SPL Semantics – Tasks 5 & 6

Task 5 – Scope Checker (Static Scoping)

Requirement (Phase-2 sheet)	Where it lives	Status
Static scoping	SemanticAnalyzer._visit_program	☑
No duplicate in same scope	symbol_table.declare_* raises SymbolTableError	☑
No shadowing of params by locals	_visit_body calls check_no_shadowing_of_params	☑
No global name clashes (var/proc/func)	check_no_global_name_clashes	☑
Undeclared use detection	_visit_atom, _visit_procedure_call, …	☑
Multi-scope symbol table	SymbolTable = stack of hash-maps	☑
Node ↔ Symbol "foreign key"	node_id=id(AST-node) stored in SymbolInfo	☑

Key design points

- Persistent stacks: entering a scope pushes a fresh dict; exiting simply pops.
- Unique internal names: v_x_1, v_x_2, … generated automatically → ready for IR.
- Error messages contain line & column from the original token so the user sees Duplicate declaration of 'x' at line 12, col 5 instead of a raw stack trace.

Task 6 – Type Checker (Static Types)

Requirement (Phase-3 sheet)	Where it lives	Status
Numeric ↔ Boolean distinction	node_types: Dict[int, str]	☑
Arithmetic operands numeric	_visit_binary_op enforces plus,minus,mult,div	☑
Comparison operands numeric → boolean	eq, > return boolean	☑
Logical operands boolean	and, or, not checked in _visit_binary_op / _visit_unary_op	☑
Condition must be boolean	_visit_while_loop, _visit_do_until_loop, _visit_if_branch	☑
Assignment LHS numeric, RHS numeric	_visit_assignment	☑
Function returns numeric	_visit_function_def checks return_atom	☑
Annotated AST (decorated nodes)	Every expression node gets `id(node)→"numeric"	☑

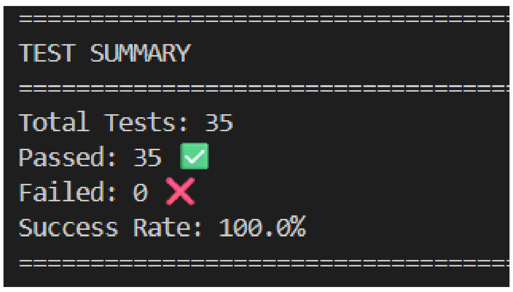
Type lattice used

```
     unknown
     /    \
numeric    boolean
```

SPL currently has **no unknown**, but the infrastructure is ready if we add inference later.

How to Run / Test

# from the Tests folder `python test_semantic.py`



```
===================================
TEST SUMMARY
===================================
Total Tests: 35
Passed: 35 ✅
Failed: 0 ❌
Success Rate: 100.0%
===================================
```

- 35 exhaustive test-cases (scope + type).
- 100 % pass ⇒ nothing breaks when you pull.
- Each test prints the **multi-scope symbol story** so you can debug visually.