

软件工程

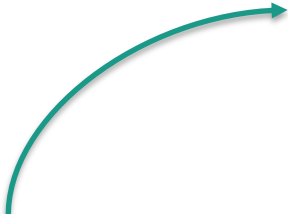
代码管理: **Git**

Spring 2022, SWUFE



复习

- 单元测试
- 效能分析 (profile)



Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their showing their absence.



Premature optimization is the root of all evil.



练习

- 使用Java或Python写一个生成斐波那契数 (Fibonacci) 的函数 , 并进行单元测试。

1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , ...

1. Git

版本管理工具

❑ 代码协作

❑ 版本控制



本公举已经不想
做了.psd



超大项目视觉初
稿.psd



超大项目视觉稿
新版.psd



超大项目视觉稿
这个一定是最终
版.psd



超大项目视觉稿
最新版.psd



超大项目视觉稿
最终版.psd



超大项目视觉稿
最终的最终版.
psd



未标题-1.psd



再让我改我就去
死版.psd



这稿坚决不改了
就这么着吧.psd

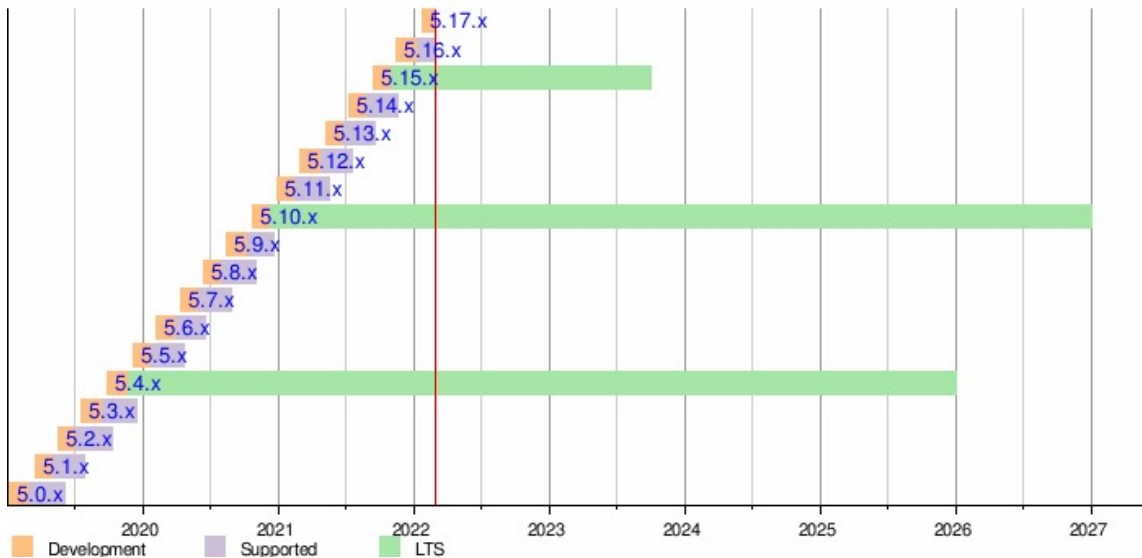
1.1 Git的历史



Linus Torvalds

Torvalds在2005年开发了Git。

几年之后，Git凭借其功能和性能的优越表现，成为代码
版本管理的最好选择。



1.2 Git的发展



git



GitHub



GitLab



gitee




Bitbucket

1.3 Git的安装

与很多软件类似，Git的使用有命令行和GUI两种。我们这里仅学习命令行的用法。

- Mac都预安装了git
- Linux可以通过包管理器安装
- Windows可以通过 <https://git-scm.com/downloads> 安装

A screenshot of a macOS terminal window. The window has a dark background and a light gray title bar with an Apple logo, a home icon, and a tilde (~) icon. The terminal shows the command 'git --version' being entered, followed by the output 'git version 2.32.0 (Apple Git-132)'.

```
git --version  
git version 2.32.0 (Apple Git-132)
```

1.4 初始设置

```
1 [user]
2     name = CHEN zhongpu
3     email = chenloveit@gmail.com
~
```

设置使用Git的名字和邮箱地址。

会出现在Git的提交日志


```
$ git config --global user.name "git_username"
$ git config --global user.email "email_address"
```

上面的内容会保存在用户目录下.gitconfig文件里面，也可以通过
`git config --global --list` 列出所有配置。



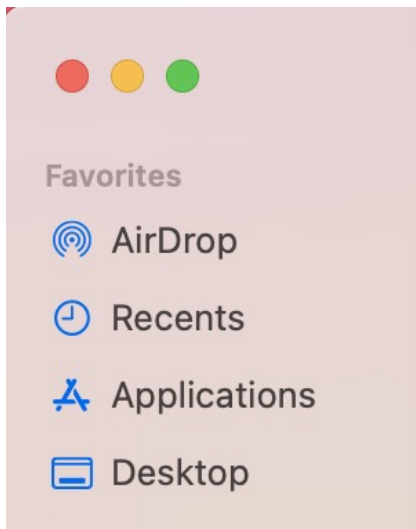
1.5 基础操作

git init: 初始化仓库



```
cd git-demo-example  
git init
```

1.5 基础操作



< > git-demo-example



fruits.py



veggies.py

```
fruits = ['apple', 'dragonfruit', 'peach', 'banana', 'grape',  
'apple', 'peach', 'watermelon', 'grape', 'grape']
```

```
veggies = ['cabbage', 'carrot', 'spinach',  
'asparagus', 'artichoke', 'pumpkin', 'lettuce']
```

git status: 查看状态

```
git-demo-example ➤ master ➤ git status
```

```
On branch master
```

```
No commits yet
```

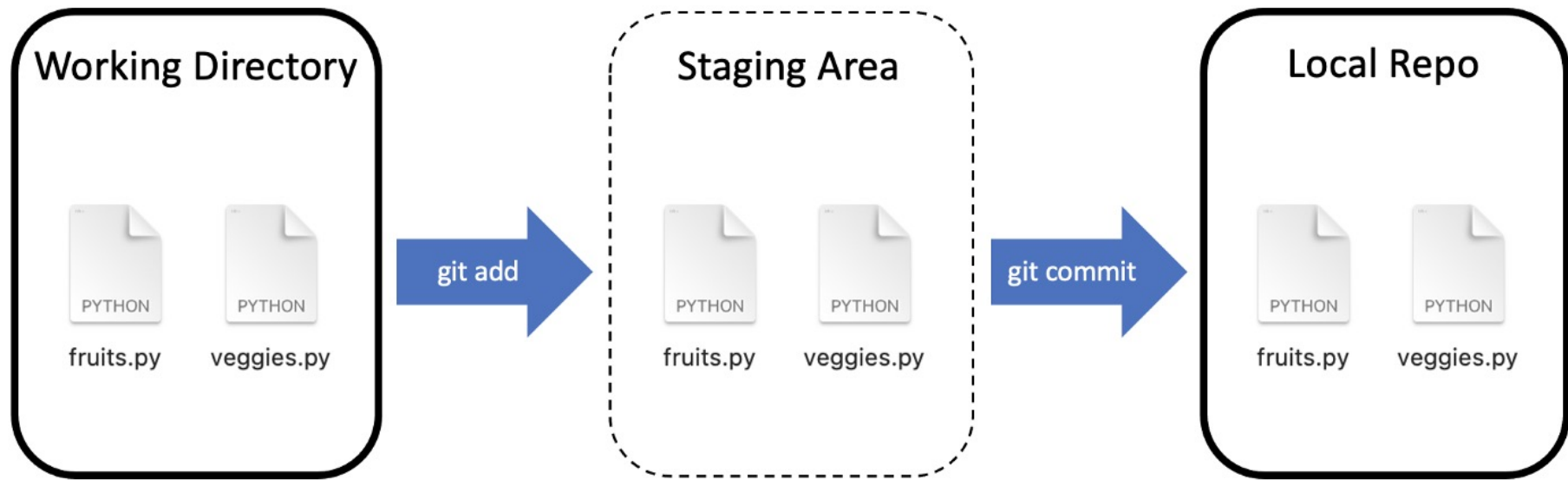
```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
    fruits.py
```


```
    veggies.py
```

```
nothing added to commit but untracked files present (use "git add" to track)
```



理解“工作目录”、“暂存区”和“本地仓库”是学习Git的重点。

git add: 提交到本地仓库



```
git add fruits.py
git add veggies.py
```

```
git-demo-example ? master + git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   fruits.py
    new file:   veggies.py
```

git commit: 提交到本地仓库

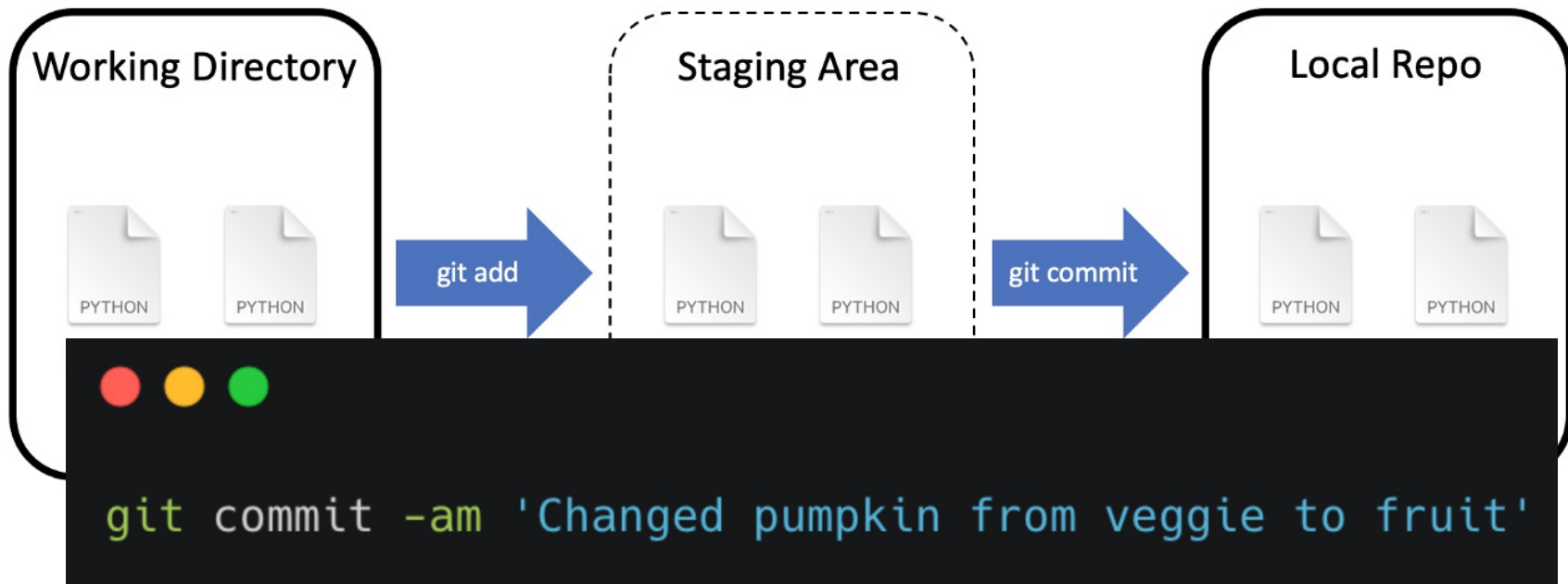


```
git commit -m 'My first commit'
```

```
git-demo-example ➤ ↗ master ➤ git status  
On branch master  
nothing to commit, working tree clean
```

```
veggies = ['cabbage', 'carrot', 'spinach', 'asparagus',  
'artichoke', 'pumpkin', 'lettuce']
```

南瓜是水果，不是蔬菜



git log: 查看提交历史

```
commit 4ebe0ec7d7f872e5cdcae390b9757abbc50a2b48 (HEAD -> master)
```

```
Author: chenzhongpu <chenloveit@gmail.com>
```

```
Date:   Fri Sep 3 23:04:00 2021 +0800
```

```
    Changed pumpkin from veggie to fruit
```

```
commit c7be7b805f85052de7191be15dd012863bcce5ed
```

```
Author: chenzhongpu <chenloveit@gmail.com>
```

```
Date:   Fri Sep 3 22:58:45 2021 +0800
```

```
    My first commit
```


git log

```
commit 47888c8f071ff87fafa882b3467624aed7ceb6fe (HEAD -> master)
Author: CHEN zhongpu <chenloveit@gmail.com>
Date: Sat Mar 5 17:00:19 2022 +0800

    remove pumpkin from veggies

diff --git a/veggies.py b/veggies.py
index 8f72bbd..aa754ed 100644
--- a/veggies.py
+++ b/veggies.py
@@ -1,1 @@
-veggies = ['cabbage', 'carrot', 'spinach', 'asparagus','artichoke', 'pumpkin','lettuce']
\ No newline at end of file
+veggies = ['cabbage', 'carrot', 'spinach', 'asparagus','artichoke', 'lettuce']
\ No newline at end of file
```

只看一个文件或目录的历史：`git log fruits.py`

显示文件的改动：`git log -p veggies.py`

仅显示一行信息：`git log -oneline`

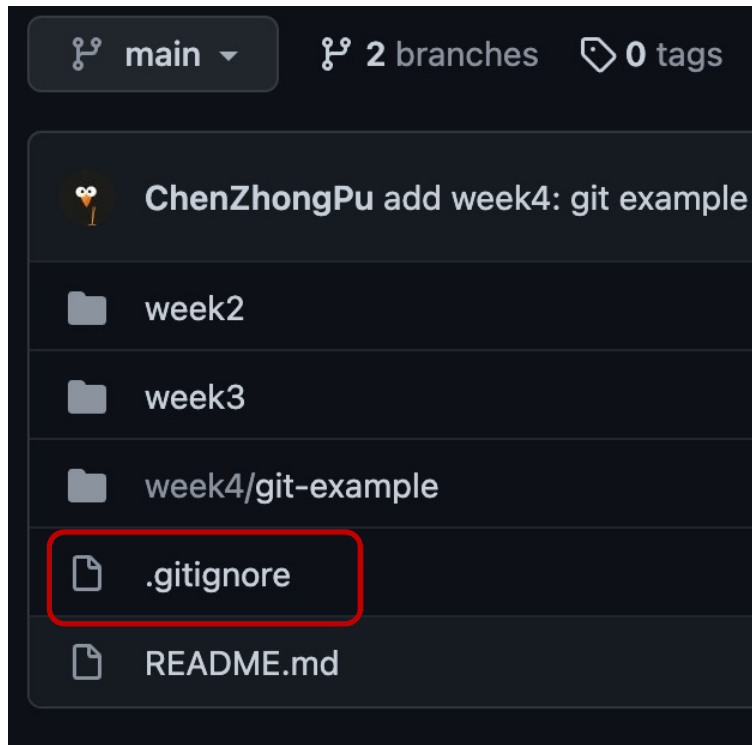
仅显示最后1次提交：`git log -n 1`

.....

.gitignore

有些文件不加入到仓库中进行代码管理。

- ❑ 比如Java的.class文件
- ❑ 比如Python的__pycache__文件夹
- ❑ 比如VSCode的.vscode文件夹
- ❑ 比如IDEA的.idea文件夹, .iml文件
- ❑ ...



```

1 #####
2 ## Java
3 #####
4 .mtj.tmp/
5 *.class
6 *.jar
7 *.war #####
8 *.ear ## Eclipse
9 *.nar #####
10 hs_err_pid* bin/
    tmp/
    .metadata
    .classpath
    .project
    *.tmp
    *.bak
    *.swp
    *~.nib
    local.properties
    .loadpath
    .factorypath

```

```

#####
## IntelliJ
#####
out/
.idea/
.idea_modules/
*.iml
*.ipr
*.iws

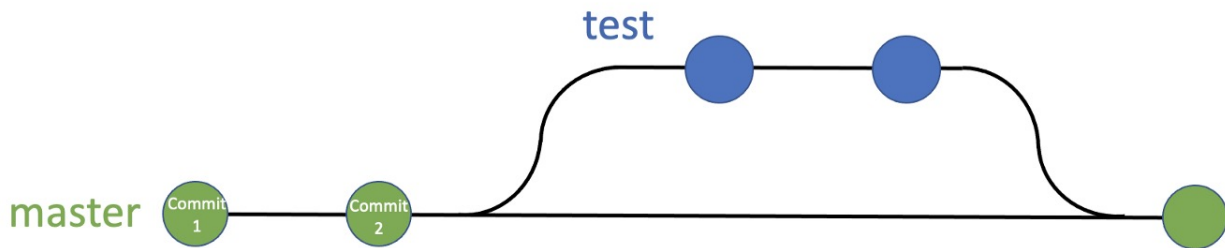
```

```

# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

```

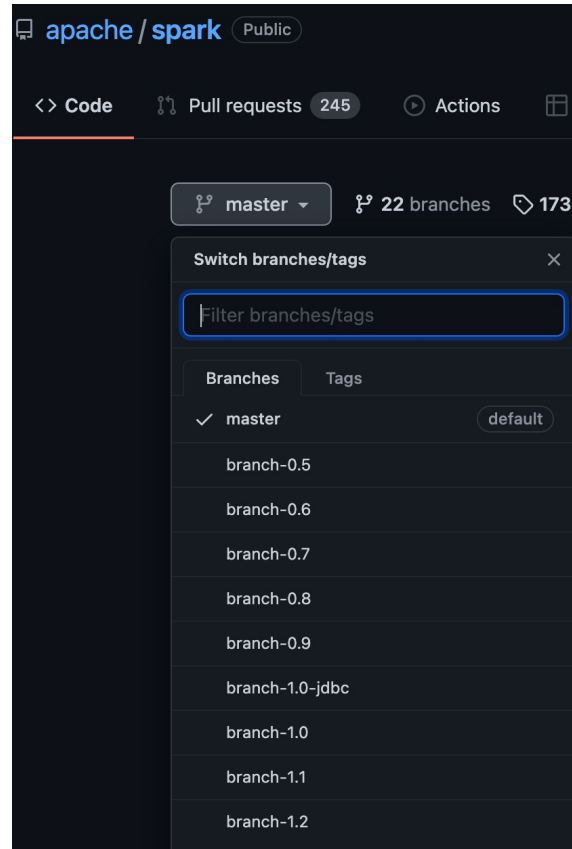
分支




```
git-demo-example ➤ master ➤ git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```



新建分支，切换分支



```
git branch test  
git checkout test
```

```
git checkout -b test
```

```
fruits = ['apple', 'dragonfruit', 'peach', 'banana', 'grape',  
          'apple', 'peach', 'watermelon', 'grape', 'grape']
```

有重复数据

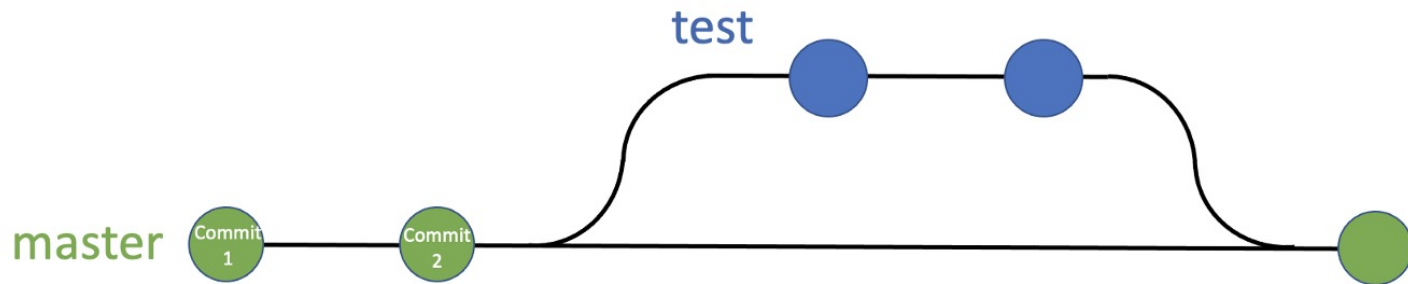
```
fruits = set(fruits)
```



课堂练习

- 在**test**分支得到水果的集合，然后提交。
- 然后再切换回**master**分支，观察**fruits.py**里面的代码。

合并分支: `git merge`

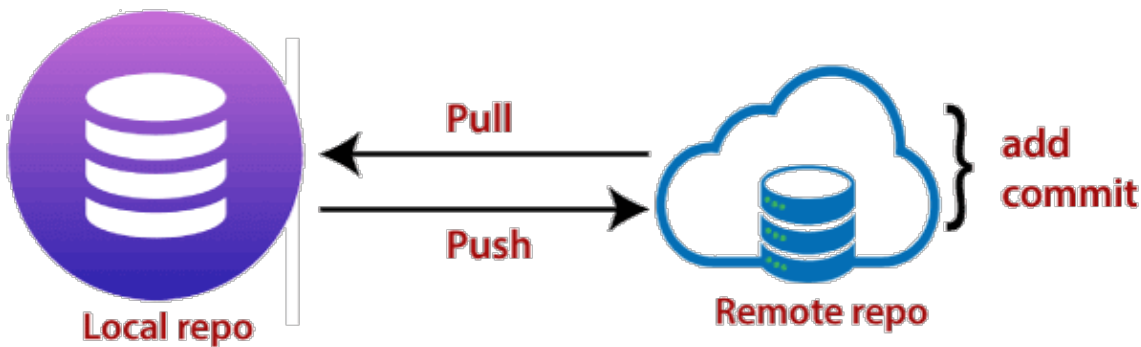


```
git checkout master  
git merge test
```

课堂任务

建议配置SSH公钥

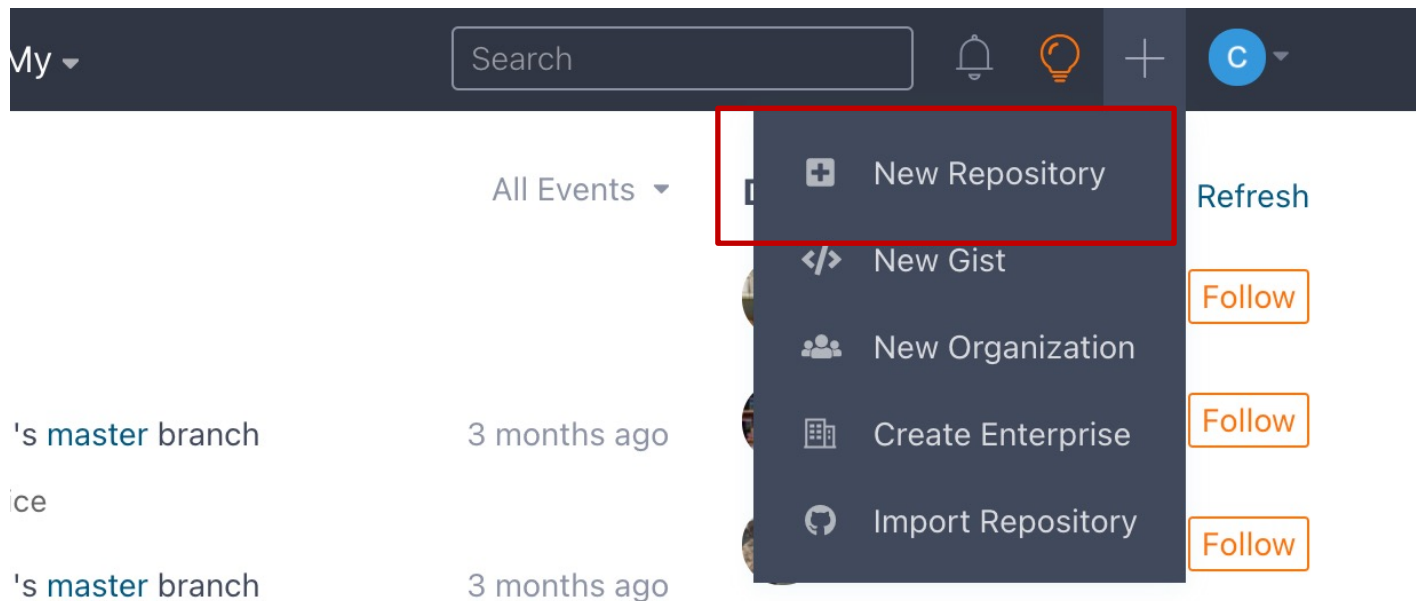
- 注册GitHub账号，并将刚才新建的仓库发布到GitHub。



<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/about-ssh>

<https://gitee.com/help/articles/4181>

远程仓库



The screenshot shows the GitHub web interface. At the top, there is a dark navigation bar with a search bar, notification bell, lightbulb icon, a plus sign, and a user profile icon. Below the navigation bar, a dropdown menu is open, listing several options: 'New Repository', 'New Gist', 'New Organization', 'Create Enterprise', and 'Import Repository'. The 'New Repository' option is highlighted with a red rectangular box. To the right of the dropdown menu, there is a 'Refresh' button and three 'Follow' buttons. Below the navigation bar, the main content area shows a list of repositories. The first two visible entries are both labeled 's master branch' and '3 months ago'.

My ▾ Search 🔔 💡 + C ▾

All Events ▾

Refresh

Follow

Follow

Follow

's master branch 3 months ago

ice

's master branch 3 months ago

New repository

Already have a project repository elsewhere?

Name * ✓

git-demo-example

Owner



chenzhongpu



Path * ✓

/ git-demo-example

Project Address: <https://gitee.com/zhongpu/git-demo-example>

Description

Describe it

- ☐ Public (Any one can see)
- ☒ Private (Only repository members can see)
- ☐ Enterprise Innersouce (Only enterprise members can see) ?

-
- ☐ Initialize this repository (Choose language, .gitignore and a license)
- ☐ Choose templates (Add Readme, issue, Pull Request templates)
- ☐ Choose branch model (Branches will be created according to the selected branch model)

Create git repository:

```
mkdir git-demo-example  
cd git-demo-example  
git init  
touch README.md  
git add README.md  
git commit -m "first commit"  
git remote add origin git@gitee.com:zhongpu/git-demo-example.git  
git push -u origin master
```

Existing repository?

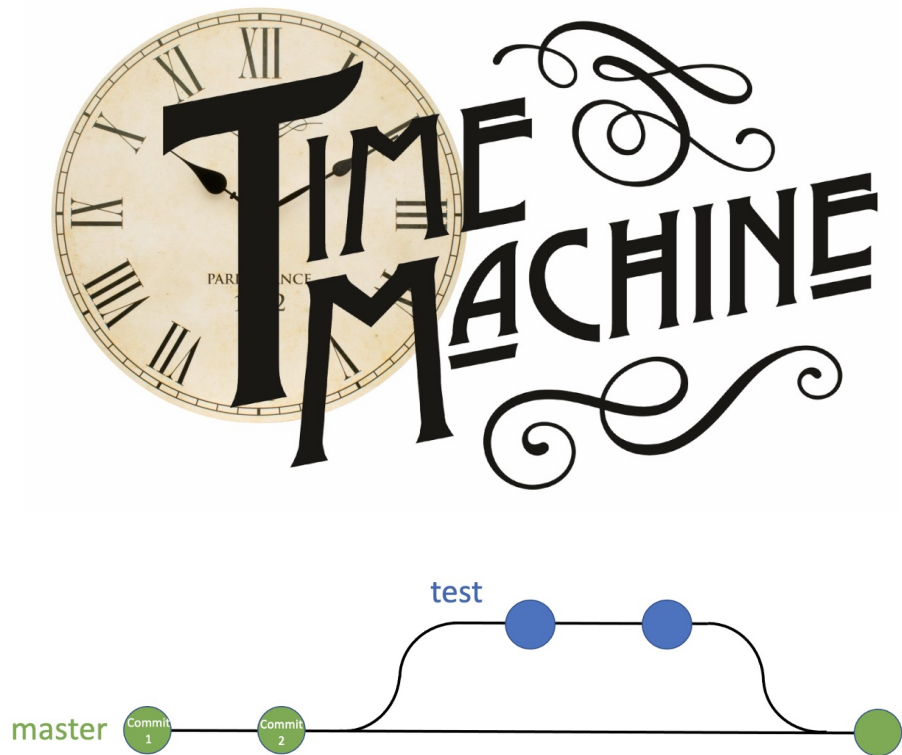
```
cd existing_git_repo  
git remote add origin git@gitee.com:zhongpu/git-demo-example.git  
git push -u origin master
```



本周作业（homework 3）

- 新建一个GitHub仓库，命名为se_homework，每次作业一个文件夹（如hw1），然后将第一次和第二次作业通过Git提交到GitHub。最后通过飞书私信老师GitHub地址。（10分）

2. Git基础（续）



2.1 蔬菜订单系统

veggies.py > ...

You, 33 seconds ago | 1 author (You)

```
1  veg_price = {'cabbage': 2.87, 'carrot': 1.59, 'spinach': 3.33, 'asparagus': 2.54, 'artichoke': 3.00, 'lettuce': 2.43}
2  order = {'cabbage': 2, 'artichoke': 5}
3
4  total = 0
5  for veggie, quantity in order.items():
6      total += veg_price[veggie] * quantity
7  print(total)
```

git commit -am "veggies order system"

市场变动：提高价格

You, 22 seconds ago | 1 author (You)

```
1 | veg_price = {'cabbage': 2.9, 'carrot': 1.8, 'spinach': 3.6, 'asparagus': 2.94, 'artichoke': 3.20, 'lettuce': 2.49}
2 | order = {'cabbage': 2, 'artichoke': 5}
3 |
4 | total = 0
5 | for veggie, quantity in order.items():
6 |     total += veg_price[veggie] * quantity
7 | print(total)
```

```
git commit -am " update price"
```

如果想查看涨价前的代码，应该如何实现？

```
9ee5c1e (HEAD -> master) update price  
51f6b89 veggies price system  
47888c8 remove pumpkin from veggies  
c676cc2 first commit  
(END)
```

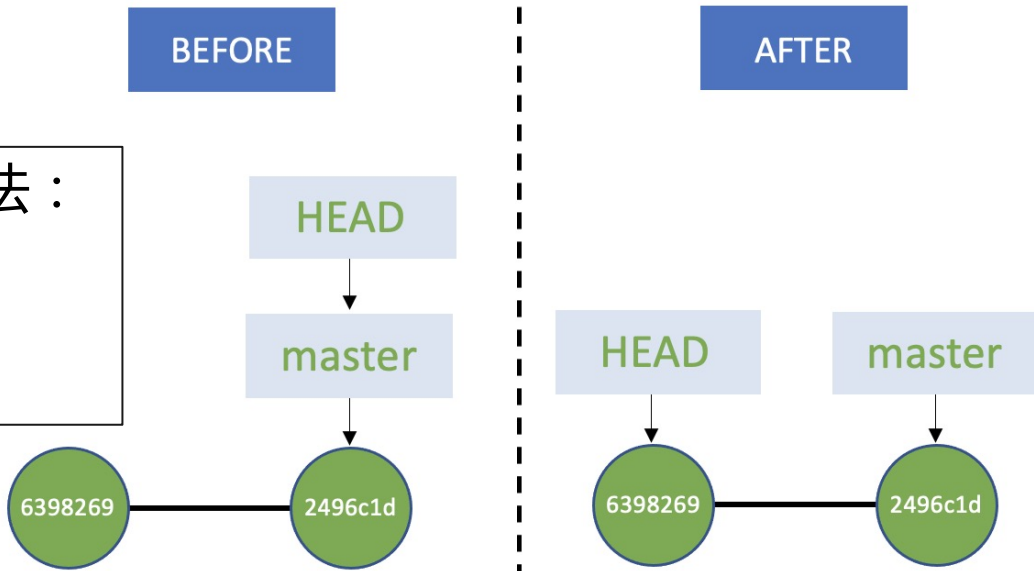
git checkout 51f6b89

再看git checkout

git checkout主要有三个用法：

- 文件
- **提交**
- 分支

查看旧快照 (snapshot)



小任务

`git log`只能查看以当前状态为终点的历史;
`git reflog`可以查看当前仓库的历史


在checkout之后，查看log的内容。

```
51f6b89 (HEAD) veggies price system
07d14ac (test) fruits set
47888c8 remove pumpkin from veggies
c676cc2 first commit
(END)
```

2.2 撤销修改

- 情况一：在现有代码的基础上，修改了一番，但最后放弃了，想撤销到上次提交的状态（即**恢复工作区**）

```

 ~/github/git-demo-example  git master  git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   veggies.py

```

2.2 撤销修改

- 情况一：在现有代码的基础上，修改了一番，但最后放弃了，想撤销到上次提交的状态（即**恢复工作区**）

```
git checkout veggies.py
```

git checkout主要有三个用法：

- **文件**
- 提交
- 分支

撤销当前工作区的修改

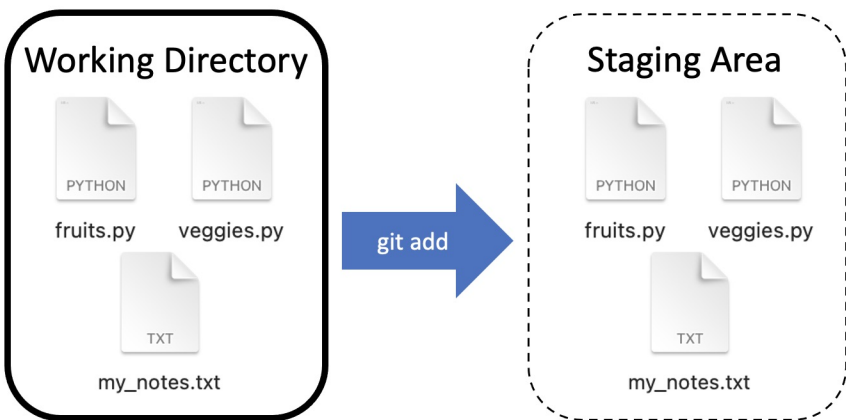


git checkout主要有三个用法：

- 文件 → git restore
- 提交
- 分支 → git switch

2.2 撤销修改

- 情况二：不小心将一个文件添加到了缓存区，想撤回添加。

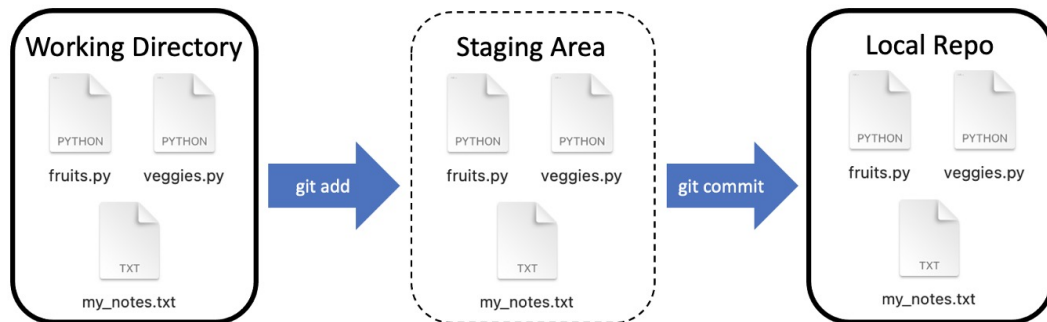


```
git restore --staged my_notes.txt
```

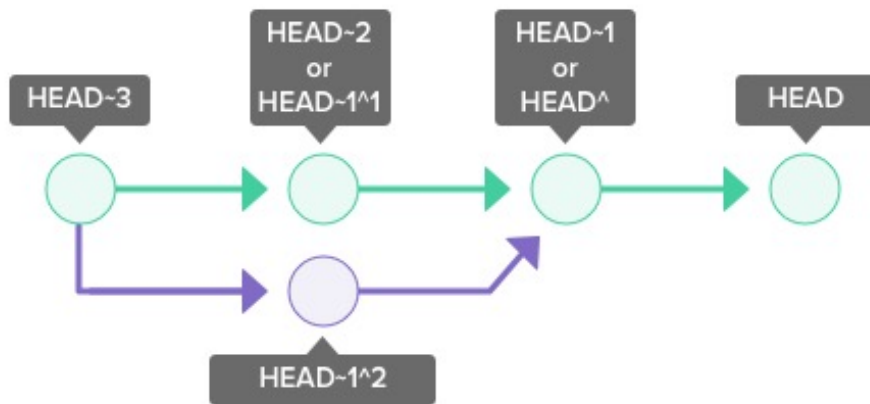
my_notes.txt 不应该被加入缓存区

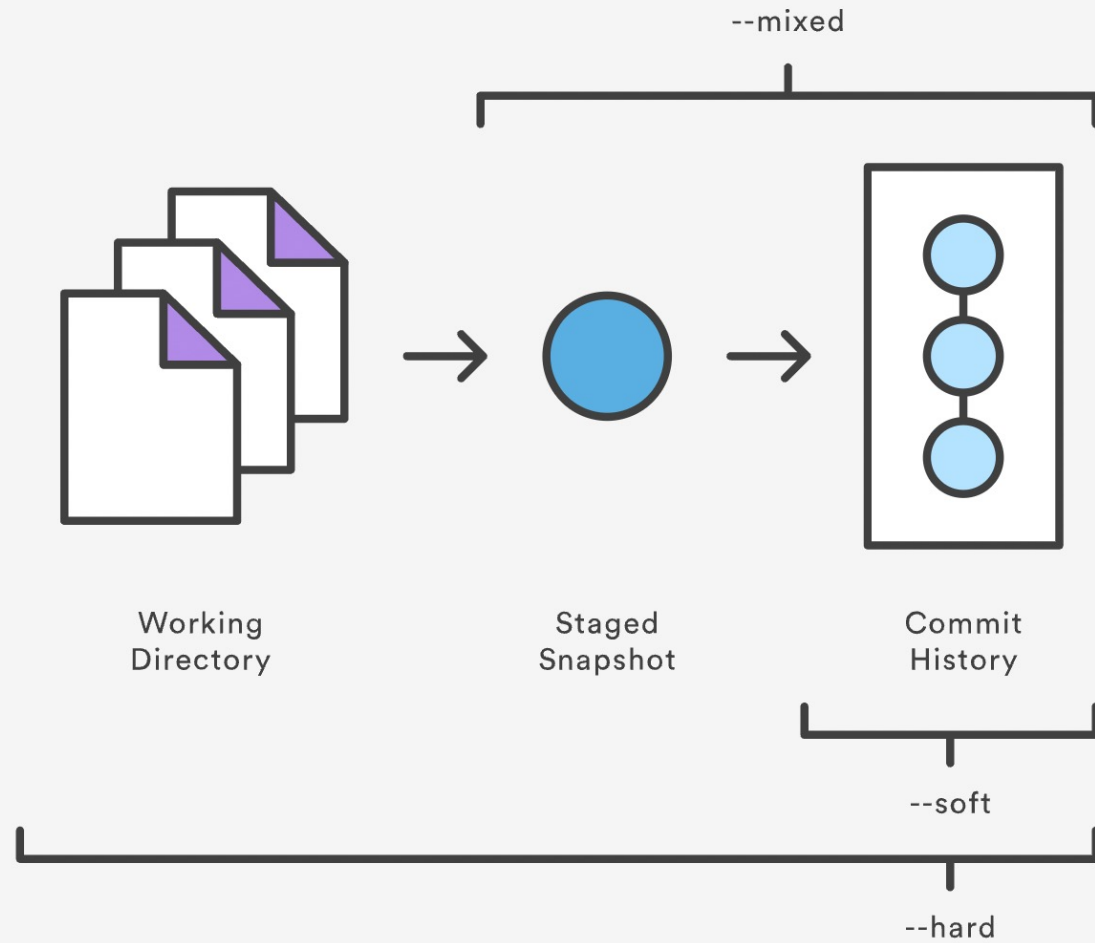
2.2 撤销修改

- 情况三：撤销提交



```
git reset --soft HEAD^
```



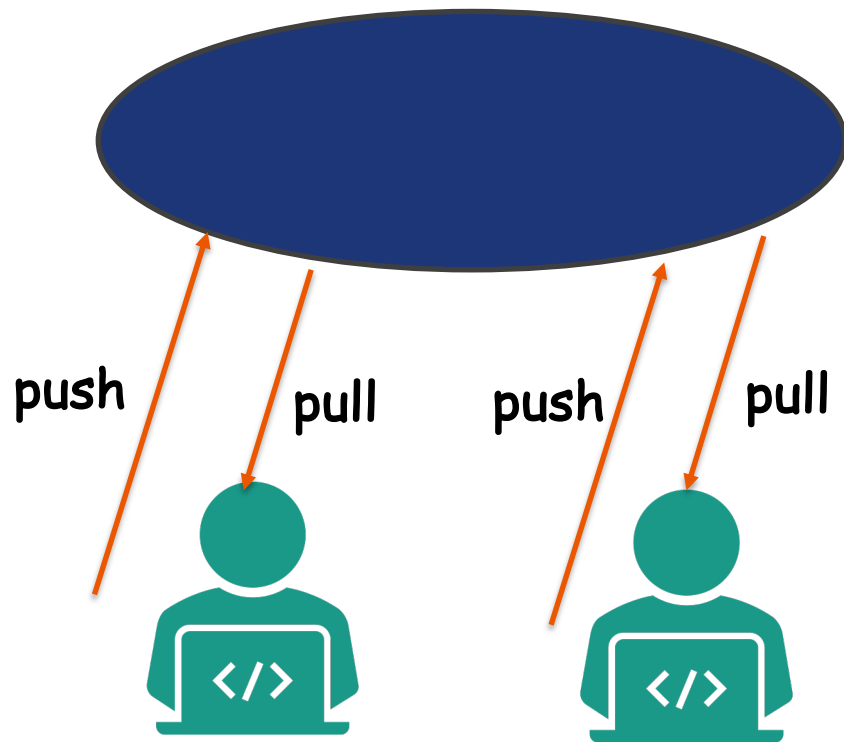




2.3 修改提交，压缩历史，修改历史

- `git commit --amend`
- `git rebase -i`
- ...

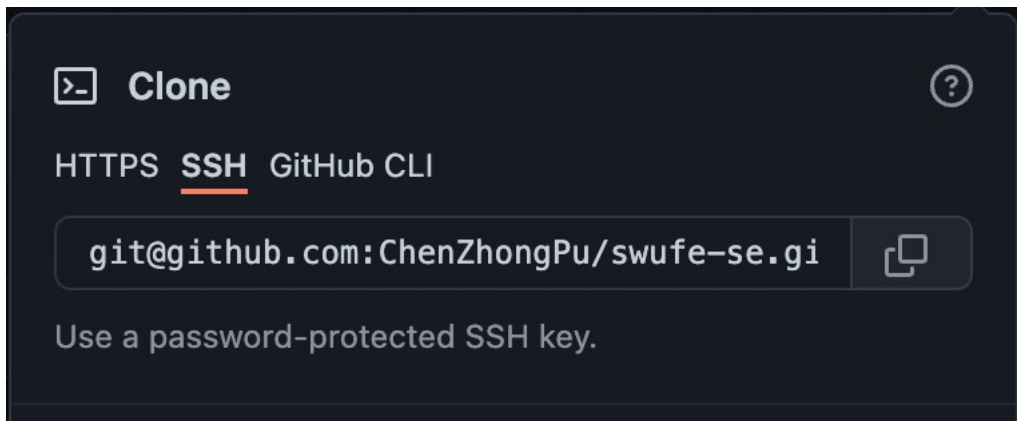
3. 合作模式



小任务

当远端仓库有更新，可以使用 `git pull` 获得更新。

- 练习 `git clone`。



git remote add <name> <url>

```
git remote add origin git@gitee.com:zhongpu/git-demo-example.git
```

git push <remote-name> <branch-name>

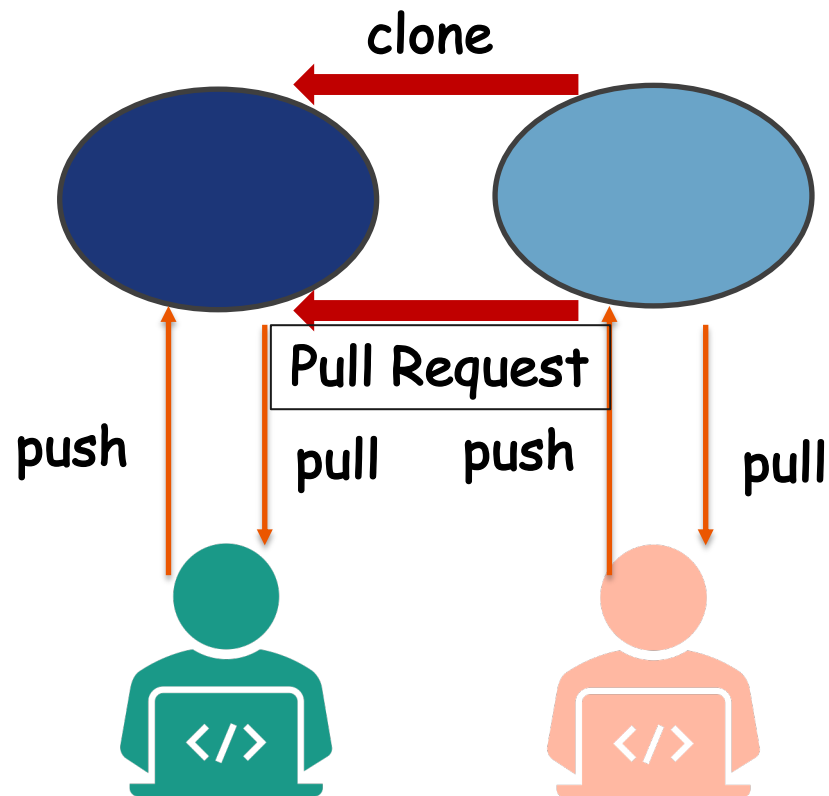
```
git push -u origin master
```



简单工作流需要的最少命令

- `git add`
- `git status`
- `git commit`
- `git clone`
- `git push`
- `git pull`
- `git checkout`

3. 合作模式



4. 设计有意义的软件工程大作业

缺乏复杂性和易变性

欢迎您来到图书管理系统

图书管理系统注册界面

学号:

设置密码:
密码由6-12位数字、字符组成

确认密码:

姓名:

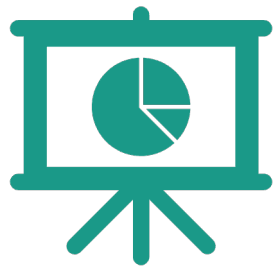
性别: ☐ 男 ☐ 女

出生日期:
格式: (****.***)**

注册

4.1 从数据方面扩展

- ❑ 数据属性：比如普通整数到大整数
- ❑ 数据规模：比如从几百到10万
- ❑ 数据维度：比如到一维到多维
- ❑ ...



4.2 从需求方面扩展



真实的需求都是复杂的，不明确的。比如考虑下面的扩展：

- ❑ 排序不仅仅是显示结果，用动画把排序过程也显示出来
- ❑ “统计程序由多少行”，把注释行、空行单独统计
- ❑ 文档编辑软件，处理100M的文本
- ❑ 图书馆管理系统模拟10万本书，而不是不到10本书的记录
- ❑ 文档编辑软件，支持Markdown语法
- ❑ ...

4.3 从用户方面扩展

- ❑ 单机变成多用户
- ❑ 用户来自全世界，如何实现多语言
- ❑ 有恶意用户怎么办
- ❑ ...



4.4 从软件构建方面

- ❑ 平台迁移
- ❑ 多语言混合
- ❑ 如何升级部分服务，而让尽量不影响其他模块
- ❑ ...





小组作业（不多于4人，不少于2人）

1. 分组讨论一个**有真实需求**的“大作业”，并由组长提交分组名单。