Google Developer Student Clubs

Git 指令與觀念 (二)

實務多人共同開發的良好習慣

RyanLin 林昕鋭
(@ryanlinjui)

上課時間: 19:00 - 20:30
2025.04.15 Tue.

本日Outline

// 19:00 - 20:30

前置準備	踏入 Git 的第一步	p3
19:00	前期回顧	p5
19:05	開發小知識	p8
19:20	你的紀錄呢？版本回溯與恢復	p16
19:40	多人專案的開發	p24
20:00	中場休息	
20:05	Merge Confilct 合併衝突處理	p37
20:28	結語	p45

踏入 Git 的第一步

前置準備

- 下載 Python3 (版本不要求，初學者也 OK)，課堂上會有開發小小實務！
- 註冊 GitHub 帳號 [Link]
- 下載 Git 相關工具 [Link]
- 得到一個好用的文件編輯器，這裡一律推薦 VSCode [Link]
- 拿取 GitHub API Token，就像你的專屬鑰匙 [Link]

(以上前置準備一定要在課程開始前準備好哦！)

前期回顧

你還記得嗎？

- 常見的名詞：Repository, Commit, Branch
- 在多人共同開發中你必須要有的習慣
- 你學會的第一組 Git 基本指令運用
- GitHub 工程師的交友平台

(如果忘記了記得前往 GDG Lecture - Git 指令與觀念(一) 再次查看哦！

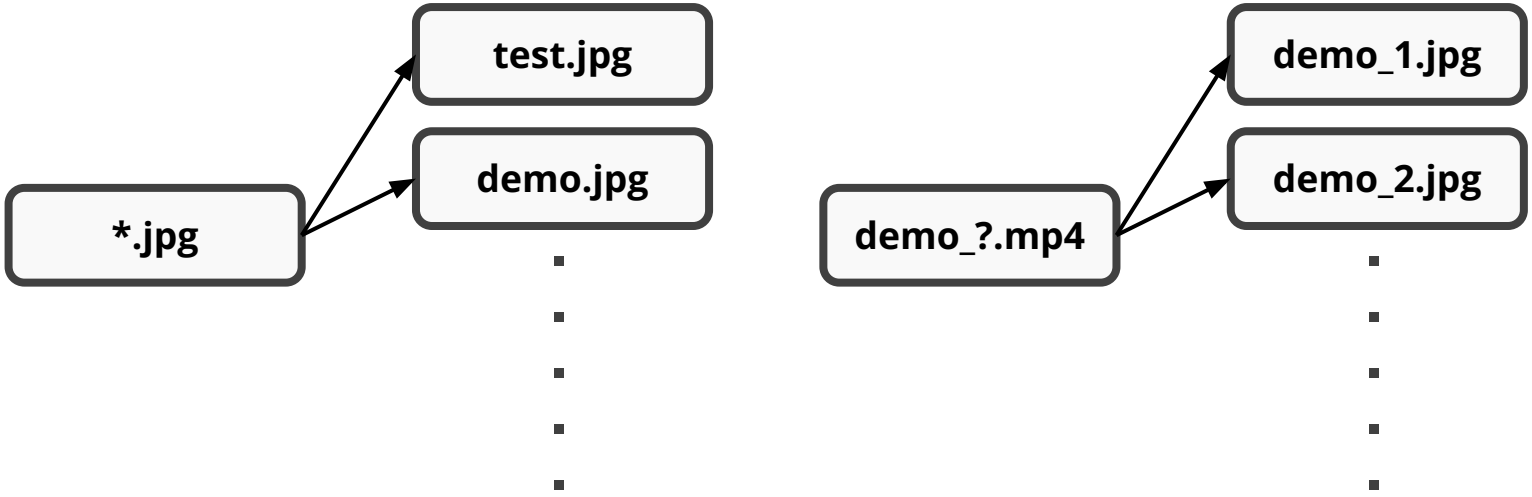
git add	檔案新增至暫存區 Adds files to the staging area.
git commit	暫存區的變更記錄到 repo Records changes from the staging area in repo.
git status	顯示目錄和暫存區狀態 Displays state of the directory & staging area.
git push	本地 repo 內容上傳到 GitHub Uploads local repo content to GitHub.
git pull	從 GitHub 提取變更到本地 repo Fetches changes from GitHub to local repo.
git branch	列出、建立或刪除 branch Lists, creates, or deletes branches.
git checkout	切換到另一個 branch Switches to a different branch or restores files.
git init	初始化一個新的 Git repo Initializes a new Git repo.
git remote	管理與 GitHub 的連線網址 Manages connections to GitHub repo URL.
git config	設定 Git 使用者特定配置 Configures user-specific settings for Git.
git log	查看 repo 的 commit 歷史 Shows the commit history of the repo.
git restore	還原目錄檔案到最近 commit Restores working tree files to their last committed state.
git reset	將當前分支移到某 commit Moves the current branch to a different commit.

開發小知識

.gitignore 忽略你的無謂檔案

- 定義哪些文件或資料夾不納入Git 的版本追蹤
- .gitignore 檔案的每一行代表一個忽略規則
  - 想忽略的檔案或資料夾路徑
  - 可用 Wildcard 忽略多個檔案 (例: * 和 ?)
  - 常忽略的檔案:
    - .env 環境變數機密檔案、package 大容量資料夾、.DS_Store 對專案無意義檔案

Wildcard 萬用字元常用示意圖



(資料參考: https://zh.wikipedia.org/zh-tw/%E9%80%9A%E9%85%8D%E7%AC%A6)

實作 : .gitignore 的簡單操作 - 前置準備

- 創建資料夾 (命名隨意), 並且用 vscode 開啟資料夾
- 開啟終端機 terminal, 初始化 Git Respository
  - **輸入 git init**
- 在專案資料夾下創建名為 .gitignore 的檔案
- 查看目前 Respository 的狀態
  - **輸入 git status**

實作 : .gitignore 的簡單操作 - * 符號

- 在專案中新增一些資料夾或檔案，並且查看目前 Respository 的狀態
  - 新增一些命名隨意的 txt 檔案: test.txt, aaa.txt, hahaha.txt……
  - 新增名為 docs 資料夾，並且放入一些隨意任何除了 txt 的檔案
- 在 .gitignore 輸入以下幾行並且儲存:
  - ***.txt**
  - **docs/***
- 再次查看目前 Respository 的狀態，檔案被忽略了

實作 : .gitignore 的簡單操作 - ? 符號

- 在專案中新增一些資料夾或檔案，並且查看目前 Respository 的狀態
  - 新增名為 system-log 資料夾，並且在資料夾下新增一些以數字命名的 log 檔案: run1.log, run2.log …… run10.log
- 在 .gitignore 輸入以下幾行並且儲存:
  - **system-log/run?.log**
- 再次查看目前 Respository 的狀態，大部分檔案被忽略了，將所有檔案放置暫存區，再查看狀態可以看到有一個檔案未被忽略
- 將暫存區全部檔案 Commit 起來

Now It’s your Turn !!

7 mins

Commit 訊息的一些淺規則

- 說明本次修改目的為標題，再寫清楚描述詳細更動內容
  - 格式範例: “<標題>: <詳細更動內容>”
- 最好控制在 50 字元以內，並使用祈使語氣
  - 例: “Add log system” 而非 “I added log system”。
- 常用的標題有:
  - **feat (新增功能)、fix (修正 bug)、docs (文件更新)、refactor (重構程式碼)、test (增加或更新測試)、chore (其他雜項)**

(資料參考: https://wadehuanglearning.blogspot.com/2019/05/commit-commit-commit-why-what-commit.html)

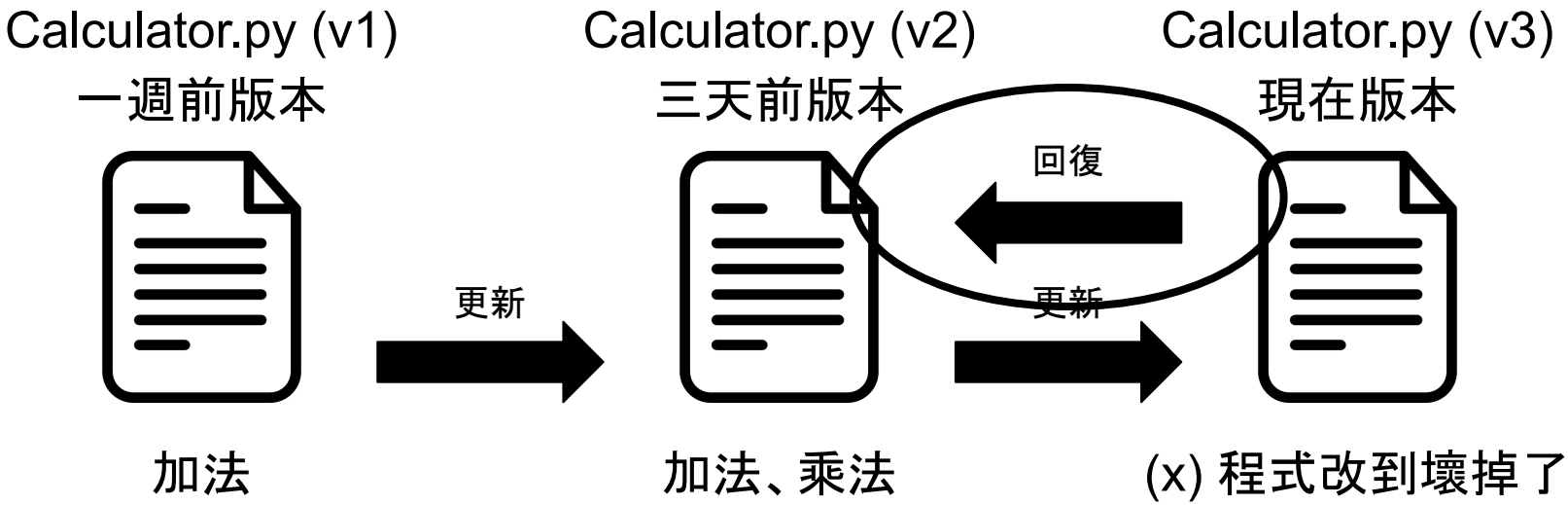
你的紀錄呢？版本回溯與恢復

我們將會學到的一些指令

- **git restore <files> / git restore --staged <files>**
  - 還原工作目錄或暫存區的檔案至指定的版本或狀態
- **git reset --soft <commit>**
  - 將專案回朔到之前的某個 commit
- **git log**
  - 用來檢視 Respository 的提交歷史(commit history)

git reset / git log

(回顧)版本控制與回復 (Commit 的回朔)



還有一個回復的情況..... (暫存區的回朔)

實作:我不是要 Commit 那些檔案

- 在 run10.log 檔案新增隨意內容後，查看 Respository 狀態
- 將 run10.log 變更檔案放置暫存區，再次查看 Respository 狀態
- 將變更檔案從暫存區移出來，再次查看 Respository 狀態
  - **輸入 git restore --staged <files>**
- 將變更檔案回復到最近的 Commit 紀錄，再次查看 Respository 狀並確認 run10.log 檔案內容
  - **輸入 git restore <files>**

Now It’s your Turn !!

7 mins

實作:版本回溯的基本操作

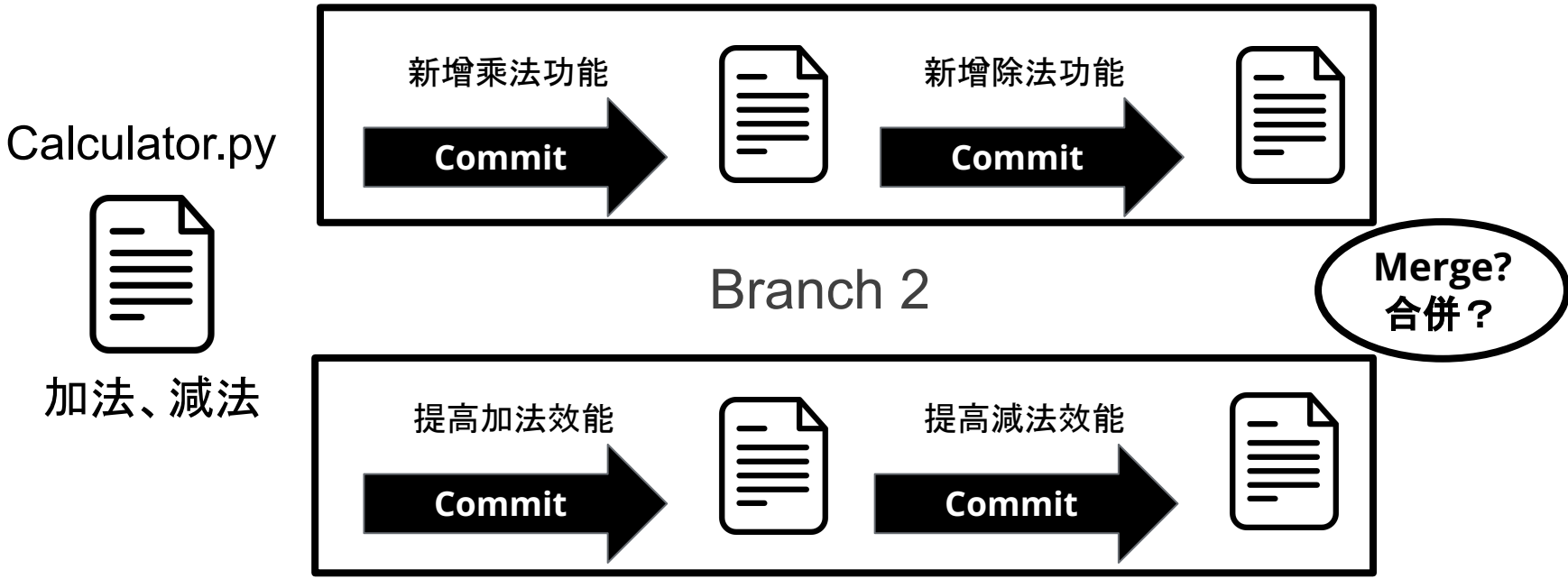
- 在 run10.log 檔案新增隨意內容後並做一次 commit
- 隨意新增一個檔案後再做一次 commit (注意別被 ignore 掉)
- 查看提交歷史，可以看到之前提交以及你剛剛提交的 commit
  - **輸入 git log**
  - **請記住前三個 commit 字串代碼 (可以記前 7 碼就好)**
- 回朔版本各三個 commit，並且在每次回朔後查看狀態與提交歷史
  - **輸入 git reset --soft <commit>**
  - **你也可以嘗試輸入看看 git reset --soft HEAD^ 看會發生什麼事**

Now It’s your Turn !!

10 mins

多人專案的開發

(回顧) Branch 示意圖



合併後將會得到？

Branch1: 高效能加減法

operation.py U

operation.py > subtract

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

Branch2: 新增乘除法

operation.py U

operation.py > multiply

```python
def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```

Merge
合併

Merged Branch

operation.py U

operation.py > divide

```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```

我們將會學到的一些指令

- **git branch / git branch -D <branch>**
  - 查看所有的 branch 分支 / 刪除指定 branch 分支
- **git checkout <branch> / git checkout -b <new-branch>**
  - 切換至指定 branch 分支 / 新增 branch 分支並切換過去
- **git merge <another-branch>**
  - 合併另一個 branch 分支到目前的 branch 分支

實作：Branch 的簡單操作

- 請做查看 Branch 列表、新增 Branch、切換 Branch 的動作
  - **名稱隨意，但是要 查看列表裡有你新增的 Branch**
  - **試試在各個 Branch 之間做切換動作（處於無需做 Commit 的狀態才能切換）**
- 請在你新增的 Branch 上做一個 Commit
- 切換至 main branch，選取 Branch 做 Merge，並查看提交紀錄 (log)
  - **輸入 git merge <another-branch>**
- 刪除新 Branch，並且查看 Branch 列表確認是否被刪除
  - **輸入 git branch -D <branch>**

Now It’s your Turn !!

7 mins

多人開發專案的前置準備

- 定好專案的一些事情:
  - 目標、核心功能、需求分析與目標使用者
  - 討論會用到的框架與技術，並建構專案基礎架構設計
  - 定義團隊規範 (例：程式碼風格、溝通工具與需求任務管理)
- 初始建立好 GitHub，並且寫 README 說明專案架構內容與開發需知
- 大家合作分配任務後開始開發！

(回顧)多人共同開發 - 你必須要有的習慣

- 除了初始化 Repository 內容之外，切勿在主要分支 (main branch) 做事
- 當你正要做一個新功能、改動時，請拉一個 branch
- 隨時更新 Repository 內容 (git pull)
- 讓其他人更懂你寫的內容
  - 像是：commit 訊息、branch 名稱、程式碼命名與註解、完整的 README
- 切勿上傳敏感、無意義或龐大資料 (.gitignore)
  - 像是：環境變數檔案 .env、package、.DS_Store

實作:組長做的第一件事

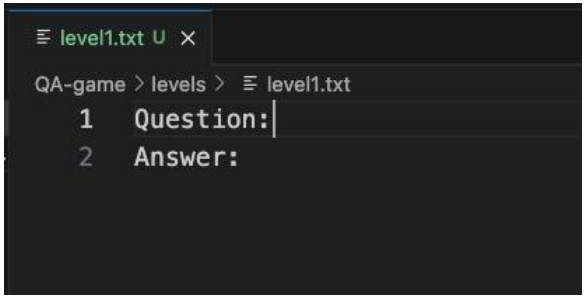
- 分組，並且指派一個人為組長
- 組長 Fork 專案並 Clone 到你的電腦內 [Link] (已經做過的可以不用再做)
- 團隊將以組長 Fork 的 Repositroy 作為組的專案，組長要邀請其他組員給予 Repository 權限
- 組員們請到 Email 接受邀請函，並且 Clone 組長的 Respository
- 組長到 Repository Setting 頁面，尋找 Branch 控制，禁止 Push 至 main branch 開關打開

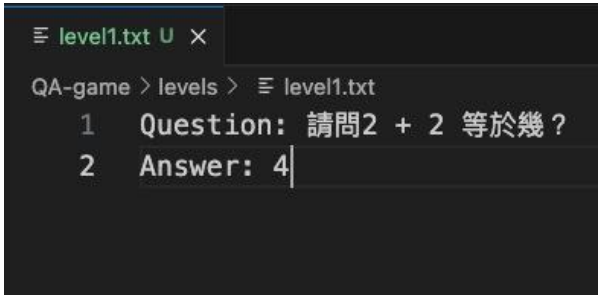
實作:Q&A 小遊戲 - 前置準備

- 組長請先做以下事情:
  - 在自己 Fork 的 Respository 下在 “NTNU-GDG/git-tutorial-2/assignments” 資料夾下新增資料夾命名為 team<你的組別號碼> 作為你們團隊的資料夾
  - 將 “NTNU-GDG/git-tutorial-2/QA-game” 資料夾複製到團隊的資料夾底下
  - 將變更紀錄 Push 至 GitHub 上
- 組員們請更新 Respository

實作:Q&A 小遊戲 - 分工新增功能

- 你們被指派要完成 Q&A 小遊戲:
  - **專案中有一個名為 “level” 的資料夾, 裡面以 level<number>.txt 讀入 Q&A 關卡**
  - **請各自設計一個關卡中的問題與答案, 請討論誰負責第幾關之對應的 txt 檔案**
  - **請記得開發時做任何改動都要拉一個 Branch 分支, 必須測試沒問題後再做 Push 動作!**
  - **創造一個 PR 並且描述清楚標題以及你做了什麼, 等待組長與其他人 查看並且合併**
- 如要執行程式, 請在專案底下開啟終端機執行 **“python main.py”**
- 組長審查完所有組員的 PR 並且允許 Merge 之後, 請更新 main branch 並且測試程式是否有問題, 最後將團隊 Fork 的 Respository 發 PR 至此地方 [Link] 即完成工作 !






```
ryanlinjui@ryan-macbook QA-game % python main.py
歡迎來到 QA Game！總共 1 關！
==========
第 1 關：
請問2 + 2 等於幾？
你的答案是：5
回答錯誤，正確答案是：4
----------
遊戲結束，你的總分是：0 / 1
```

Now It’s your Turn !!

中場休息: 10 + 5 mins

Merge Confilct 合併衝突處理

Conflict 衝突？

- Git 衝突發生於多人協作或分支合併時，當兩個或多個不同的修改同時作用於同一個檔案的相同部分，Git 就無法自動判斷要保留哪一個修改，這時就需要使用者介入進行手動整合。
- 為什麼會有衝突？
  - 多人協作：多個開發者同時修改同一檔案的相同區段。
  - 分支合併：不同分支之間在相同程式碼區域有不一致的變更。
  - 缺乏同步更新：本地與遠端更新長期不對齊，導致修改重疊。

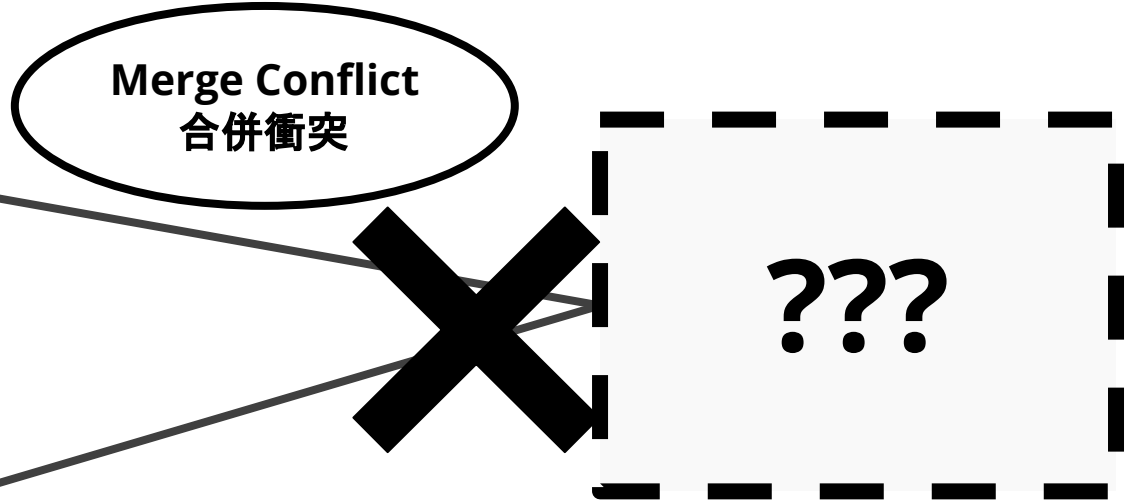
Conflict 衝突示意圖 - 實作次方函式

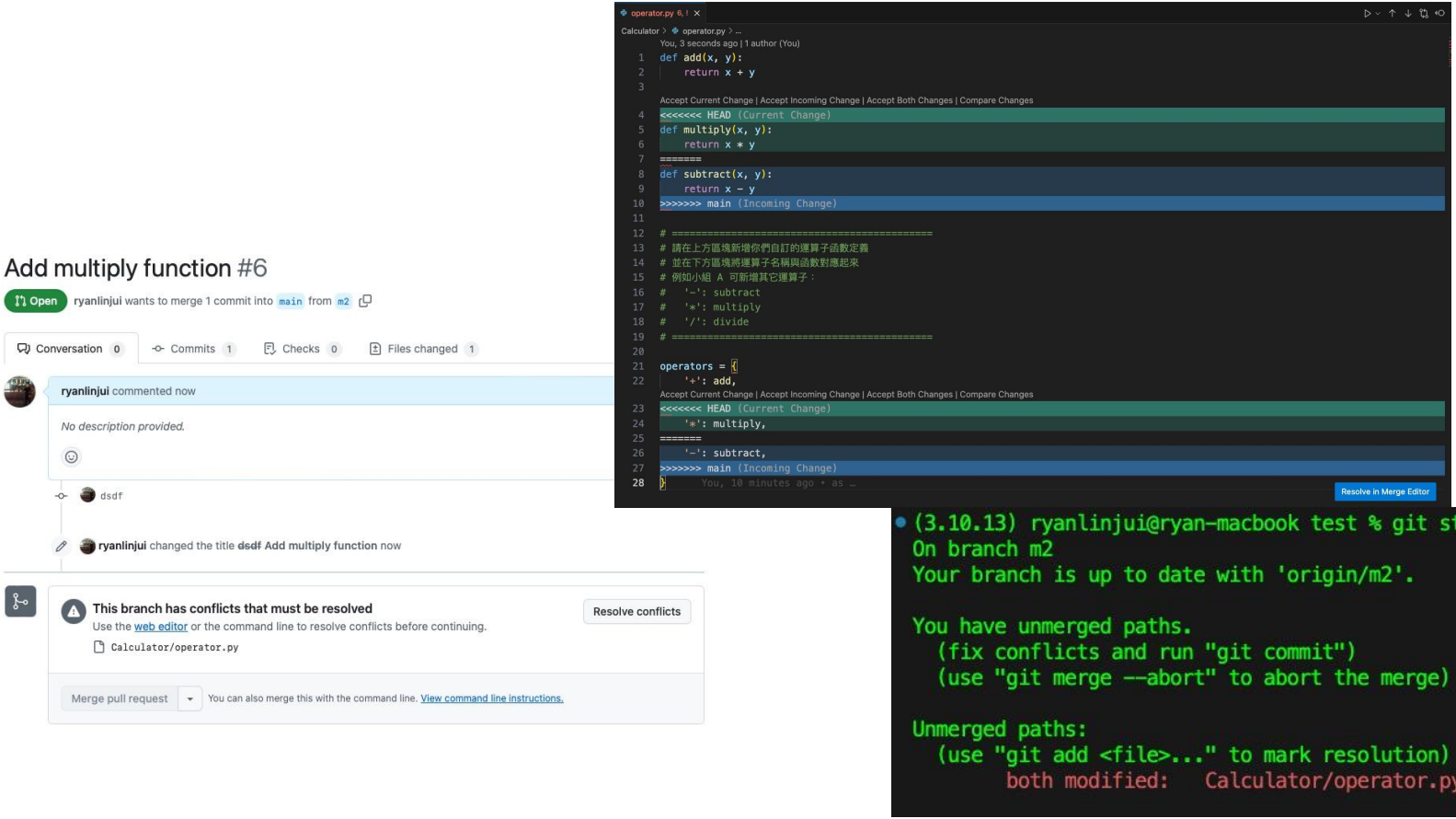
Branch1: 使用 math 函式庫實作

```python
import math

def pow(a, b):
    return math.pow(a, b)
```

Branch2: 直接計算實作

```python
import math

def pow(a, b):
    return a ** b
```




operator.py 6, !
Calculator > operator.py > ...
You, 3 seconds ago | 1 author (You)
def add(x, y):
return x + y
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
def multiply(x, y):
return x * y
=======
def subtract(x, y):
return x - y
>>>>>>> main (Incoming Change)
# ==========
# 請在上方區塊新增你們自訂的運算子函數定義
# 並在下方區塊將運算子名稱與函數對應起來
# 例如小組 A 可新增其它運算子：
# '-': subtract
# '*': multiply
# '/': divide
# ==========
operators = {
'+': add,
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<<< HEAD (Current Change)
'*': multiply,
=======
'-': subtract,
>>>>>>> main (Incoming Change)
}
You, 10 minutes ago • as …
Resolve in Merge Editor
Add multiply function #6
Open
ryanlinjui wants to merge 1 commit into main from m2
Conversation 0
Commits 1
Checks 0
Files changed 1
ryanlinjui commented now
No description provided.
dsdf
ryanlinjui changed the title dsdf Add multiply function now
This branch has conflicts that must be resolved
Use the web editor or the command line to resolve conflicts before continuing.
Calculator/operator.py
Resolve conflicts
Merge pull request
You can also merge this with the command line. View command line instructions.
(3.10.13) ryanlinjui@ryan-macbook test % git status
On branch m2
Your branch is up to date with 'origin/m2'.
You have unmerged paths.
(fix conflicts and run "git commit")
(use "git merge --abort" to abort the merge)
Unmerged paths:
(use "git add <file>..." to mark resolution)
both modified: Calculator/operator.py

實作:計算機小程式 - 前置準備

- 組長請先做以下事情:
  - 將 “NTNU-GDG/git-tutorial-2/Calculator” 資料夾複製到團隊的資料夾底下
  - 將變更紀錄 Push 至 GitHub 上
- 組員們請更新 Respository

實作:計算機小程式 - 分工新增功能

- 你們被指派要完成計算機小程式:
  - **專案中有一個名為 “operations.py” 檔案是提供計算機功能的各個運算 function 的程式碼**
  - **請各自實作一種運算 function, 你們將面臨大家在同一檔案上傳成果的 Merge Conflict 狀況**
  - **請記得開發時做任何改動都要拉一個 Branch 分支, 必須測試沒問題後再做 Push 動作!**
  - **創造一個 PR 並且描述清楚標題以及你做了什麼, 等待組長與其他人 查看並且合併**
- 如要執行程式, 請在專案底下開啟終端機執行 **“python main.py”**
- 組長審查完所有組員的 PR 並且允許 Merge 之後, 請更新 main branch 並且測試程式是否有問題, 最後將團隊 Fork 的 Respository 發 PR 至此地方 [Link] 即完成工作 !

operator.py U

Calculator > operator.py > ...

```python
def add(x, y):
    return x + y

# ============================================
# 請在上方區塊新增你們自訂的運算子函數定義
# 並在下方區塊將運算子名稱與函數對應起來
# 例如小組 A 可新增其它運算子：
#   '-': subtract
#   '*': multiply
#   '/': divide
# ============================================

operators = {
    '+': add
}
```

```
ryanlinjui@ryan-macbook Calculator % python main.py
目前支援的運算子：+
請輸入第一個數字: 1
請輸入第二個數字: 3
請輸入運算子： +
結果: 4.0
```

Now It’s your Turn !!

15 mins

結語

- 目前大家經過了兩堂系列課的學習，其實有兩件大事實要跟你說：
  - 你事實上只學到 Git 中大約 3 % 的指令
  - 但是 **3 % 的指令卻能在多人開發實務中做到 80% 情況**
- **Git 指令不用一次學到好學到滿學到 100 %**
- **請多多跟找其他人一起做專案累積經驗與機會** ，在專案開發學習中自然會有需求的情況，你也將會自動的去學習其他 Git 技能與知識！！

Thank you for Your Participants !!!