

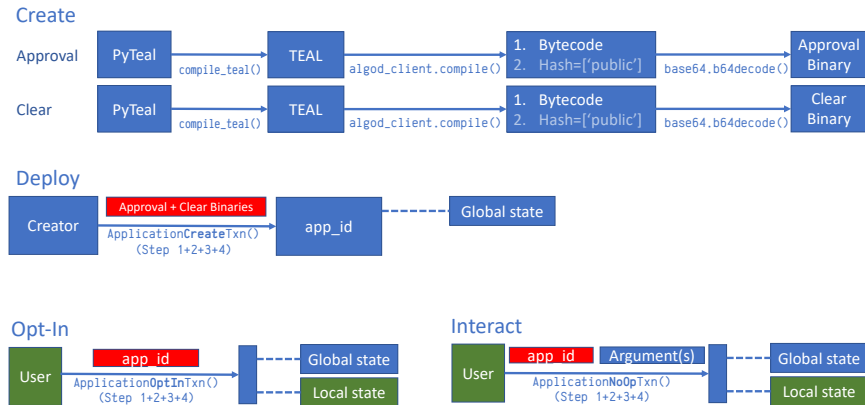
Writing Smart Contracts

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

Example: guest list of a restaurant

Global storage

For entire SC

total_visits	42
total_veg	3
total_meat	6

Local storage

For each user

Alice		Bob		Charlie		Dina		...
visits	7	visits	2	visits	5	visits	9	
meal	veg	meal	meat	meal	veg	meal	meat	

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

Variables in PyTEAL

- Key – Value pairs
 - ▶ Key = name of the variable, e.g. Bytes("x")
 - ▶ Value = which value it gets, e.g. Int(1)

Global variables

	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))

Local variables

- Need to specify user
- Int(0) = "current user"

	Python	PyTEAL
Get value	x	App.local <u>Get</u> (Int(0), Bytes("x"))
Set value	x=1	App.local <u>Put</u> (Int(0), Bytes("x"), Int(1))

Two useful 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

Elements of a smart contract

- 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: everything except clear state

```
approval_pyteal = Cond(  
    [Txn.application_id() == Int(0), creation_fn],  
    [Txn.on_completion() == OnComplete.OptIn,          optin_fn],  
    [Txn.on_completion() == OnComplete.CloseOut,       closeout_fn],  
    [Txn.on_completion() == OnComplete.UpdateApplication, update_fn],  
    [Txn.on_completion() == OnComplete.DeleteApplication, delete_fn],  
    [Txn.on_completion() == OnComplete.NoOp,           noop_fn]  
)
```

Clear State Program: handle forced opt out

```
clearstate_pyteal = closeout_fn
```

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)
 - ▶ + 0.0035 Algos for integer entries
 - ▶ + 0.025 Algos for byte entries
 - ▶ + 0.025 Algos for state 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
- Dynamic per-byte fee depending on congestion

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

Limitations

Computational Cost

- Fees do **not** depend on computational cost (unlike Ether)
 - ▶ 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 16

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