

國立中山大學
National Sun Yat-sen University

資訊工程學系
Department of Computer Science and Engineering

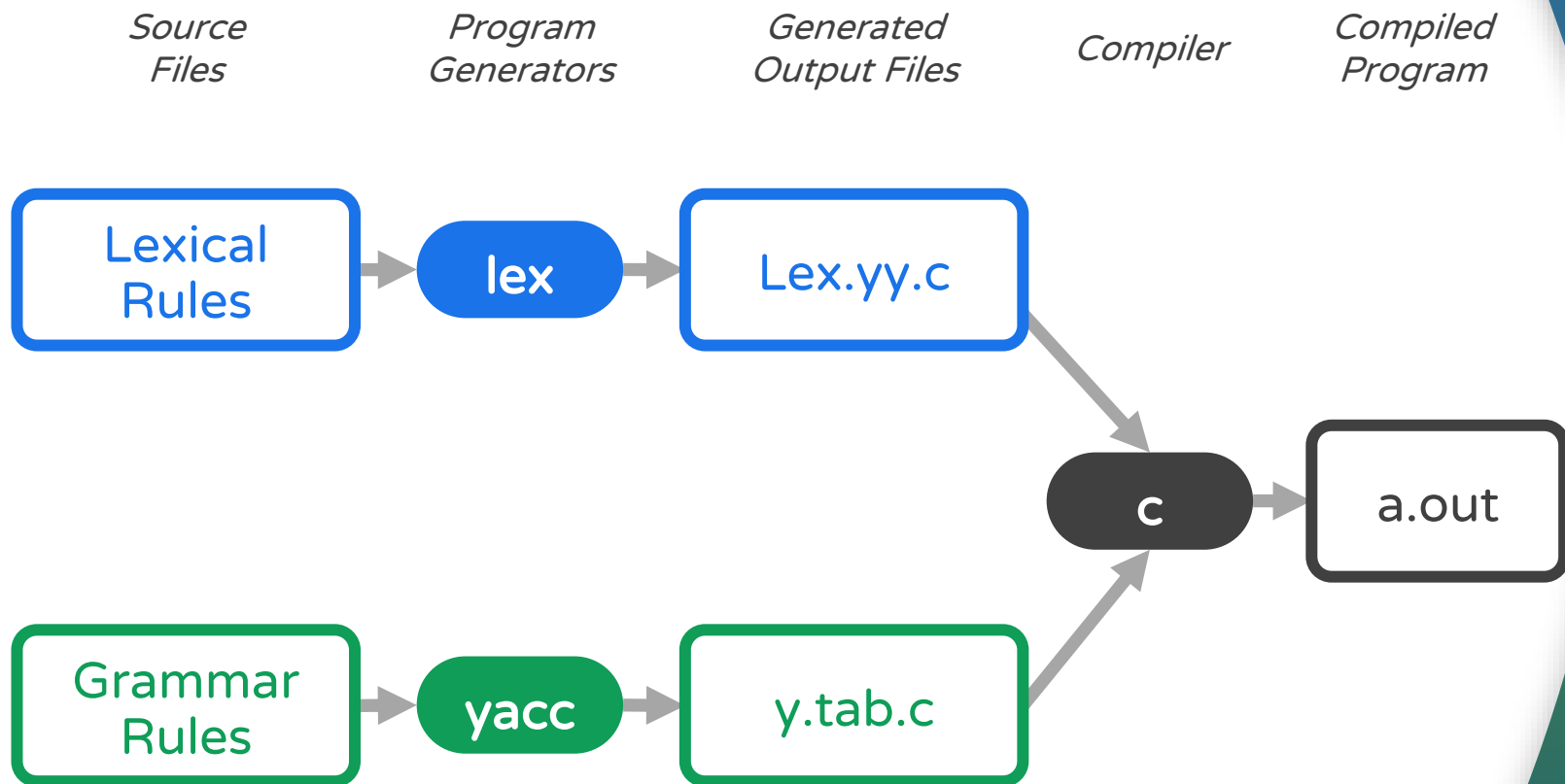
111 學年度
編譯器製作

編譯器製作

Lex Tutorial

助教:鄭子瀚

Compiler 的分工



Lex 的工作

運作

Lex 會把 input 當作 a sequence of characters

- 一個以上連續的 character 會形成一個 token

目的

Lex 的目的是檢查 token 是否合法

- 例如不合法的變數名稱 (identifier)

條件

Lex 必須事先定義規則


- Regular Expression
 - ▣ 可以被辨識的 token



Lex 的 Input

以 Pascal 為例

```
program test;  
var  
    3i : string;  
begin  
    3i := 'ab; //invalid  
end;
```



Lex 的格式

分成三部分，每個部分以 %% 區隔開來

Definition

%%

Lex Rules

%%

User code



Definition

demo.l

```
%{  
#include <stdio.h>  
unsigned charCount=1, idCount=0, lineCount=1;  
%}  
operator [\+\-\*\//]  
space [ \t]  
eol \n  
  
/* You should write your own regular expression. */  
reserved_word  
symbol  
id  
  
%%
```

Rules

demo.l

```
%%
```

```
{operator} {  
    printf("Line: %d, 1st char: %d, \"%s\" is an  
        \"operator\".\n", lineCount, charCount, yytext);  
    charCount += yyleng;  
}  
{space} {  
    charCount++;  
}  
{eol} {  
    lineCount++;  
    charCount = 1;  
}  
{reserved_word} {  
    /* You should write your own code */  
}
```

```
%%
```

Rules

- Scanner 所匹配規則的優先順序
 - 會 scan 出**長度最長**的 token 去進行匹配
 - 如果匹配長度一樣，
則看被定義的先後順序 (由上到下)




```
%%  
int main(){  
    yylex();  
    return 0;  
}
```

來看看 Test Cases

Test File

test1.pas

```
int main()
{
    yylex();
    printf("\n\nThe number of characters: %d\n", charCount);
    printf("The number of lines: %d\n", lineCount);
    return 0;
}

void print_char()
{
    printf("[In print_char]Symbol: '%s'\n", yytext);
    charCount++;
}
```

Output

```
wangyc@wangyc-ubuntu:~/Desktop/lex/testfile_lab1_2022
/example$ ./a.out < test.in
String : 'This'
String : 'is'
String : 'a'
String : 'Lex'
String : 'Example'
[In print_char]Symbol:','
'In print_char]Symbol:'
String : 'I'
String : 'love'
String : 'compiler'
[In print_char]Symbol:','
String : 'I'
String : 'love'
String : 'coding'
[In print_char]Symbol:','

The number of characters: 45
The number of lines: 1
```

Lex 的特殊字元

- 這些字元在regular expression中有特殊意義，
如果要當成一般字元，
請在前面加上\這一個跳脫字元(Escape character)
 - ? * + | () ^ \$. [] { } " \
- Digit [0-9]
- Letter [a-zA-Z]
- Operator [\+ \- *]



使用 Lex

如何使用 Lex File

- 我們的目的要將 demo.l 編譯成可以執行的 scanner
- 首先必須安裝 flex 這個程式來編譯我們的 lex file

```
sudo apt-get install flex
```

<- 以ubuntu為例

- 透過 flex 將 demo.l 編譯成 C source file ，
這個 C source file 就是我們的 scanner

```
flex demo.l
```

- C source file 預設檔名為 lex.yy.c ，
最後我們可以利用 gcc 將其編譯成可執行檔

```
gcc lex.yy.c -o demo -lfl
```

- 執行檔為 demo ， 假設我們要 scan 的檔案為 test1.pas

```
./demo < test1.pas
```

- 也可以直接執行 demo ， <Ctrl+D> 可以送出 EOF



Regular Expression

常用字元符號

.	任意 字元 (不含換行)
\d	任意 數字
\D	任意非 數字
\w	任意 文字、數字、底線
\W	任意非 文字、數字、底線
\s	任意 空白字元 (空白、定位、換行)
\S	任意非 空白字元 (空白、定位、換行)
\n	換行字元 (NewLine)
\t	定位字元 (Tab)
\r	回車字元 (Carriage return)
\0	Null 字元



特殊字元符號

\.	. 字元
\?	? 字元
*	* 字元
\+	+ 字元
\\	字元
\^	^ 字元
\\$	\$ 字元
\"	" 字元

\((字元
\)) 字元
\[[字元
\]] 字元
\{	{ 字元
\}	} 字元
\\	\ 字元
\/	/ 字元



常用列舉規則

規則	說明	合法舉例
[12abc]	A single character of 1, 2, a, b, c	1, 2, a, b, c
[^12abc]	A single character except 1, 2, a, b, c	3, 4, d, e, f
[0-9A-Z]	A character in the range of 0-9 or A-Z	0, 1, A, B, C
[ab] [0-9]	A single character of a, b, or in the range of 0-9	a, b, 0, 1, 2



常用次數符號 (加在尾巴)

符號	說明
*	重複 $0 \sim \infty$ 次
+	重複 $1 \sim \infty$ 次
?	重複 $0 \sim 1$ 次
{n}	重複 恰好 n 次
{n,}	重複 $n \sim \infty$ 次
{n,m}	重複 $n \sim m$ 次



Regular Expression

- 當然不止這一些
- 如果你對正規語言不熟悉，
網路上對有相當豐富的資源，利如
 - <https://www.vixual.net/blog/archives/211>
- Online Regular Expression Tester
 - <https://regex101.com>
 - <https://regexpr.com>



REGULAR EXPRESSION

1 match (143 steps, 0.1ms)

/ \/*(.|\n)**\//

/ gm



TEST STRING

/* OMG!
... 棒呆惹~
... 這是一個合法的註解喔
*/
~但是這個不是，好難過嗚嗚嗚~

EXPLANATION

▼ / \/*(.|\n)**\// / gm
\/ matches the character / with index 47₁₀
(2F₁₆ or 57₈) literally (case sensitive)
\/ matches the character * with index 42₁₀

MATCH INFORMATION

Match 1	0-32	/* OMG! ... 棒呆惹~ ... 這是一個合法的註解喔 */
---------	------	---

QUICK REFERENCE

Search reference

All Tokens

★ Common Tokens ✓

A single c... [abc]

A chara... [^abc]

A charact... [a-z]

A chara... [^a-z]

這裡有相當豐富的
Regular Expression Rules 可以參考喔



Untitled Pattern

Save (ctrl-s)

New



by gskinner

GitHub

Sign In



Expression

JavaScript

Flags



```
/\/\*(.|\n)*\*\/g
```

Text

Tests

NEW

1 match (0.1ms)

```
/* OMG!
... 棒呆惹~
→ 這是一個合法的註解喔
*/
--但這裡就不是
|
```

Tools

Replace

List

Details

Explain



關於作業I



推薦的環境

■ Cygwin

- 一個類 Linux 環境
- 直接在 Windows 上操作，不須虛擬機
- 記得安裝 flex 和 m4 這兩個 Packages

■ 在虛擬機上裝 Ubuntu

- Ubuntu 22.04



關於 Cygwin 的提示

- 請記得在安裝 Cygwin 時，
一併安裝以下這些 Packages

- tcsh
- ncurses
- dos2unix
- Emacs
- nano
- gcc-core
- gcc-g++
- make

- flex

- m4



這兩個不在 UNIX 課程的安裝教學裡
請特別留意



關於 Cygwin 的提示

- 如果你已經安裝好 Cygwin ，
但是有 Packages 沒有安裝到 ，
別擔心！**不用**重新安裝！
- 請到 Cygwin 官網下載安裝檔 (setup-x86_64.exe)
 - <https://www.cygwin.com/install.html>
- 執行安裝檔，下一步直到 “Select Packages”
- View 選 “Full”，右邊選擇 “Keep”
接下來透過 Search，搜尋並選取所有需要的內容
 - 可參考下頁



關於 Ubuntu 的提示

- 請安裝Ubuntu 22.04: <https://releases.ubuntu.com/focal/>，選擇 Desktop image
- 虛擬機推薦: <https://www.virtualbox.org/>
- 下載好後記得更新一些基本的東西:
 - `sudo apt update`
 - `sudo apt-get update`
 - `sudo apt install build-essential`
- 安裝好後，確認是否成功:
 - `gcc --version`



作業繳交注意事項

- **DUE DATE: 2023年04月07日 23:59**
- 程式 Demo 環境是 Cygwin / Ubuntu 22.04 LTS，因此請保證你們的程式碼能夠在至少其中一個編譯執行
- 請參考課程網頁中的測試檔案來驗證你的程式
- 助教會自行設計額外的隱藏測資，因此請保證你所寫的 Regular Expression 可以 match 到大部分的 cases
 - 例如一些複雜的變數名稱、浮點數必須要可以是負數...
- 請準時繳交作業，作業遲交一天打七折
- 請把作業包成一個壓縮包，上傳至網路大學，檔名命為「學號_hw1.zip」。
學號輸錯，作業分數-10；沒輸學號，作業分數-50。
- 在繳交截止後，會安排時間 Demo (4/8~4/12)，請準時到 EC5023 資料庫系統實驗室 找助教 Demo。



常見問答 與提示



題目檔範例輸出更新 (Page5)

```
// print hello world
{
    print("hello world");
    int a = 5 + 5.5;
}
```

你的scanner 將會輸出下列的結果：

```
Line: 1, 1st char: 1, "// print hello world" is a "comment".
Line: 2, 1st char: 1, "{" is a "symbol".
Line: 3, 1st char: 3, "print" is a "reserved word".
Line: 3, 1st char: 8, "(" is a "symbol".
Line: 3, 1st char: 10, "hello world" is a "string".
Line: 3, 1st char: 22, ")" is a "symbol".
Line: 3, 1st char: 23, ";" is a "symbol".
Line: 4, 1st char: 3, "int" is a "reserved word".
Line: 4, 1st char: 7, "a" is an "ID".
Line: 4, 1st char: 9, "=" is a "operator".
Line: 4, 1st char: 11, "5" is an "integer".
Line: 4, 1st char: 13, "+" is an "operator".
Line: 4, 1st char: 15, "5.5" is a "float".
Line: 4, 1st char: 18, ";" is a "symbol".
Line: 5, 1st char: 1, "}" is a "symbol".
The symbol table contains:
a
```



Error Handling

- 遇到錯誤該做什麼！？
- 只要可以讓程式遇到錯誤時，還能繼續執行下去，不致中止，就可以囉。
- 而如果你的錯誤處理，是包含輸出錯誤相關資訊，甚至能加以分類的話，當然就會拿到比較高的分數喔！



可以被識別的資料型態

- 合法的 Pascal 資料型態，都應該要被識別
 - 畢竟，這是一個 Pascal 的 Scanner
 - 可以自己到 Online pascal Compiler 試試看
https://www.onlinegdb.com/online_pascal_compiler
 - 反之，Pascal 中 不合法的宣型態，則都該被視為錯誤
(如 字串錯誤、數字溢位、數字開頭的變數名稱、...)

請善用 User Code

- 以 整數 為例
 - 0x14 (Hexadecimal)
 - 024 (Octal)



可以被識別的資料型態

- 以 浮點數 為例
- 分為 float 和 double 兩種浮點數型態

double .2 , 2. , 2.0

float .2f, 2.f, 2.0f

- 本次作業允許 float 和 double 都納入 {float}



Return Value of User Code

- 既然是放在 “User Code” 裡面的 function，顧名思義，就是你想要有一些額外的功能，在利用 Lex Rules 掃描檔案的時候，呼叫他來達成某些特定目的。
- 所以在本次作業，你的程式需要 Return 什麼值，就 Return 他吧。
 - Yacc Parser 會需要有特定的回傳值，這部分下次 Lab 作業會再講解



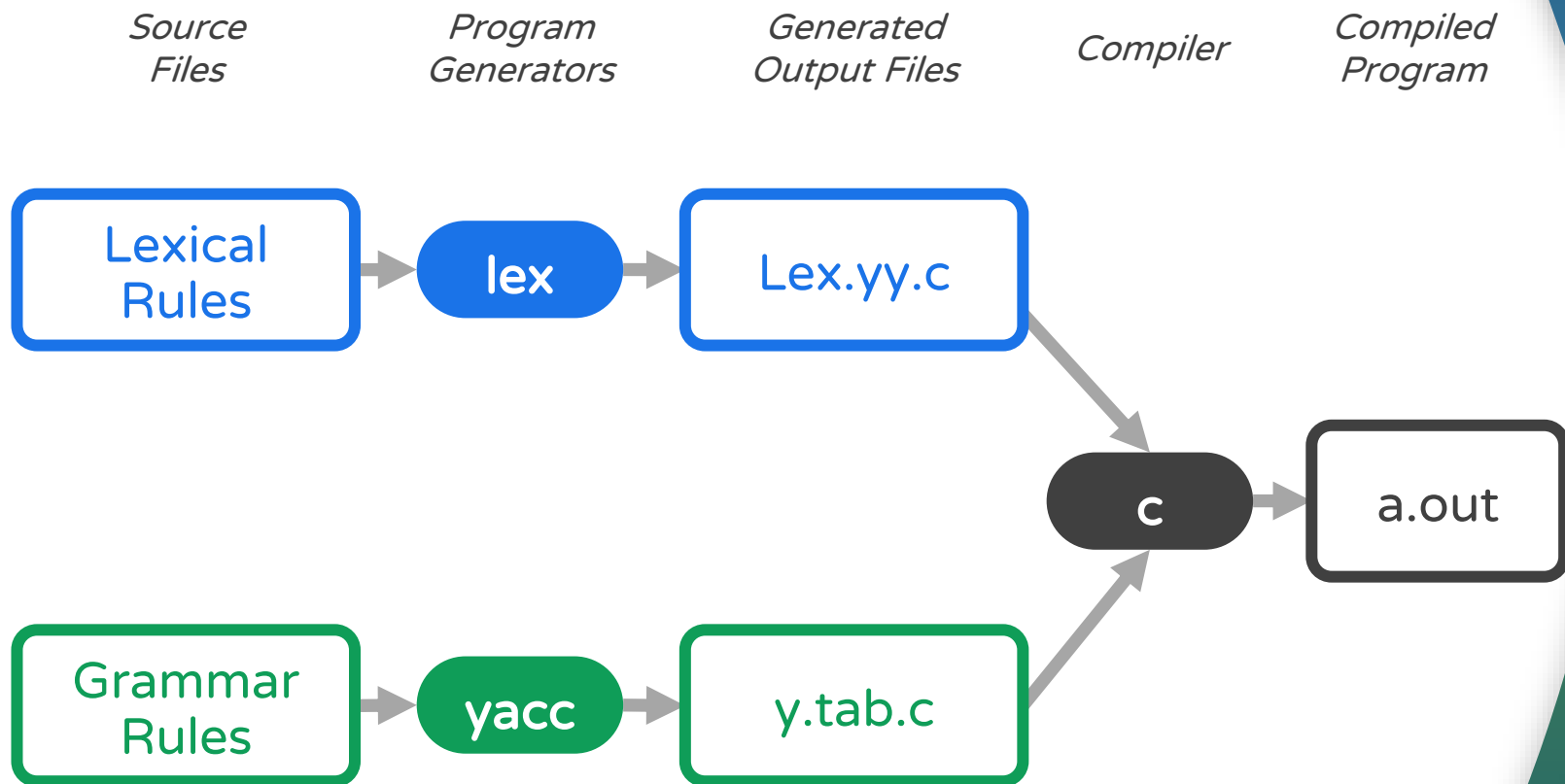
這個文法錯了，要報錯嗎

- 請記得你現在實作的是
檢查 Lexical Rules 的 Yacc Scanner，
並不是檢查 Grammar Rules 的 Yacc Parser。
- 所以，如果他是個合法的 Token，就請印出他，
不需要報錯。
- 例如以下 程式雖文法有誤，無法被編譯，
但因語彙無誤，因此仍可被正常辨識輸出。

```
int x = 2 + 0.3f ;
```



Compiler 的分工



Symbol Table

- 請不要重複存入 Symbol ID !!
 - Symbol Table 只要存入相異的 ID 即可，
這樣才符合 Symbol Table 的目的與精神喔！
 - 請善用 lookup function !!
- 按照讀入順序，依序輸出即可
 - 要使用其他順序排列也可以，
但請在PDF報告書裡說明



這個邏輯要被識別嗎

- 作業說明裡面列舉的定義，
都是 Minimum Requirements。
- 請記得在報告 PDF 檔裡面，以及 Demo 的時候讓我知道，才能給你較高的分數。



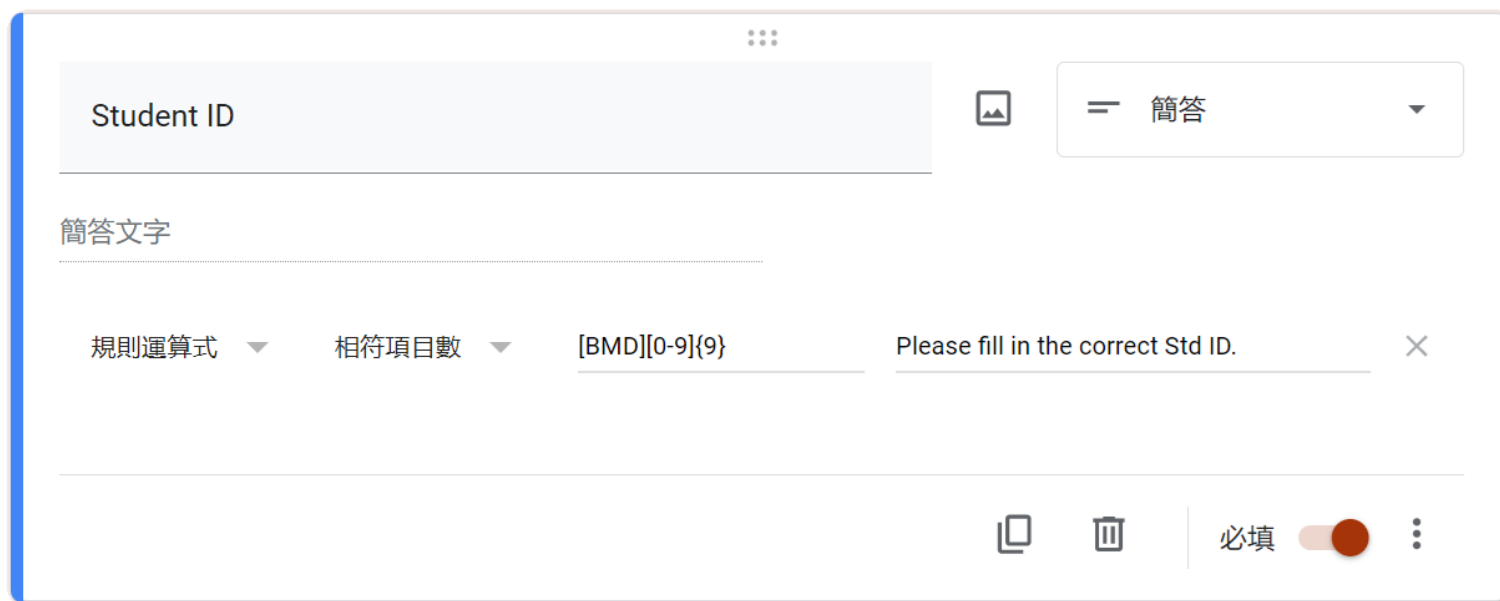
正負號還是加減號

- 如果 $+/-$ 左側**不是**數字的話，
那他就代表**正負記號**，對嗎？
- 如果 $+/-$ 左側**是**數字的話，
那他就代表**運算子**，對嗎？
- 想想看，也許可以善用變數來實作喔
- 這只是提供一個可能的想法，
還請自己多加思考是否符合所有邏輯



正規語言的實際應用

■ Google Form 回應驗證



The screenshot shows a Google Form question titled "Student ID". The question type is set to "簡答" (Short answer). Below the title, there is a text input field. Underneath the input field, the validation settings are displayed: "規則運算式" (Regex) with a dropdown arrow, "相符項目數" (Number of matches) with a dropdown arrow, and the regex pattern "[BMD][0-9]{9}" in a text box. To the right of the text box, the validation message "Please fill in the correct Std ID." is shown with a close button (X). At the bottom right of the question card, there are icons for copying and deleting the question, a "必填" (Required) toggle switch which is currently turned on (indicated by a red dot), and a vertical ellipsis menu icon.

Student ID

簡答

簡答文字

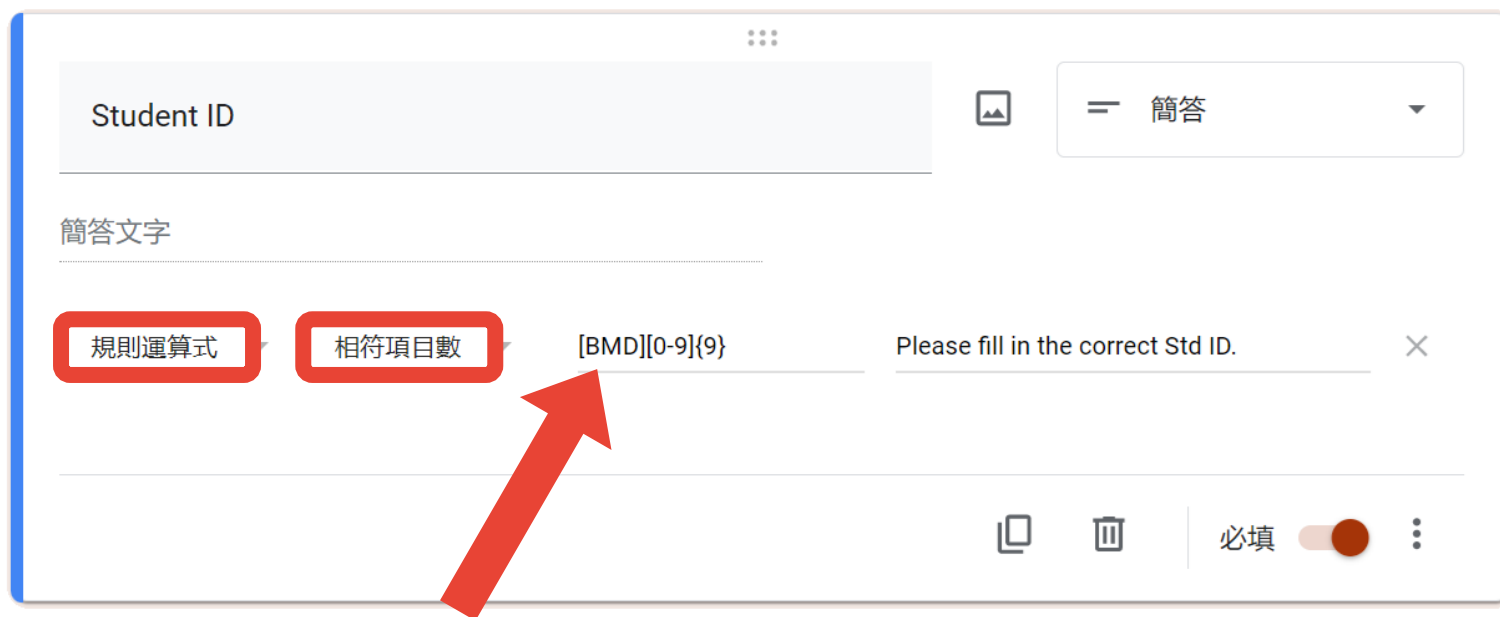
規則運算式 ▼ 相符項目數 ▼ [BMD][0-9]{9} Please fill in the correct Std ID. X

必備 必填 ☒ ⋮



正規語言的實際應用

■ Google Form 回應驗證



Student ID

簡答

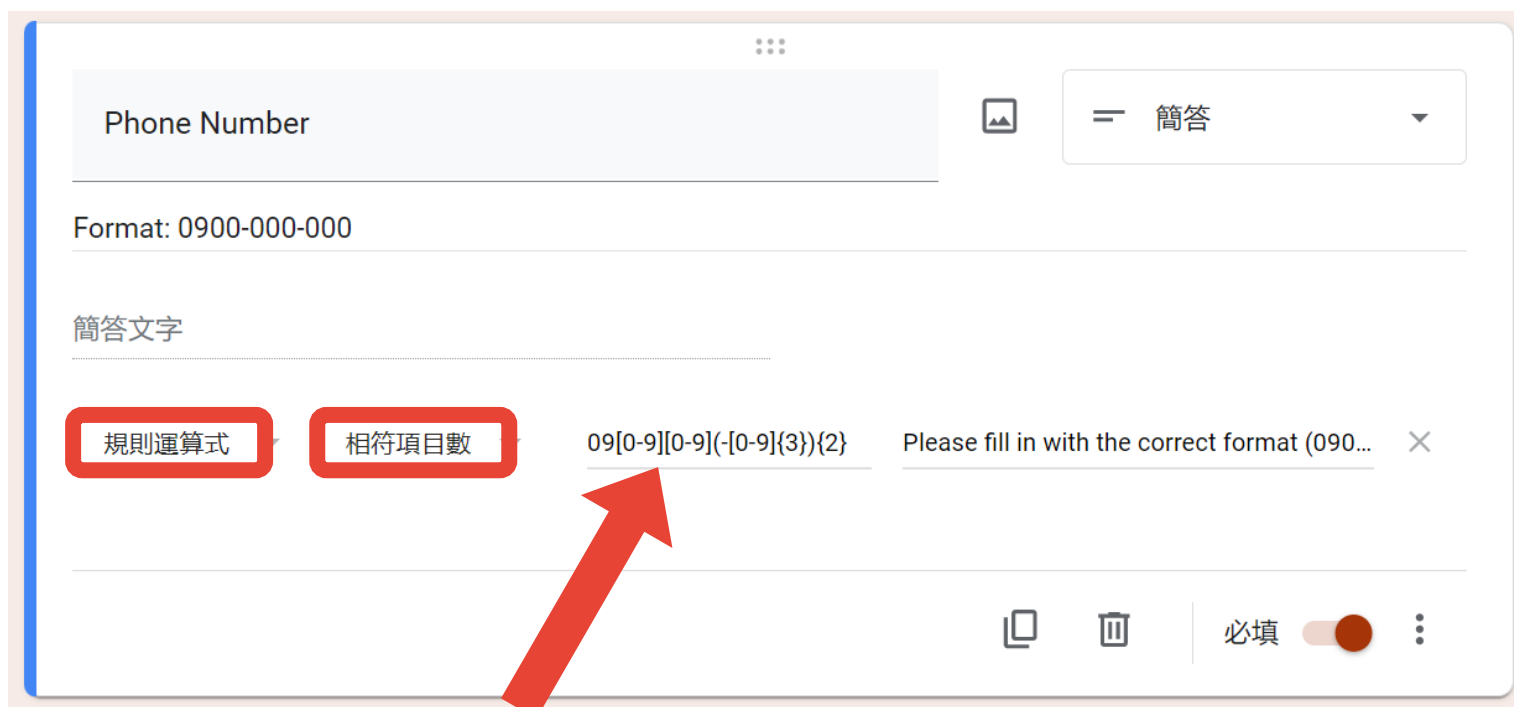
簡答文字

規則運算式 相符項目數 [BMD][0-9]{9} Please fill in the correct Std ID. X

必填

正規語言的實際應用

■ Google Form 回應驗證



The screenshot shows the validation settings for a Google Form field titled "Phone Number". The field type is set to "簡答" (Short answer). The format is specified as "0900-000-000". The validation rule is set to "規則運算式" (Regular expression), and the matching criteria is set to "相符項目數" (Number of matches). The regular expression is "09[0-9][0-9](-[0-9]{3}){2}", and the error message is "Please fill in with the correct format (090...". A red arrow points to the regular expression field.

Phone Number

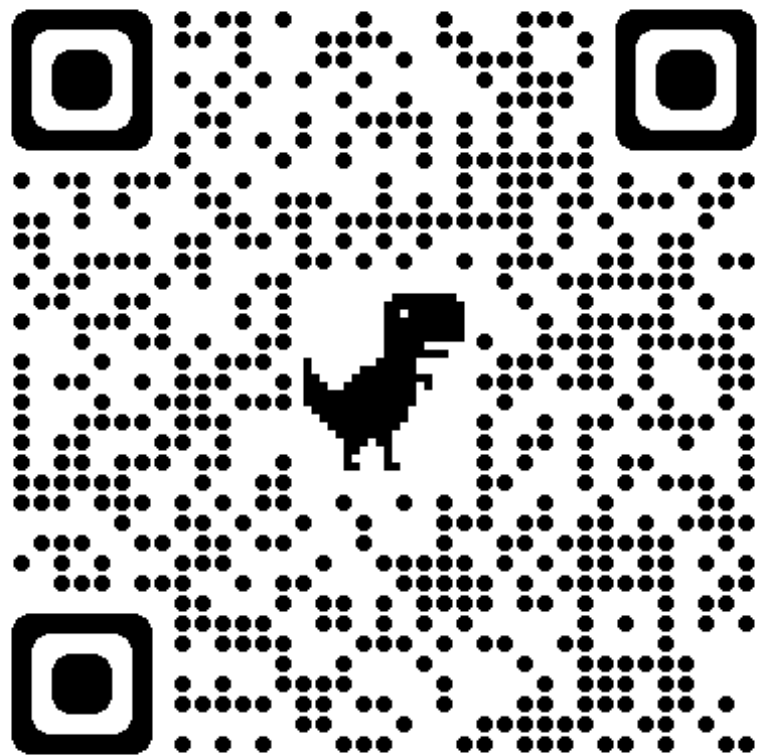
Format: 0900-000-000

簡答文字

規則運算式 相符項目數 09[0-9][0-9](-[0-9]{3}){2} Please fill in with the correct format (090... X

必填 ☒

DEMO 時段登記



- 帳號：學號
預設密碼：
sesame
- 登入後請盡速
變更密碼
- 程式 Demo
及 口頭問答

