SmartPy Cheat sheet

This document doesn't contain everything about the syntax of the language, but should contain everything that is covered during this training, and needed to solve the exercises.

Important: in all the examples below, when some text is between square brackets and italics, *[like this]*, all of it should be replaced by the value you need. In particular, you shouldn't type these square brackets.

Links

SmartPy IDE	https://smartpy.io/ide

Constructor, storage, entrypoints

Constructor, storage initialization	`import smartpy as sp`
	
	`@sp.module`
	`def main():`
	
	`    class [Name of the class](sp.Contract):`
	`        def __init__(self, [parameter 1], [...]):`
	`            self.[name of field] = [value]`

Acces so the storage	`self.data.[name of the field]`

Entrypoint	`@sp.entrypoint`
	`def [name of the entrypoint](self, [parameter 1], [...]):`
	`    [code]`

Note: There are two underscores on each side of the word "init": `__init__`, not `_init_`

Basic types

Integer	`sp.int`	`-34, -12, sp.int(42)`
Natural	`sp.nat`	`sp.nat(12)`

Integer or Natural	`sp.int_or_nat`	`42`
Tez Token	`sp.mutez`	`sp.tez(12), sp.mutez(12000000)`
Boolean	`sp.bool`	`True, False`
String	`sp.string`	`"Hello World!"`
Address	`sp.address`	`sp.address("tz1YtuZ4vhzzn7ssCt93Put8U9UJDdvCXci4")`

Variables

Variable creation or modification	*`[variable] = [initial value]`*
~~Access to the value of the variable~~	*`[variable name]`*

Helping the type inference

Explicit type	`sp.int(4), sp.nat(4)`
Annotating the type	*`sp.cast([expression], [type])`*

Examples of annotations	*`@sp.entrypoint`*
	`def my_entrypoint(self, x, y, z):`
	`    sp.cast(x, sp.int)`
	`    sp.cast(y, sp.string)`

Arithmetic operators

```
i = sp.int(5)
n = sp.nat(3)
a = n + 2   # a has type sp.nat
b = n - 2   # b has type sp.int
c = i * n   # c has type sp.int
d = i / n   # integer division (unlike python)
```
~~`e = i // n # integer division`~~

```
f = sp.ediv(i, n) # returns an option of a pair that contains the
result
                   # of the integer division, and the remainder
g *= 2      # equivalent to g = g * 2
h -= 1      # equivalent to h = h - 1
i = "Hello" + "World" # concatenates two strings

# computation of a ratio between tez
sp.split_tokens(amount, quantity, totalQuantity)
# Computes amount * quantity / totalQuantity where amount is of type
# sp.TMutez, and quantity and totalQuantity are of type sp.nat.
# for example, sp.split_tokens(amount, 30, 100)
# computes 30% of amount
```

Test scenarios

Definition of a test	`@sp.add_test(name = "`*`[name of the test]`*`")`
	`def test():`

Contract instantiation	`c1 = main.StoreValue(`*`[initial value of the storage]`*`)`

Scenario creation	`scenario = sp.test_scenario([module name])`
	`scenario = sp.test_scenario(main)`

Adding some html	`scenario.h1("`*`[some text]`*`")`
	`scenario.h2("`*`[some text]`*`")`
	`scenario.p("`*`[some text]`*`")`

Adding the contract to the scenario	`scenario += c1`
Adding a call to an entrypoint with no parameter	`c1.`*`[entrypoint]`*`()`
Adding a call to an entrypoint with one parameter	`c1.`*`[entrypoint]`*`(`*`[value]`*`)`
Call to an entrypoint with several parameters	`c1.`*`[entrypoint]`*`(`*`[param 1 name] = [value], [param 2 name] = [value], ...`*`)`
Verification about the storage content	`scenario.verify(c1.data.`*`[field name] == [value]`*`)`

Timestamps

Seconds since 01/01/1970	`sp.timestamp([number of seconds])`

List of parameters	`sp.timestamp_from_utc(year, month, day, hours, minutes, seconds)`
Date and time of the current block	`sp.now`

Adding some time	`d = sp.add_seconds(a, 42)`
	`e = sp.add_minutes(b, 15)`
	`f = sp.add_hours(c, 24)`
	`g = sp.add_days(a, 365)`

Difference, in seconds	`h = sp.now - g`

Pairs

Creation of a pair	`p = (`*`[value 1]`*`, `*`[value 2]`*`)`
First element	`sp.fst(p)`
Second element	`sp.snd(p)`
Extracting the two values into two python variables	`(x1, x2) = p`

Options

Creation of an option with no value	`o = None`
Creation of an option with a value	`o = sp.Some(`*`[value]`*`)`
Extract the value of an option Triggers an error if there is none	`v = o.unwrap_some()`
Test if an option has a value	`if (o != None):`

Addresses, transactions

Transfer of tokens	`sp.send(`*`[address]`*`, `*`[value in tez]`*`)`

Address of the direct caller of the contract	`sp.sender`
Address of the indirect initial caller of the chain of contracts	`sp.source`
Address of the contrat	`sp.self_address`
Amount transferred to the contract	`sp.amount`
Current balance of the contract	`sp.balance`

Verifications, booleans, errors

Error	`raise "`*`[message]`*`"`
Verification, without a message	`assert `*`[condition]`*
Verification, with a message	`assert `*`[condition], [message]`*

Boolean operators	`a = True`
	`b = not a        # Not`
	`c = a or b    # Logical or`
	`d = a and b    # Logical and`
	`e = a ^ c    # Exclusive or`

Comparisons	`<, >, <=, >=, ==, !=`

Conditional instructions	`if `*`[condition]`*`:`
	   `[code to run if true]`
	`else:`
	   `[code to run if false]`

~~**Warning**~~ ~~Too many parenthesis~~	~~`sp.if (a == b):`~~ ~~`# This won't work. At time of writing, Smartpy doesn't accept parenthesis around the whole condition`~~
~~**Warning**~~ ~~Combining boolean operators and comparisons~~	~~`a < b | b < c # Causes an error`~~ ~~`(a < b) | (b < c) # Works`~~ ~~`# This is due to the mix between smartpy code and python code`~~

Maps

Empty map	`{}`

Pre-filled map	`varMap = {`
	`    [key 1]: [value 1],`
	`    [key 2]: [value 2],`
	`    [...]`
	`}`

Reading an entry	`v = varMap[ [key] ]`
Adding or updating an entry	`varMap[ [key] ] = [value]`
Testing if an entry exists for a given key	`varMap.contains([key])`
Removal of the entry for a key	`del varMap[ [key] ]`

Records

Creation of a record	`varRecord = sp.record(`
	`    [field 1] = [value 1],`
	`    [field 2] = [value 2],`
	`    [...]`
	`)`

Access to a field (read/write)	`varRecord.[field 1] = [value]`
~~Modification of several fields~~	~~`sp.modify_record(varRecord,`~~ ~~`    [field 1] = [value 1],`~~ ~~`    [field 2] = [value 2]`~~ ~~`)`~~

Advanced tests

Creation of a test account	`account1 = sp.test_account("[name of the account]")`
Getting the address of an account	`address1 = account1.address`
In one line	`address1 = sp.test_account("[name of account]").address`

Context of a call to an entrypoint	`c1.entrypoint1().run(sender = [address],`
	`                      amount = [value in tez],`
	`                      now = [a timestamp],`
	`                      valid = False)`

Specify who calls the entrypoint	`sender = [address]`

Specify what amount is transferred	`amount = [value in tez]`
Specify what date is simulated during the call	`now = [a timestamp],`
Specify that the test should fail	`valid = False`
Specify that the test should fail with expected message	`valid = False, exception = [exception]`
Check the balance of the contract	`scenario.verify(c1.balance == sp.tez([value]))`

Serialization, Hashing

Serialization of a value, returns TBytes	`sp.pack([value])`
Deserialization of a value	`sp.unpack([value in TBytes])`
Hashing of a value of type TBytes	`hashedValue = sp.blake2b([value in TBytes])`
Hashing of a value that is not of type TBytes	`hashedValue = sp.blake2b(sp.pack([value])`