Michelson Language Cheat Sheet

Type Taxonomy

int, nat, timestamp ,mutez ..... numeric
string, bytes ..... sequences
list, set, map, big_map ..... structural

Types

address ..... address of untyped contract
bigmap kty vty ..... big map
bool ..... true or false
bytes ..... sequence of bytes
chain_id ..... chain identifier
contract type address of contract w param type type
int ..... arbitrary precision integer
key ..... public key
key_hash ..... hash of public key
lambda ty1 ty2 .... lambda w. given param & return types
list type ..... single immutable homogenous list
map ktv vty ..... immutable map from kty to vty
mutez ..... type for manipulating tokens
nat ..... arbitrary precision natural number
operation ..... internal operation emitted by contract
option type ..... optional value
or ty1 ty2 ..... union of two types
pair ty1 ty2 ..... pair of two values
set cty ..... immutable set of comparable ctys
signature ..... cryptographic signature
string ..... string of characters
timestamp ..... real-world date
unit ..... the type whose only value is unit

Instructions

ABS ..... absolute value of integer
ADD ..... add two numerical values
ADDRESS ..... push the address of a contract
AMOUNT ..... push amount of current tx
AND ..... boolean bitwise AND
APPLY ..... partially apply tuplified fn from stack
BALANCE ..... push current mutez of contract
BLAKE2B ..... compute blake2b hash
CAR ..... access left part of pair
CDR ..... access right part of pair
CHAIN_ID ..... push chain identifier
CHECK_SIGNATURE .. verify signature of bytes of key
COMPARE ..... compare two values
CONCAT ... concatenate string, byte sequence, string list or sequence list
CONS ..... prepend element to list
CONTRACT ty ..... cast address to typed contract
CREATE_CONTRACT ty1 ty2 code . push contract creation operation
DIG n ..... retrieve nth element of stack
DIP n code ..... run code protecting top of stack
DROP n ..... drop the top n elements of stack
DUG n ..... insert top element at depth n
DUP ..... duplicate top of stack
EDIV ..... Euclidian division
EMPTY_BIG_MAP kty vty .... new empty big map from kty to vty
EQ ..... check that top of stack EQuals zero
EXEC ..... execute function from stack
FAILWITH ..... abort current program
GE ..... check top of stack $\geq$ zero
GET ..... access element in map or big_map
HASH_KEY ..... compute base58check of pub key
IF code1 code2 ..... conditional branching
IF_CONS ..... inspect a list
IF_LEFT ..... inspect value of a union
IF_NONE ..... inspect optional value
IMPLICIT_ACCOUNT ... create an implicit account
INT ..... convert nat to int
ISNAT ..... convert non-negative int to nat
ITER code ..... iterate over texttt set list or map
LAMBDA ty1 ty2 code .... push a lambda onto stack
LE ..... check that top value on stack ≤ 0
LEFT ty2 ..... wrap a value in a union (left case)
LOOP code ..... a generic loop
LOOP_LEFT code ..... loop with accumulator
LSL ..... logically shift left a nat
LT ..... check top of stack is < 0
MAP code ..... apply body to each ele of map or list
MEM ..... check for key in map, set or big_map
MUL ..... multiply two numerical values
NEG ..... negate a numerical value
NEQ ..... check top of stack $\neq$ zero
NIL ty1 ..... push empty list of type ty1
NONE ty1 ..... push absent optional value
NOT ..... boolean and bitwise complement
NOW ..... push block timestamp
OR ..... boolean and bitwise OR
PACK ..... serialize data
PAIR .... build a pair from the stack's top 2 elements
PUSH ty1 x ..... push constant value x of type ty1
RIGHT ty1 ..... wrap a value in a union (right case)
SELF ..... push the current contract
SENDER push contract that initiated current internal transaction
SET_DELEGATE ..... push a delegation operation
SHA256 ..... compute a SHA-256 cryptographic hash
SIZE .... obtain size of a string, list, set, map or byte sequence
SLICE .... obtain sub{string|sequence} of string|bytes
SOME ..... wrap an existing optional value
SOURCE push the contract that initiated the current transaction
SUB ..... subtract two numerical values
SWAP ..... swap the top two elements of the stack
TRANSFER_TOKEN ..... push a transfer operation
UNIT ..... pushes the unit value onto the stack
UNPACK ty1 ..... deserialize data, if valid
UPDATE ..... add or remove an element in
XOR ..... boolean and bitwise eXclusive OR
code1; code2 ..... instruction sequence
.{}. ..... empty instruction sequence