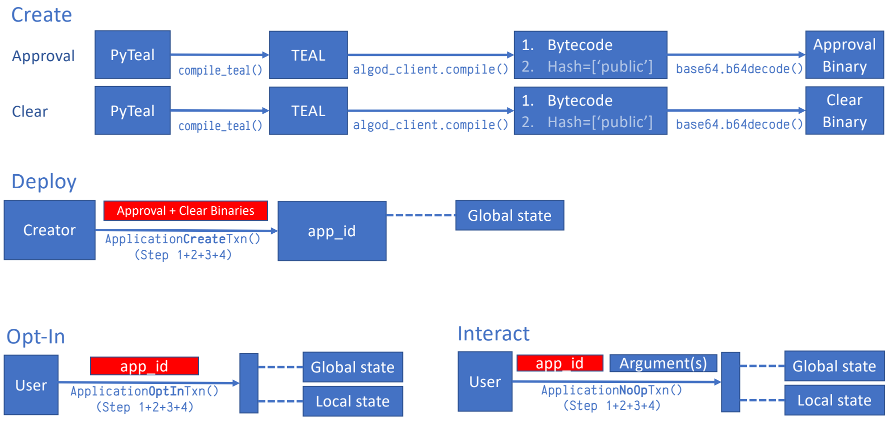
# Writing Smart Contracts
# 07 Smart Contracts

Peter H. Gruber

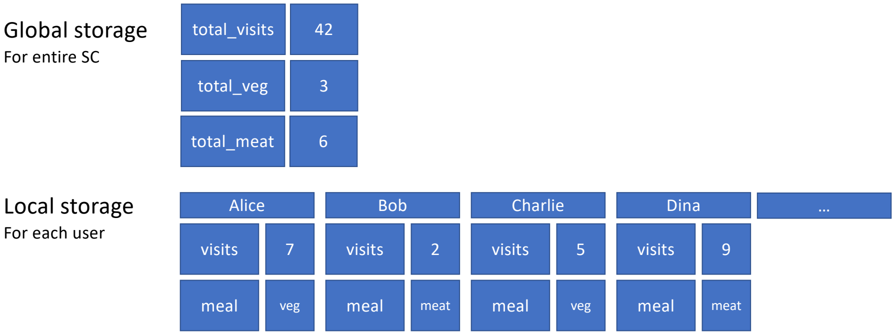
Supported by the Algorand Foundation

## Life of a Smart Contract



- Read from and write to the blockchain

## Variables in PyTEAL



- Global = for the entire smart contract
- Local = (different) values for each user

## Variables in PyTEAL

| | Python | PyTEAL |
|---|---|---|
| Get value | `x` | `App.global`<u>`Get`</u>`(Bytes("x"))` |
| Set value | `x=1` | `App.global`<u>`Put`</u>`(Bytes("x"), Int(1))` |
| Add one | `x=x+1` | `App.global`<u>`Put`</u>`(Bytes("x"), App.global`<u>`Get`</u>`(Bytes("x"))+Int(1))` |

- Variables are read from and written to the blockchain
- Key – Value pairs
  - Key = `Bytes("x")`
  - Value = `Int(1)`
- Local context: need to specify user
  - `Int(0)` = "current user"
  - `App.local`<u>`Get`</u>`(Int(0), Bytes("x"))`
  - `App.local`<u>`Put`</u>`(Int(0), Bytes("x"), Int(1))`

## Two PyTEAL commands

```
Seq (
    [
        first_command,
        second_command,
        third_command
    ]
)
```

```
Cond (
    [condition_1, what_to_do_1],
    [condition_2, what_to_do_2],
    [condition_3, what_to_do_3],
)
```

- Define variables *outside* of `Cond, Seq`

## Things a smart contract must cover

- Creation
  - Initialize variables
  - `Txn.application_id() == Int(0)`
- Opt-in
  - Initialize local variables
  - `Txn.on_completion() == OnComplete.OptIn`
- Normal interaction
  - What happens in the Smart Contract
  - `Txn.on_completion() == OnComplete.NoOp`
- Opt-out
  - Update number of active users
  - Delete local variables
  - `Txn.on_completion() == OnComplete.CloseOut`
- Update and delete
  - Who can change/delete the smart contract? (nobody?)
  - `Txn.on_completion() == OnComplete.UpdateApplication`
  - `Txn.on_completion() == OnComplete.DeleteApplication`

## Life cycle of a smart contract

**Approval Program:** cover everything except clear state

```
approval_pyteal = Cond(
   [Txn.application_id() == Int(0), handle_creation],
   [Txn.on_completion() == OnComplete.OptIn, handle_optin],
   [Txn.on_completion() == OnComplete.CloseOut, handle_closeout],
   [Txn.on_completion() == OnComplete.UpdateApplication, handle_updateapp],
   [Txn.on_completion() == OnComplete.DeleteApplication, handle_deleteapp],
   [Txn.on_completion() == OnComplete.NoOp, handle_noop]
)
```

**Clear State Program:** cover forced opt out

```
clearstate_pyteal = handle_closeout
```

**Difference**

- Approval program can say "no" to opt-out request
- Clear state program *must* clean up local user's state

https://developer.algorand.org/docs/get-details/dapps/pyteal/

## Transaction costs and limitations

## Transaction Costs

**Minimum Balance**

- Contract creation 0.1 Algo per page (=2kB)
  - + 0.0285 Algos for integer entries
  - + 0.05 Algos for byte entries
- Min balance for opt-in 0.1 Algo (flat)
- Per ASA each 0.1 Algo (creator *or* opt-in)

**Transaction Fees**

- Min fee 0.001 Algo
- Dynmaic per-byte fee depending on congestion

`https://developer.algorand.org/docs/get-details/parameter_tables/`

## Limitations

### Computational Cost

- Fees/holdings **not** depend on computational cost (unlike Ether)
    - There is a max. computational cost of 20’000 units
    - Most operations have a cost of 1 unit, price list at `https://developer.algorand.org/docs/get-details/dapps/avm/teal/opcodes/`

### Smart signatures

- Max size 1000B
- Max computational cost 20’000

### Smart contracts

- Max size 1+3 pages = 8kB
- Max computational cost 700
- Max global variables 64
- Max local variables 15

`https://developer.algorand.org/docs/get-details/parameter_tables/`