商管程式設計 108-1 Project Development

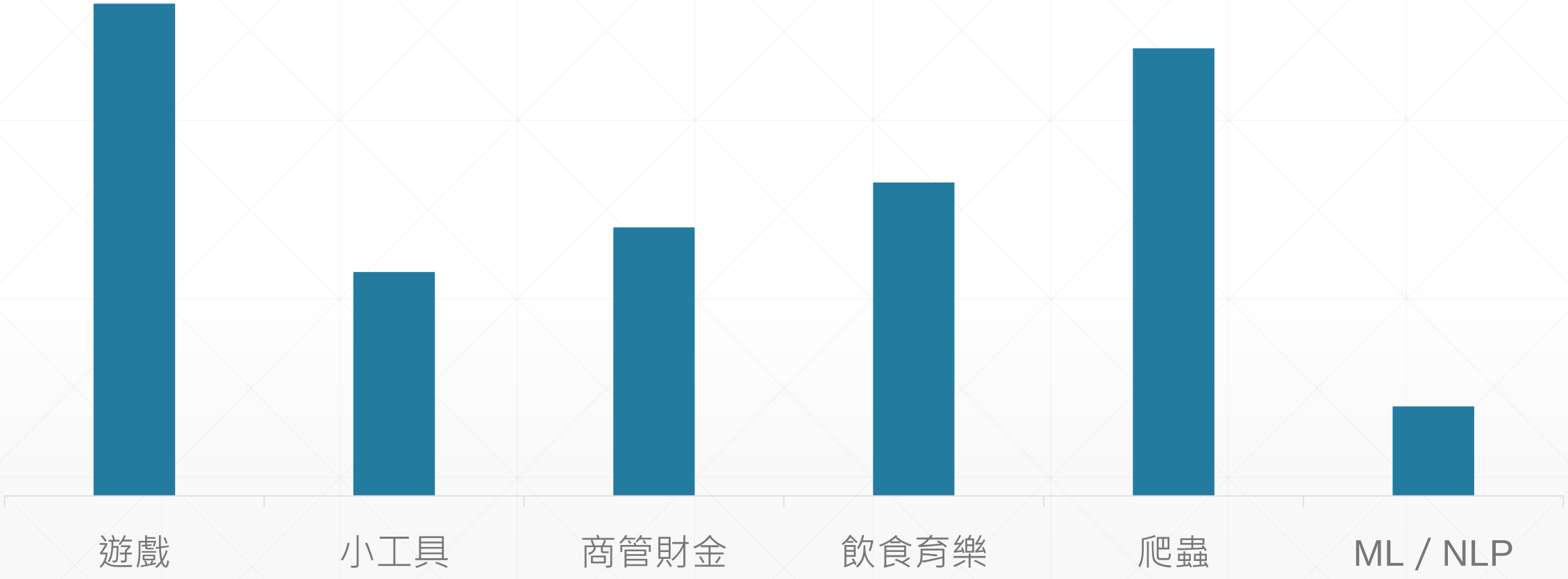
2019/12/11-12

Agenda

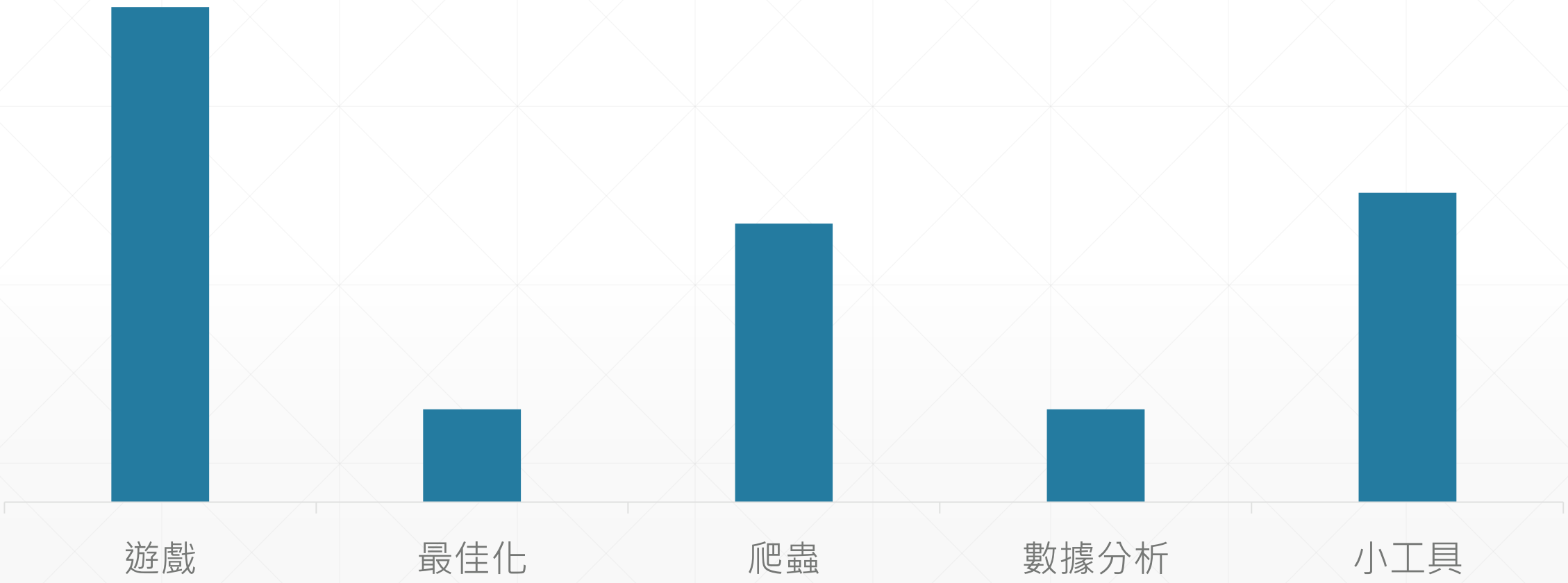
- 引言
- 軟體專案分工
- 開發流程
- 結語

引言

大家大概都做些甚麼？（107-2）



大家大概都做些甚麼？（108-1）



大家的主題好像大概就長那樣

- 這也不怎麼奇怪
  - ~~因為助教隨便亂歸類~~
  - ~~所學就這樣，同溫層就長這樣，選擇類似的主題也很正常~~
  - 同樣的主題，有些組就能做得比較好，然後得到高分；有些組做出來的東西就像是拼裝車
- 那應該如何才能把報告做好？
  - 而且只剩下短短一個月 (不到)？

軟體專案分工

第一個問題：怎麼分工？

- 每一種任務，都需要不同的作業與分工方式
  - 製作一個程式產品，照著平常做報告的思維下去分工，肯定就不會有很好的效果
    - 兩個人寫 code、一個人上台、一個人做投影片、一個人書面 = GG
- 專業的分工長甚麼樣子呢？
  - 不知道怎麼做，那就去參考別人怎麼做

專業的分工與開發流程長甚麼樣子呢？

- PM (Product Manager)
- UI/UX (User Interface/User Experience)
- Dev (Developer)
  - Frontend vs. Backend
- QA (Quality Assurance)

- 不是每個人都只忙著寫 code

PM

- 領導、規劃整個專案的進行
  - 要做甚麼？怎麼分工？目標是甚麼？哪些事情優先？
  - 大家可能都在瞎子摸象，但 PM 必須知道大象的全貌
  - 建立共同工作的平台，促進成員之間的溝通
- 實作能力不一定滿分，但責任感必須滿分
  - 有人出狀況 → 跳下去幫忙做
  - 有東西沒做 → 跳下去幫忙扛 (然後想辦法找人幫忙)

UI/UX

- UI 跟 UX 不一樣，但常常被放在一起
  - 有時候還要身兼美宣
- UI：使用者介面
  - 存檔按鈕要甚麼顏色？怎麼排版比較好看？
- UX：使用者體驗
  - 下一個畫面應該要有哪些內容？按鈕放哪邊，用起來比較順？
- 需要有一點美感細胞、一點想像力、一點設身處地

Dev

- Team 中的技術強者們
  - 想盡辦法把設計好的程式用 code 實現出來
  - 根據 QA 的回報，將 bug 予以清除
- Frontend vs. Backend
  - 前端 = 使用者看得到的部分
    - 按鈕、視窗、介面、圖形...
  - 後端 = 使用者看不到的部分
    - 系統架構、演算法、資料庫、函式...

QA

- Debug：找到不足之處並協助修正
  - 程式上的錯誤或者是設計上的錯誤都包含在內
- 建立適當的測資及測試步驟，統整後進行回報與討論
  - 該用甚麼測資？為甚麼會有這個 bug？可能是哪裡錯了？
  - 現在的程式流程有甚麼問題？使用者會不會弄到一半，結果不知道怎麼繼續下一步？
- 需要細心、縝密的思考與觀察

其他

- Game Designer
  - Game Balance
- Analyst
- Data Engineering
- Marketing
  - Customer Relationship
- Legal (?!)

開發流程

第二個問題：該怎麼動手？

- 先想好要做甚麼，一次搞定？
  - 先做一個 (或多個) 雛型出來，再討論下一步？
- 要怎麼做才能弄出一個厲害的期末報告？
- 要如何順暢的合作？

規劃→實作→測試回饋→修正規劃→……

- 決定目標
- 動手做
- 測試、回饋
- 調整目標
- 動手做
- …

規劃

- 制定合適的目標
  - 你的「客戶」只有老師、助教和同學；他們會喜歡甚麼東西？
  - 使用者需求！
- 跟其他組做出差異化
  - 能做出別人沒想到的東西，絕對是很猛的一件事情
  - 多討論，多腦力激盪
- 你們有辦法做出來的東西，比你們想像中的還要多

實作

- 選擇適當的工具與方法
  - 請善用搜尋引擎，用英文搜尋會更好
  - Python 最強大的就是有很多人在寫，到處都找得到資源
- 適當的進度管理、工作分配、版本控制與合作溝通
  - 一群人一起寫 code 並不容易，版本混亂很麻煩
  - 讓大家彼此知道各自在做甚麼、進度做到哪裡

實作：Code Review

- 觀念：好的程式碼的關鍵不是簡潔，是**好讀**
  - Python 已經是最好讀的程式語言之一
  - 如果你真的完全看不懂別人的 code，那就是他寫得太複雜了
- Write a clean code!
  - 適當的註解、良好且一致的命名、不過度複雜的 code
- 利用 code review 互相監督程式碼的品質
  - 保持至少有兩個人知道一段程式碼在寫甚麼
  - 便於溝通、便於修改、便於以防萬一

測試、回饋

- 寫了這麼多作業，大家都知道 debug 很痛苦
  - 貢獻方式有很多種，code 寫得少更應該要努力幫忙 debug
  - 報告的時候，在台上當機就好笑啦
- 發現錯誤或不足之處，適當的回饋並討論
  - QA 不是一兩個人的工作，是每個人都要參與的過程
  - 晚改的風險：修好這裡的一個 bug，製造出那裡的兩個 bug
  - 問題越晚發現，越容易牽一髮而動全身

結語

重要：請學會使用你的電腦

- 請不要到台上來然後不會操作你的電腦
  - 很拖時間，很不專業，老師助教會很無言，印象分數會很差
- 請準備好你的投影工具
  - 簡報筆、轉接頭.....
- 投影模式有【延伸】【同步】
  - 拜託請學會使用！
- 不要讓所有努力敗在最後報告的時刻

小小的建議

- 每一種任務，都需要不同的作業與分工方式
  - 製作不同的程式，肯定也會產生不同的分工狀況
- 現在做的是期末報告，不一定要完全照這套下去分工
  - 今天介紹的是專業人士的作法，殺雞未必需要用牛刀
  - 人少好解決，但人多就需要仔細想一下怎麼分工好
  - 身兼多職是很常見（必定會出現）的事情 XD
- The most important thing: **COMMUNICATION**