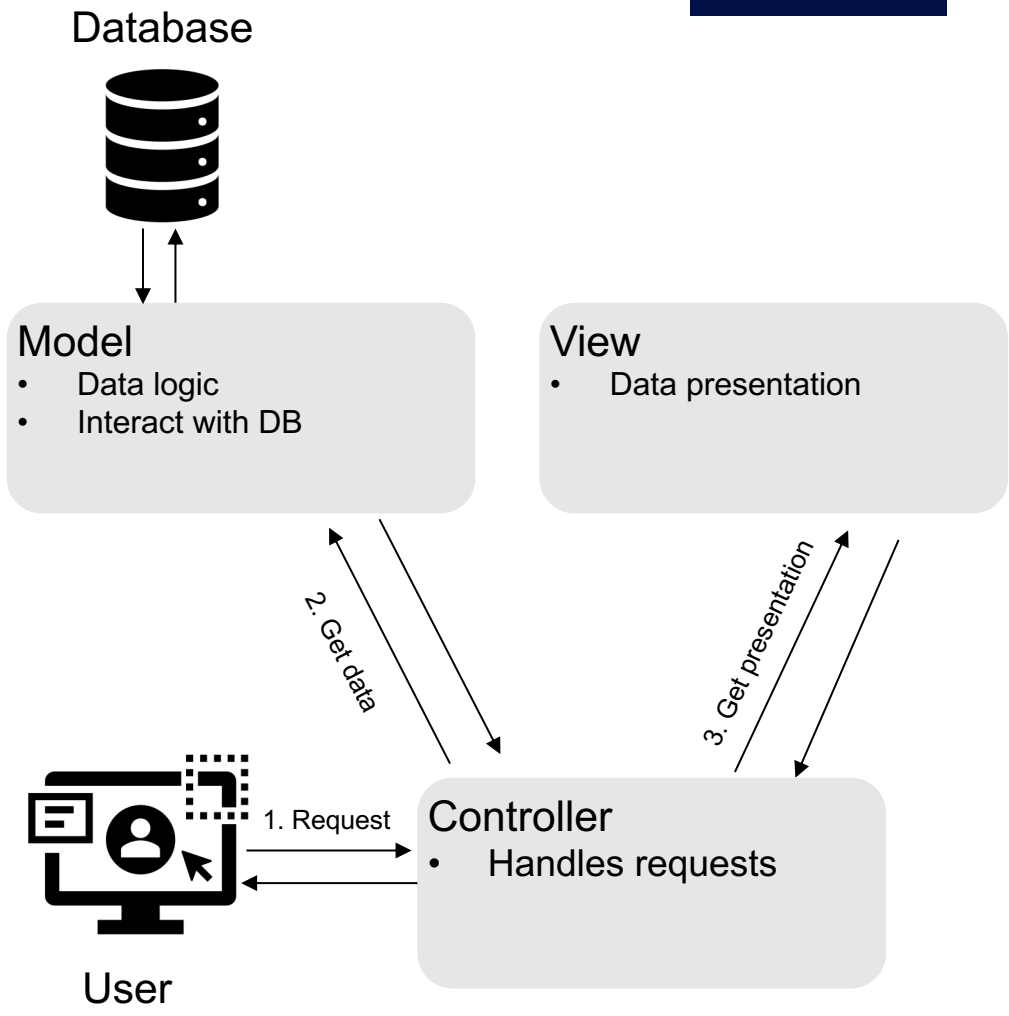
o-Tree 3 vs o-Tree 5

- o-Tree 3 was based on **Django**, a popular package for implementing webpages.
- o-Tree 5 is based on **o-Tree Lite**, a self-contained framework
  - Detailed changes can be found at https://otree.readthedocs.io/en/latest/misc/otreelite.html
- Why should I use o-Tree 5?
  - Easier installation
  - Compatible with more version of Python
- **The material in this workshop will be based on o-Tree 5**

o-Tree Structures

oTree conceptual framework

- oTree is based on the **Model-View-Controller** (MVC) framework
- MVC is a popular, if not the most popular, approach for building web applications
- MVC separates an application into three main components:
  - Model: manage data, logic
  - View: display data on screen (data presentation)
  - Controller: manage user interactions with Model and View
- What is the benefit of MVC?
  - Separation
  - Reusability



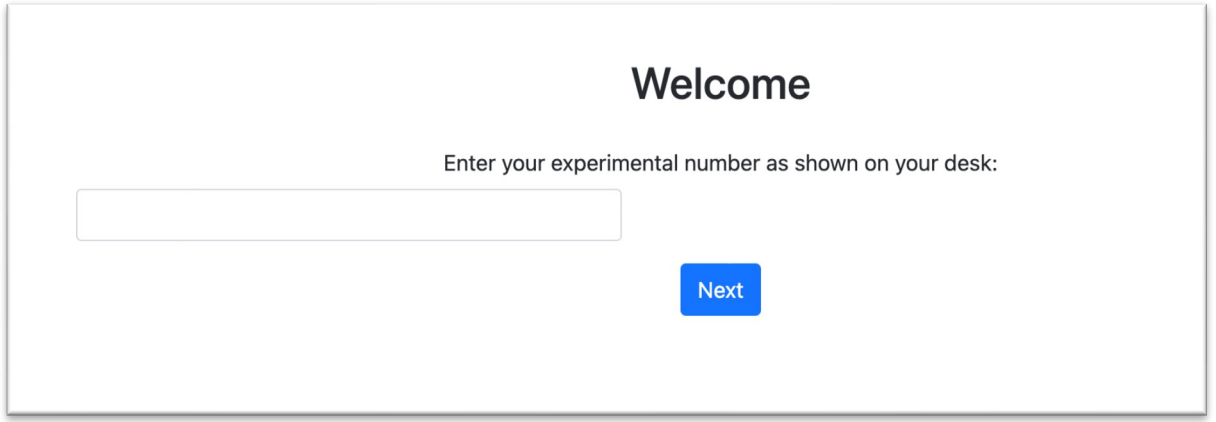
oTree conceptual framework

MVC framework	oTree structure
Model • Data • Interact with DB	**Model** • Each model is a database table • Interact with DB
View • Data presentation	**Pages** • Programs on the page • User interface (HTML)
Controller • Handles requests	Pre-defined by oTree Don't need to worry much

The reason why we need oTree rather than writing a program with Python directly!

Player Model

ID	age	gender
N0001	24	M
…		
…		



oTree conceptual framework

MVC framework

Model
- Data
- Interact with DB

View
- Data presentation

Controller
- Handles requests

oTree structure

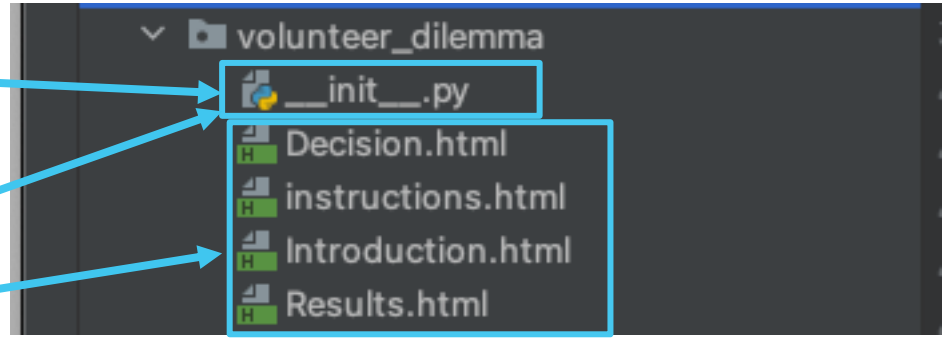
Model
- Each model is a database table
- Interact with DB

Pages
- Programs on the page
- User interface (HTML)

Pre-defined by oTree
Don't need to worry much

The reason why we need oTree rather than writing a program with Python directly!

Structure in PyCharm

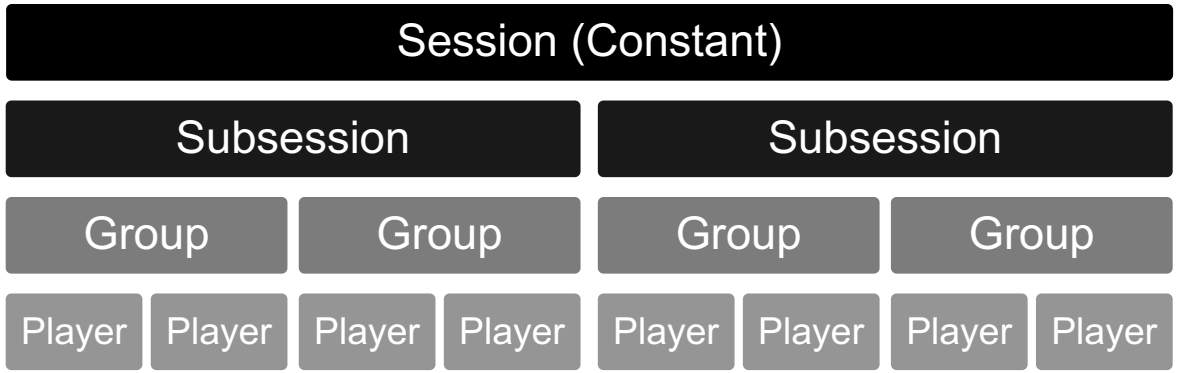


oTree conceptual framework

MVC framework	oTree structure
Model • Data • Interact with DB	**Model** • Each model is a database table • Interact with DB
View • Data presentation	**Pages** • Programs on the page • User interface (HTML)
Controller • Handles requests	Pre-defined by oTree Don't need to worry much

The reason why we need oTree rather than writing a program with Python directly!

- oTree has Pre-defined Models with hierarchy



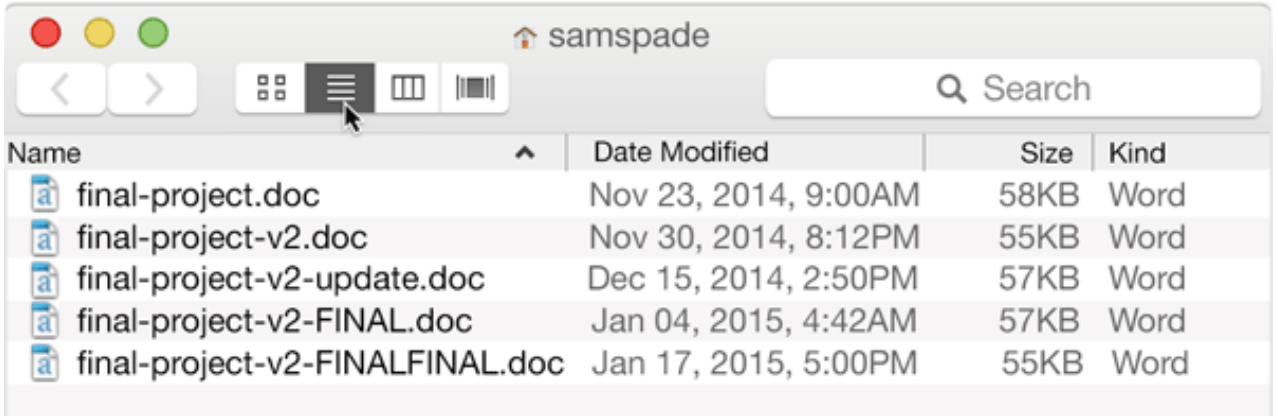
- You only need to define variables (fields) in each Model
- How do I know what fields to put in what table?
  - An object stored in a higher model can be accessed from a lower model, but not verse visa
  - Example: in a public good game, each player's contribution will be stored in the Player model, the group total contribution will be stored in the Group model

Now you are ready to start programming!

Git Basics

Local version control

- **Does this look like your folder?**

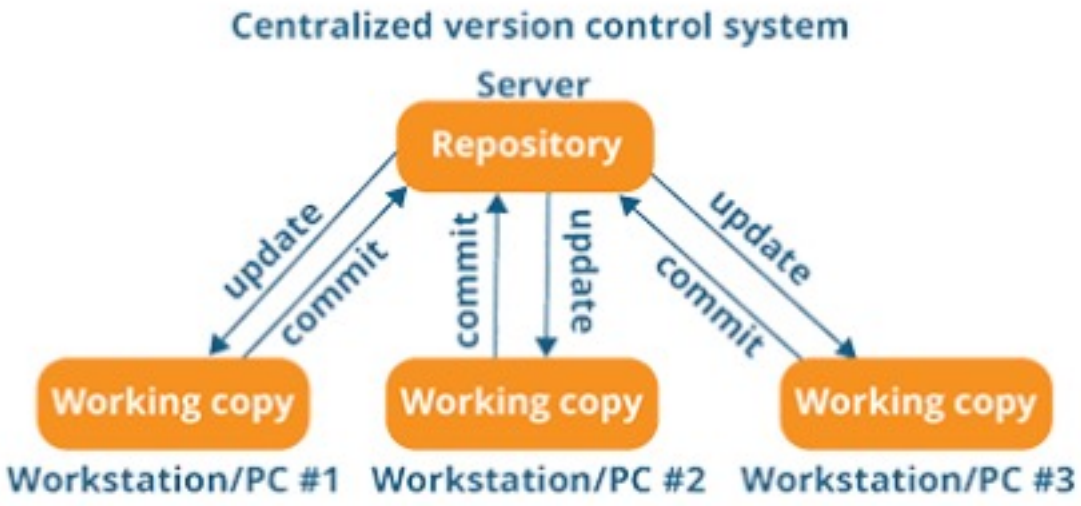


Source: https://zahidhasan.github.io/2020/11/24/version-control-system.html

- **What is the problem with this way of version control?**
  - It becomes unmanageable very quickly
  - Impossible for history tracking (what did I change from v2.0 – v2.1?)
  - Hard to collaborate

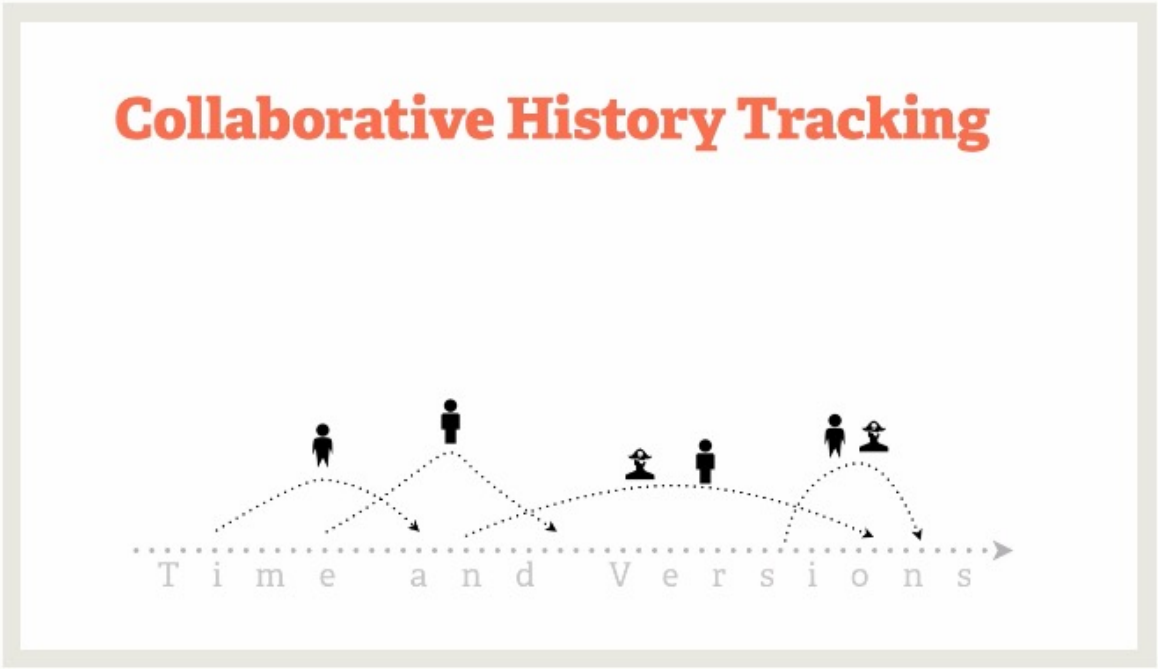
Centralized version control system

- **Imaging using Drobox on different PCs**



Each user has a working copy, but there is just one central repository. As soon as you commit, it is possible for your co-workers to update and to see your changes. For others to see your changes

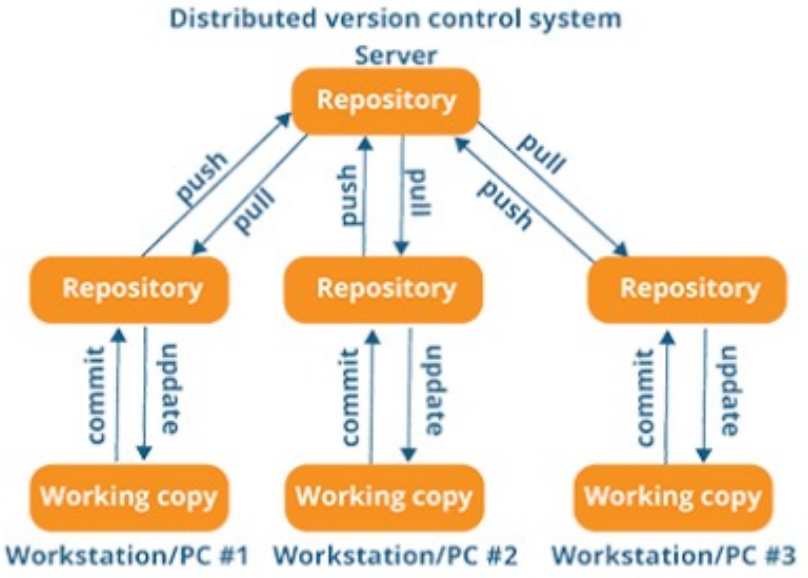
- **Version control with overlaps are more complicated**



Source: https://homes.cs.washington.edu/~mernst/advice/version-control.html#best-practices

Distributed version control system - Git

- **How Git works**



Each user gets their own repository *and* working copy. After you commit, others have no access to your changes until you push your changes to the central repository. When you update, you do not get others' changes unless you have first fetched those changes into your local repository.

- **Benefit of Git**
  - Allows simultaneous changes
  - Handles merge automatically
  - Easily go back to the previous version
  - No server is needed if only used locally
  - Work offline
  - Open source

- **Limitation of Git**
  - Only track-changes your text-based files, for other files, it only saves the snapshot of different versions, but don't know what's changed.

Source: https://homes.cs.washington.edu/~mernst/advice/version-control.html#best-practices

Git, GitHub, GitHub Desktop

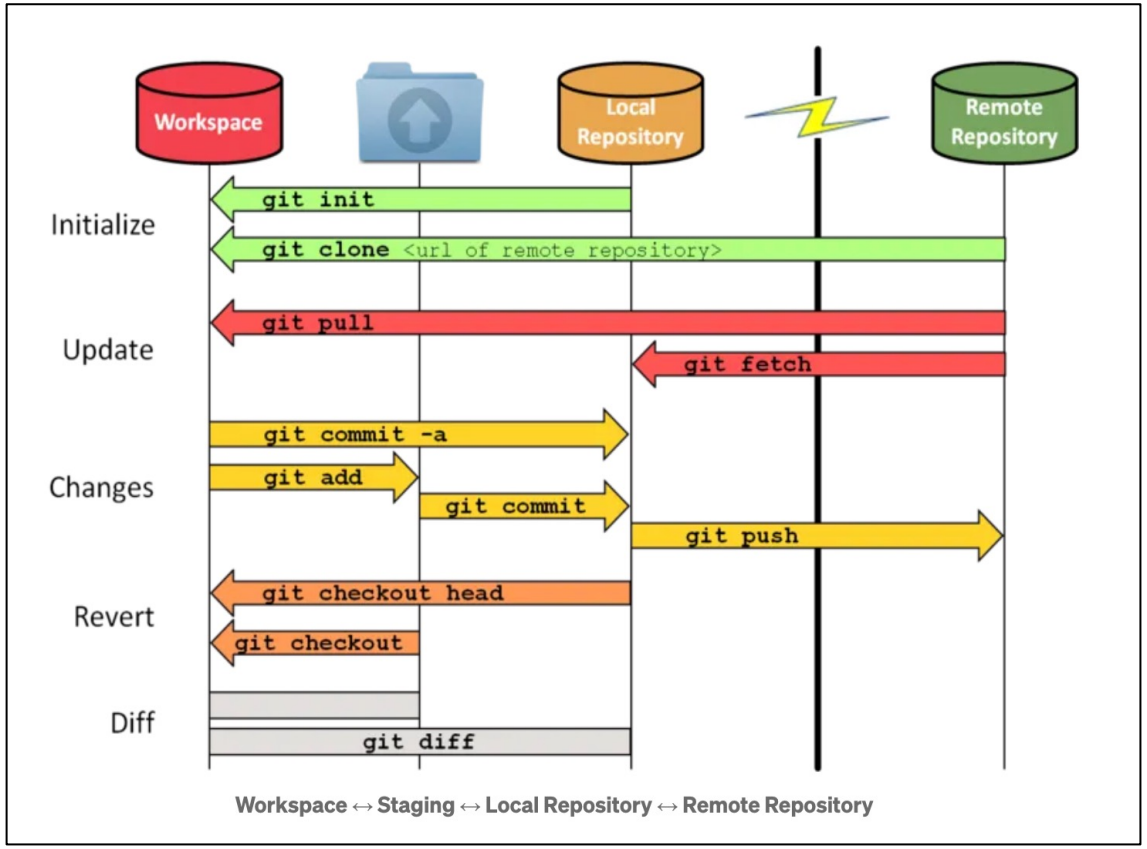
- **Git is the software**
  - Runs on the command line (Terminal/Shell)
- **GitHub is (one of) the server(s)**
  - Collaboration, save all documents
- **GitHub Desktop is an app**
  - Run Git using a more friendly graphical interface
  - Also support files saved on servers other than GitHub!

❖ We will use Git via the command line and GitHub in this workshop.

❖ GitHub Desktop is also recommended if you prefer tracking to be more visual

Most often used Git(Hub) concepts

- **Repository (repo): a ‘folder’ that holds your project**
  - Workspace: current version files you work on
  - Local repo: repo on your computer
  - Remote repo: repo on server (GitHub)
- **Frequently used actions**
  - Start a repo in local repo (git init)
  - **Clone** a repo from a remote repo (git clone)
  - **Pull** from the remote repo (git pull)
  - **Staging** prepare for committing to local repo (git add)
  - **Commit** changes to the local repo (git commit)
  - **Push** to the remote repo (git push)



oTree, Python, PyCharm, Git Installation

Installing oTree

"If you are an advanced programmer, you can use oTree with a text editor"

--oTree

"You just need to learn a bit of Python programming to use oTree!"

--Most experimentalists

Installing oTree

1. **Installing Python 3.12.3** https://www.python.org/downloads/
2. **Install PyCharm**
   - Integrated development environment (IDE) for software development using Python
   - Free educational licenses you can apply for (https://www.jetbrains.com/community/education/#students/)
3. **Install oTree using Terminal**
   - `$ pip3 install -U otree`
   - More on: https://otree.readthedocs.io/en/latest/install-nostudio.html#install-nostudio

Programming for Experiments with oTree

27 – 28 May 2024

Laura Liu

DAY 2

Git set up

- **You need to first install Git**
  - Check if you already have Git in Terminal/CMD
    - `$ git --version`
  - Git: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
  - Tutorial: https://www.atlassian.com/git/tutorials/install-git (recommended)
  - Use an installer is the easiest way: https://sourceforge.net/projects/git-osx-installer/files/

- **Set up your Git username and email**
  - `$ git config --global user.name “Your name”`
  - `$ git config --global user.email “Your email”`

Preparation for tomorrow - Set up Heroku

- **Create an account**
  - Need to download Salesforce authenticator on your phone
- **Add payment method**
  - Has to be done before creating an app

Now you are ready to start programming!

THE UNIVERSITY OF MELBOURNE

Convert Experiment to oTree Program

Let’s start a project

- **Create an o–Tree project in PyCharm**
  - `$ otree startproject otreedemoproject`
- **Create a local repo using Git to track all your changes in the project folder**
  - `$ cd otreedemo otreedemoproject`
  - `$ git init`
  - `$ git add .`
  - `$ git commit –m’’start otreedemoproject’’`
- **Check your folder for git files.**
  - You might need to review hidden files: Command + Shift + . (the period key) for Mac

Push code to GitHub

- **Create a remote repo on GitHub**
  - Same name as your local Git repo
  - Private repo
  - Do not choose ‘add a README file’ if you want to push an existing local repo to GitHub (this will cause a difference between your local and remote)

- **Follow the prompt on GitHub to push your local repo to GitHub**
  - Username for GitHub:
  - When you enter password, your will see error message, this is because GitHub has change security protocol
    - GitHub – settings – developer settings – Personal access tokens – Generate new token – classic -- Scopes: repo
    - Copy token to terminal and somewhere else [important it only going to show once!]

Programming for project

- **Open your project in PyCharm**
  - File –> open project -->find the oTree project you created

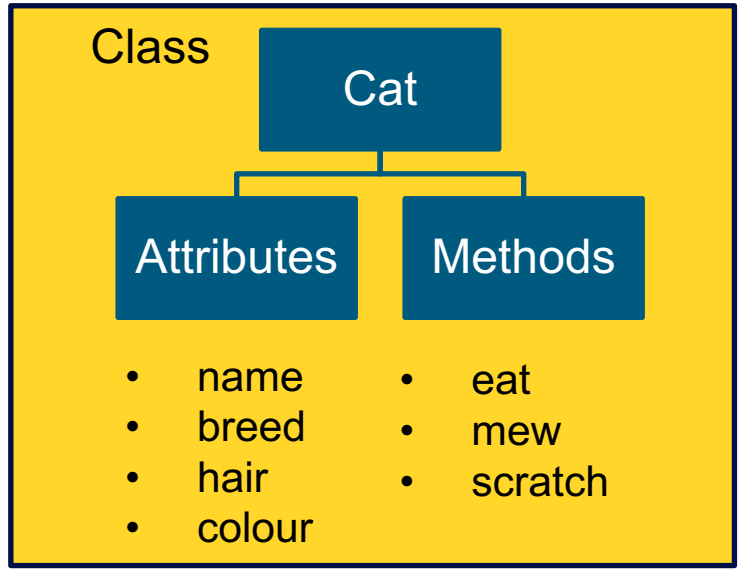
- **Creating virtual environment**
  - **Virtual environment:** self-contained directory that contains a Python interpreter and its associated libraries and scripts
  - **Benefit of using a virtual environment:** create an isolated environment for projects where you can install specific versions of packages without affecting the global Python installation.
  - Starting from Python 3.3, the ‘venv’ module is included in the standard library

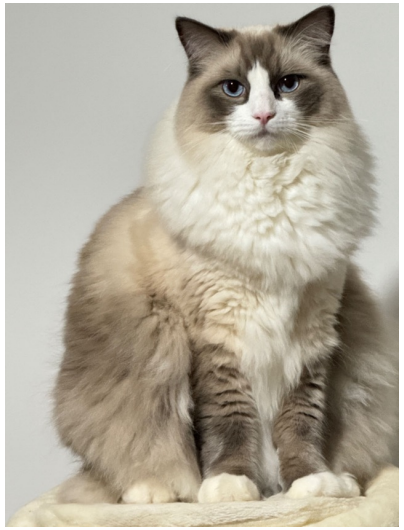
Structure of an oTree project

- **Files and what they do**
  - App folders: each contains an app
    - Each part of your experiment can be an individual app for easy reuse and modification.
  - In each app folders:
    - __init__.py: back-end Python program
    - Xx.html: front-end user interface
  - settings.py: configuration when you run your project (not only for the single app)
    - Define session: what app is available

Object oriented programming (OOP)

- **Python is one of the languages that use OOP**
- **OOP focuses on the objects rather than the logic that manipulates them**
- **Structure of OOP**
  - Classes: blueprints or templates for objects/instances.
    - Attributes - columns in the table
    - Methods – behaviour of objects in class
- **oTree models and pages are defined using Python Class**





```
cat1.name = ‘Leo’
cat1.bread = ‘Ragdoll’
cat1.hair = ‘LongHair’
cat1.color = ‘blue-bi’
```

Test the program on local server

- You can test your program locally
  - `$ otree devserver`

- If oTree ask you to remove your db.sqlite3 files, you can type
  - `$ rm db.sqlite3`

Save your changes using Git

- git add .
- git commit –m”add real effort task”
- git push

- Check on GitHub

Deploy your experiment to Heroku

Deploy your experiment to Heroku

- **Heroku is a commercial cloud hosting provider where you can run your apps.**

- **There are three ways to deploy your experiment to Heroku**
  - oTree Hub (least coding but only with limited free projects and public!)
  - GitHub (free, almost no coding but still depends on third party)
  - Command line (full control but need coding)

Deploy to Heroku

- **Heroku is a commercial cloud hosting provider where you can run your apps.**

- **There are three ways to deploy your experiment to Heroku**
  - oTree Hub (least coding but only with limited free projects and public!)
  - GitHub (free, almost no coding but still depends on third party)
  - Command line (full control but need coding)

- In this course we will deploy the experiment using GitHub.

Deploy your app to Heroku

- **Create an app**
  - **Choose a deployment method – GitHub**
  - **Connect your GitHub**
    - Log in to your GitHub
    - Pop-up window for your GitHub account to authorize
  - **Connect your GitHub Repository**
    - Manual Deploy is recommended
    - Auto deploy will automatically deploy your program whenever you do a new push to GitHub (even for the changes you don't want to deploy yet)
  - **Click 'More' on the top right then 'run console'**
    - Type 'otree resetdb'
  - **Click 'open app' to check your program**

Deploy your app to Heroku

- **Allocate resources to your app**
  - Dyno: Compute Resource
  - Install add-ons: Heroku Postgres – database

- **Heroku charges the resources your app use by time, remember to remove the resources if you are not using them**
  - Make sure you download all your data before removing them!

Deploy your app to Heroku

- **Now you can open up your app**

- **Remove debug info**
  - On Heroku –> app–> setting –> Config Vars, you will be able to add these:
    - OTREE_ADMIN_PASSWORD
    - OTREE_AUTH_LEVEL
    - OTREE_PRODUCTION (1, remove debug; 0 default, with degub)

Let’s test your program on Heroku

- **Please create a session and send the link to your group member to test it!**

The full programming cycle:

The whole programming cycle:

Preparation:

Create an oTree project → initial tracking using Git →create a remote repo on GitHub

Programming iterations:

program and changes → commit to the local repo → push to the remote repo →testing

Deploy experiment:

Deploy to Heroku → allocate resources → test run → run experiment

Workshop Resources

- **All workshop materials are in GitHub Repository**
  - https://github.com/lauraliuyang/otree_workshop.git
  - You can clone it to your local repo

- **The demo project code in the shared repository is the full experiment**
  - It contains both the real effort game and the public good game.
  - The endowment in public good game is calculated based on the earnings in real effort game
  - You can run it first to see how it works, then try to replicate it by using the basic games we built together during the workshop.

More resources:

- **Running online experiment GitHub Repo by Christian Peters**
  - https://github.com/CPHPeters/Guest_Lecture_Experiments?tab=readme-ov-file#heroku
- **oTree tutorials on YouTube by Jonas Frey**
  - https://www.youtube.com/playlist?list=PLBL9eqPcwzGPli11Yighw5LWwzIifEFd_
- **Git commands**
  - https://www.atlassian.com/git/tutorials/setting-up-a-repository

THE UNIVERSITY OF
MELBOURNE

Thank you!

Email: yang.liu@unimelb.edu.au

Website: https://www.lauraliu.net/

The University of Melbourne (Australian University)
PRV12150/CRICOS 00116K