逢甲大學資訊工程學系

基於變異測試之遊戲化程式學習開發

指導教授：薛念林教授

學生姓名：莊鎮維

目錄

[第一章 緒論 3](#_Toc55655700)

[1.1 研究背景 3](#_Toc55655701)

[1.2 研究動機 3](#_Toc55655702)

[第二章 相關技術及使用工具說明 4](#_Toc55655703)

[2.1 相關技術 4](#_Toc55655704)

[2.1.1 單元測試 4](#_Toc55655705)

[2.2.2 變異測試 5](#_Toc55655706)

[2.2 使用工具 6](#_Toc55655707)

[2.2.1 Django 6](#_Toc55655708)

[2.2.2 Python 6](#_Toc55655709)

[2.2.3 subprocess 7](#_Toc55655710)

[2.2.4 Pytest 7](#_Toc55655711)

[第三章 研究架構 8](#_Toc55655712)

[3.1 使用者與MuPy系統之互動關係 8](#_Toc55655713)

[3.2 系統架構 9](#_Toc55655714)

[3.2.1 用戶端 9](#_Toc55655715)

[3.2.2 應用服務端 9](#_Toc55655716)

[第四章 系統實作 11](#_Toc55655717)

[4.1 執行環境 11](#_Toc55655718)

[4.2 介面實作 11](#_Toc55655719)

[4.3 模組實作 14](#_Toc55655720)

[4.4 題目設計 17](#_Toc55655721)

[4.5 實作結果 19](#_Toc55655722)

[第五章 結論與建議 21](#_Toc55655723)

[第六章 參考文獻 22](#_Toc55655724)

# 緒論

## 研究背景

隨著現代資訊科技的發達，程式語言的種類也越來越多，而為了確保程式執行正確，程式開發者會進行所謂的「測試」，測試的種類很多，比如說：驗收測試、系統測試、整合測試、等等種類繁多的測試，而在這之中用來測試程式最底層的單位則是被稱為單元測試。透過單元測試，程式開發者能夠在開發週期的早期發現錯誤進行修復並節省成本，並且幫助程式研發團隊了解程式代碼。

## 研究動機

透過學校老師的教導、以及在網路上所接觸到與單元測試及變異測試有關的資訊後，我了解到了單元測試對於一個程式開發者的重要性，但是單元測試也是並非總是有效的，程式開發者一不小心就有可能寫出不夠完善的單元測試，不完善的單元測試自然無法找出程式有問題的地方，到最後整份程式完成後才發現程式出錯了，就必須花更多的精力去修正這些問題，而為了避免以上的事情發生，就必須要提高單元測試的正確性，這時候可以藉由變異測試來檢測單元測試的正確性，以提高單元測試的正確性，而為了能讓更多人去認識到這件重要的事情，我希望能透過開發出一個遊戲系統，可以讓人們更加理解變異測試是如何運行的以及理解它的原理。

# 相關技術及使用工具說明

## 相關技術

### 單元測試

在電腦編程中，單元測試（Unit Testing）又稱為模組測試，是針對[程式模組](https://zh.wikipedia.org/wiki/%E6%A8%A1%E7%B5%84_(%E7%A8%8B%E5%BC%8F%E8%A8%AD%E8%A8%88))（[軟體設計](https://zh.wikipedia.org/wiki/%E8%BD%AF%E4%BB%B6%E8%AE%BE%E8%AE%A1)的最小單位）來進行正確性檢驗的測試工作。程式單元是應用的最小可測試部件。在程序化編程中，一個單元就是單個程式、函式、過程等；對於物件導向程式設計，最小單元就是方法，包括基礎類別（超類）、抽象類、或者衍生類別（子類）中的方法。

以下為單元測試範例，藍框區域的程式是要接受單元測試的程式，此程式的目的為找出最小的數字，而紅框部分就是進行單元測試的地方(此assert函式是由Python第三方函式庫"Pytest"所提供)，assert函式的第一個參數是預期的執行結果值，第二個參數是要進行測試的程式以及測試用輸入、函數等等，以圖2.1來看，此單元測試就是在確認當minima函數輸入(10,20)時，是否會正確地回傳10這個預期的結果，如果回傳值與預期值相同表示此受測程式在此輸入的情況下是正確的，則Pytest就會回傳"pass"表示通過單元測試，反之則表示此受測程式可能有邏輯上的錯誤，那Pytest就會回傳" failed "表示失敗。

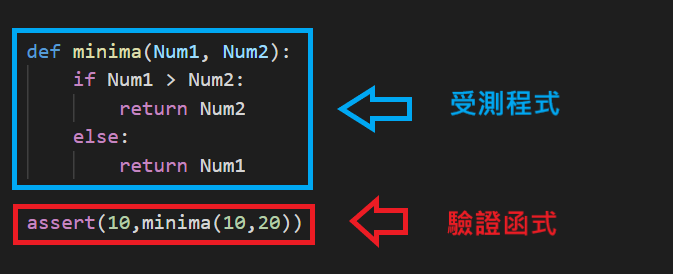


圖2.1 單元測試說明圖

### 變異測試

變異測試（Mutation Testing）是一種在細節方面改進程序源代碼的軟件測試方法。以單元測試為例，以下詳述變異測試的步驟與原理:

1. 將原有的運算子換成其他同類型的運算子(">" => ">=" , ">" => "<="...等等，如圖2.3)，基於此原則產生數個變異程式(此處後通稱P')。

而變更運算子後的P'的程式邏輯會變得和以原始的程式邏輯不同，所以兩者在輸入同樣的測試資料時獲得的結果也理當會不相同，因此當兩者的結果不同時就會稱此P'被殺掉了(Killed)，而結果相同時則稱此P'存活(Survive)。

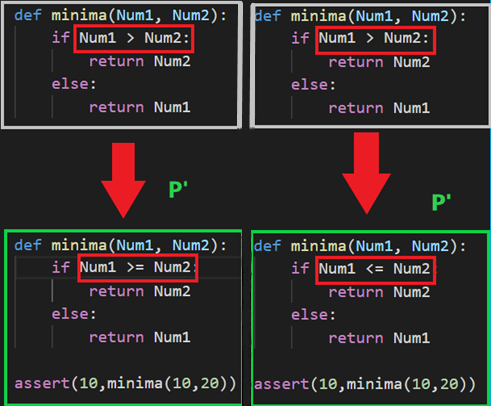


圖2.2 變異測試說明圖1

1. 以和單元測試相同的輸入執行這些P'並與原先的單元測試結果進行比對，若是結果與先前的單元測試結果不同則會稱此P'被殺掉(Kill，圖2.3)，若有相同結果的則稱存活(Survive，圖2.4)。

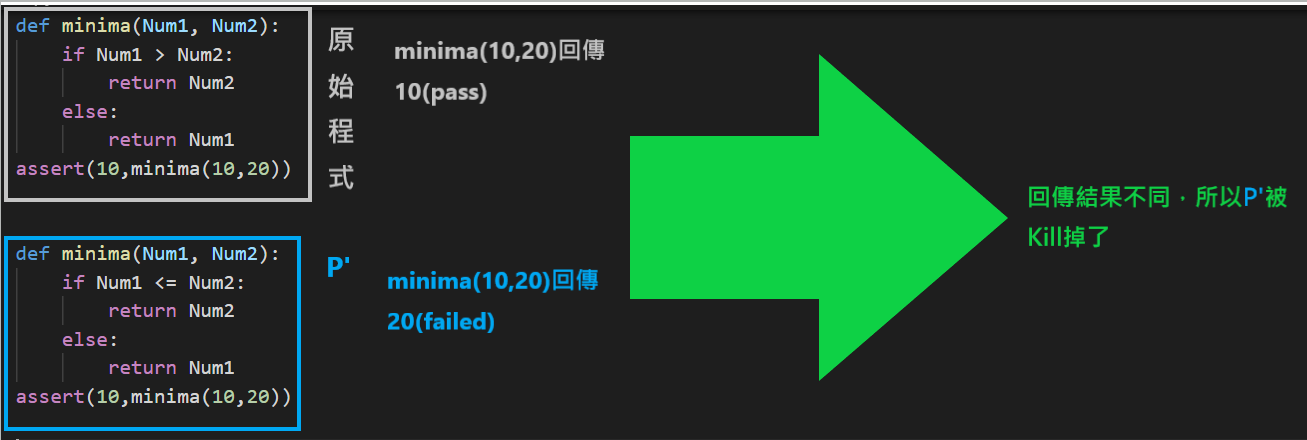


圖2.3 變異測試說明圖2

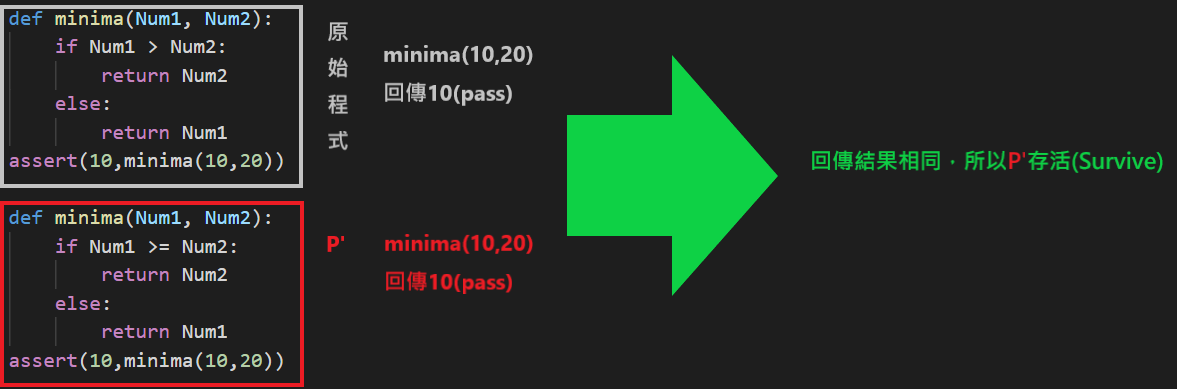


圖2.4 變異測試說明圖3

※測試資料通常不會只有一筆而是多筆的測試資料(如圖2.5)，而只要有其中一筆測試資料的回傳值與原先的單元測試的回傳值不同，就能通過變異測試。

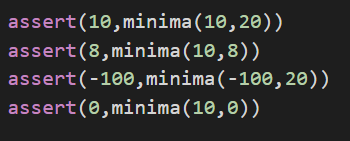


圖2.5 變異測試說明圖4

## 使用工具

### Django

基於 Python 寫成的免費而且開放原始碼的 Web 應用程式框架，可以快速開發安全和可維護的網站。有活躍繁榮的社區、豐富的文檔、以及很多免費和付費的解決方案。可以（並已經）用於構建幾乎任何類型的網站—從內容管理系統和維基，到社交網絡和新聞網站。它可以與任何客戶端框架一起工作，並且可以提供幾乎任何格式（包括 HTML、RSS、JSON、XML等）的內容。

### Python

Python是一種易學、功能強大的程式語言。它有高效能的高階資料結構，也有簡單但有效的方法去實現物件導向程式設計。Python 優雅的語法和動態型別，結合其直譯特性，並且其本身擁有一個巨大而廣泛的函式庫，除了內建的庫外，Python還有大量的第三方函式庫，使它成為眾多領域和大多數平臺上，撰寫腳本和快速開發應用程式的理想語言。

### subprocess

Python的內建函式，執行Python程式時，可以引用此函式庫內的run函式以建立一個新的process並執行指定的指令，並且能用特定的參數操控執行時資料傳遞的走向，比如說設定"capture\_output=True"就能將以run函式建立之process執行並print至shell的結果回傳至一開始執行run函式的Python程式。。

### Pytest

Pytest是Python 的一種單元測試框架，與Python自帶的Unittest測試框架類似，但是比Unittest 框架使用起來更簡潔，效率更高，方便的編寫測試用例。適合從簡單的單元到複雜的功能測試，執行測試過程中能將某些測試跳過，或者是對某些預期失敗的case進行標記，更支援執行由nose、unittest編寫成的測試case。

# 研究架構

本研究為遊戲系統(此後通稱MuPy系統)的開發與建置，將利用Django、HTML，CSS，JavaScript、Python等程式語言開發MuPy系統。在本章節中主要分為兩個部分。第一部分為使用者與MuPy系統之互動關係，第二部分將說明系統研究架構。

## 使用者與MuPy系統之互動關係

圖3.1為互動關係圖，在此系統中，使用者只需要輸入測試資料，MuPy系統會根據使用者輸入自動產生相對應的驗證函式，並自動執行變異測式，最後輸出執行結果供使用者查看。

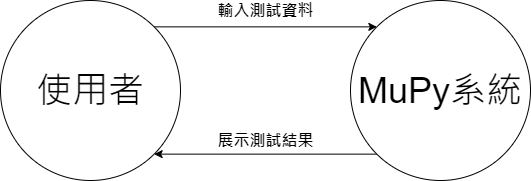


圖3.1 使用者與MuPy系統互動關係圖

## 系統架構

圖3.2為系統架構圖，主要為用戶端及應用服務端兩部分，以下兩部分將在另以文字做詳細說明。

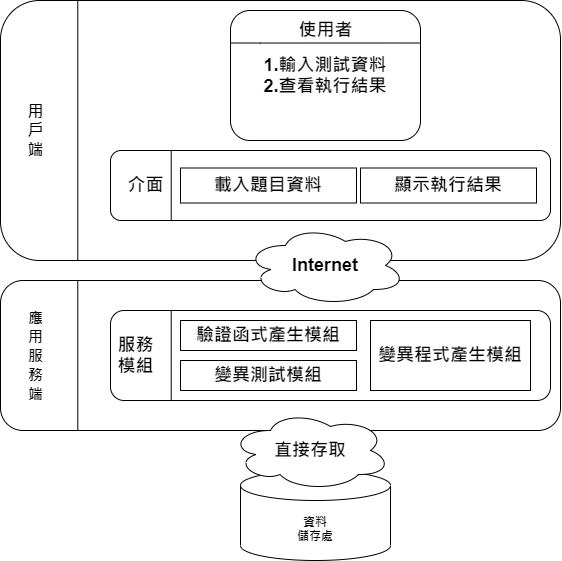


圖3.2系統架構圖

### 用戶端

使用者：進入使用者介面後，輸入測試資料，經由使用者介面傳輸資料，待應用服務端執行變異測試完成後再透過使用者介面查看執行結果。

### 應用服務端

介面：本系統之使用者介面主要功能有三個

1. **載入題目資料(原始及經過變異的程式碼)：**使用者一進入使用者介面後，使用者介面後會先傳送請求至應用服務端要求給予題目資料，等收到題目資料的回應後再顯示至使用者介面。
2. **顯示執行結果：**應用服務端執行完變異測後會傳送一包含結果的回應至使用者介面，使用者介面收到此回應後就會將結果輸出至介面，以供使用者查看。
3. **讓使用者輸入測試資料：**使用者輸入測試資料後，資料會由使用者介面發送請求傳送至應用服務端進行變異測式。

服務模組：服務模組包含四個主要功能

1. **驗證函式產生模組：**此模組的功能為產生驗證函式。當使用者透過使用者介面傳送測試資料到應用服務模組時，以此測試資料執行原始未經變異的程式碼並記錄其輸出結果，再以此結果產生驗證函式。
2. **變異程式產生模組：**此模組主要功能有兩個
   * + 1. 第一個是根據使用者進入的介面產生難度不同的變異程式碼，並傳送到使用者介面供使用者瀏覽。
       2. 第二個功能是，當變異測試模組要進行變異測式時，提供需要的變異程式碼供變異測試模組使用。
3. **變異測試模組：**此模組的功能為執行變異測式。此模組會以驗證函式產生模組所產生的驗證函式與變異程式產生模組產生的變異程式碼進行變異測試並在執行結束後分析執行結果，記錄每個變異程式的測試結果，並計算Kill rate。

# 系統實作

## 執行環境

本研究使用Django3.0.8開MuPy系統前後端，Python3.8.0開發變異測式相關模組並以開源套件Pytest5.4.3輔助開發，MuPy系統前後端之資料傳遞則是透過Ajax技術及Django內建方法進行。

## 介面實作

圖4.1為題目選擇頁面，一進到此頁面可以看到有三個Level可以選擇，分別是3個、6個、9個P'，逐漸提升完全kill掉所有P'的難度。

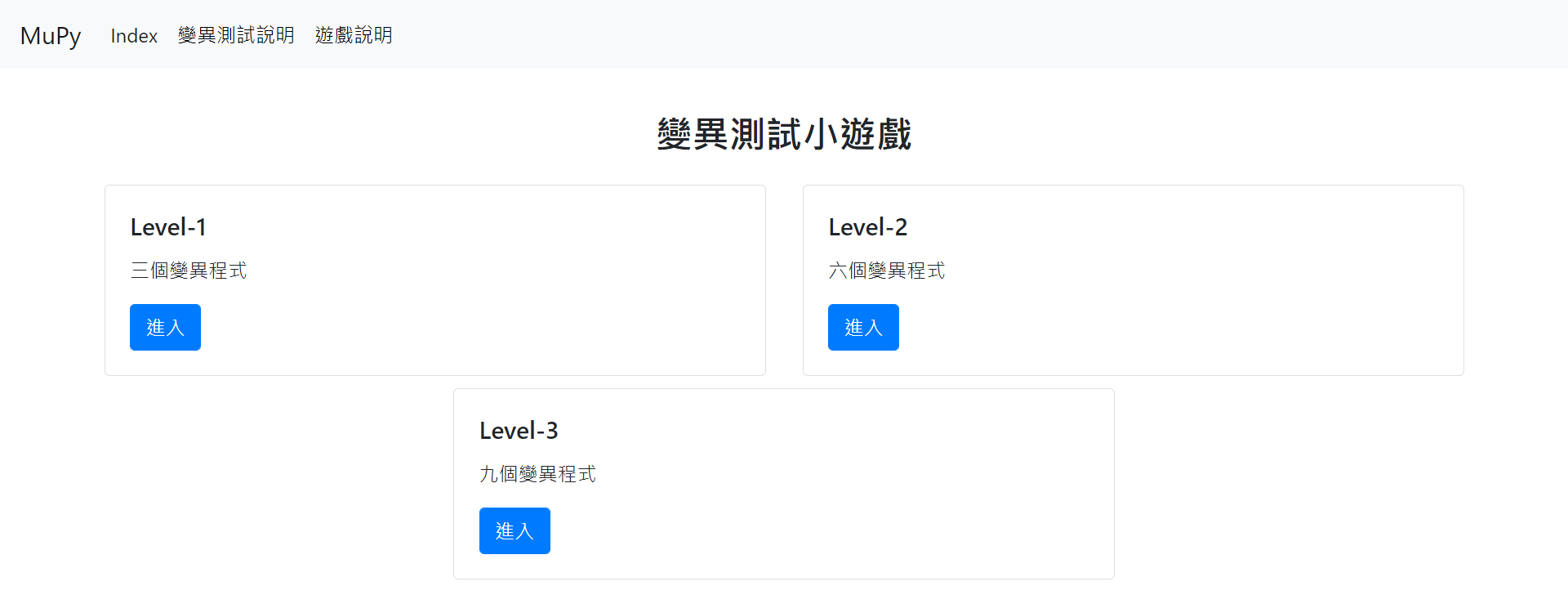


圖4.1 題目選擇頁面

在圖4.1的上方可以看到有一條Navbar，點選"index"會回到圖4.1的頁面，點選"變異測試說明"及"遊戲說明"，可以查看相關說明，如圖4.3、4.4所示。

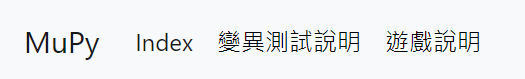


圖4.2 Navbar



圖4.3 變異測試部分說明圖

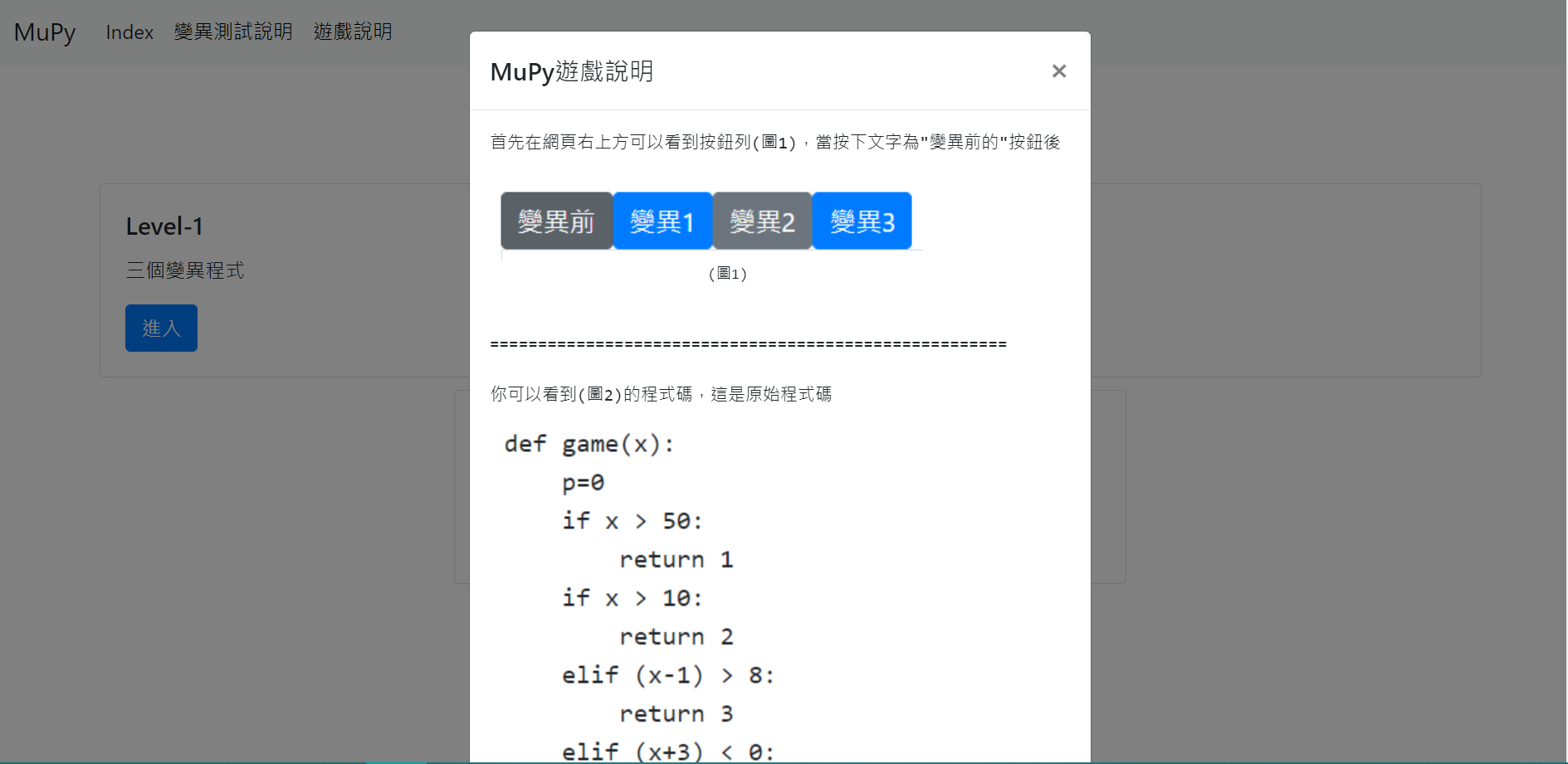


圖4.4 遊戲系統部分說明圖

圖4.5到圖4.8為Level-1頁面(與其他兩個Level差別只有P'的數量)，圖4.5為題目畫面，可以看到在這個區塊可以檢視原始的程式碼，並透過Button變更以查看其他的P'程式碼。

圖4.6為測試資料輸入畫面，一個欄位就代表一筆輸入，可以根據需求使用Add column Button來新增測試資料的欄位數量 (如圖4.7)，不小心新增了多餘的欄位也沒關係，傳送資料到服務端時會過濾掉非法的輸入，再執行變異測式。

圖4.8是執行結果畫面，使用者介面接收到來自服務端的執行結果後，會使用JavaScript將結果顯示至此部分，可以看到kill rate以及每個P'是killed還是Survived。

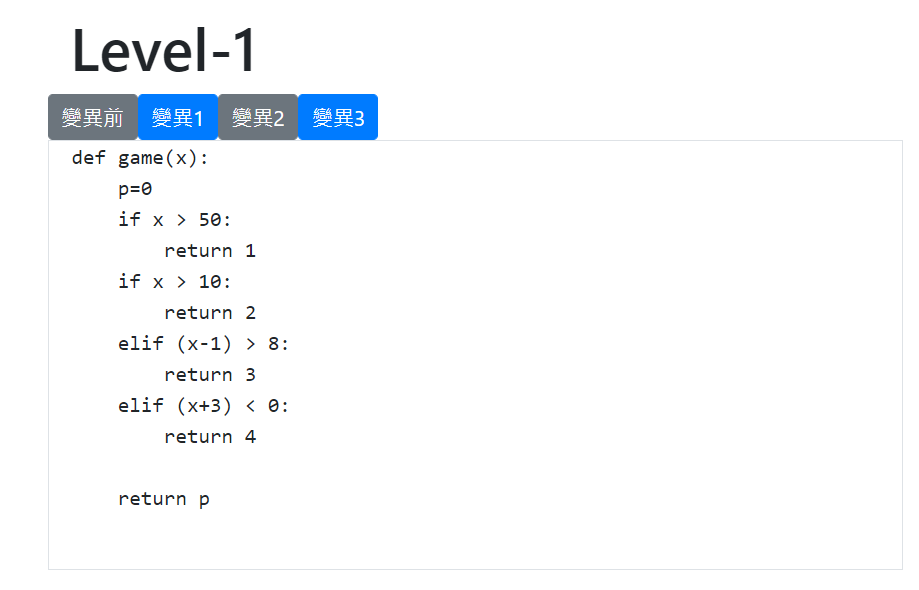


圖4.5 題目畫面

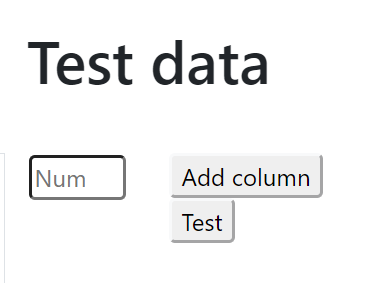


圖4.6 輸入畫面

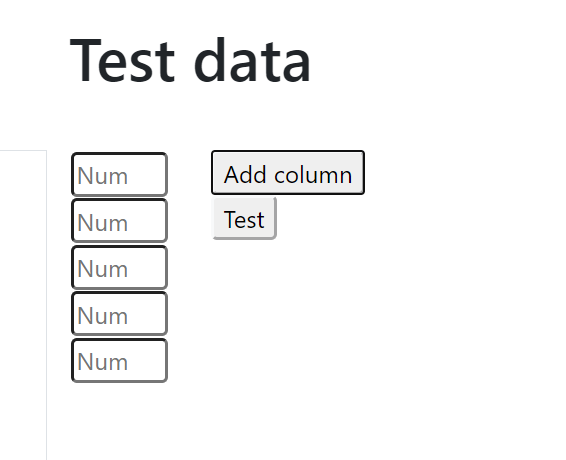


圖4.7 新增欄位後的輸入畫面

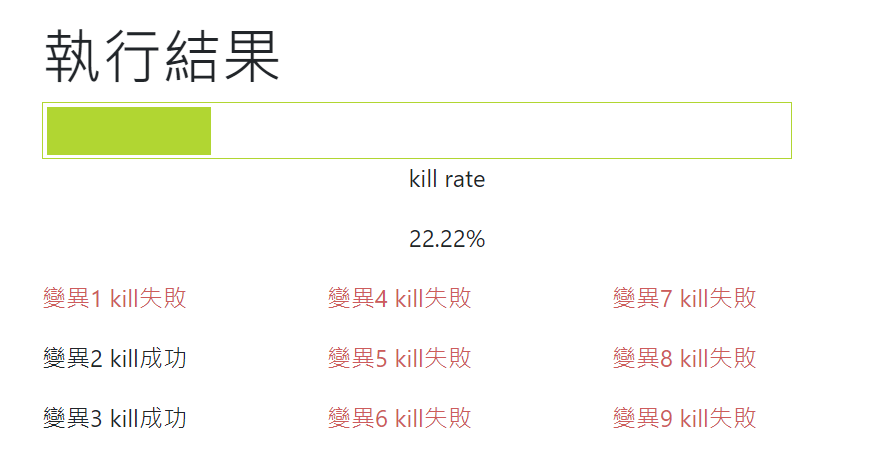


圖4.8 執行結果畫面

## 模組實作

1. **變異測試模組：**

圖4.9為變異程式及驗證函式產生完成後會執行的程式碼區塊，此區域執行內容為利用Python內建函式subprocess來執行Pytest，對每個變異程式進行單元測試，然後再紀錄執行結果(圖4.10)。

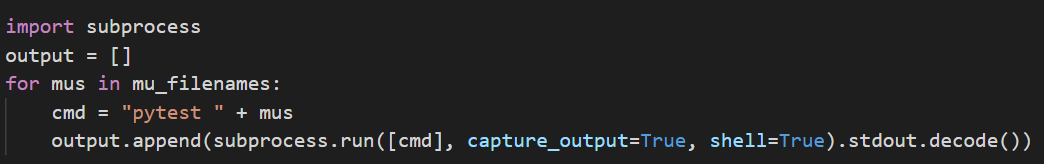


圖4.9 變異測試模組部分程式碼

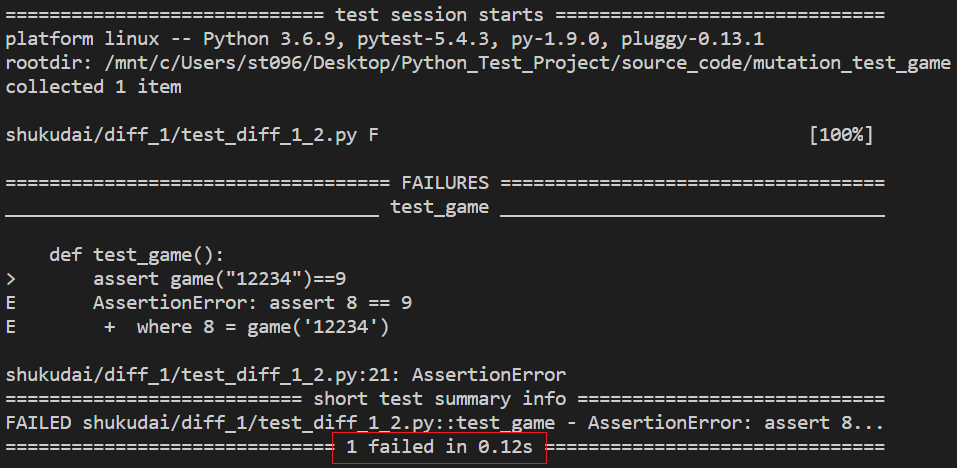


圖4. 10 Pytest 輸出

圖4.11是處理執行結果的程式碼區塊，此處會對執行結果字串(圖4.10)進分割處理，然後記錄每個變異程式的測試結果是failed(Kill)還是pass(Suvive)及計算Kill rate，最後再以JSON的格式傳送至使用者介面。

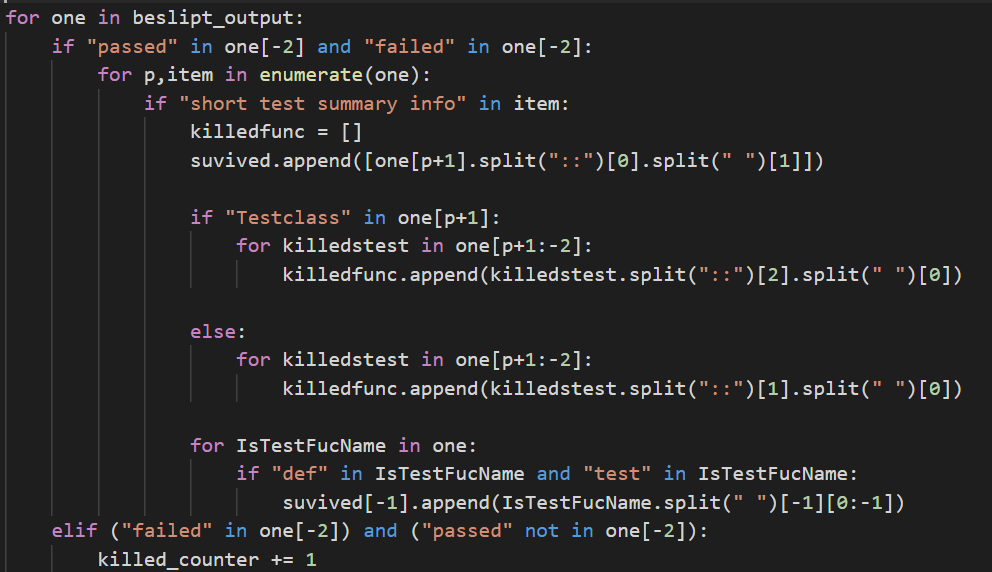


圖4.11 變異測試模組部分程式碼

1. **變異程式產生模組：**

此模組會將要進行變異的程式碼讀入，而變異的順序會依Symbol分組依序進行(表4.1)，每一種的Symbol type會變異出多少變異程式與此Symbol type的Symbol數量有關，以Priority最高的Relation symbols為例，若程式碼內包含">"這個Symbol，此模組會先找出所有">"的索引位置，然後以同Symbol type內其他不同的Symbol和此索引位置的">"進行替換並產生新檔案(圖4.12、圖4.13)。

|  |  |  |
| --- | --- | --- |
| Priority | Symbol type | Symbols |
| 1 | Relation symbols | (< , > ,<= ,>= ,== ,!=) |
| 2 | Operations symbols | (+ , - , \* , /) |
| 3 | Logic symbols | (and , or) |
| 4 | Binary symbols | (& , | , ^ , >> , <<) |

表4.1

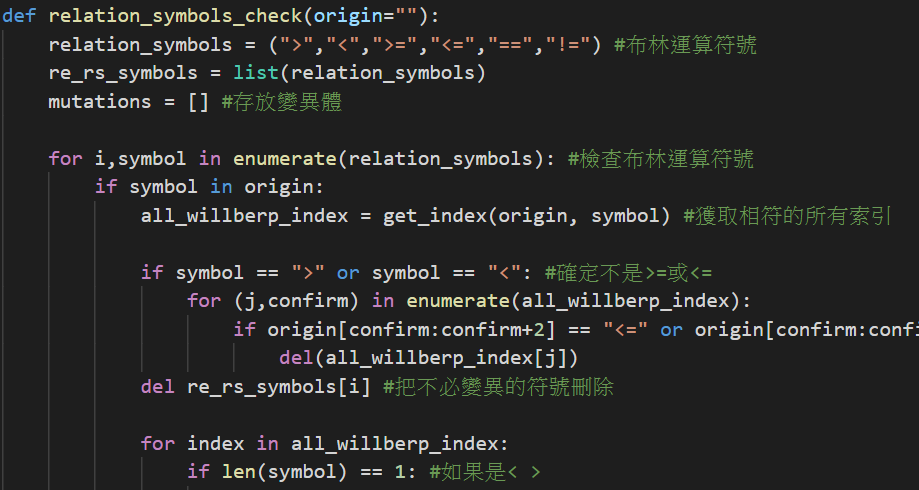


圖4.12

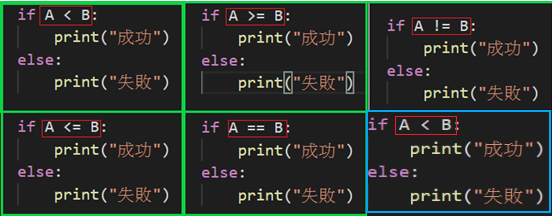


圖4.13

1. **驗證函式產生模組**

當使用者輸入的資料透過使用者介面傳送過來後，會先用for迴圈把所有輸入檢查過一輪，過濾掉非數字的輸入。

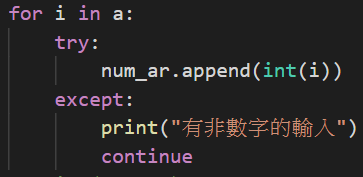


圖4.14 非數字過濾程式碼

呼叫此模組開始產生驗證函式，以經過濾的輸入對原始程式碼個別進行一次單元測試(圖4.15)，並以List記錄其輸出，最後再以此List去組成驗證函式的程式碼(圖4.16)。

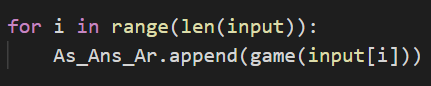


圖4.15 單元測試結果紀錄

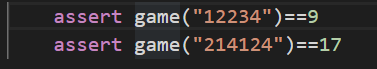


圖4.16 驗證函式範例

## 題目設計

從4.3模組設計可以看出，大部分的P'其實都非常容易被Kill掉，以">"這個符號為例子，除了變成">="之外，變成其他符號的時候通常都能輕易的影響輸出結果這也就代表這個P'被kill的機率非常的大，因此在設計題目載入的P'時，Level-1會以載入容易Kill的P'為主，Level-2& Level-3則以新增不容易被Kill的P'為主。

1. **Level-1：**以較容易被Kill掉的P'組成，共有三個P'，判斷參數x的數值，改變第一條if內的Relation symbols，以較容易Kill掉的Relation symbols替換。



圖4.17 Level-1題目

1. **Level-2：**以Level-1的P'為基底再加上三個新的P'，但增加的是較不容易被Kill的項目。

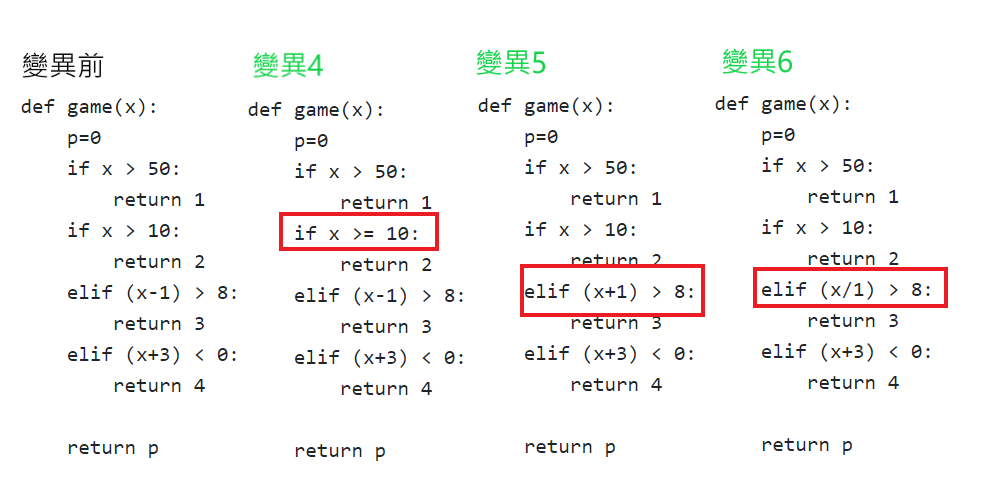


圖4.18 Level-2新增P'

1. **Level-3：**以Level-2的P'為基底，再加上三個不容易被Kill的新P'。

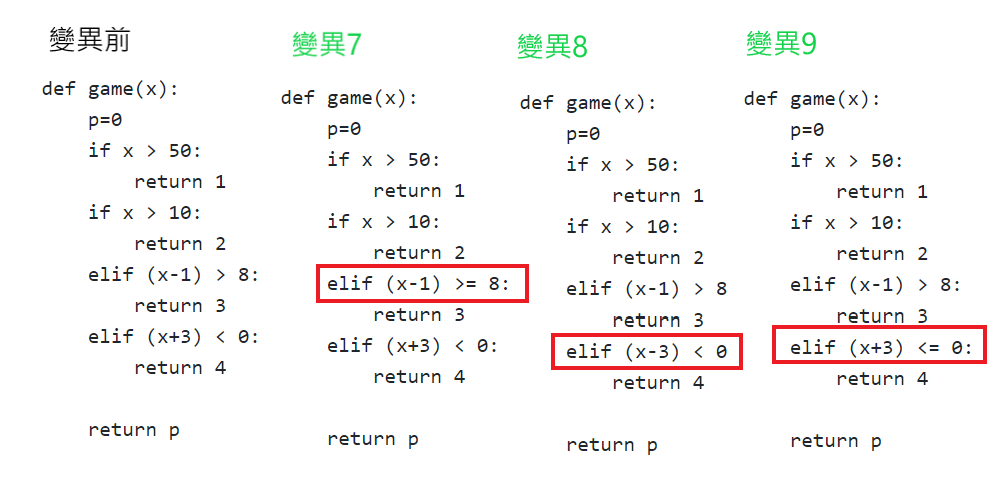


圖4.19 Level-3新增P'

## 實作結果

設計一回饋問卷，內含五個問題，並尋找三位測試者實際測試本系統，並在測試後填寫回饋問卷，提供使用心得及改善建議，以下為回饋問卷之題目及結果。

1. **問卷題目：**詢問內容包含，"遊戲說明是否完整"、"MuPy是否有幫助到您理解變異測試"、"MuPy的頁面設計"、"MuPy的題目設計"、"是否有任何其他的意見?"，共四個選擇題，一個開放式回答問題。

****

圖4.20問卷題目圖

1. **問卷結果：**根據問卷的結果可以看出測試者對於本次問卷題目都有偏向正面的回答，因此可以認為本系統確實能做到"讓使用者更加了解變異測試"這件事情，但還是有許多可以改善的地方，比如說可以把遊戲及變異測試說明做得更加直觀，幫助使用者能夠更快進入狀況。

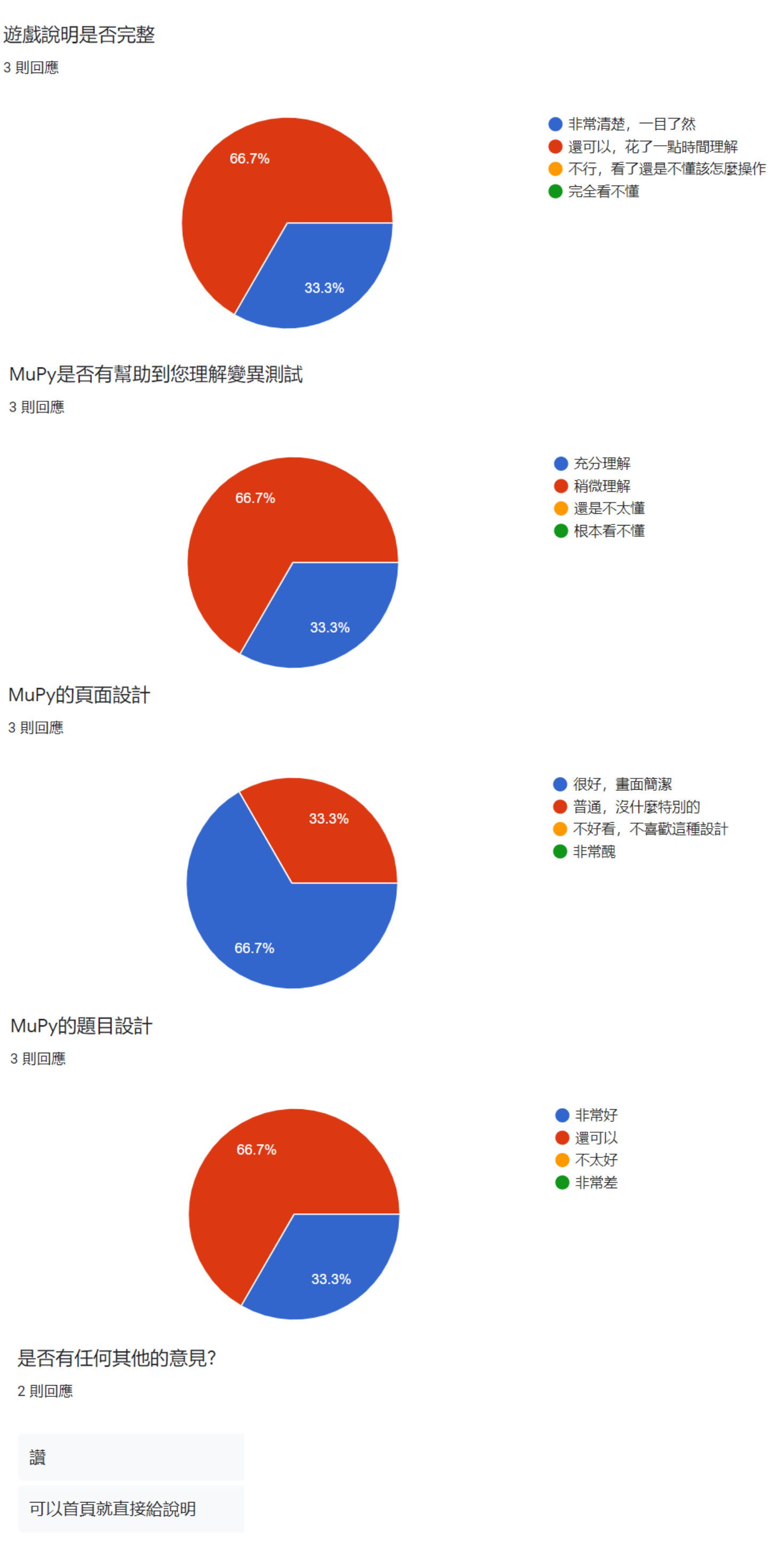


圖4.21問卷結果圖

# 結論與建議

使用者遊玩MuPy透過觀察每一個Level的P'，想像可能的執行結果，然後輸入測試資料，並透過執行結果檢視自身的邏輯是否正確，我想這確實能有效的提升自身對變異測試理解程度，但其實MuPy對程式進行變異的種類還是稍嫌過少，還有其他更多能夠進行變異的地方，像是直接對數字做更改(+1、-1等等)，然後是以我自己設計變異方法來看，大部分的P'其實都非常容易被kill掉，以">"這個符號為例子，除了變成">="之外，變成其他符號的時候通常都能輕易的影響輸出結果這也就代表這個P'被kill的機率非常的大，也因此要設計出一個有難度遊戲關卡變得不太容易，使用者介面也有可以加以改善的部分，像是加入更多的動畫特效，讓使用者更有在玩遊戲的感受，可以更放鬆的學習。

因此我想未來也許可以再將變異的方式做修改，減少容易被kill的部分，增加不容易kill的部分，並增加更多的變異種類，並嘗試提高使用者介面的互動性。

# 參考文獻

1. <https://pitest.org/quickstart/mutators/>
2. <https://stackoverflow.com/questions/9935151/popen-error-errno-2-no-such-file-or-directory>
3. <https://sites.google.com/site/coolteazhuanti/css/padding>
4. <https://andy6804tw.github.io/2018/01/06/bootstrap-tutorial(1)/#%E4%BD%BF%E7%94%A8-container>
5. <https://stackoverflow.com/questions/34450586/justify-buttons-with-bootstrap-v4>
6. <http://www.eion.com.tw/Blogger/?Pid=1158>
7. <https://zh-tw.learnlayout.com/position.html>
8. <https://dotblogs.com.tw/caubekimo/2018/09/17/145733>
9. <https://pjchender.blogspot.com/2015/03/cssmultiple-selectorsspace.html>
10. <https://code.ziqiangxuetang.com/try/tryajax_first/>
11. <https://www.w3schools.com/jsref/dom_obj_paragraph.asp>
12. <https://www.runoob.com/jsref/met-win-clearinterval.html>
13. <https://www.itread01.com/content/1543975213.html>