自定義型別與函式

Justa from Bucket Protocol

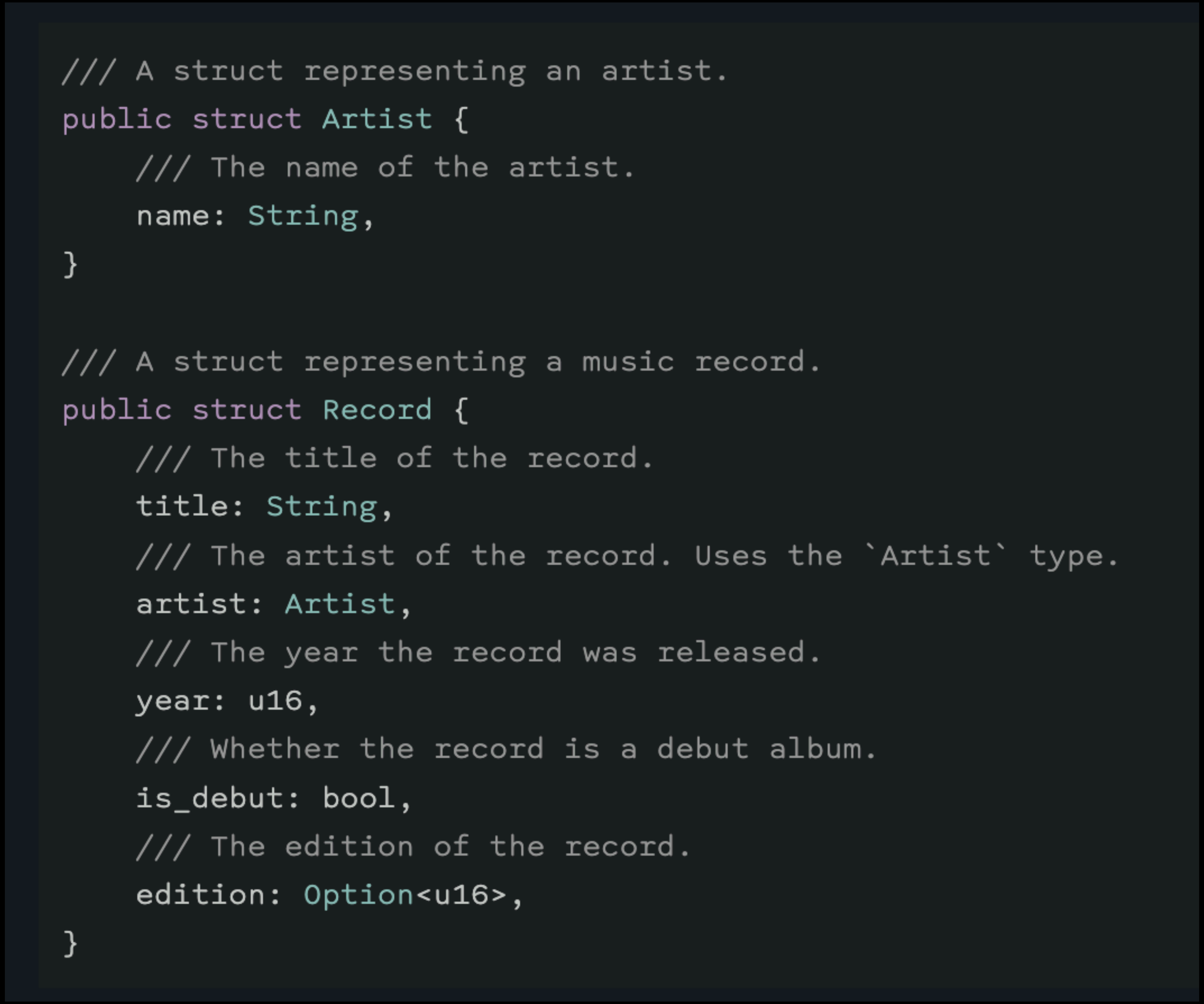
Bucket

大綱

- 自定義型別（struct）與行為（ability）
- 自定義函式（function）與能見度（visibility）
- 物件表現（display）

自定義型別

struct



```rust
/// A struct representing an artist.
public struct Artist {
    /// The name of the artist.
    name: String,
}

/// A struct representing a music record.
public struct Record {
    /// The title of the record.
    title: String,
    /// The artist of the record. Uses the `Artist` type.
    artist: Artist,
    /// The year the record was released.
    year: u16,
    /// Whether the record is a debut album.
    is_debut: bool,
    /// The edition of the record.
    edition: Option<u16>,
}
```

型別能力
abilities

- copy - 此型別可被複製
- drop - 此型別可被任意丟棄
- key - 此型別可被持有或分享
- store - 此型別可被儲存
- key (without store) - 此型別可被擁有或分享但不能被任意轉移
- key + store - 此行別可被擁有或分享且可被任意轉移

自定義函式與能見度

function & visibility

- fun - only used in module
- public fun - can used anywhere
- public(package) fun - only used in the same package
- entry fun - public but can’t packed into PTB

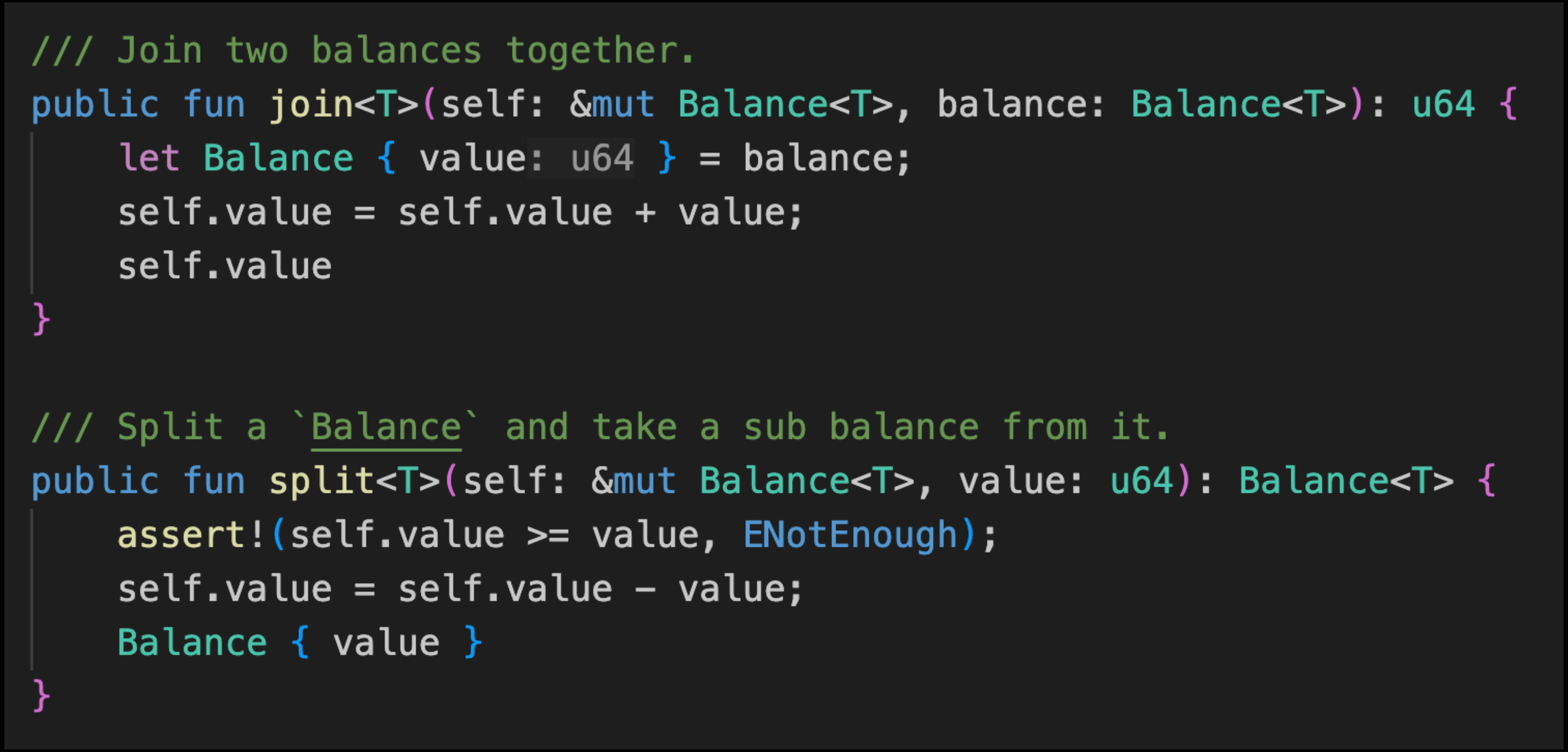
官方範例一

Balance has store

```
/// Storable balance - an inner struct of a Coin type.
/// Can be used to store coins which don't need the key ability.
public struct Balance<phantom T> has store {
    value: u64
}
```

官方範例一

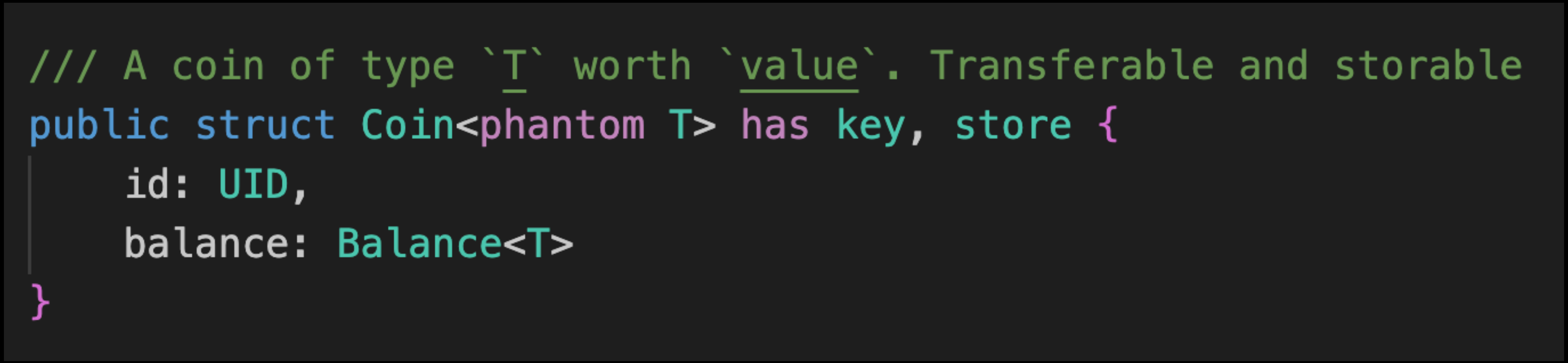
Balance methods



```
/// Join two balances together.
public fun join<T>(self: &mut Balance<T>, balance: Balance<T>): u64 {
    let Balance { value: u64 } = balance;
    self.value = self.value + value;
    self.value
}

/// Split a `Balance` and take a sub balance from it.
public fun split<T>(self: &mut Balance<T>, value: u64): Balance<T> {
    assert!(self.value >= value, ENotEnough);
    self.value = self.value - value;
    Balance { value }
}
```

官方範例二

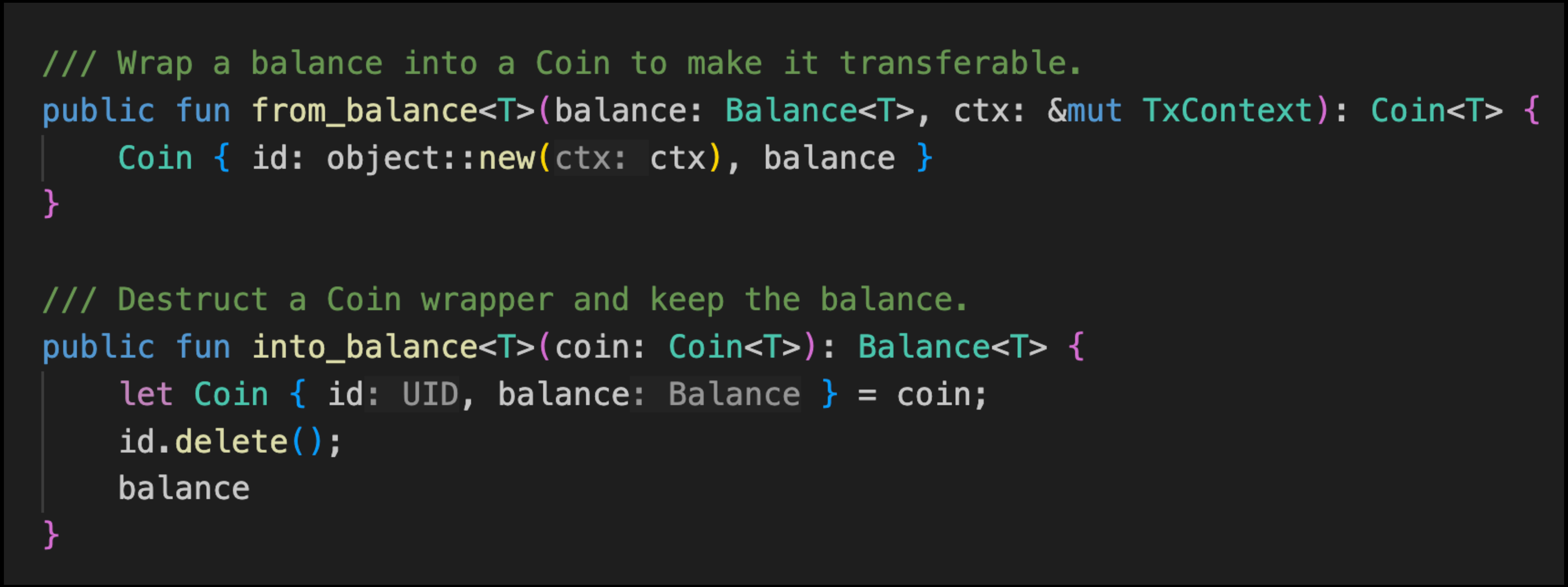
Coin has key + store



```
/// A coin of type `T` worth `value`. Transferable and storable
public struct Coin<phantom T> has key, store {
    id: UID,
    balance: Balance<T>
}
```

官方範例二

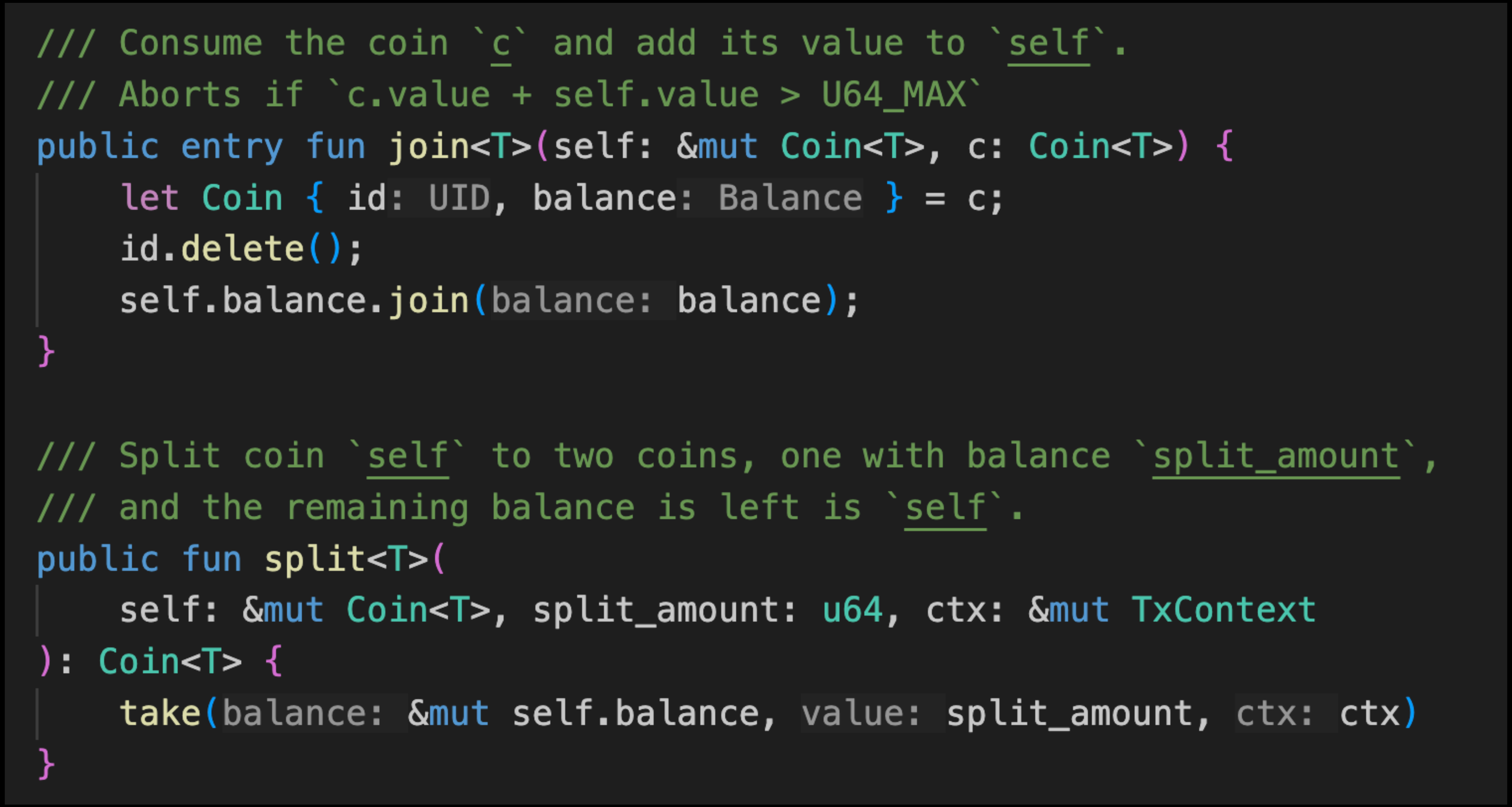
Coin methods



```
/// Wrap a balance into a Coin to make it transferable.
public fun from_balance<T>(balance: Balance<T>, ctx: &mut TxContext): Coin<T> {
    Coin { id: object::new(ctx: ctx), balance }
}

/// Destruct a Coin wrapper and keep the balance.
public fun into_balance<T>(coin: Coin<T>): Balance<T> {
    let Coin { id: UID, balance: Balance } = coin;
    id.delete();
    balance
}
```

官方範例二

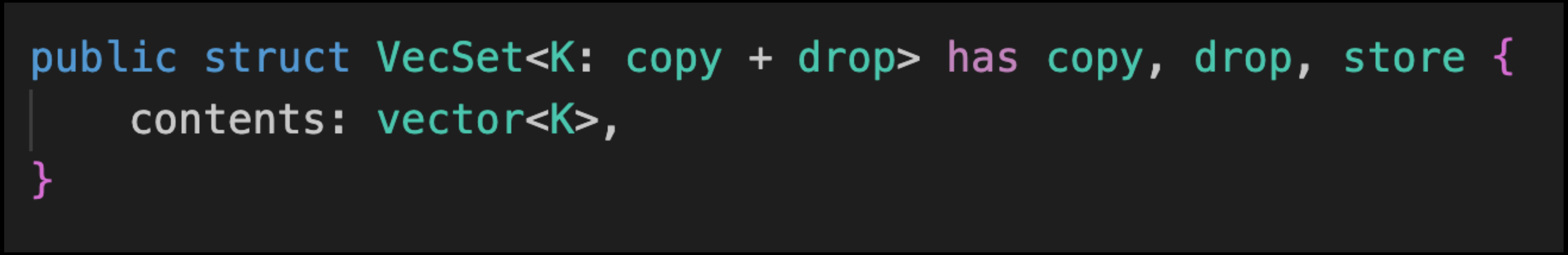
Coin methods



```
/// Consume the coin `c` and add its value to `self`.
/// Aborts if `c.value + self.value > U64_MAX`
public entry fun join<T>(self: &mut Coin<T>, c: Coin<T>) {
    let Coin { id: UID, balance: Balance } = c;
    id.delete();
    self.balance.join(balance: balance);
}

/// Split coin `self` to two coins, one with balance `split_amount`,
/// and the remaining balance is left is `self`.
public fun split<T>(
    self: &mut Coin<T>, split_amount: u64, ctx: &mut TxContext
): Coin<T> {
    take(balance: &mut self.balance, value: split_amount, ctx: ctx)
}
```

官方範例三

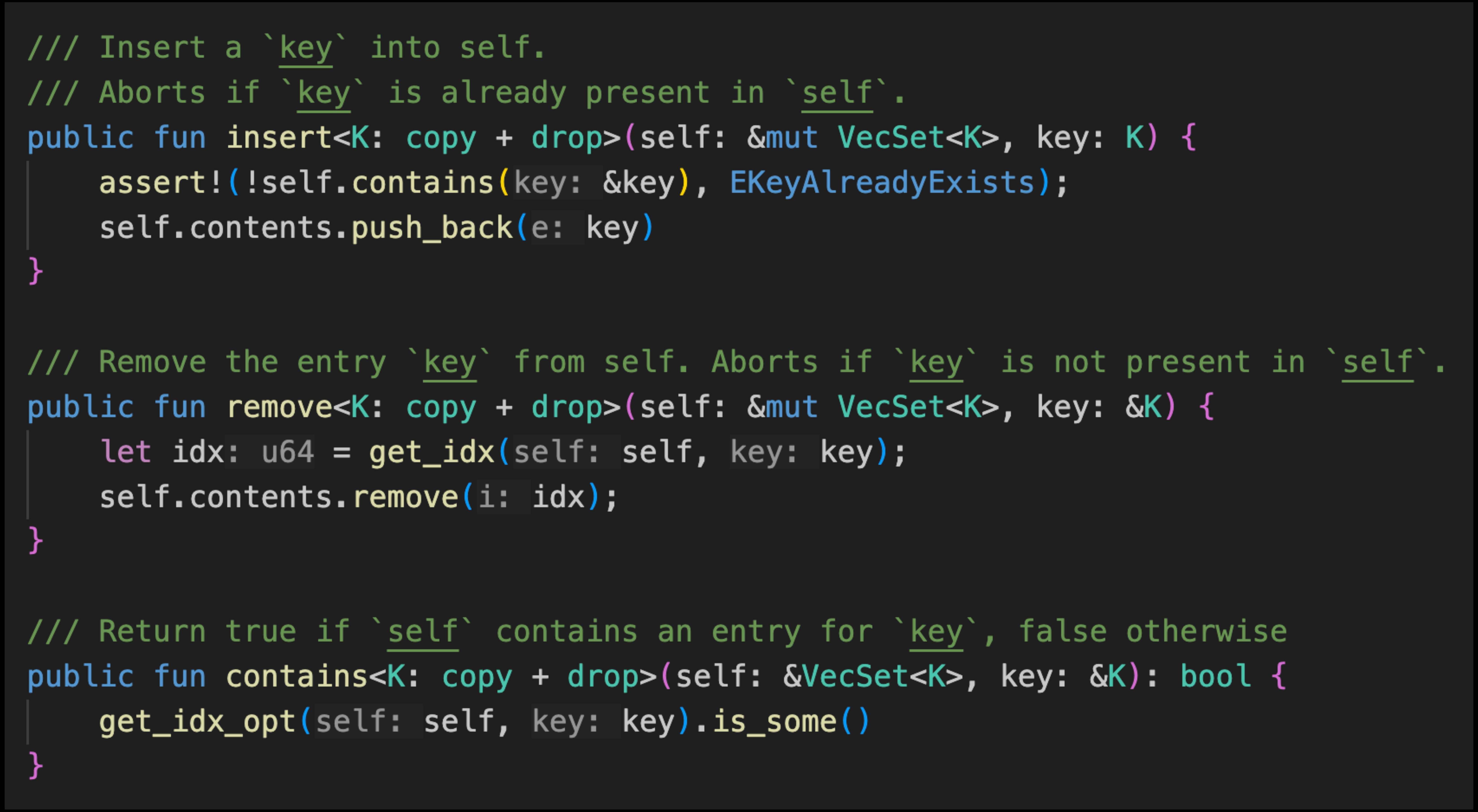
VecSet has copy + drop + store



```
public struct VecSet<K: copy + drop> has copy, drop, store {
    contents: vector<K>,
}
```

官方範例三

VecSet methods

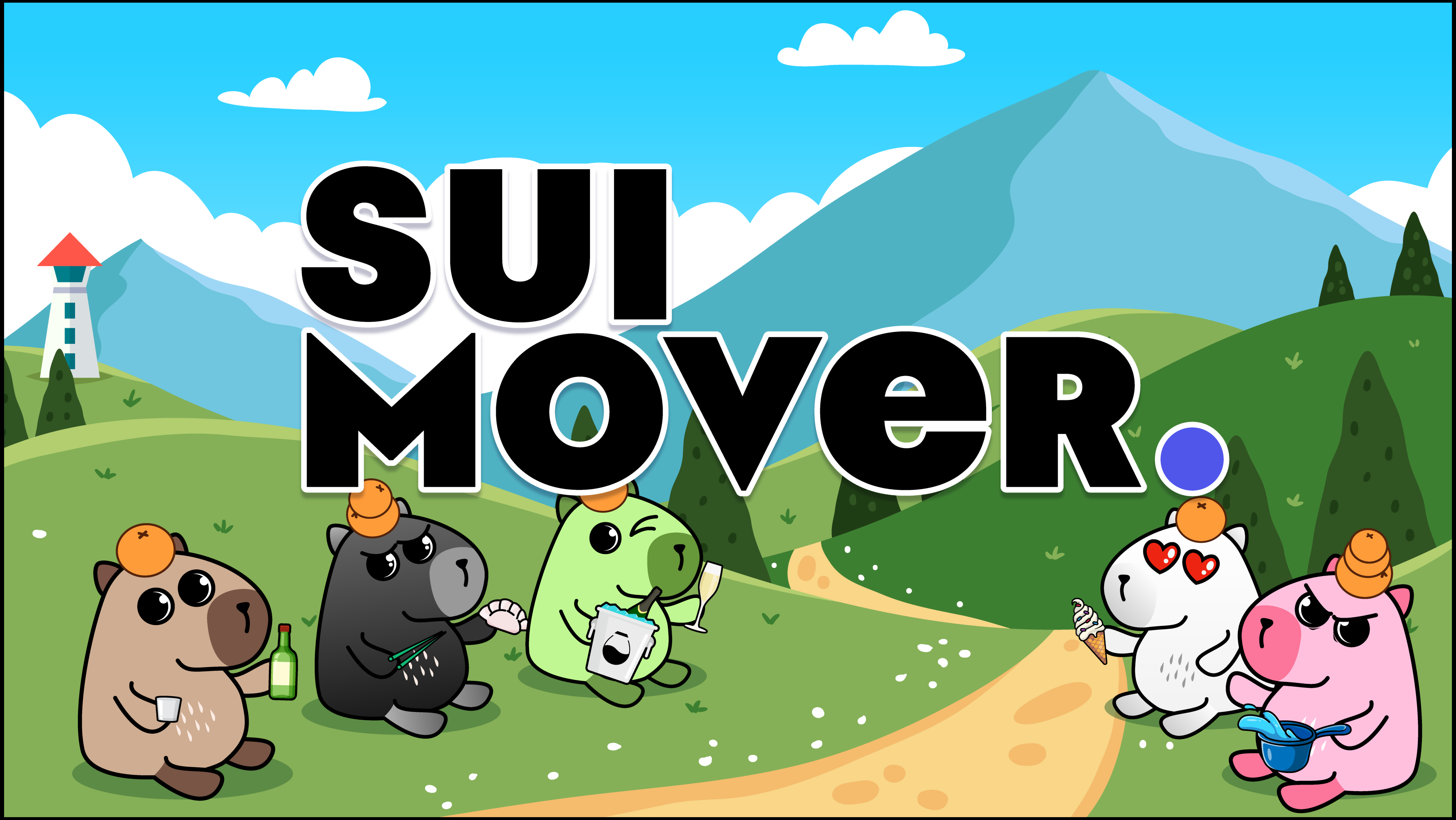


```
/// Insert a `key` into self.
/// Aborts if `key` is already present in `self`.
public fun insert<K: copy + drop>(self: &mut VecSet<K>, key: K) {
    assert!(!self.contains(key: &key), EKeyAlreadyExists);
    self.contents.push_back(e: key)
}

/// Remove the entry `key` from self. Aborts if `key` is not present in `self`.
public fun remove<K: copy + drop>(self: &mut VecSet<K>, key: &K) {
    let idx: u64 = get_idx(self: self, key: key);
    self.contents.remove(i: idx);
}

/// Return true if `self` contains an entry for `key`, false otherwise
public fun contains<K: copy + drop>(self: &VecSet<K>, key: &K): bool {
    get_idx_opt(self: self, key: key).is_some()
}
```

物件表現

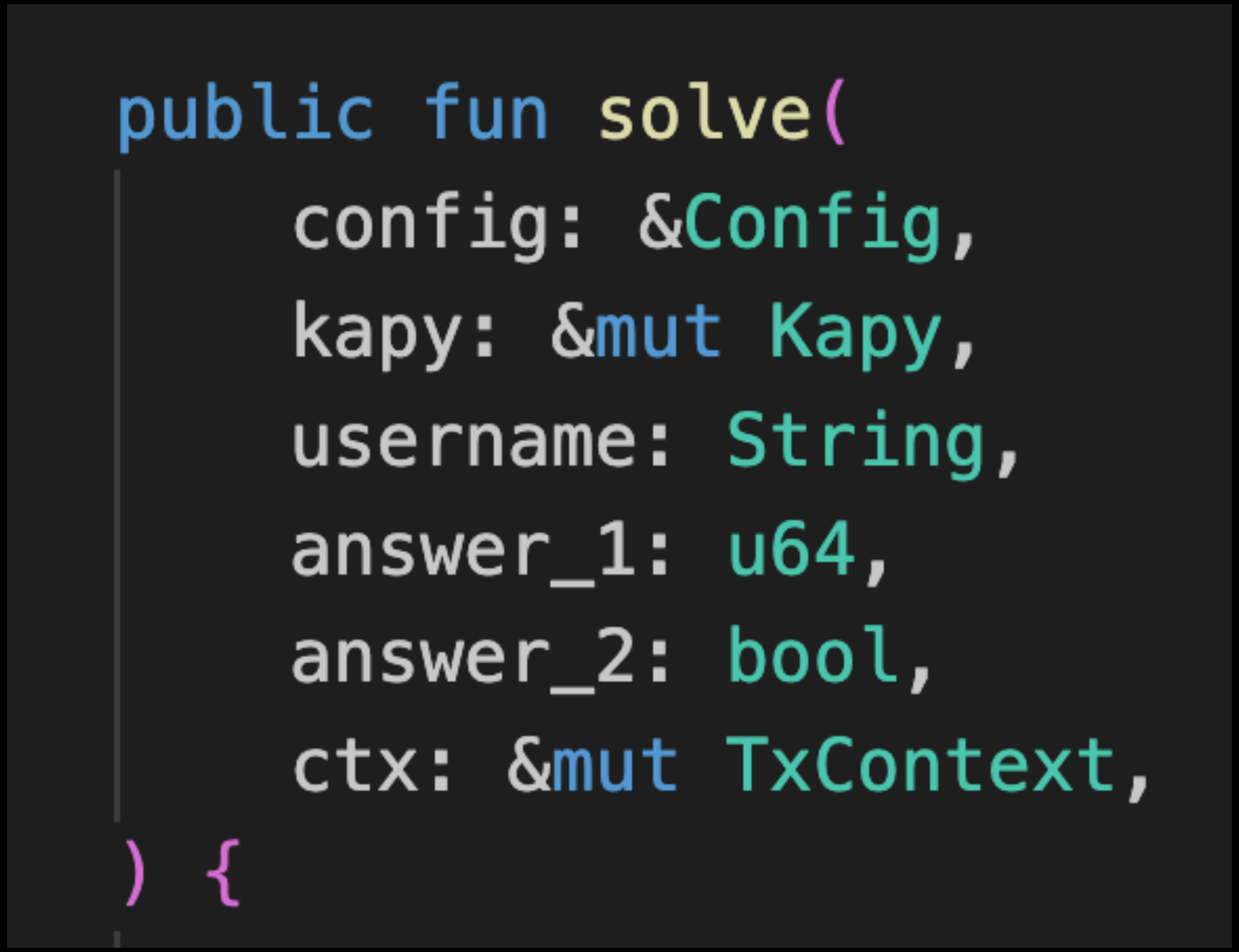
- 可以讓物件(有key的)有metadata
  - name - A name for the object.
  - description - A description for the object.
  - link - A link to the object to use in an application.
  - image_url - A URL or a blob with the image for the object.
  - thumbnail_url - A URL to a smaller image to use in wallets, explorers…
  - project_url - A link to a website associated with the object or creator.
  - creator - A string that indicates the object creator.

Sui Mover Kapy!



Exercise 1

Basic Types and Operators



```
public fun solve(
    config: &Config,
    kapy: &mut Kapy,
    username: String,
    answer_1: u64,
    answer_2: bool,
    ctx: &mut TxContext,
) {
```

Exercise 1

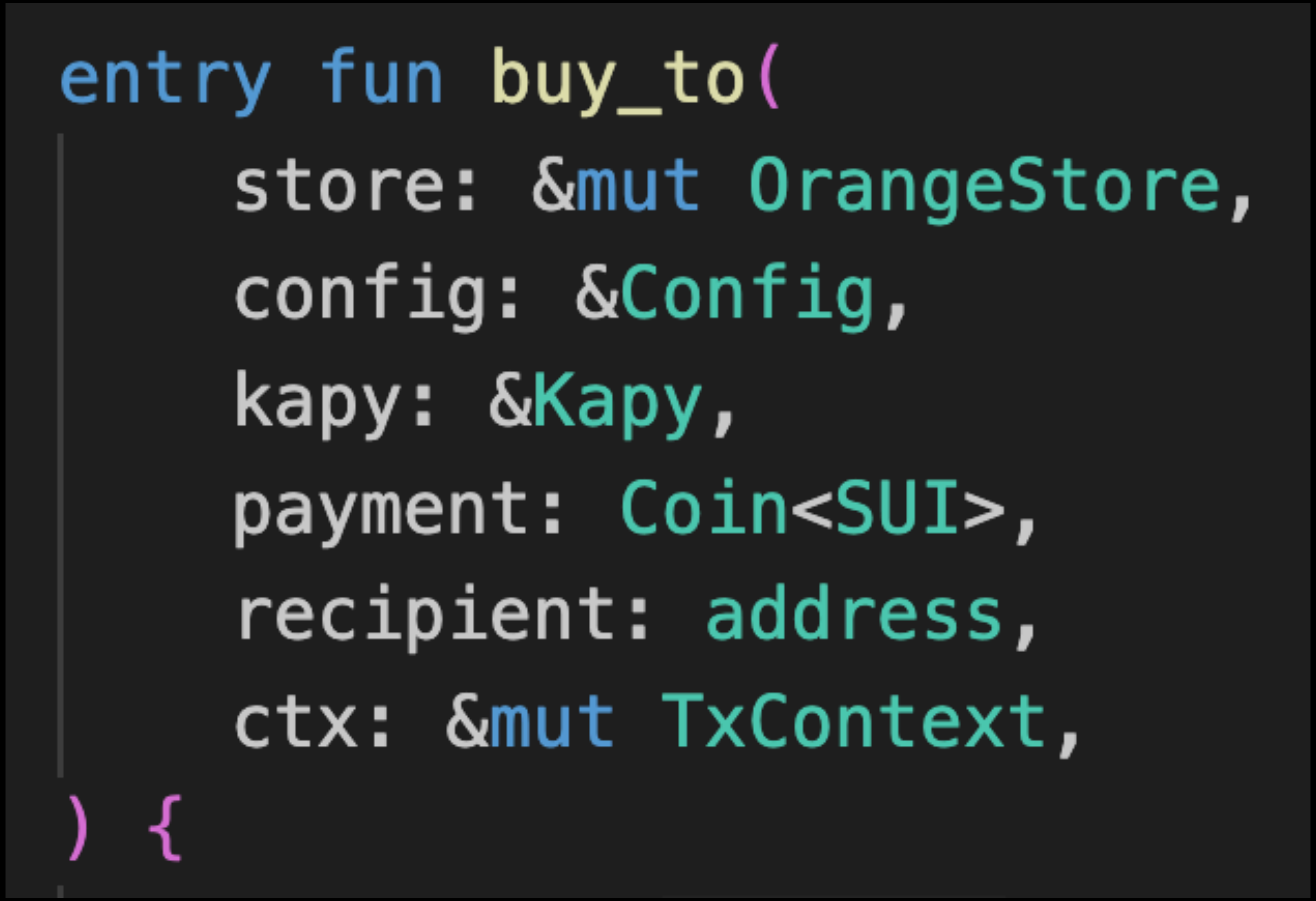
Submit with CLI

- sui client switch --address [kapy-owner-account] --env mainnet
- sui client call --package [package-id-of-exercise-1] --module exercise_1 --function solve --args [config-id] [your-kapy-id] [username] [some-u64-number] [true | false]

- PS: check your Kapy's change after submit successfully!

Exercise 2

Buy orange and put it on Kapy



```
entry fun buy_to(
    store: &mut OrangeStore,
    config: &Config,
    kapy: &Kapy,
    payment: Coin<SUI>,
    recipient: address,
    ctx: &mut TxContext,
) {
```

Exercise 2

Submit with CLI

- sui client switch --address [kapy-owner-account] --env mainnet
- split your SUI coin to required amount (how?)
- sui client call --package [package-id-of-exercise-2] --module exercise_2 --function buy --args [store-id] [config-id] [your-kapy-id] [sui-coin-id] [recipient-address]
- put the orange on the your Kapy's head (how?)

- PS: check your Kapy's change after submit successfully!