



# 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
  - 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

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

# WebIDE → Simulation

The screenshot displays the SMARTPY WebIDE interface, which is used for developing and simulating Tezos smart contracts. The interface is divided into several sections:

- Header:** The SMARTPY logo and "BY SMART CHAIN ARENA" are on the left. On the right, there are links for "Editor", "Tezos Faucet Importer", "Explorer", "Prettifier", and "Help".
- Menu Bar:** Below the header, there is a menu bar with "File", "Editor", "Tests", and "Help".
- Code Editor:** The main area on the left contains a Python script for a smart contract. The script defines a class `HelloWorld` that inherits from `sp.Contract`. It includes an `__init__` method to initialize a memory variable `mem`, an `entryPoint` decorator, and a `remember` method to update the memory. A test function `test()` is also defined, which creates a `HelloWorld` instance and calls the `remember` method with the strings "Hello" and "World".
- Simulation Results:** On the right side, there are two panels showing the state of the contract during simulation.
  - Contract State:** This panel shows the current state of the contract. It lists "Balance: tez(0)" and "Storage: Mem". Below this, the "Entry points" section shows the `__init__` method being called with `self` and an empty string `''`. The `entryPoint` decorator is also shown, along with the `remember` method being called with `self` and the parameter `params`.
  - Transaction Log:** This panel shows the results of two transactions. The first transaction, labeled "Transaction [OK] by [] at time [timestamp(0)] (line 15)", shows the `remember` method being called with the parameter "Hello". The second transaction, labeled "Transaction [OK] by [] at time [timestamp(0)] (line 16)", shows the `remember` method being called with the parameter "World".

# WebIDE → Annotated Michelson

The screenshot displays the SMARTPY WebIDE interface, which is used for developing and testing smart contracts. The interface is divided into several sections:

- Header:** The SMARTPY logo and name are on the left. On the right, there are links for "Editor", "Tezos Faucet Importer", "Explorer", "Prettifier", and "Help".
- Menu Bar:** Below the header, there is a menu bar with options: "File", "Editor", "Tests", and "Help".
- Code Editor:** The main area on the left contains Python code for a smart contract. The code defines a class `HelloWorld` that inherits from `sp.Contract`. It includes methods for `__init__`, `remember`, and `test`. The `test` method calls `addTest` with the name "Test".
- Michelson Code Panel:** On the right, there is a panel titled "Michelson Code". It has tabs for "Contract", "SmartPy", "Storage", "Types", "All", "Michelson", and "X". The "Michelson" tab is selected. Below the tabs, there is a "Deploy contract" button and a "Generated Michelson" section. The "Generated Michelson" section has tabs for "Storage", "Code", "Storage JSON", and "Code JSON". The "Code" tab is selected, showing the Michelson code generated from the Python code. The code includes comments explaining the operations, such as `DUP`, `CDR`, `SWAP`, `CAR`, `ENTRY POINT`, `PAIR`, `CONCAT`, `DROP`, `NIL operation`, and `PAIR`.
- Transaction Panel:** At the bottom right, there is a panel titled "Transaction [OK] by [] at time [timestamp(0)] (line 15)". It shows a transaction record with a "Remember" operation and a value of "Hello".

# WebIDE → Dark Mode

**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.entry_point
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, 'World!'



# 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

```
1  @sp.entryPoint
2  def setCurrentValue(self, params):
3      thingToSign = sp.pack(
4          sp.record(
5              o = self.data.currentValue,
6              n = params.newValue,
7              a = sp.self,
8              c = self.data.counter))
9      sp.verify(
10         sp.checkSignature(
11             self.data.bossPublicKey, # Only tz1 in browser for now
12             params.userSignature,
13             thingToSign))
14     self.data.currentValue = params.newValue
15     self.data.counter = self.data.counter + 1
```

## Example 2: Some OO

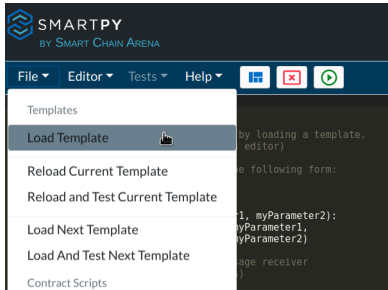
```
1  class MultiSigFactory(sp.Contract):
2      def __init__(self):
3          # ...
4
5      @sp.entryPoint
6      def checkSigsAndDo(self, params):
7          # ...
8          self.onOK(contract)
9
10     def onOK(self, contract):
11         pass
12
13 class MultiSigFactoryWithPayment(MultiSigFactory):
14     def onOK(self, contract):
15         sp.send(contract.owner, contract.amount)
```

## Example 3: Some Meta-programming

```
1  class NimGame(sp.Contract):
2      def __init__(self, size, bound = None, winnerIsLast = False):
3          self.bound = bound
4          self.winnerIsLast = winnerIsLast
5          self.init(deck = range(1, size + 1), size = size,
6                  nextPlayer = 1, claimed = False, winner = 0)
7
8      @sp.entryPoint
9      def remove(self, params):
10         # [...]
11         sp.verify(params.cell < self.data.size)
12         sp.verify(1 <= params.k)
13         if self.bound is not None:      # -----> NOT AN sp.if !
14             sp.verify(params.k <= self.bound)
15         sp.verify(params.k <= self.data.deck[params.cell])
```

# WebIDE → Templates

The NimGame, and more full / didactic examples:



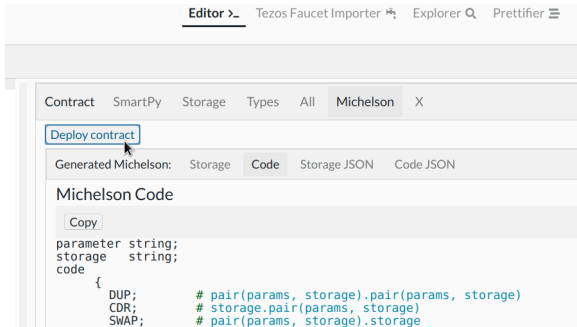
# Non-Hello-World Examples

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](https://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:  [Switch Node ↗](#)



# WebIDE Faucet Importer

 SMARTPY  
BY SMART CHAIN ARENA

Editor >\_ Tezos Faucet Importer  Explorer 🔍 Prettifier ≡ Help ?

## Tezos Faucet Importer

Tezos test network keys importer with [ConseilJS](#) by [Cryptonomic](#).

To obtain free tokens for a test network such as Babylonnet and Zeronet, you can use the faucet.

1. Download data from the [Faucet](#).
2. Copy it here.

# WebIDE Contract Explorer

## SmartPy Contract Explorer

Node:

[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:

[Explore 🔍](#)

### Contract Data

Address	Balance	Manager
KT1GvgQwPwo8ZdYojFr0yjs1QtjRKjn52cbV	sp.tez(0)	

### Storage

Administrator	Balances	Paused	TotalSupply
tz1YB12JHvHw9GbN66wyfakGYgdTBvokmXQK	KeyApprovalsBalance	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
tz1YB12JHVHw9G	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
  - *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](https://smartpy.io)
- Slides: [wr.mondet.org/slides/20191121-smondet-smartpy.pdf](https://wr.mondet.org/slides/20191121-smondet-smartpy.pdf)
- Me: [seb.mondet.org](https://seb.mondet.org)
- TQ: [tqtezos.com](https://tqtezos.com)