Version Controlling and Git

Informatics I

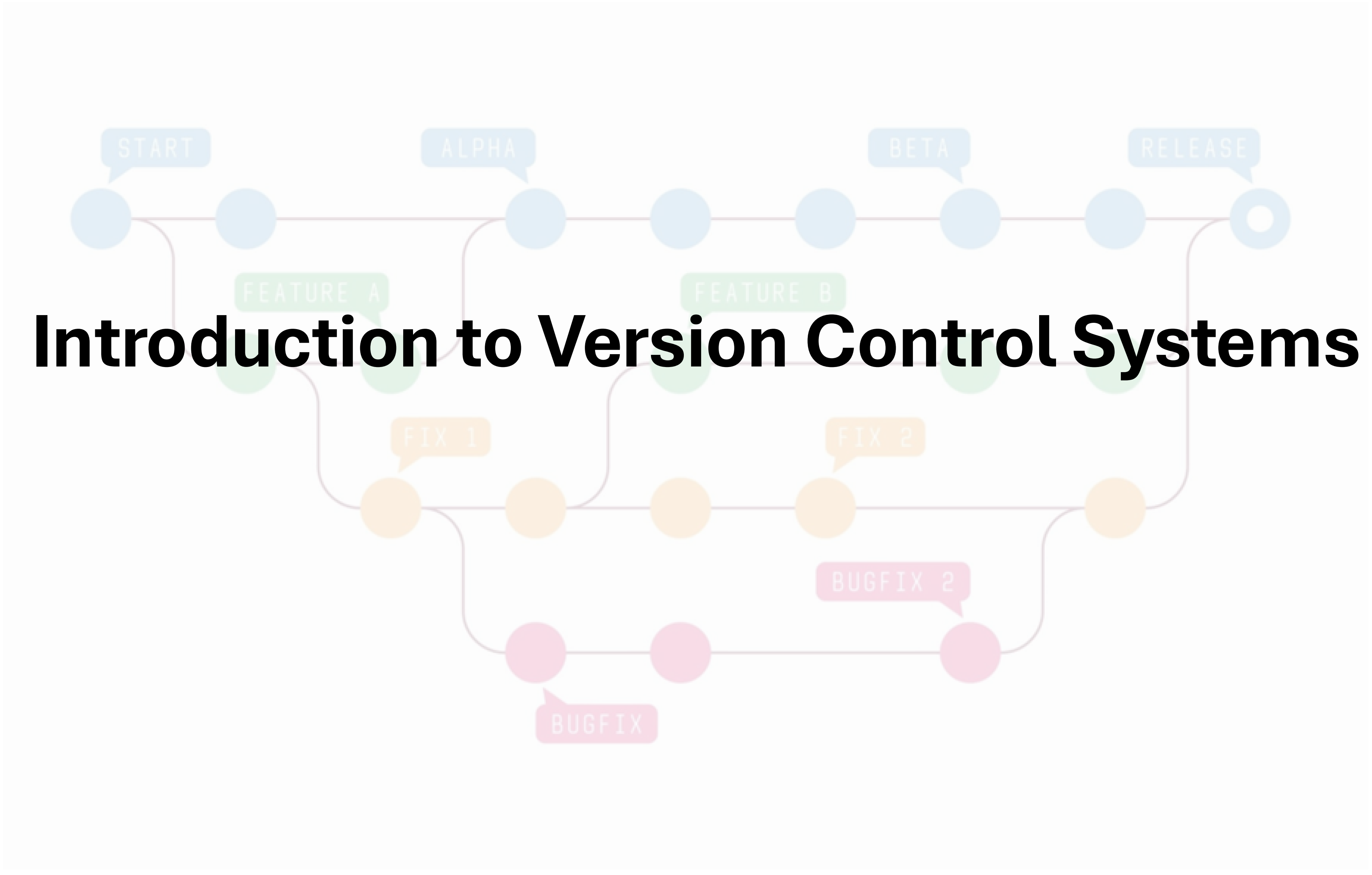
Fall 2024

Melih Catal - catal@ifi.uzh.ch

Outline

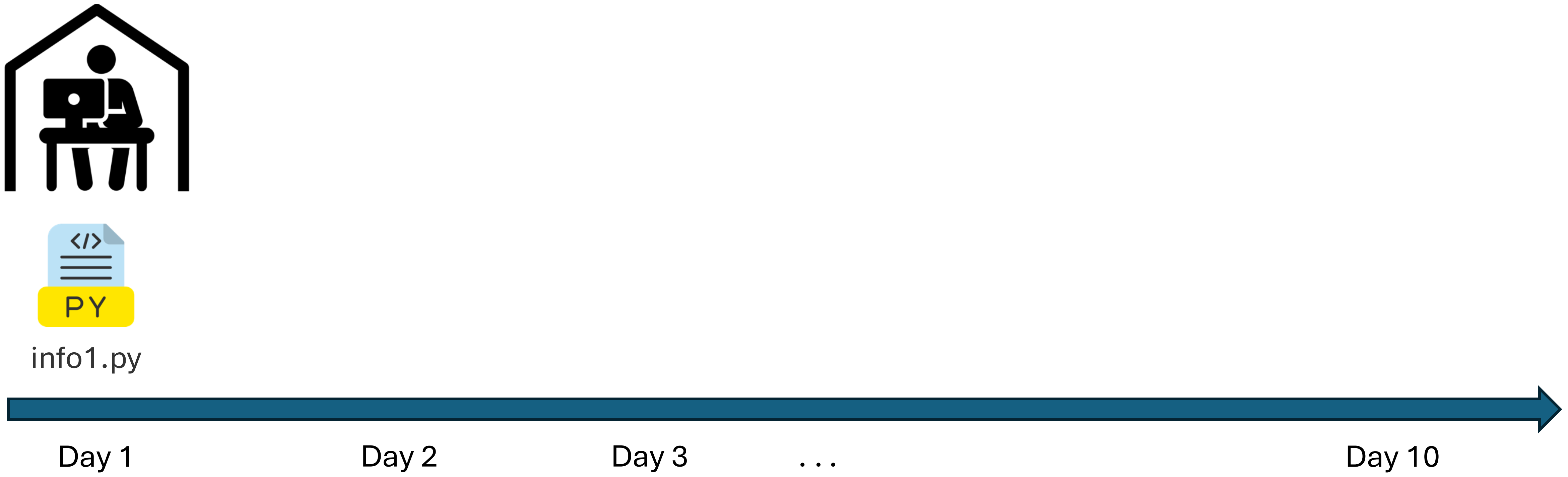
1. Introduction to Version Control Systems
2. Overview of Git
    1. Getting Started with Git
    2. Fundamental Concepts in Git
    3. Basic Git Commands
    4. Working with Branches
    5. Collaborating with Others

    $\}$ Will be covered in the Jupyter Notebook
3. Best Practices
4. Git Hosting Platforms
5. Further Resources

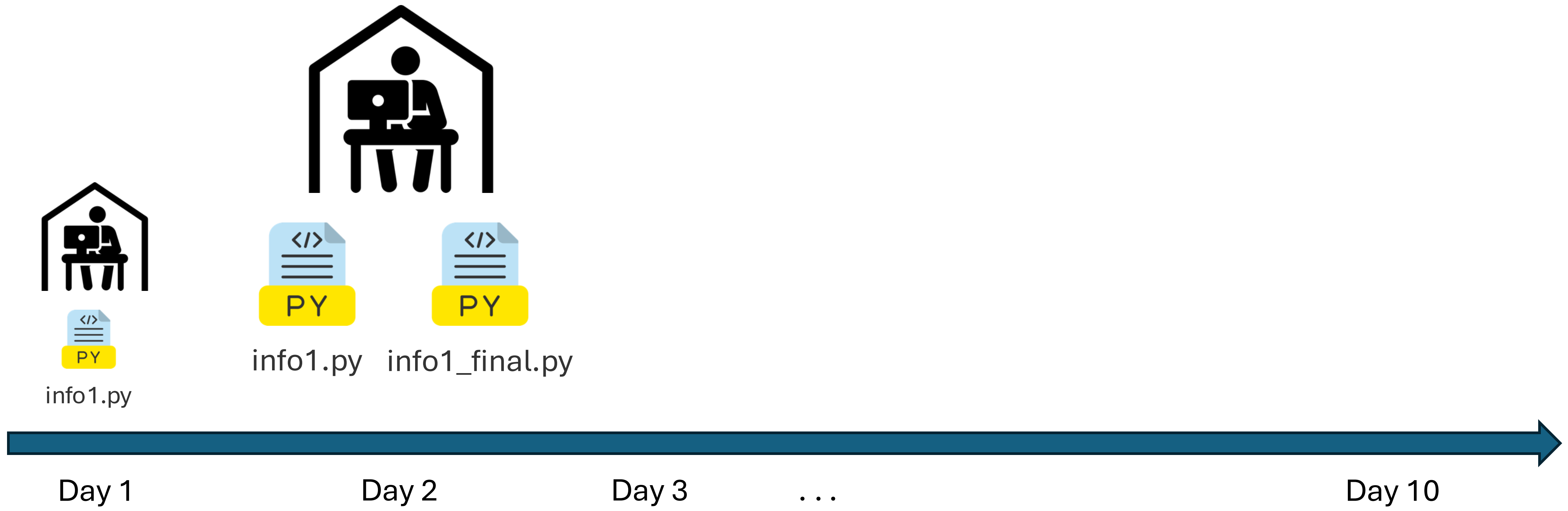


Introduction to Version Control Systems

Motivation - Avoid the File Chaos



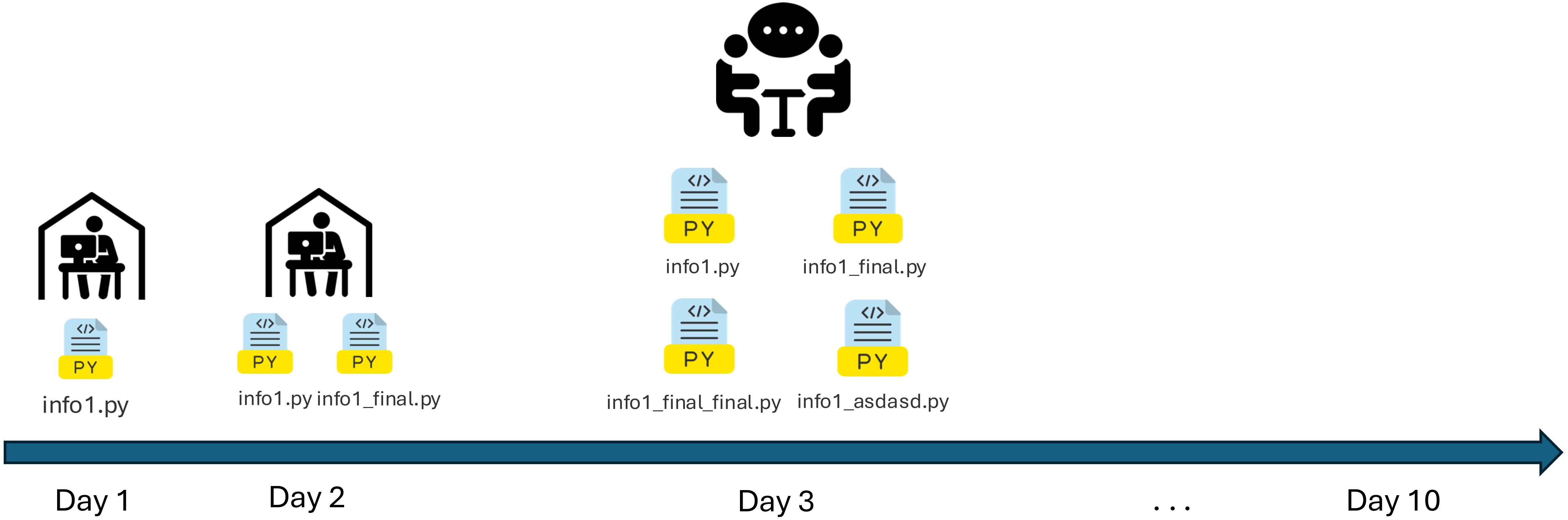
Motivation - Avoid the File Chaos



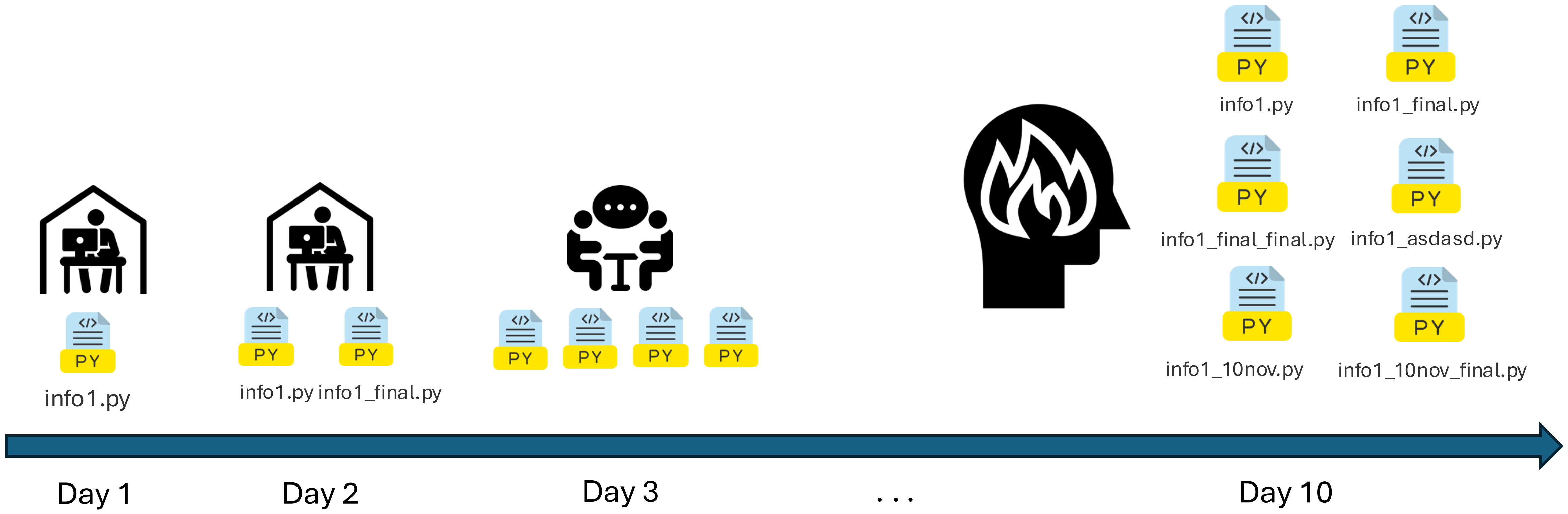
Universität Zürich^UZH

s.e.a.l. software evolution & architecture lab

Motivation - Avoid the File Chaos

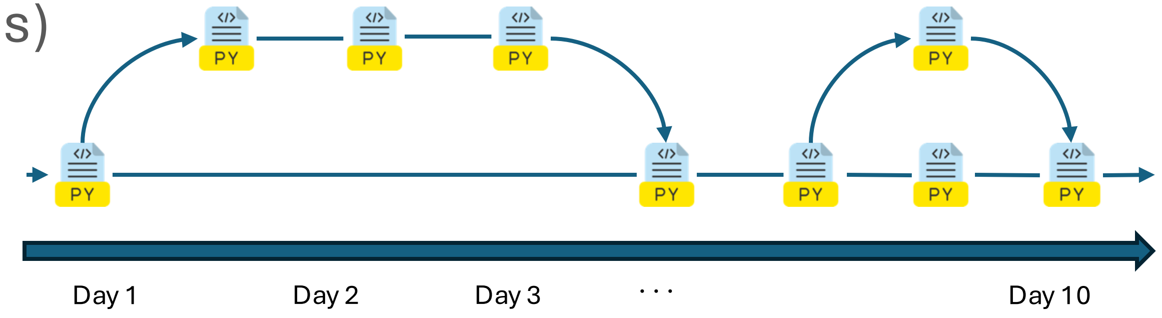


Motivation - Avoid the File Chaos



Solution: Version Control Systems

- What is it?
  - A tool to track and manage file changes over time
- Key Features:
  - Tracks every change (additions, edits, deletions)
  - Labels milestones (e.g., fixes, completions)
  - Enables reverting to previous versions
  - Simplifies collaboration without overwriting
- Why Use It?
  - Avoid file confusion (e.g., info1_last_final_v2.py)
  - Save time and work efficiently
  - Collaborate seamlessly with others



Version Control System Types

Centralized Version Control Systems

- All version history and the main repository are stored on a central server
- Developers get the files to their local machines and then commits changes back to the central server
- Examples: Subversion (SVN)
- Advantages:
  - Simple to set up and manage
  - Clear visibility
  - Easy to enforce policies
- Disadvantages:
  - Single Point of Failure
  - Limited offline capabilities

Distributed Version Control Systems

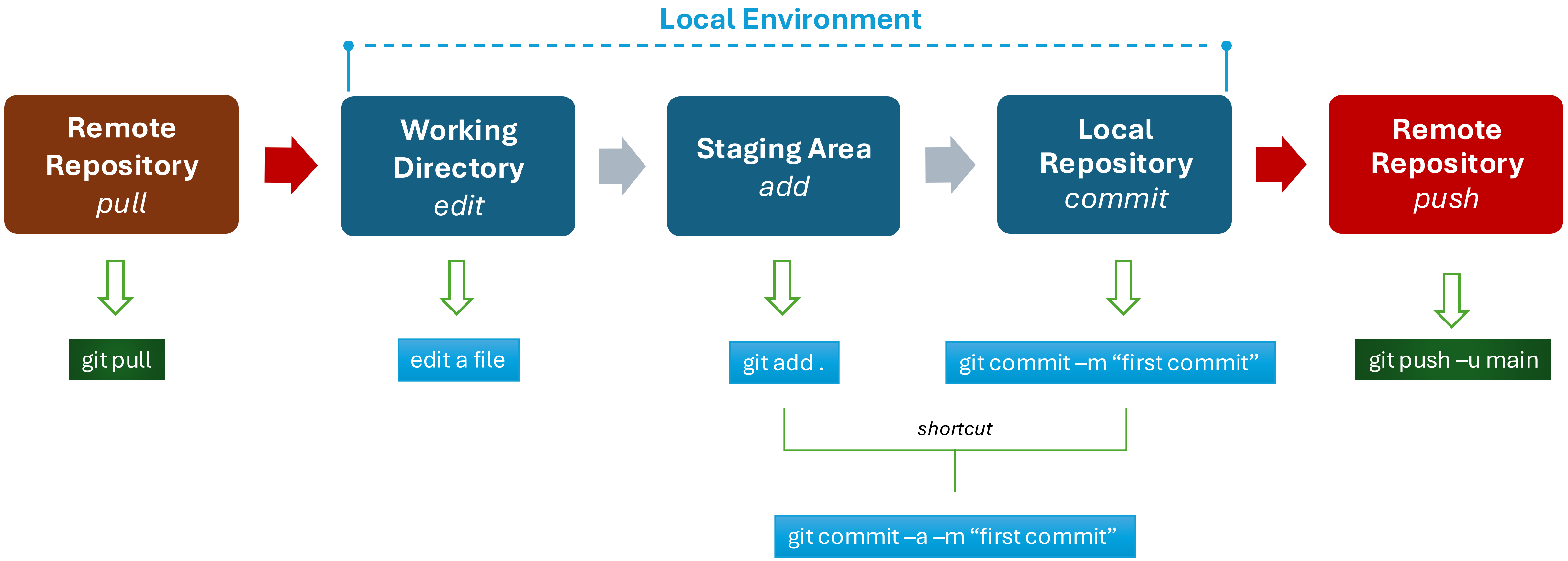
- Replicate the entire repository, including version history, on each user's local machine
- Developers can work offline and sync changes with a central repository
- Examples: Git, Mercurial, Bazaar
- Advantages:
  - Offline access to version history, and the ability to make commits locally
  - Improved performance for local operations
- Disadvantages:
  - Synchronization and conflict resolution can be more challenging in large teams

Overview of Git

Fundamental Concepts in Git

- **Repository:** A storage space where your project's files and their version history are tracked
  - **Local Repository:** A version of the repository stored on your local machine for development
  - **Remote Repository:** A shared repository hosted on a server for collaboration and synchronization
- **Version History:** A chronological record of all changes made to the files in a repository
- **Commits:** Snapshots of changes saved to the repository with a descriptive message
- **Branches:** Parallel lines of development used to work on features or fixes independently
- **Working Directory:** The local folder where you create, edit, and delete files
- **Staging Area:** A space where changes are prepared and reviewed before committing

How Does Git Work?



Universität Zürich^UZH

s.e.a.l. software evolution & architecture lab

Continue with the Notebook

Best Practices

Write Descriptive Commit Messages

Follow a Branching Strategy

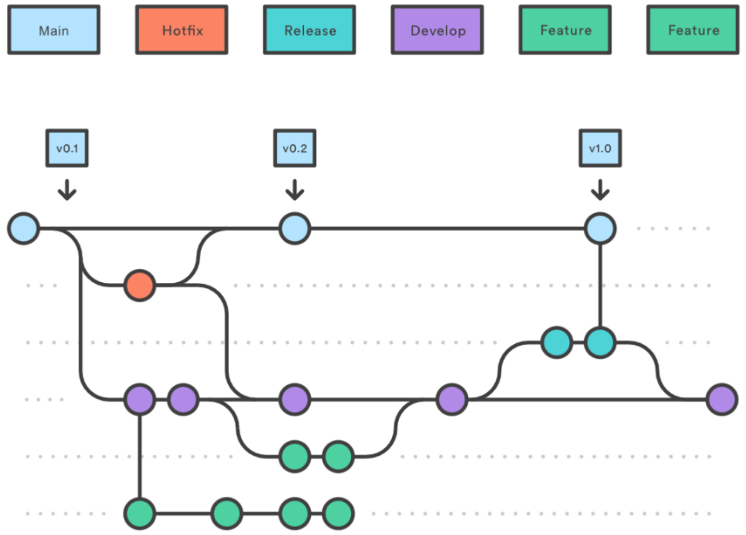
Protect critical branches and use Pull Requests

Keep commits small and atomic

Pull changes frequently to stay updated

Branching Strategies

- Structured workflows for using Git branches effectively
- Each branch has specific purpose
- Git Flow:
  - **Branches**:
    - main: Always production-ready
    - develop: Integration branch for new features
    - feature/*: Prepares for a production release
    - hotfix/*: For critical fixes in production
  - **Workflow:**
    - Features are developed in feature/* branches and merged into develop
    - A release/* branch is created for testing and polishing
    - After release, changes are merged into main and tagged
  - High complexity
  - Good for large projects with clear release cycles



Git Hosting Platforms

- A central place to share your code
- Think it as a central train station
- Examples:
    - GitHub
    - GitLab
    - Bitbucket
    - Azure DevOps
    - AWS CodeCommit

Further Resources

- Atlassian
- w3 Schools
- Git Scm

Universität Zürich^UZH

s.e.a.l. software evolution & architecture lab

Thank You and Q&A