

Project 1

- 互動式直譯器
- 介紹 OurScheme
- 抽象語法樹 (Abstract Syntax Tree)
- 格式化輸出 (Pretty Print)
- 處理錯誤

互動式直譯器？

REPL (Read-Eval-Print Loop)

- Read : 讀取使用者輸入的指令
- Eval : 解析並執行該指令
- Print : 輸出執行結果
- Loop : 等待下一個輸入，形成迴圈

Welcome to OurScheme!

```
> (1 . (2 . 3)
)
(1
2
.
3
)
> (exit)
```

Thanks for using OurScheme!

```
# Step 1: 印出歡迎訊息
Print "Welcome to OurScheme!"
Print "\n"
Print "> " # 提示使用者輸入

# Step 2: 進入 REPL 迴圈
Repeat:
    # 讀取使用者輸入的 S-Expression
    expr = ReadSExp()

    # 如果讀取到 EOF (檔案結尾)，跳出迴圈
    If expr == END-OF-FILE:
        Print "ERROR (no more input) : END-OF-FILE encountered"
        Break

    # 若使用者輸入的是 '(exit)'，結束迴圈
    If expr == "(exit)":
        Break

    # 格式化並輸出 S-Expression
    PrintSExp(expr)

    # 顯示新的提示符號，等待下一次輸入
    Print "> "

# Step 3: 結束程式
Print "\n"
Print "Thanks for using OurScheme!"
```

什麼是 OurScheme？

- OurScheme 是基於 Lisp/Scheme 的一種語言
- 語法以 **S-Expression** (S-Exp) 為基礎
- 由原子 (Atoms) 和清單 (Lists) 組成
- 區分大小寫 apple != Apple
- 支援數字、字串、符號 (Symbols)、布林值等

OurScheme tokens

Token 類型	對應字元	備註
LEFT-PAREN	(左括號，代表 list 開始
RIGHT-PAREN)	右括號，代表 list 結束
INT	123, +123, -123	整數型別，允許正負號
STRING	"string's (example)." " " " " " "	字串，必須使用雙引號包圍，不允許跨行
DOT	.	可作為 Dotted Pair 分隔符號
FLOAT	123.567, 123., .567, +123.4, -.123	浮點數，輸出需統一小數點後三位 ("%.3f")
NIL	nil, #f	代表 `false`，必須統一輸出為 `nil`
T	t, #t	代表 `true`，必須統一輸出為 `#t`
QUOTE	'	單引號，表示 `quote` 簡寫
SYMBOL	abc, a.b, a-B!c? (區分大小寫)	符號 (變數名稱)，大小寫區分

Syntax of OurScheme

<S-exp> ::= <ATOM>

| LEFT-PAREN <S-exp> { <S-exp> } [DOT <S-exp>] RIGHT-PAREN
| QUOTE <S-exp>

<ATOM> ::= SYMBOL | INT | FLOAT | STRING | NIL | T | LEFT-PAREN RIGHT-PAREN

| : 或

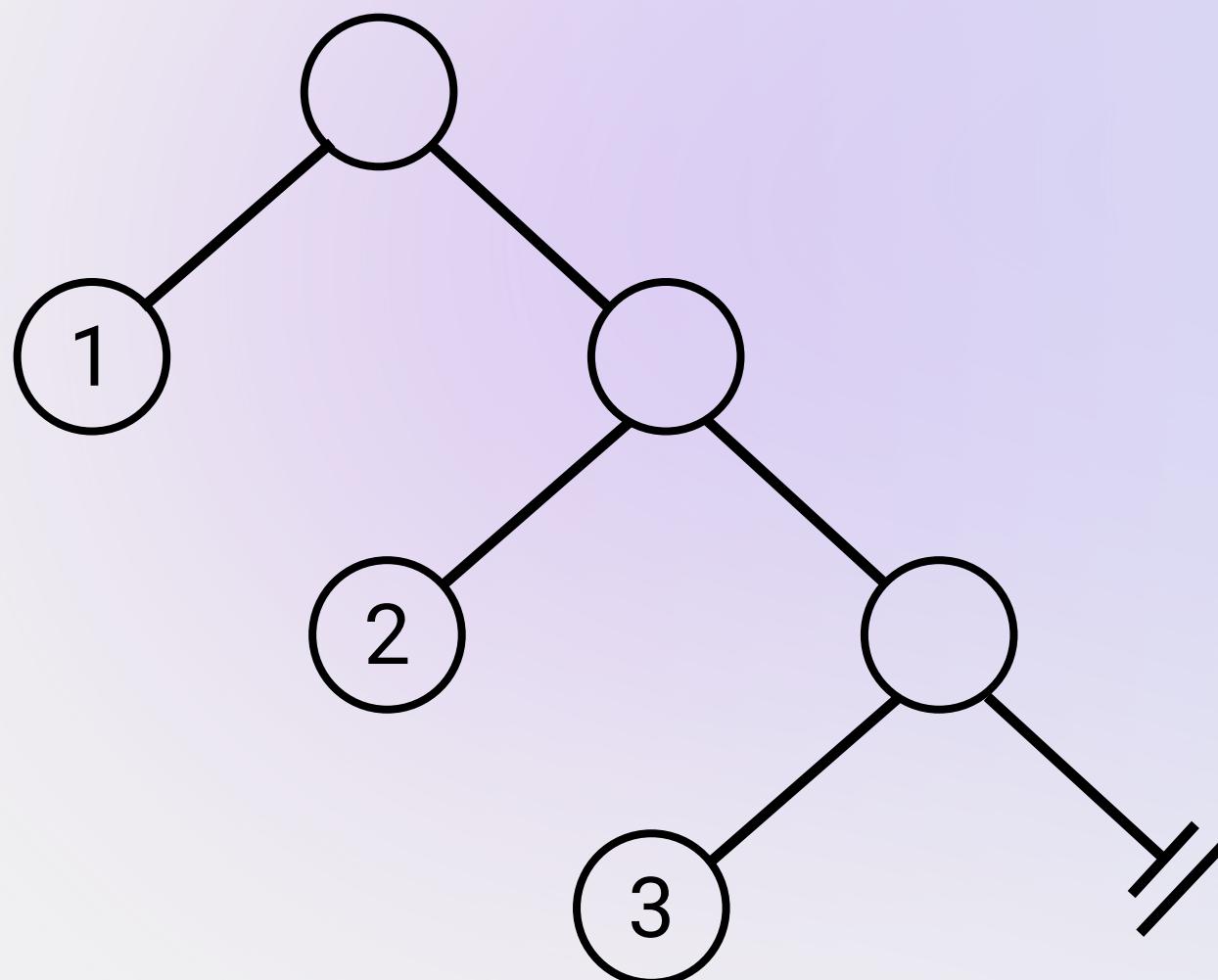
{ } : 出現0次或多次

[] : 出現0次或1次

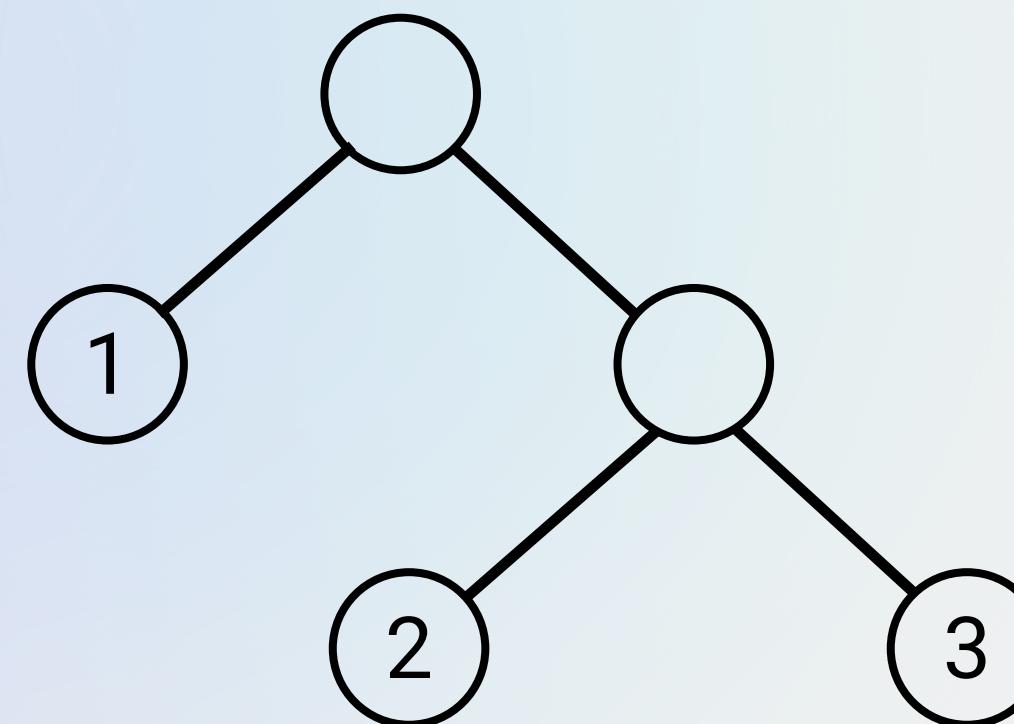
抽象語法樹 AST

AST 是一種樹狀結構，用來表示 OurScheme 的語法結構，並且搭配遞迴下降解析器（ recursive decent parser ）使得解析、評估（Evaluation）、優化 等操作更加容易處理。
例如：

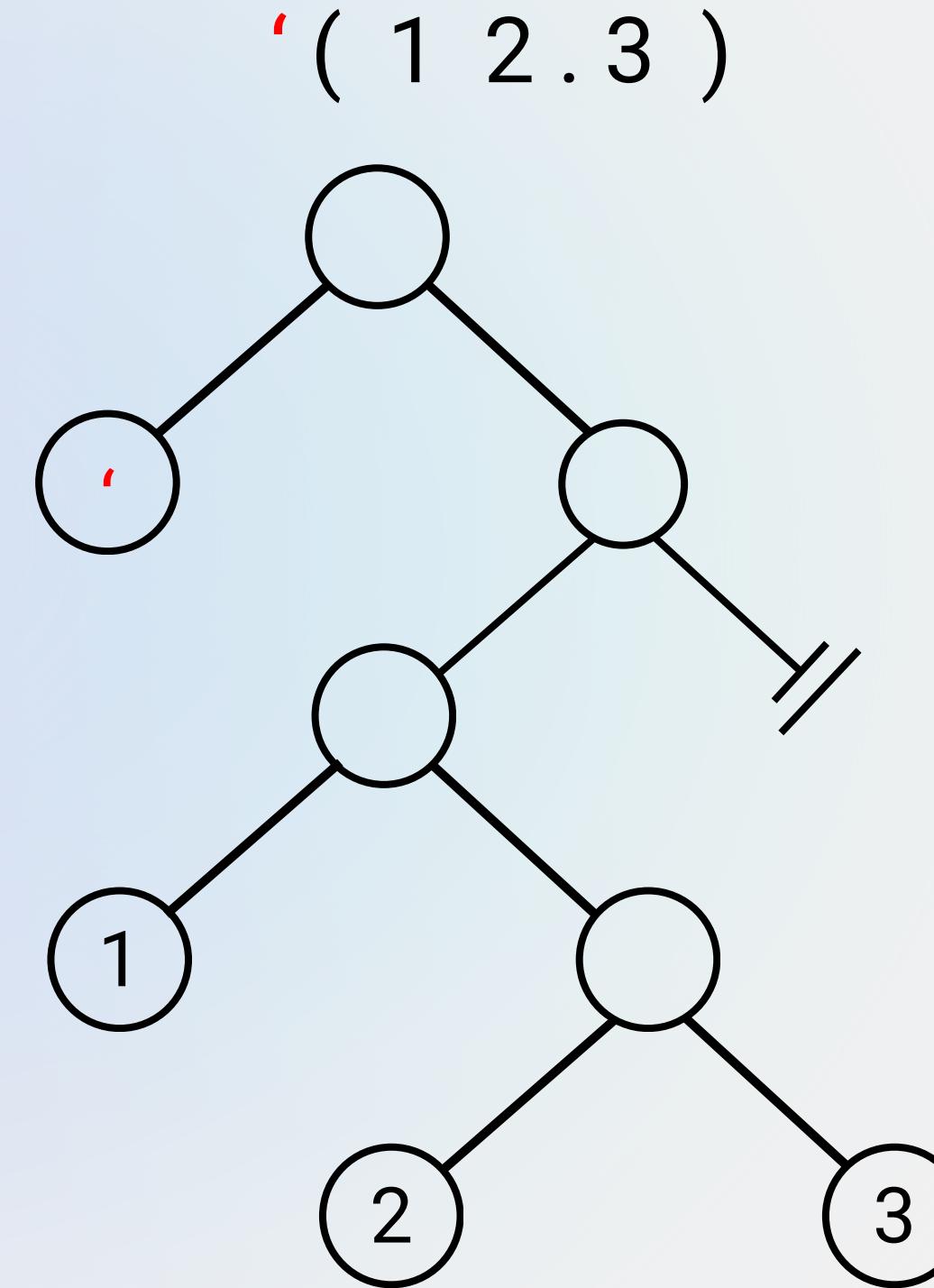
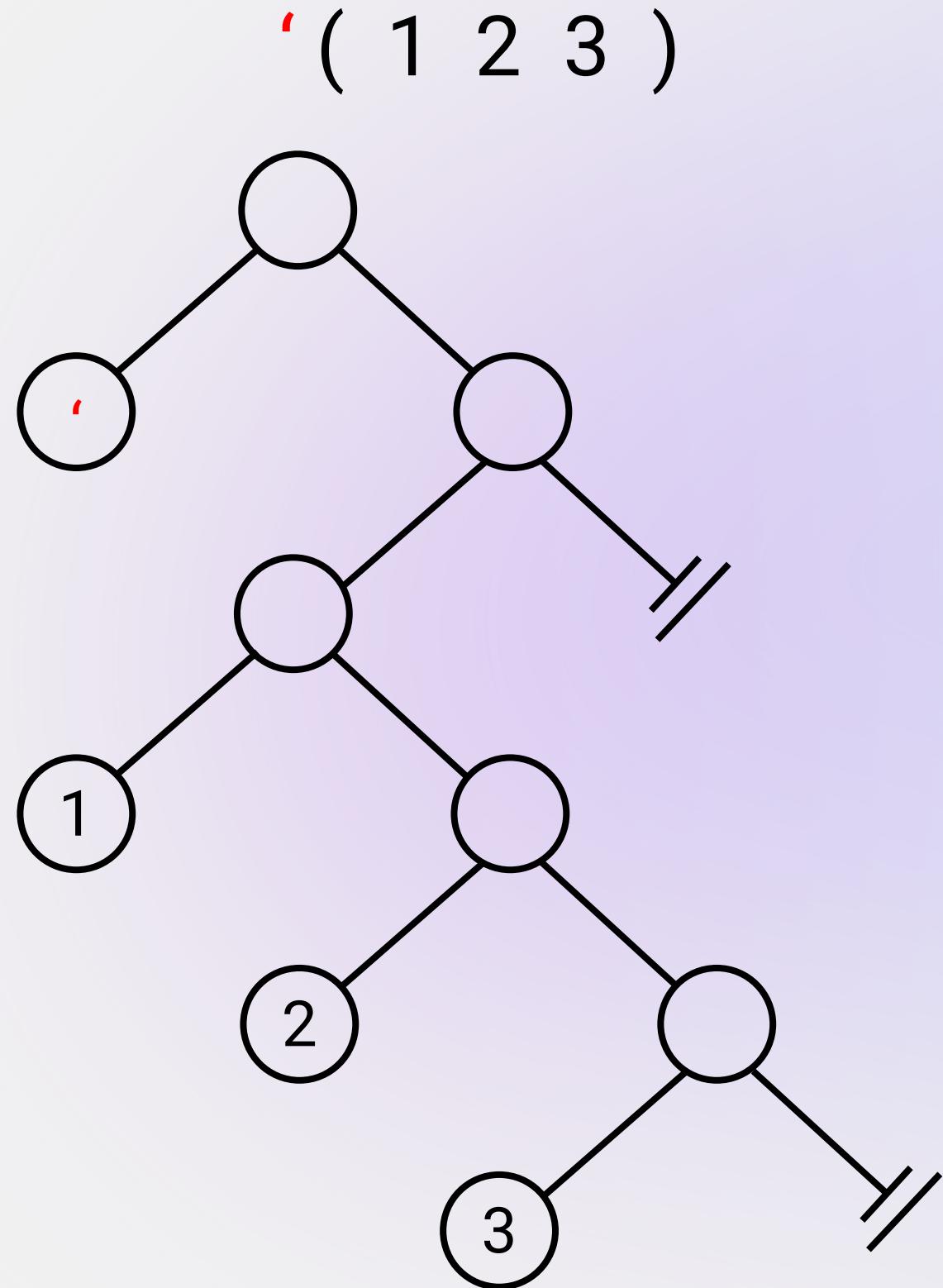
(1 2 3)



(1 2 . 3)



抽象語法樹 AST



格式化輸出 Pretty print

(1 2 3)

```
(  
 1  
 2  
 3  
)
```

(1 2 . 3)

```
(  
 1  
 2  
 ·  
 3  
)
```

'(1 2 . 3)

```
('quote  
(  
 1  
 2  
 ·  
 3  
)  
)
```

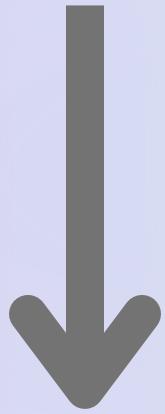
處理錯誤

錯誤類型	發生條件	對應錯誤訊息	舉例
未預期的 Token	碰到不符合文法的token (如 . 開頭的錯誤輸入)	ERROR (unexpected token) : atom or '(' expected when token at Line X Column Y is >>...<<	(1 2 .)
缺少右括號)	S-Exp 未完整封閉，遇到 . 或) 時語法錯誤	ERROR (unexpected token) : ')' expected when token at Line X Column Y is >>...<<	(1 2 . 3 4)
未關閉字串	遇到 " 開頭但沒有對應 " 結束，且換行	ERROR (no closing quote) : END-OF-LINE encountered at Line X Column Y	("hello)
檔案結束 (EOF)	讀取輸入時沒有 (exit) ，但檔案提前結束	ERROR (no more input) : END-OF-FILE encountered	-

處理錯誤

Once the attempt to read in an S-expression fails, the line containing the error-token is ignored.

The system starts to read in an S-expression from the next input line.



一旦嘗試讀取 S-expression 失敗，含有 error-token 的那一行將被忽略。系統會從下一個輸入行開始讀取 S-expression。

Proj1 完成步驟

1. 建立 REPL (Read-Eval-Print-Loop)
 - while 迴圈不斷讀取輸入直到 (exit) 或 EOF。
2. 實作 ReadSExp()
 - 解析 token，構建 AST (Abstract Syntax Tree)。
3. 實作 PrintSExp()
 - 遵循 pretty-print 規則 來輸出 S-Expression。
4. 實作錯誤處理
 - 未預期的 token、括號錯誤、EOF、字串錯誤都要報錯並跳過當行。

PL線上測試系統



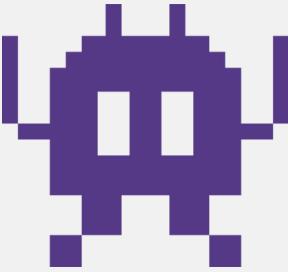
歡迎來到 PL 可視化

中原大學資訊工程學系「程式語言」課程學習輔助工具，用於程式碼的可視化和執行追蹤。

OurScheme

<https://cycu-ice-pl.github.io/website/#/>

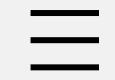
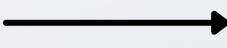
自動化測試工具



- 助教本人開發
- 一鍵測試所有自訂測資
- 支援測資匯入/匯出
- 「知識只有在分享時才有價值。」

互相分享測試資料有助於感情增溫





祝各位順利~