Feature Mapping: Solidity vs Algorand Python

Category	Solidity	Algorand Python
Contract Structure	`contract X { ... }`	`class X(arc4.ARC4Contract):`
	Single inheritance	Multiple base classes
Basic Types	`uint256`, `uint8`	`UInt64`
	`address`	`Account`
	`string`	`String`
	`bytes`, `bytes32`	`Bytes`
	`bool`	`bool`
	`mapping`	`GlobalState`
Complex Types	Arrays	`Box`, `BoxRef`
	Structs	Multiple boxes
	Enums	Constants
Function Modifiers	`public`	`@arc4.abimethod()`
	`view`	`@arc4.abimethod(readonly=True)`
	`pure`	`@subroutine`
	`payable`	Check `Txn.amount` in function
Operations	`require(x, msg)`	`assert x, msg`
	`+=`, `-=`, `*=`, `/=`	Direct operators
	`revert("msg")`	`assert False, "msg"`
Global Variables	`msg.sender`	`Txn.sender`
	`msg.value`	`Txn.amount`
	`block.timestamp`	`Global.latest_timestamp`

	`address(this)`	`Global.current_application_address`
Data Storage	State Variables	`GlobalState`
	Mappings	`GlobalState` with mapping
	Arrays	`Box`, `BoxRef`
	Local Variables	Method variables
Special Functions	`constructor()`	`__init__()` and `create()`
	`fallback()`	`approval_program()`
	`receive()`	Handle in `approval_program()`
Visibility	`public`	`@arc4.abimethod()`
	`private`	`@subroutine`
	`internal`	`@subroutine`
	`external`	`@arc4.abimethod()`
Events	`event X(...)`	`log(...)`
	`emit X(...)`	`log(Bytes(...))`