Introduction to SMARTPY

Tezos Smart Contracts Workshop, 2019-11-21

Seb Mondet, TQ Tezos

SmartPy

Python **library** for writing DApps on Tezos

- Generates Michelson + Tests + ...
- Tooling:
  - WebIDE and CLI tools
  - Simulate & analyze
  - Deploy & interact

Why Python

- One of the most popular languages in the world
- Intuitive syntax
- Good meta-programming capabilities
- New users believe they already know it
  $\rightarrow$ Tezos Gateway Drug :)

Python script to Simulation/Michelson

SmartPy programs generate **SmartML** contracts

- SmartML is an imperative, type-inferred intermediate representation
- SmartML is also an OCaml library:
  - Compiled to Native (tests, CLI tools) and to Javascript (WebIDE, more CLI tools)
  - Typing, Analysis
  - Interpreter
  - Compiler to Michelson (+ Michelson to Michelson optimizations)

Example 0: Full Contract

```
import smartpy as sp
class HelloWorld(sp.Contract):
    def __init__(self): self.init(mem = "")

    @sp.entryPoint
    def remember(self, param):
        self.data.mem += param

@addTest(name = "Test")
def test():
    c = HelloWorld()
    s = sp.testScenario()
    s += c
    s += c.remember("Hello, ")
    s += c.remember("World!")
```

WebIDE → Simulation

SMARTPY
BY SMART CHAIN ARENA

Editor >_ | Tezos Faucet Importer | Explorer | Prettifier | Help ?

File | Editor | Tests | Help

```
1  import smartpy as sp
2
3  class HelloWorld(sp.Contract):
4      def __init__(self): self.init(mem = "")
5
6      @sp.entryPoint
7      def remember(self, param):
8          self.data.mem += param
9
10 @addTest(name = "Test")
11 def test():
12     c = HelloWorld()
13     s = sp.testScenario()
14     s += c
15     s += c.remember("Hello, ")
16     s += c.remember("World!")
17
```

Contract | SmartPy | Storage | Types | All | Michelson | X

Balance: tez(0)
Storage:

Mem
''

Entry points:

```
def __init__(self):
  self.init(mem = '')

@sp.entryPoint
def remember(self, params):
  self.data.mem += params
```

Transaction [OK] by [] at time [timestamp(0)] (line 15)
Remember 'Hello, '
Balance: tez(0)
Effects:
Storage:

Mem
'Hello, '

Transaction [OK] by [] at time [timestamp(0)] (line 16)
Remember 'World!'
Balance: tez(0)
Effects:
Storage:

Mem

WebIDE → Annotated Michelson

SMARTPY
BY SMART CHAIN ARENA

Editor >_ | Tezos Faucet Importer | Explorer | Prettifier | Help ?

File | Editor | Tests | Help

```
1  import smartpy as sp
2
3  class HelloWorld(sp.Contract):
4      def __init__(self): self.init(mem = "")
5
6      @sp.entryPoint
7      def remember(self, param):
8          self.data.mem += param
9
10 @addTest(name = "Test")
11 def test():
12     c = HelloWorld()
13     s = sp.testScenario()
14     s += c
15     s += c.remember("Hello, ")
16     s += c.remember("World!")
17
```

Contract | SmartPy | Storage | Types | All | Michelson | X

Deploy contract

Generated Michelson: Storage | Code | Storage JSON | Code JSON

Michelson Code

Copy

```
parameter string;
storage   string;
code
      {
        DUP;         # pair(params, storage).pair(params, storage)
        CDR;         # storage.pair(params, storage)
        SWAP;        # pair(params, storage).storage
        CAR;         # params.storage
        # Entry point: remember # params.storage
        # self.data.mem += params # params.storage
        DUP;         # params.params.storage
        DIG 2;       # storage.params.params
        DUP;         # storage.storage.params.params
        DUG 3;       # storage.params.params.storage
        CONCAT;      # string.params.storage
        DUG 2;       # params.storage.string
        DROP;        # storage.string
        DROP;        # string
        NIL operation; # list operation.string
        PAIR;        # pair (list operation) string
      } # pair (list operation) string;
```

Transaction [OK] by [] at time [timestamp(0)] (line 15)

Remember 'Hello, '

WebIDE → Dark Mode

SMARTPY
BY SMART CHAIN ARENA

Editor >_ | Tezos Faucet Importer | Explorer | Prettifier | Help ?

File | Editor | Tests | Help

```
import smartpy as sp

class HelloWorld(sp.Contract):
    def __init__(self): self.init(mem = "")

    @sp.entryPoint
    def remember(self, param):
        self.data.mem += param

@addTest(name = "Test")
def test():
    c = HelloWorld()
    s = sp.testScenario()
    s += c
    s += c.remember("Hello, ")
    s += c.remember("World!")
```

Contract | SmartPy | Storage | Types | All | Michelson | X

Deploy contract

Generated Michelson: Storage | Code | Storage JSON | Code JSON

Michelson Code

Copy

```
parameter string;
storage   string;
code
      {
        DUP;          # pair(params, storage).pair(params, storage)
        CDR;          # storage.pair(params, storage)
        SWAP;         # pair(params, storage).storage
        CAR;          # params.storage
        # Entry point: remember # params.storage
        # self.data.mem += params # params.storage
        DUP;          # params.params.storage
        DIG 2;        # storage.params.params
        DUP;          # storage.storage.params.params
        DUG 3;        # storage.params.params.storage
        CONCAT;       # string.params.storage
        DUG 2;        # params.storage.string
        DROP;         # storage.string
        DROP;         # string
        NIL operation; # list operation.string
        PAIR;         # pair (list operation) string
      } # pair (list operation) string;
```

Transaction [OK] by [] at time [timestamp(0)] (line 15)

Remember 'Hello, '

WebIDE: Implementation

What happens:

- Python code executed with the Brython interpreter.
- Constructs SmartML: S-Expression.
- Contract enters `js_of_ocaml` world:
  - Type inference / checking
  - Simulation
  - Compilation
  - Back to the UI to construct the HTML "right pane"

Example 1: Tezos Primitives

```
@sp.entryPoint
def setCurrentValue(self, params):
    thingToSign = sp.pack(
        sp.record(
            o = self.data.currentValue,
            n = params.newValue,
            a = sp.self,
            c = self.data.counter))
    sp.verify(
        sp.checkSignature(
            self.data.bossPublicKey, # Only tz1 in browser for now
            params.userSignature,
            thingToSign))
    self.data.currentValue = params.newValue
    self.data.counter = self.data.counter + 1
```

Example 2: Some OO

```
class MultiSigFactory(sp.Contract):
    def __init__(self):
        # ...

    @sp.entryPoint
    def checkSigsAndDo(self, params):
        # ...
        self.onOK(contract)

    def onOK(self, contract):
        pass

class MultiSigFactoryWithPayment(MultiSigFactory):
    def onOK(self, contract):
        sp.send(contract.owner, contract.amount)
```

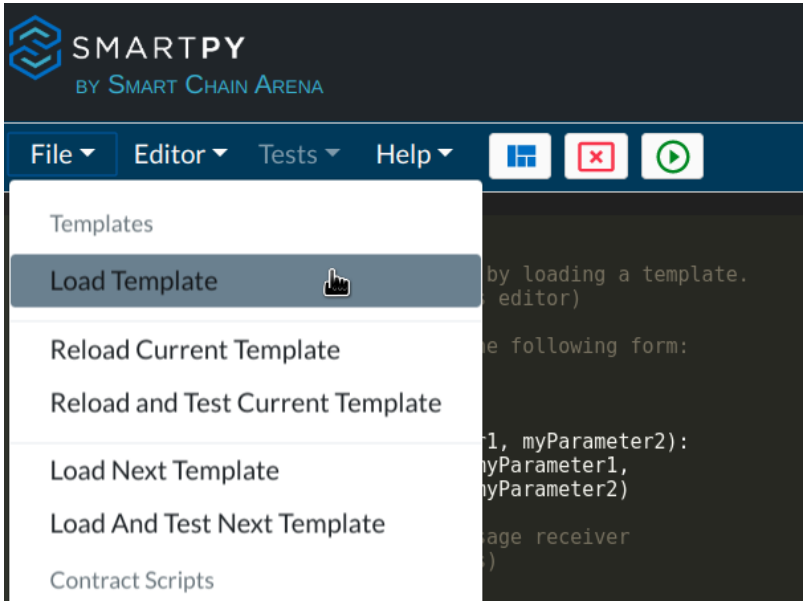
Example 3: Some Meta-programming

```
class NimGame(sp.Contract):
    def __init__(self, size, bound = None, winnerIsLast = False):
        self.bound        = bound
        self.winnerIsLast = winnerIsLast
        self.init(deck = range(1, size + 1), size = size,
                  nextPlayer = 1, claimed = False, winner = 0)

    @sp.entryPoint
    def remove(self, params):
        # [...]
        sp.verify(params.cell < self.data.size)
        sp.verify(1 <= params.k)
        if self.bound is not None:       # ------>  NOT AN sp.if !
            sp.verify(params.k <= self.bound)
        sp.verify(params.k <= self.data.deck[params.cell])
```

WebIDE → Templates

The `NimGame`, and more full / didactic examples:



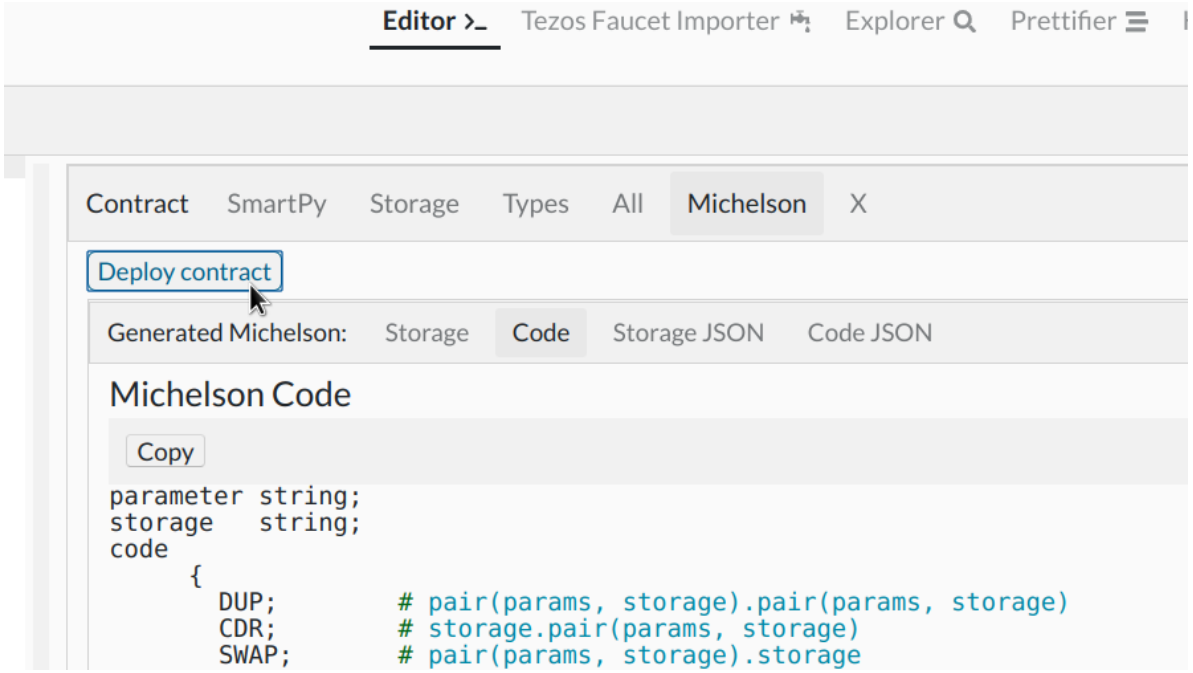
Non-Hello-World Examples

Within the WebIDE:

- Calculator
- Fungible and non-fungible assets
- Multisig contracts
- Escrow contract
- State channels (under development)
- Games: tic-tac-toe, nim, chess

See also on SmartPy.io.

WebIDE Origination Button



WebIDE Origination Form

SMARTPY
BY SMART CHAIN ARENA

Network: Test Network Contract Origination | Mainnet Contract Origination

Test Network Contract Origination

Origination of the contract with ConseilJS by Cryptonomic on test networks: babylonnet, zeronet and other test networks.

1. Private Key and Account

Warning: this is for test networks such as Babylonnet and Zeronet, do not input Mainnet Private Keys here.
A private key is needed - you can use the Tezos Faucet Importer to retrieve one.

Private Key:

With a private key, you can check credentials and compute account public key hash *and obtain a public account.*

Account public key hash:

2. Contract Origination Parameters

Determine deployment parameters.

Node: `https://rpcalpha.tzbeta.net` Switch Node ↺

WebIDE Faucet Importer

WebIDE Contract Explorer

SmartPy Contract Explorer

Node: https://rpczero.tzbeta.net Switch Node ↺

Please select the corresponding protocol for this node.
Tezos mainnet is currently on Protocol 005 - Babylon.

- ○ Protocol 004 - Athens
- ◉ Protocol 005 - Babylon

Contract address: KT1GvgQwPwo8ZdYojFrQyjs1QtjRKjn52cbV Explore

Contract Data

Address	Balance	Manager
KT1GvgQwPwo8ZdYojFrQyjs1QtjRKjn52cbV	sp.tez(0)	

Storage

Administrator	Balances			Paused	TotalSupply
tz1YB12JHVHw9GbN66wyfakGYgdTBvokmXQk	Key	Approvals	Balance	False	0

View on other explorers: Better Call Dev Mainnet, Babylonnet, or Zeronet **-or-** TzStats Mainnet, Babylonnet, or Zeronet.

WebIDE Message Builder

Message Builder

This message builder helps build a command for the Tezos Client to interact with the contract.
This is very experimental.
It may be correct for SmartPy generated contracts. It's certainly not in full generality.
Please use cautiously and at your own risk.

Mint

Address	Amount
tz1YB12JHVHw9	1000

Build Message

```
Message OK.
Value: mint(sp.record(address = sp.address('tz1YB12JHVHw9GbN66wyfakGYgdTBvokmXQk'), amount = 1000))
Michelson: (Left (Left (Left (Right (Pair "tz1YB12JHVHw9GbN66wyfakGYgdTBvokmXQk" 1000)))))
```

Interact with the contract | With the Tezos Client | Direct Access to Test Networks

Default account name: __my_account__

If you have a personal node and a tezos client, you can execute this command in a terminal.

Copy

```
# ./tezos-client transfer 0 from __my_account__ to KT1GvgQwPwo8ZdYojFrQyjs1QtjRKjn52cbV -arg '(Left (Left
(Left (Right (Pair "tz1YB12JHVHw9GbN66wyfakGYgdTBvokmXQk" 1000)))))'
```

Out-of-browser SmartPy

TL;DR:

```
sudo apt install --yes python3 nodejs curl

sh <(curl -s https://SmartPy.io/SmartPyBasic/SmartPy.sh) \
    local-install ~

~/SmartPyBasic/SmartPy.sh help

~/SmartPyBasic/SmartPy.sh compile \
    dapp00.py "TakeOverWorld(42,42)" /tmp/out

...
```

Roadmap / WIP

- Enabling *all* capabilities of Michelson
- Open sourcing
- Generate/Check JSON specifications
  $\rightarrow$ *cf. next talk: Michael Klein*
- Making sandbox testing available to end-users
- Decompilation
- Other analyses:
  - Abstract Interpretation: ownership, value domains, etc.
  - Gas prediction
- Other generation targets:
  - Storage schema / parsing code
  - WhyML?

The End

Thanks ! 🙏

- Website, docs, WebIDE: `smartpy.io`
- Slides: `wr.mondet.org/slides/20191121-smondet-smartpy.pdf`
- Me: `seb.mondet.org`
- TQ: `tqtezos.com`