簡易fomo3D

1072928 江毓晴
1072911 陳元娣

玩法介紹

Buy key

- 選擇隊伍，key中30%分給同隊隊友
- 5%給空投池，5%給遊戲方，55%在獎金池
- 剩餘5%為轉移用資金
- key price = 平均價格 + 調漲10%
- 秒數 +30sec
- 空投概率 +5%

簡介

- 一局 : 10分鐘 起跳
- 鑰匙 : 0.001ETH 起跳
- 當有人購買鑰匙，已加入這場遊戲的玩家
  且與其同隊將會獲得分紅
- 倒數結束時，最後購買鑰匙者為贏家，
  與其同隊者會獲得分紅
- 每次購買鑰匙會獲得一次抽獎機會

獲得空投

- 有人獲得空投，所有人概率重置

- 獲得金額 (ETH):

  0.001 ~ 0.01 – 空投池25%
  0.01x ~ 0.10 – 空投池50%
  0.1xx ~ – 空投池75%

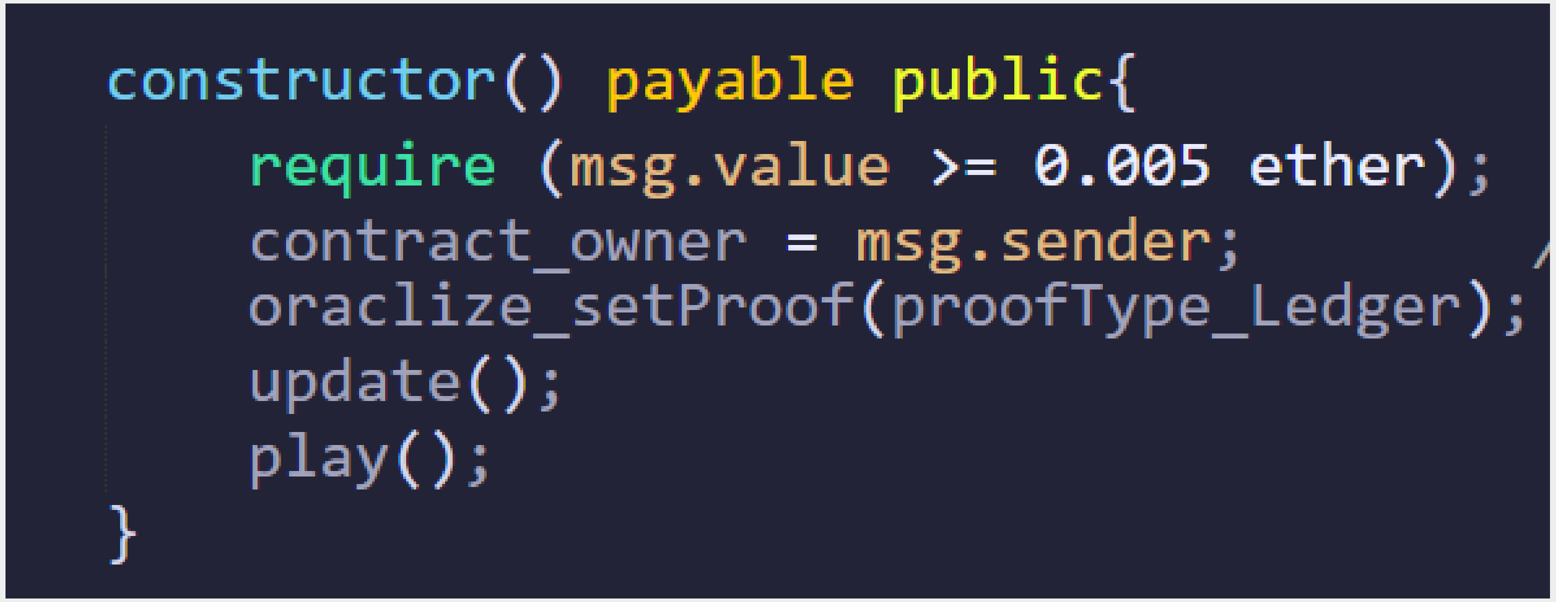
遊戲結束

- 當倒數結束，結算遊戲
- 贏家獲得獎金池的48%
- 2%給遊戲方，50%依照選擇隊伍分配
- 隊伍：

	留存獎金池	分予玩家
天使隊	10%	40%
惡魔隊	30%	20%

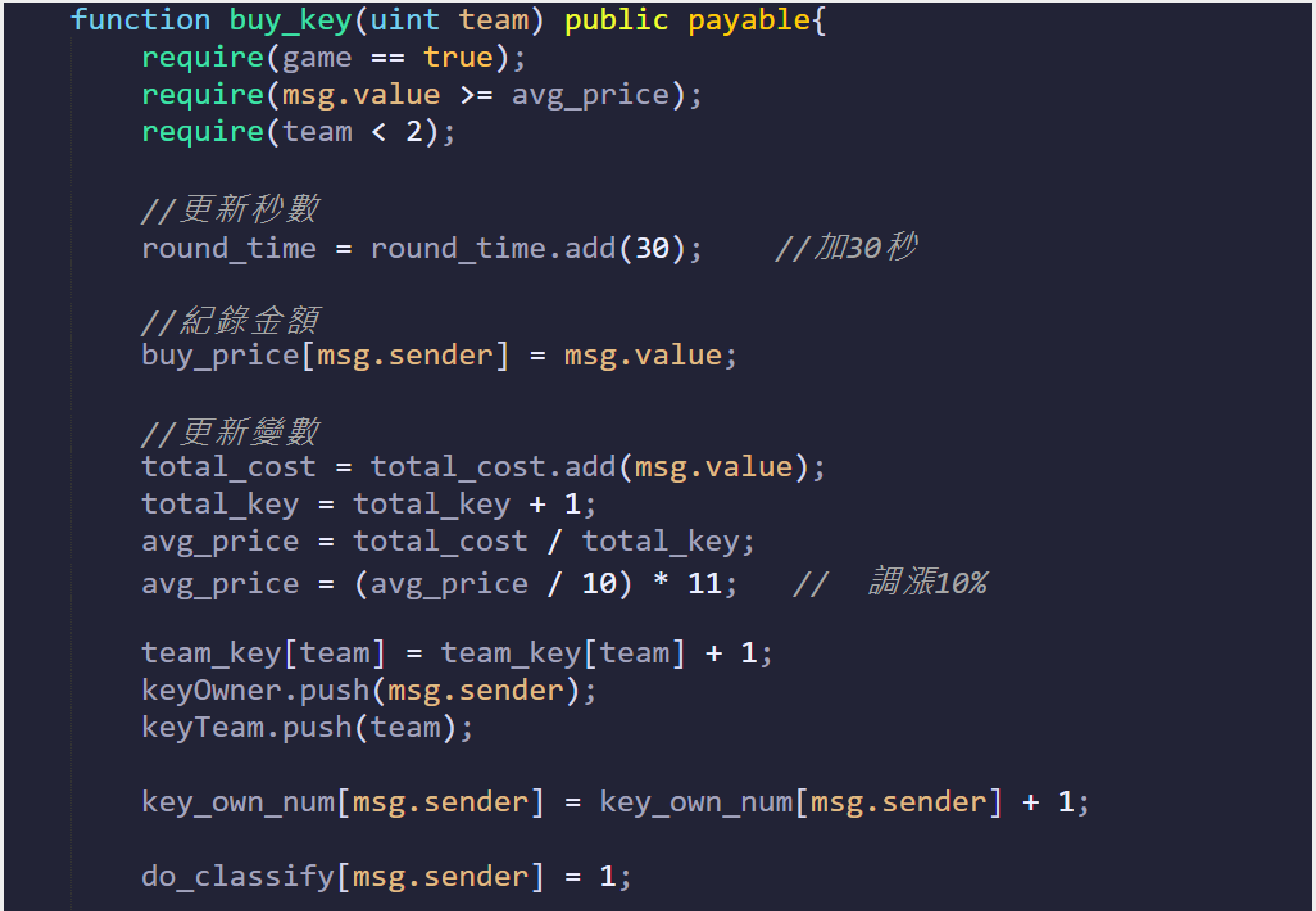
Code 說明

佈署時



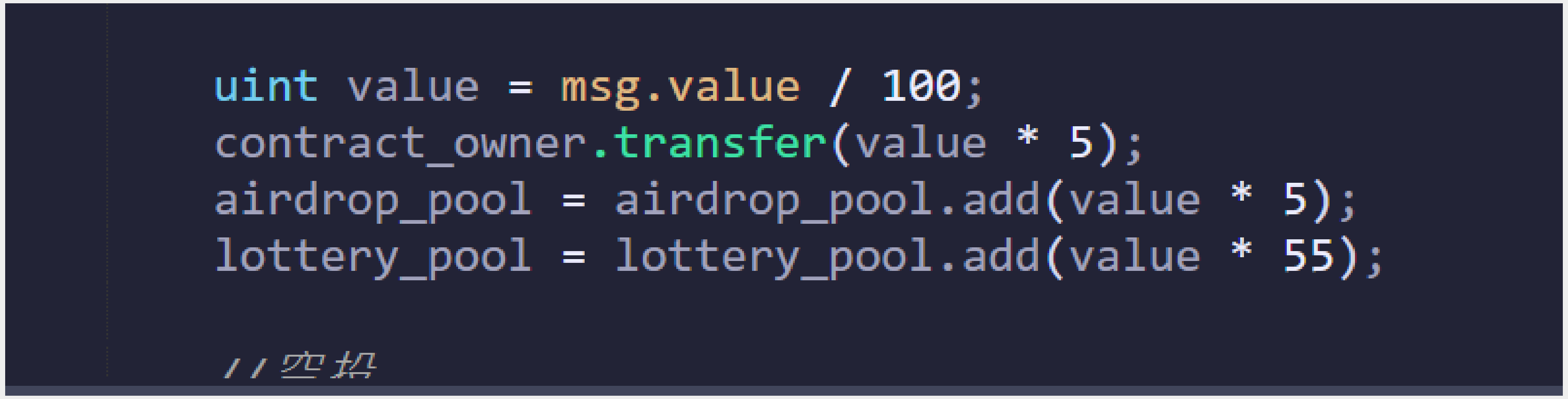
```
constructor() payable public{
    require (msg.value >= 0.005 ether);
    contract_owner = msg.sender;
    oraclize_setProof(proofType_Ledger);
    update();
    play();
}
```

購買鑰匙

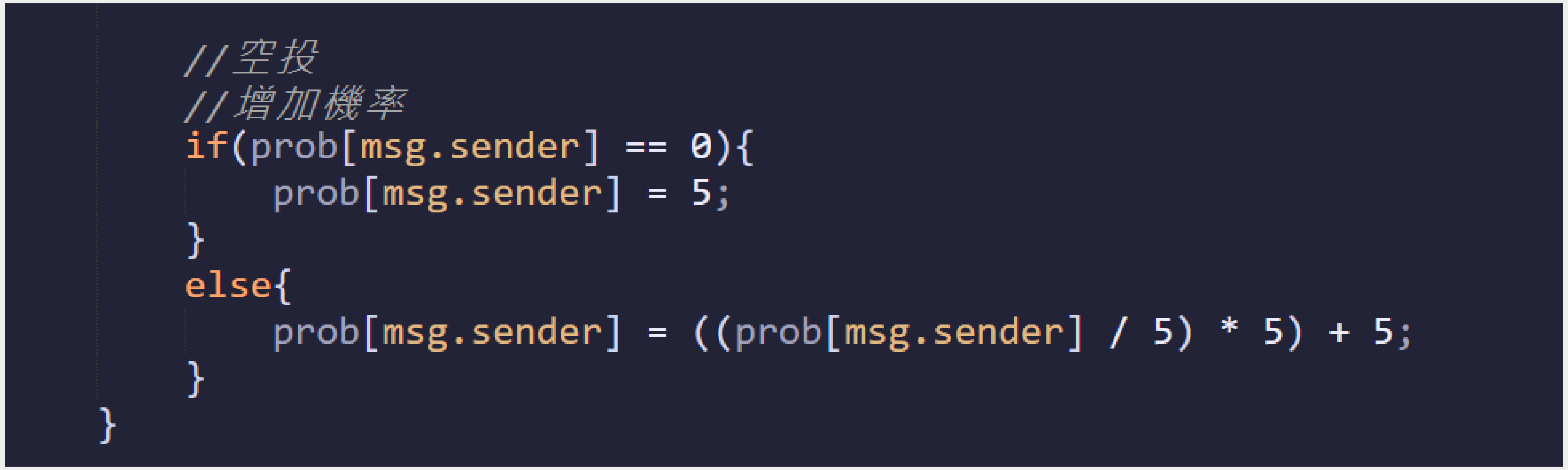


```
    function buy_key(uint team) public payable{
        require(game == true);
        require(msg.value >= avg_price);
        require(team < 2);

        //更新秒數
        round_time = round_time.add(30);    //加30秒

        //紀錄金額
        buy_price[msg.sender] = msg.value;

        //更新變數
        total_cost = total_cost.add(msg.value);
        total_key = total_key + 1;
        avg_price = total_cost / total_key;
        avg_price = (avg_price / 10) * 11;    //   調漲10%

        team_key[team] = team_key[team] + 1;
        keyOwner.push(msg.sender);
        keyTeam.push(team);

        key_own_num[msg.sender] = key_own_num[msg.sender] + 1;

        do_classify[msg.sender] = 1;
```

購買鑰匙

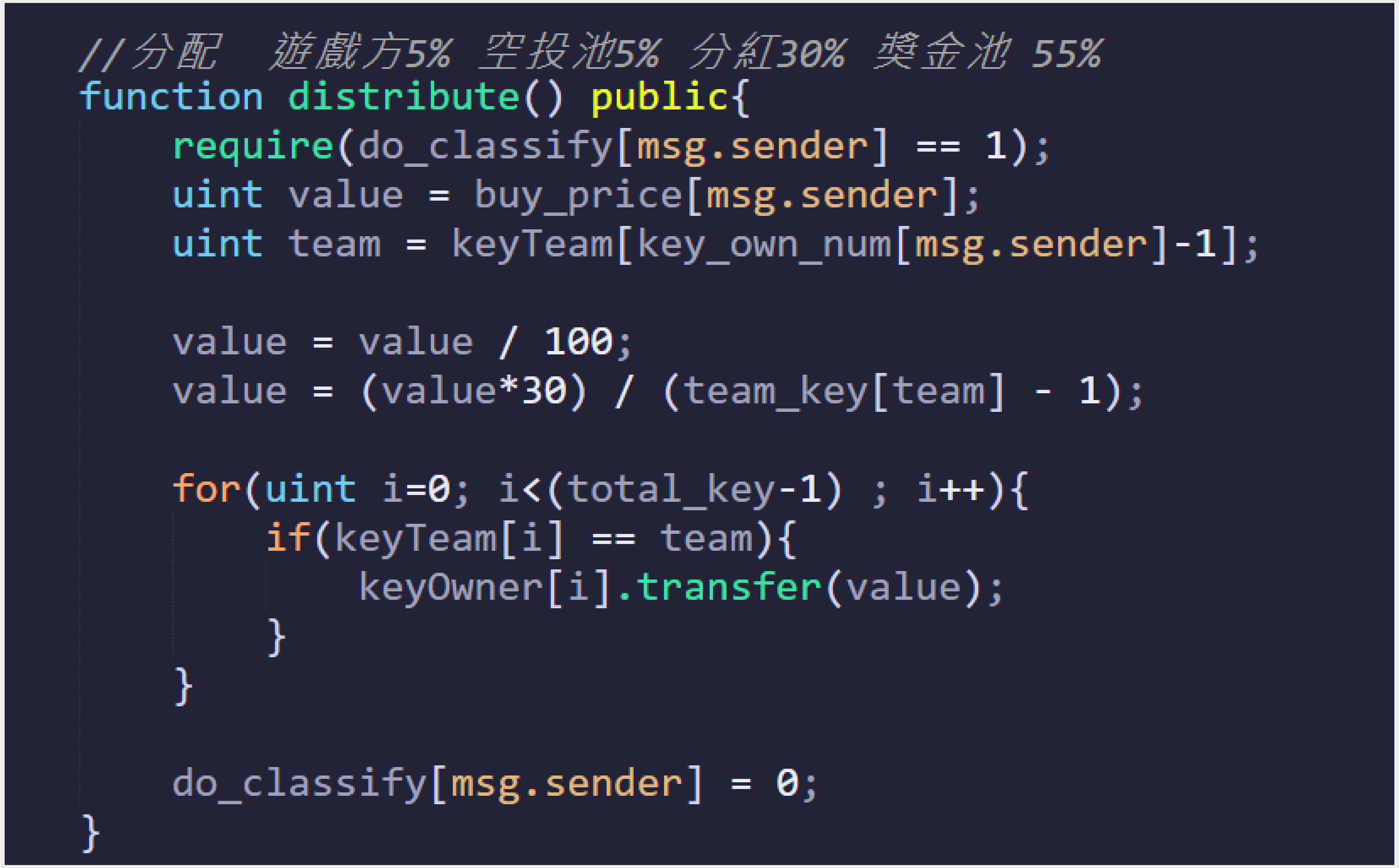


```
uint value = msg.value / 100;
contract_owner.transfer(value * 5);
airdrop_pool = airdrop_pool.add(value * 5);
lottery_pool = lottery_pool.add(value * 55);

//空投
```



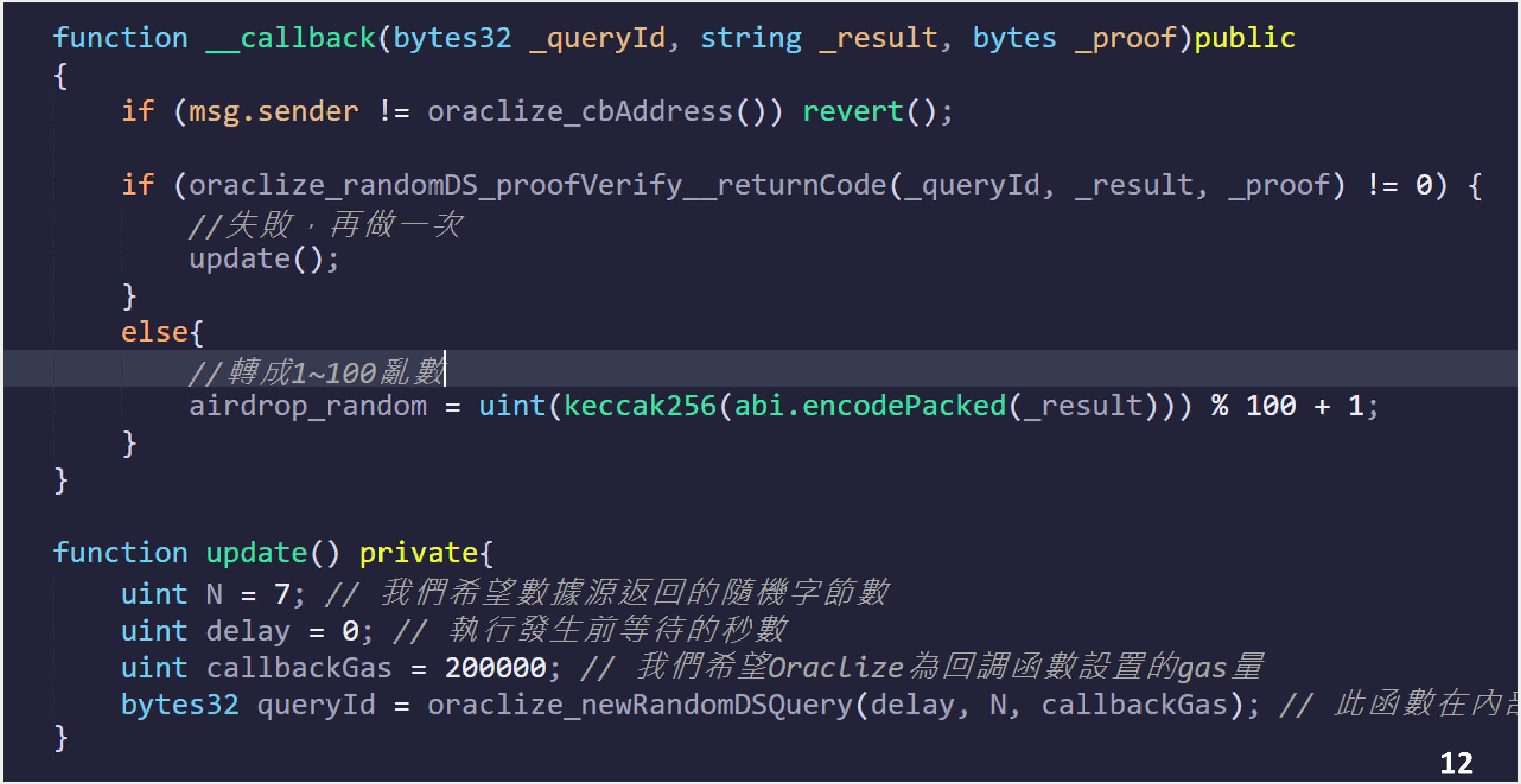
```
    //空投
    //增加機率
    if(prob[msg.sender] == 0){
        prob[msg.sender] = 5;
    }
    else{
        prob[msg.sender] = ((prob[msg.sender] / 5) * 5) + 5;
    }
}
```

分紅



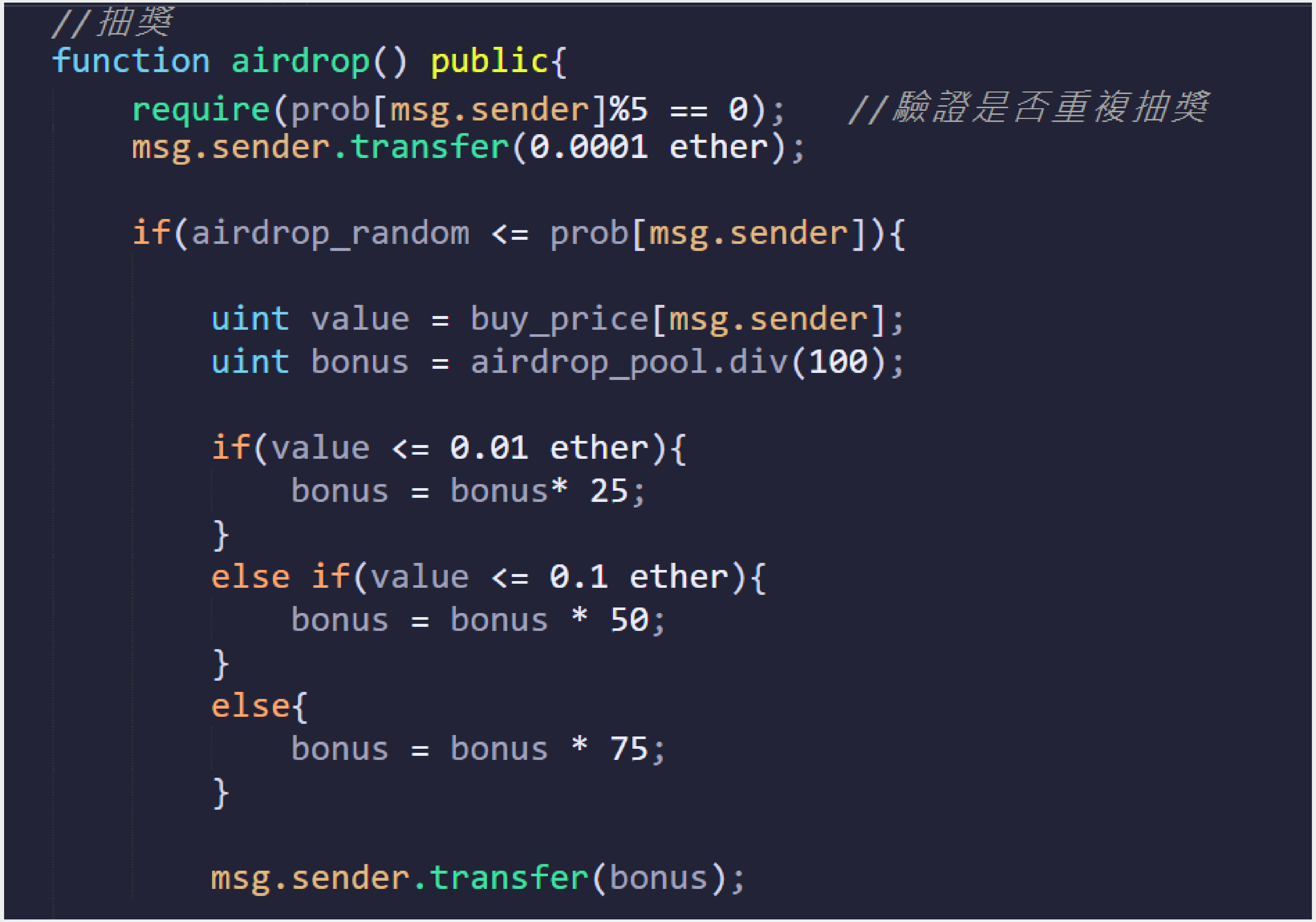
```
//分配  遊戲方5% 空投池5% 分紅30% 獎金池 55%
function distribute() public{
    require(do_classify[msg.sender] == 1);
    uint value = buy_price[msg.sender];
    uint team = keyTeam[key_own_num[msg.sender]-1];

    value = value / 100;
    value = (value*30) / (team_key[team] - 1);

    for(uint i=0; i<(total_key-1) ; i++){
        if(keyTeam[i] == team){
            keyOwner[i].transfer(value);
        }
    }

    do_classify[msg.sender] = 0;
}
```

取得亂數



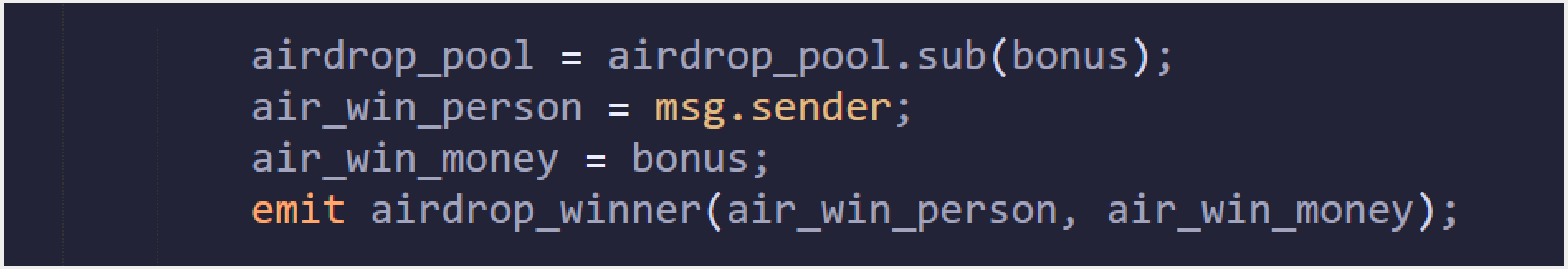
```
function __callback(bytes32 _queryId, string _result, bytes _proof)public
{
    if (msg.sender != oraclize_cbAddress()) revert();

    if (oraclize_randomDS_proofVerify__returnCode(_queryId, _result, _proof) != 0) {
        //失敗，再做一次
        update();
    }
    else{
        //轉成1~100亂數
        airdrop_random = uint(keccak256(abi.encodePacked(_result))) % 100 + 1;
    }
}

function update() private{
    uint N = 7; // 我們希望數據源返回的隨機字節數
    uint delay = 0; // 執行發生前等待的秒數
    uint callbackGas = 200000; // 我們希望Oraclize為回調函數設置的gas量
    bytes32 queryId = oraclize_newRandomDSQuery(delay, N, callbackGas); // 此函數在內
}
```

抽空投

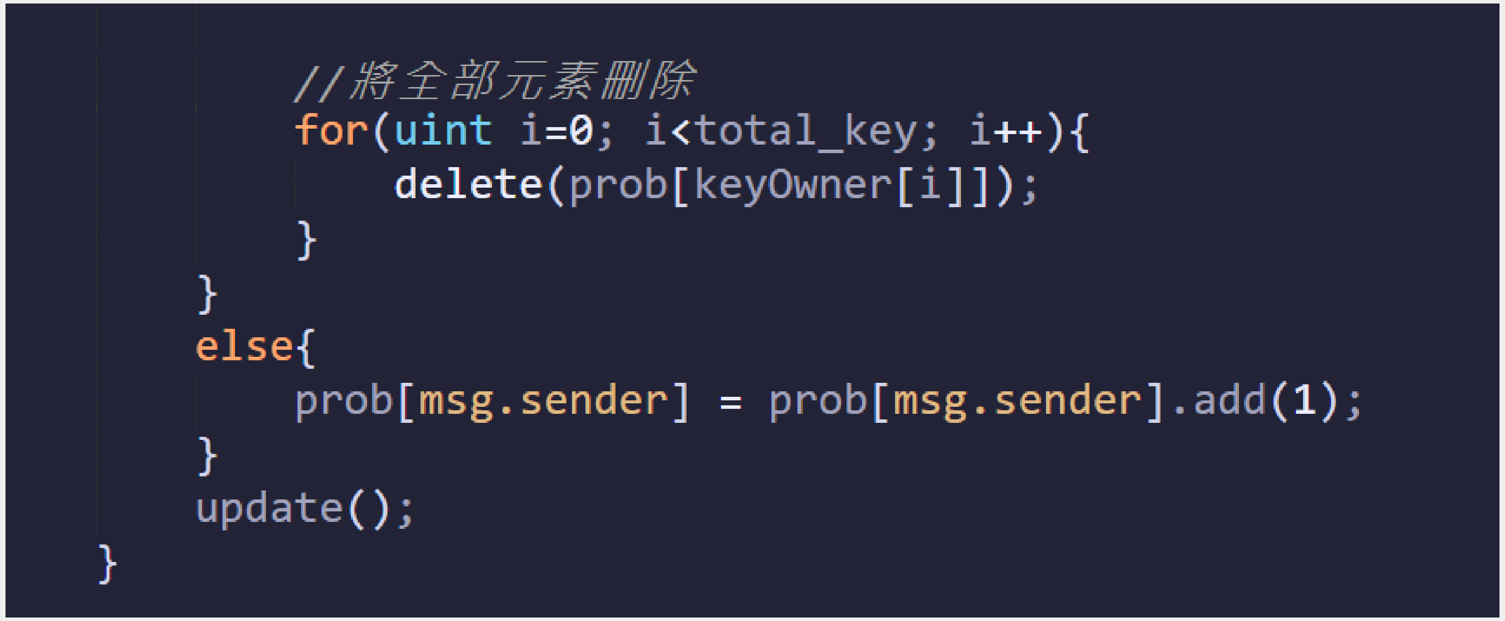


```
//抽獎
function airdrop() public{
    require(prob[msg.sender]%5 == 0);   //驗證是否重複抽獎
    msg.sender.transfer(0.0001 ether);

    if(airdrop_random <= prob[msg.sender]){

        uint value = buy_price[msg.sender];
        uint bonus = airdrop_pool.div(100);

        if(value <= 0.01 ether){
            bonus = bonus* 25;
        }
        else if(value <= 0.1 ether){
            bonus = bonus * 50;
        }
        else{
            bonus = bonus * 75;
        }

        msg.sender.transfer(bonus);
```

抽空投

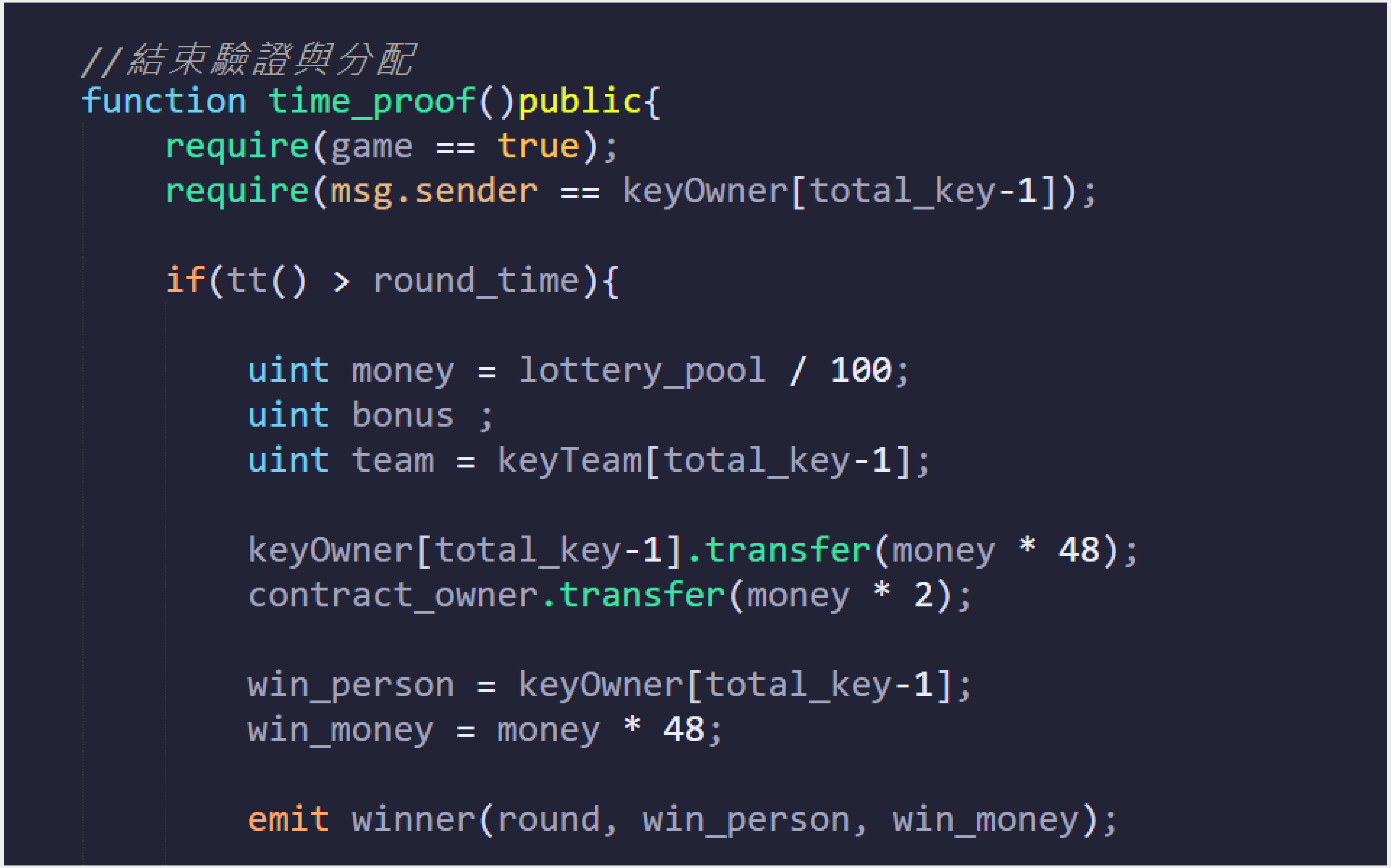


```
        airdrop_pool = airdrop_pool.sub(bonus);
        air_win_person = msg.sender;
        air_win_money = bonus;
        emit airdrop_winner(air_win_person, air_win_money);
```



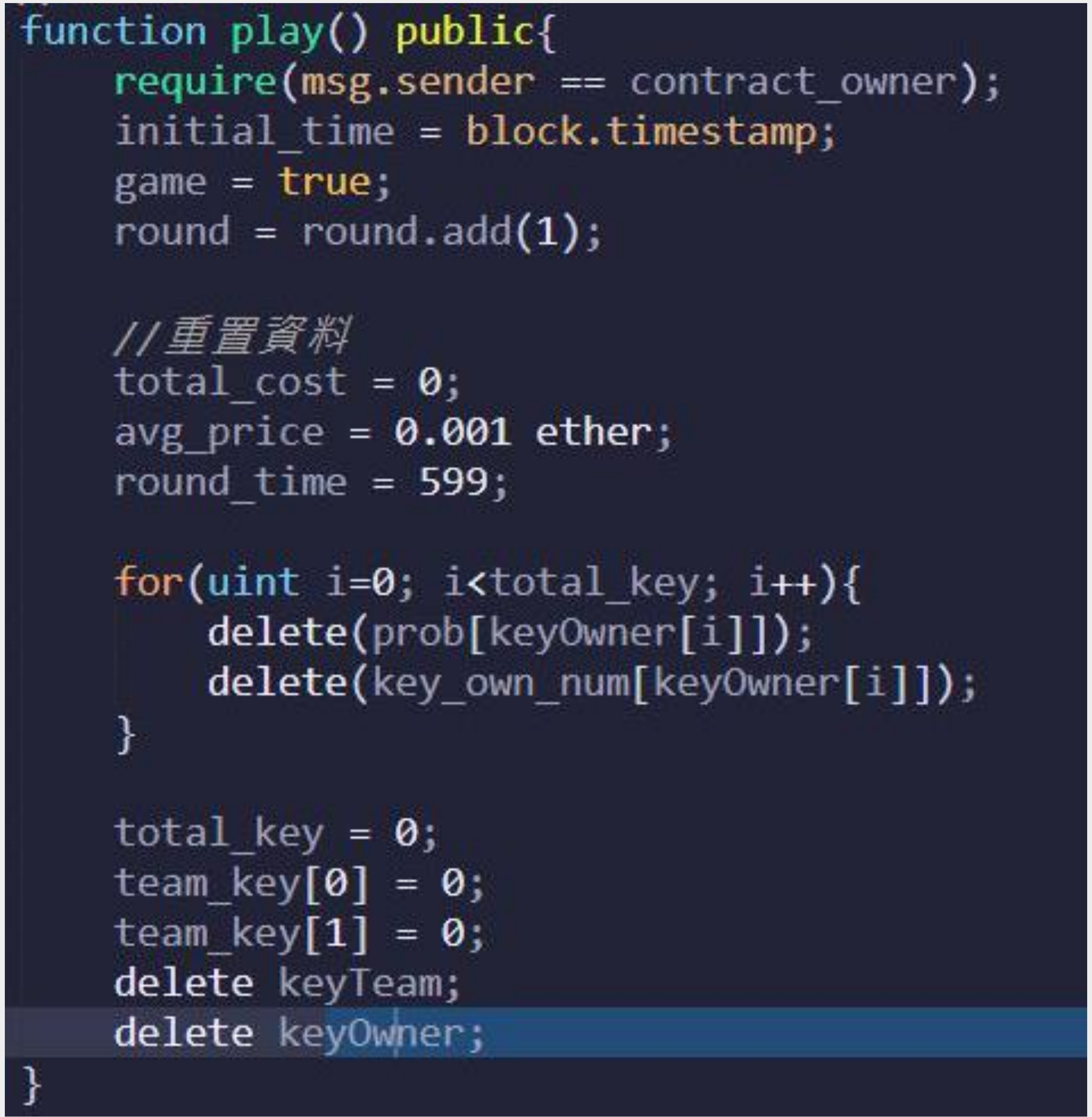
```
            //將全部元素刪除
            for(uint i=0; i<total_key; i++){
                delete(prob[keyOwner[i]]);
            }
        }
        else{
            prob[msg.sender] = prob[msg.sender].add(1);
        }
        update();
    }
```

驗證時間並結算



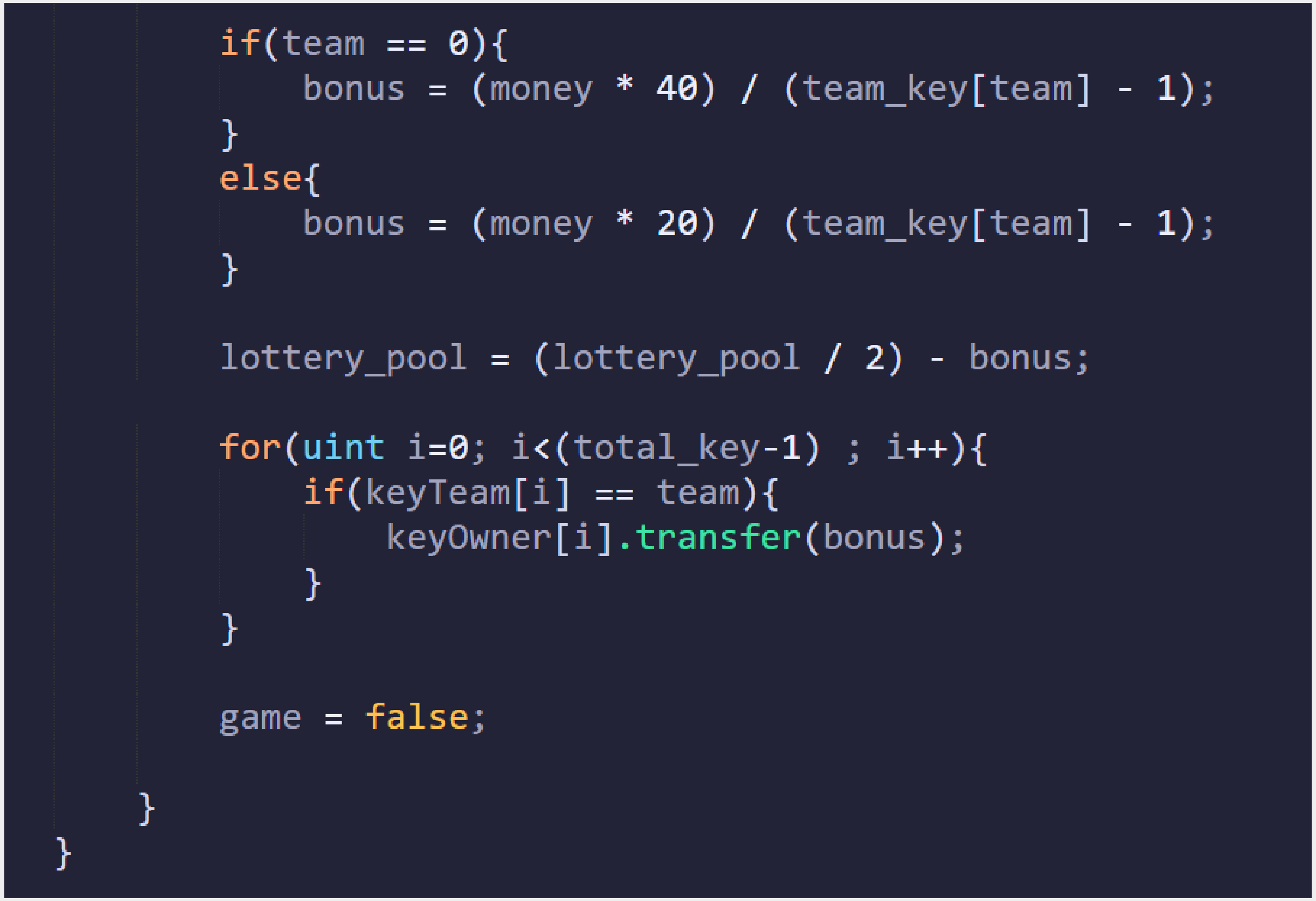
```
//結束驗證與分配
function time_proof()public{
    require(game == true);
    require(msg.sender == keyOwner[total_key-1]);

    if(tt() > round_time){

        uint money = lottery_pool / 100;
        uint bonus ;
        uint team = keyTeam[total_key-1];

        keyOwner[total_key-1].transfer(money * 48);
        contract_owner.transfer(money * 2);

        win_person = keyOwner[total_key-1];
        win_money = money * 48;

        emit winner(round, win_person, win_money);
```

新的一局



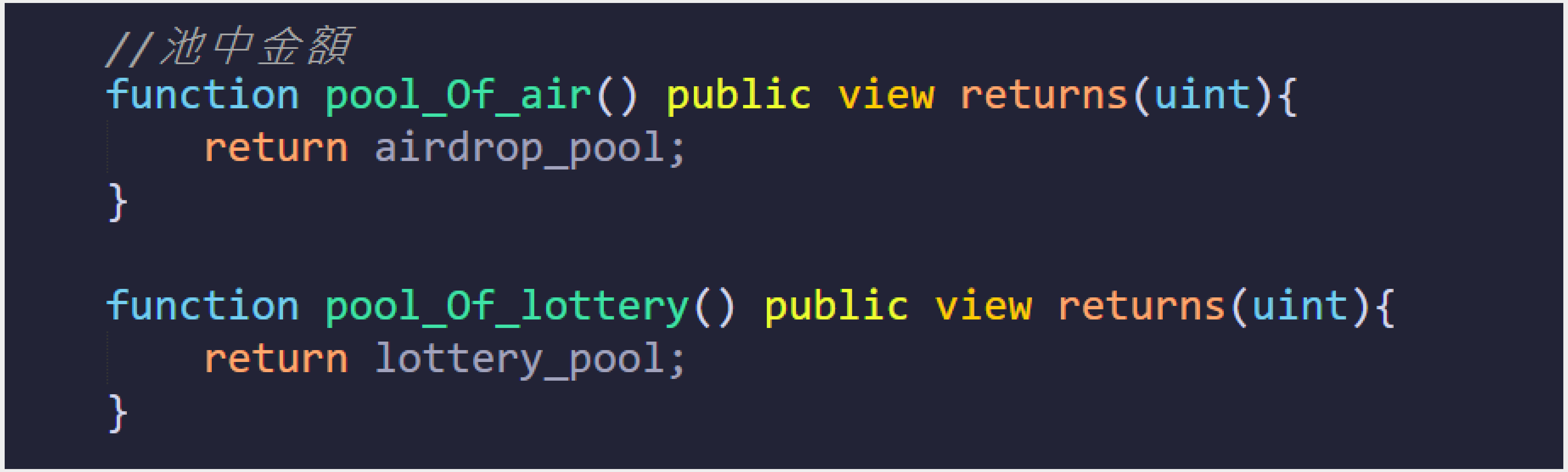
```
function play() public{
    require(msg.sender == contract_owner);
    initial_time = block.timestamp;
    game = true;
    round = round.add(1);

    //重置資料
    total_cost = 0;
    avg_price = 0.001 ether;
    round_time = 599;

    for(uint i=0; i<total_key; i++){
        delete(prob[keyOwner[i]]);
        delete(key_own_num[keyOwner[i]]);
    }

    total_key = 0;
    team_key[0] = 0;
    team_key[1] = 0;
    delete keyTeam;
    delete keyOwner;
}
```

驗證時間並結算



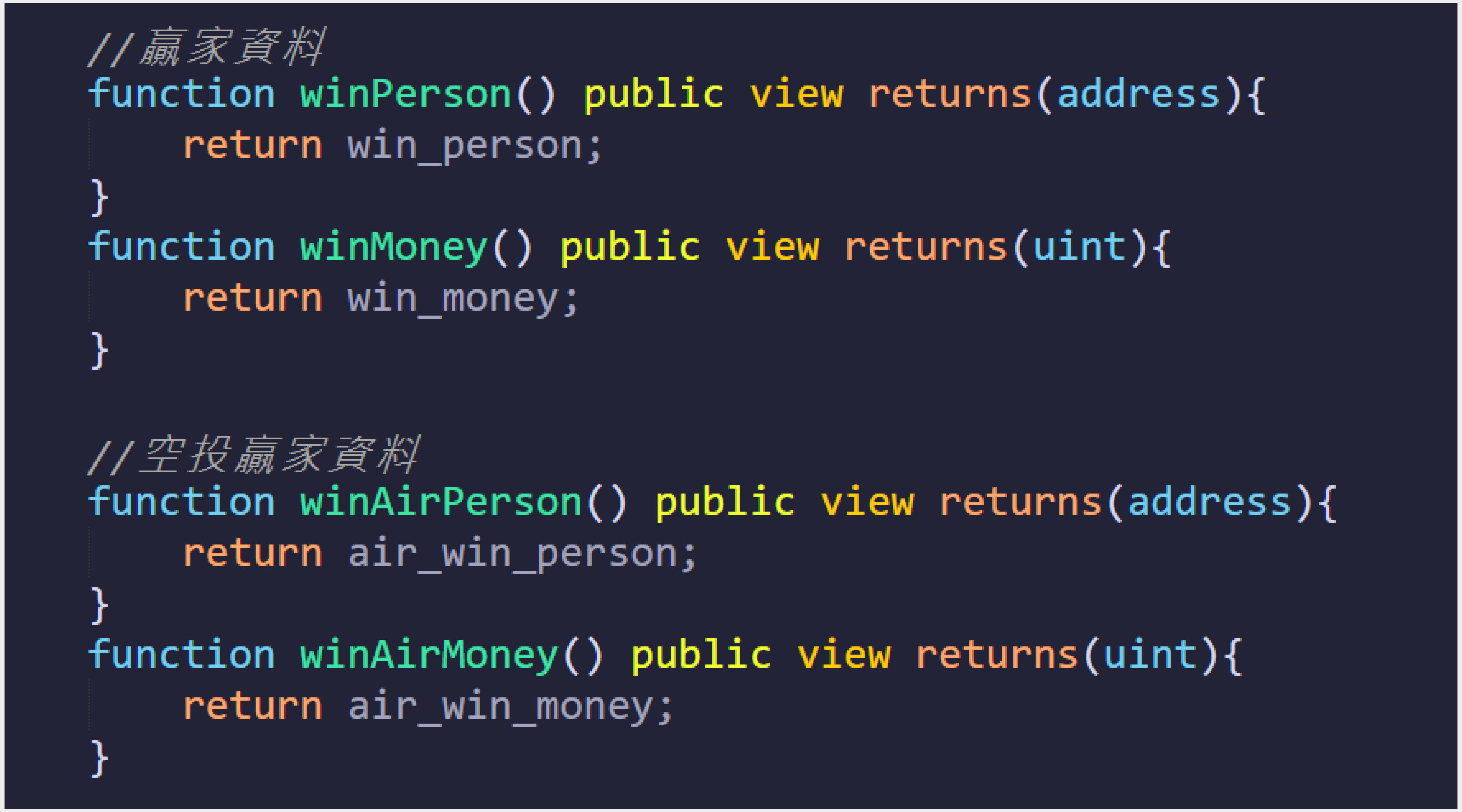
```
        if(team == 0){
            bonus = (money * 40) / (team_key[team] - 1);
        }
        else{
            bonus = (money * 20) / (team_key[team] - 1);
        }

        lottery_pool = (lottery_pool / 2) - bonus;

        for(uint i=0; i<(total_key-1) ; i++){
            if(keyTeam[i] == team){
                keyOwner[i].transfer(bonus);
            }
        }

        game = false;

    }
}
```

查看資料



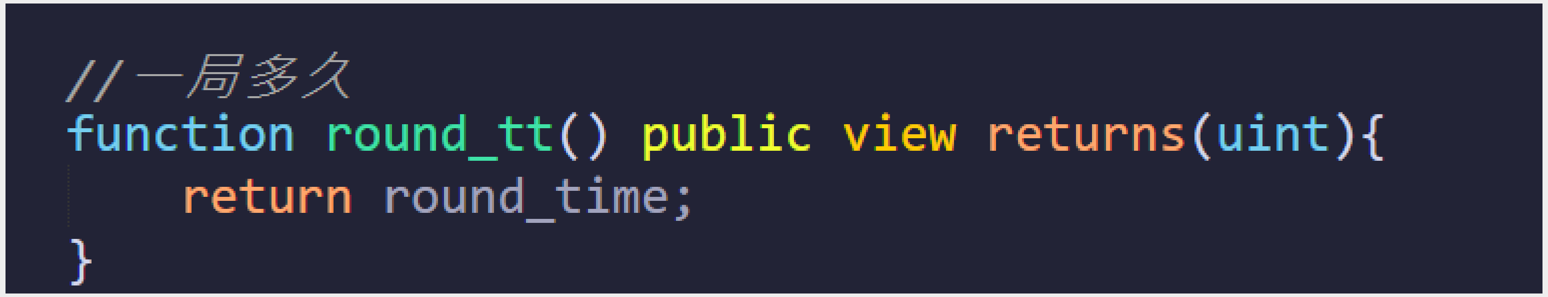
```
//池中金額
function pool_Of_air() public view returns(uint){
    return airdrop_pool;
}

function pool_Of_lottery() public view returns(uint){
    return lottery_pool;
}
```

遊戲結束

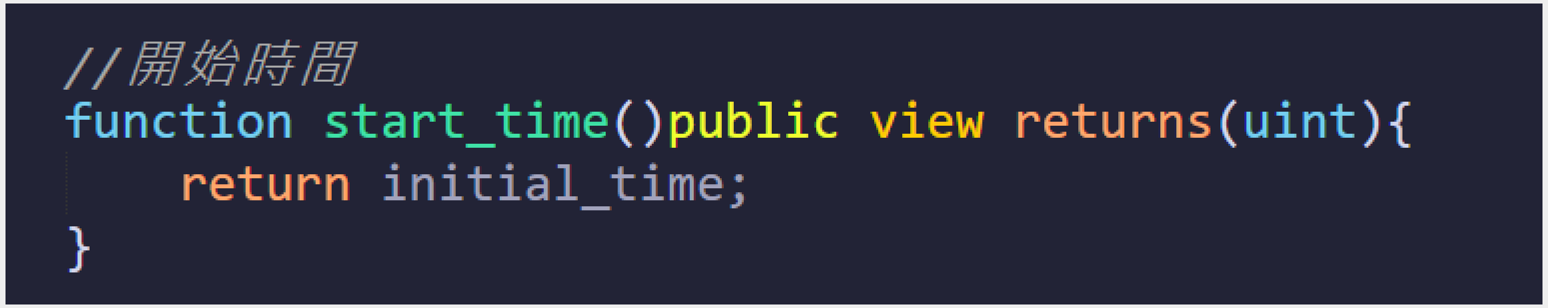


```
//贏家資料
function winPerson() public view returns(address){
    return win_person;
}
function winMoney() public view returns(uint){
    return win_money;
}

//空投贏家資料
function winAirPerson() public view returns(address){
    return air_win_person;
}
function winAirMoney() public view returns(uint){
    return air_win_money;
}
```

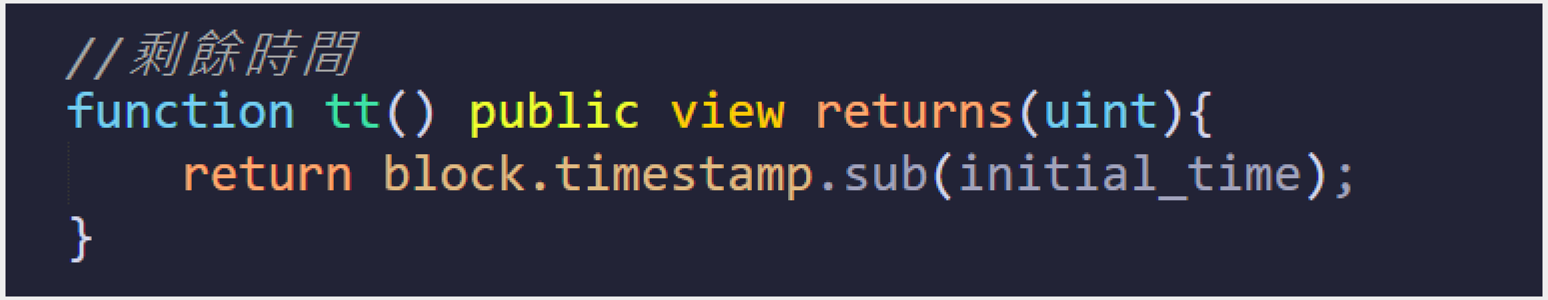
查看資料



```
//一局多久
function round_tt() public view returns(uint){
    return round_time;
}
```

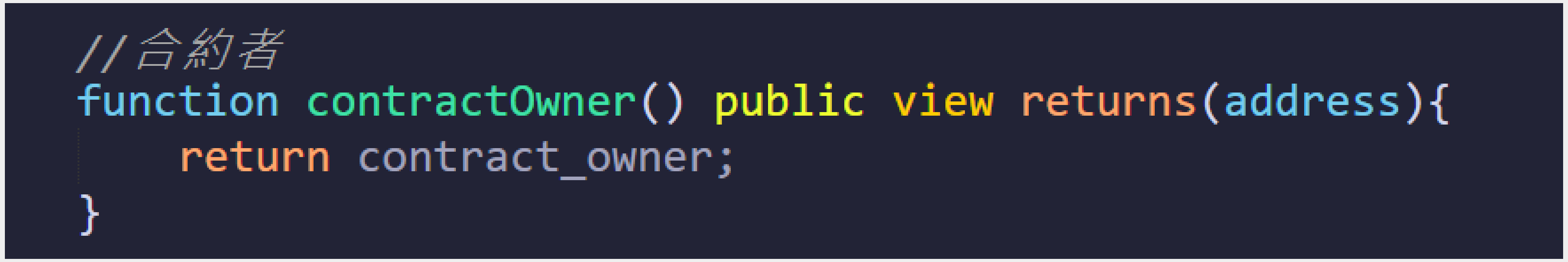


```
//開始時間
function start_time()public view returns(uint){
    return initial_time;
}
```

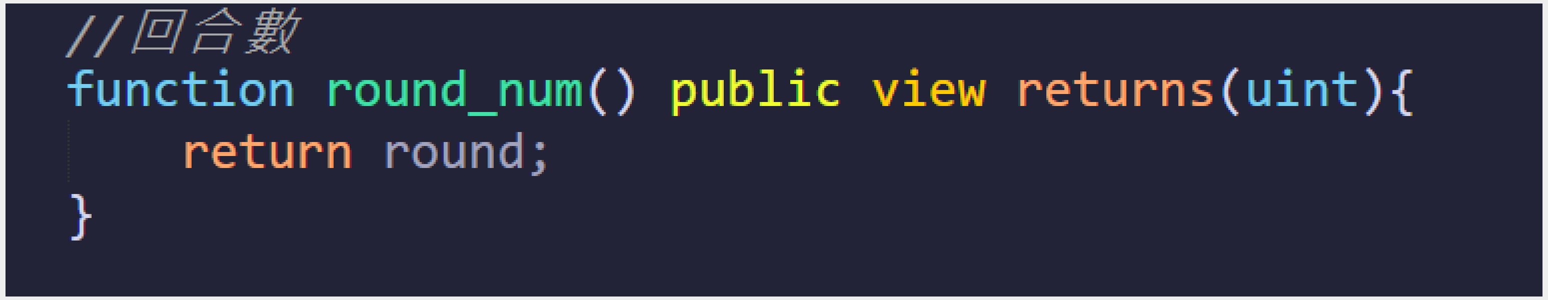


```
//剩餘時間
function tt() public view returns(uint){
    return block.timestamp.sub(initial_time);
}
```

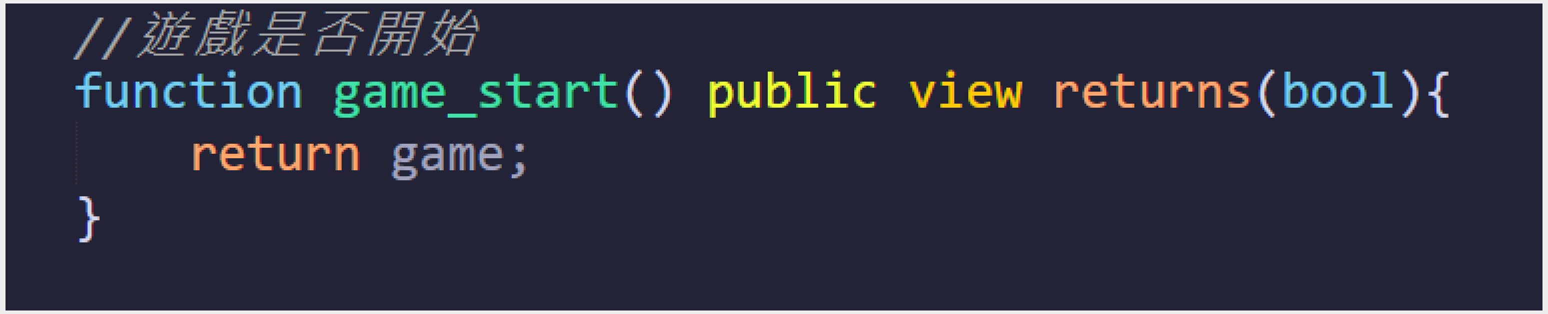
查看資料



```
//合約者
function contractOwner() public view returns(address){
    return contract_owner;
}
```

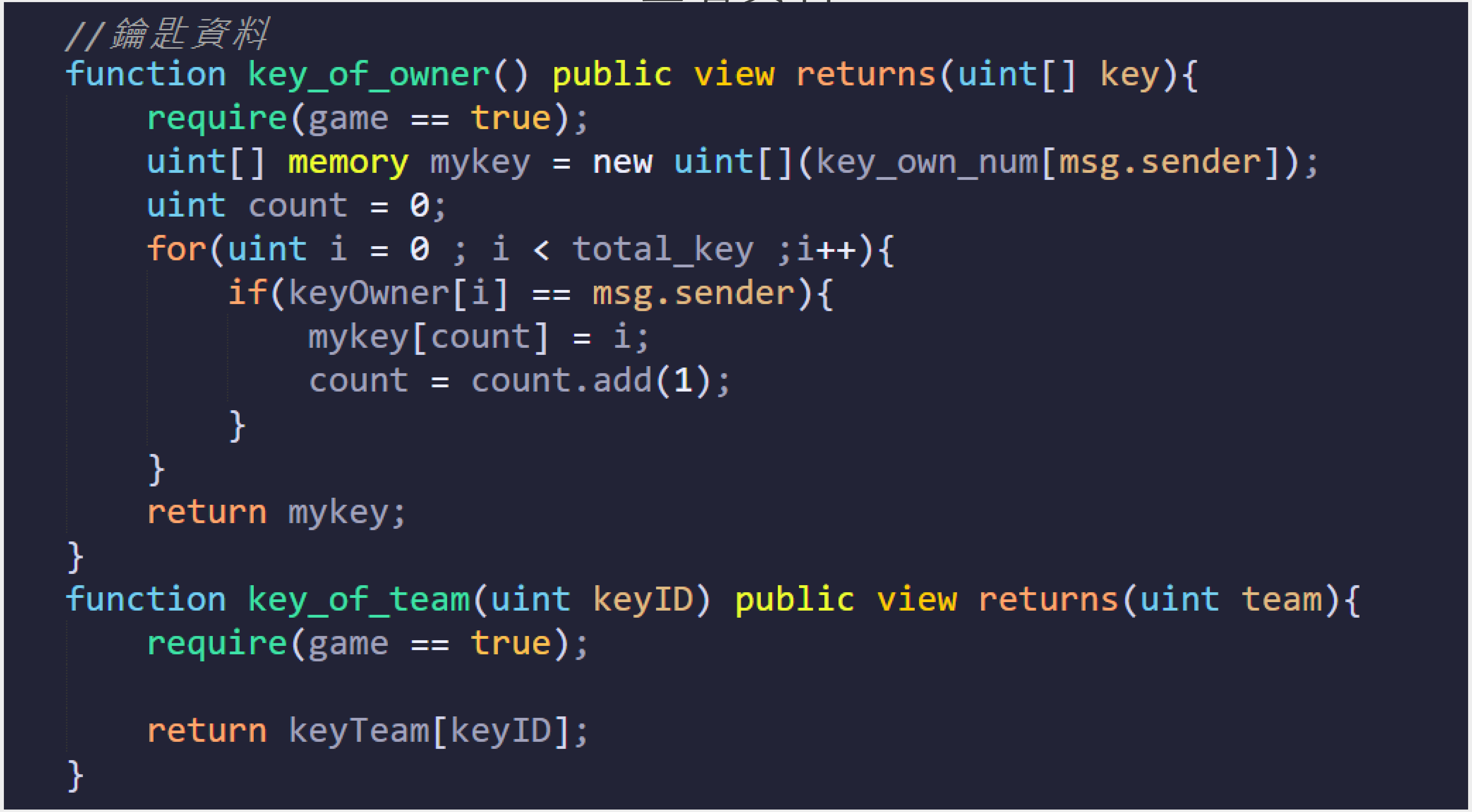


```
//回合數
function round_num() public view returns(uint){
    return round;
}
```



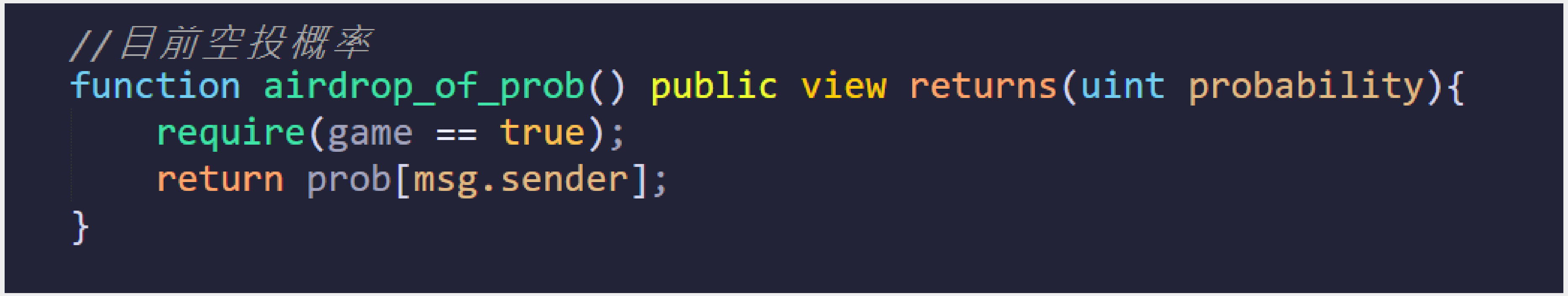
```
//遊戲是否開始
function game_start() public view returns(bool){
    return game;
}
```

查看資料

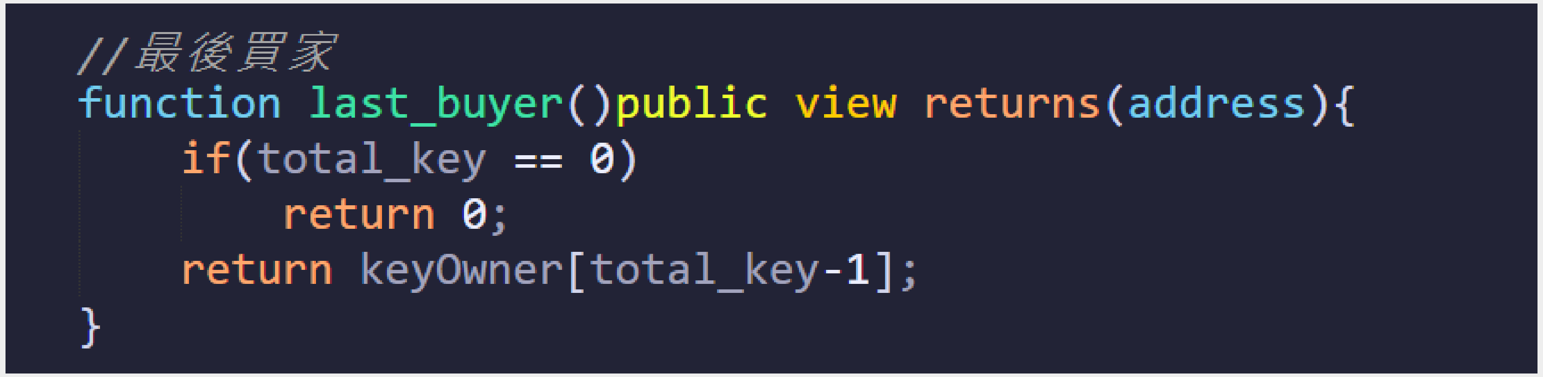


```
//鑰匙資料
function key_of_owner() public view returns(uint[] key){
    require(game == true);
    uint[] memory mykey = new uint[](key_own_num[msg.sender]);
    uint count = 0;
    for(uint i = 0 ; i < total_key ;i++){
        if(keyOwner[i] == msg.sender){
            mykey[count] = i;
            count = count.add(1);
        }
    }
    return mykey;
}
function key_of_team(uint keyID) public view returns(uint team){
    require(game == true);

    return keyTeam[keyID];
}
```

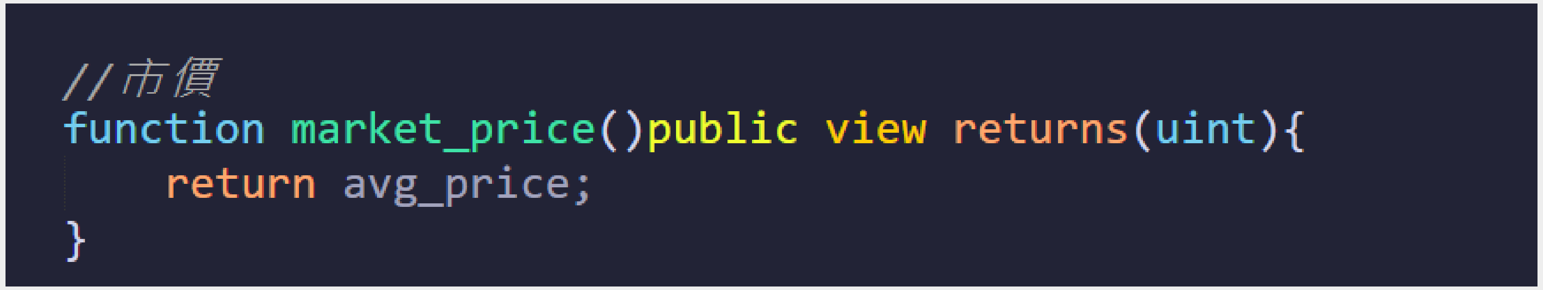
查看資料



```
//目前空投概率
function airdrop_of_prob() public view returns(uint probability){
    require(game == true);
    return prob[msg.sender];
}
```

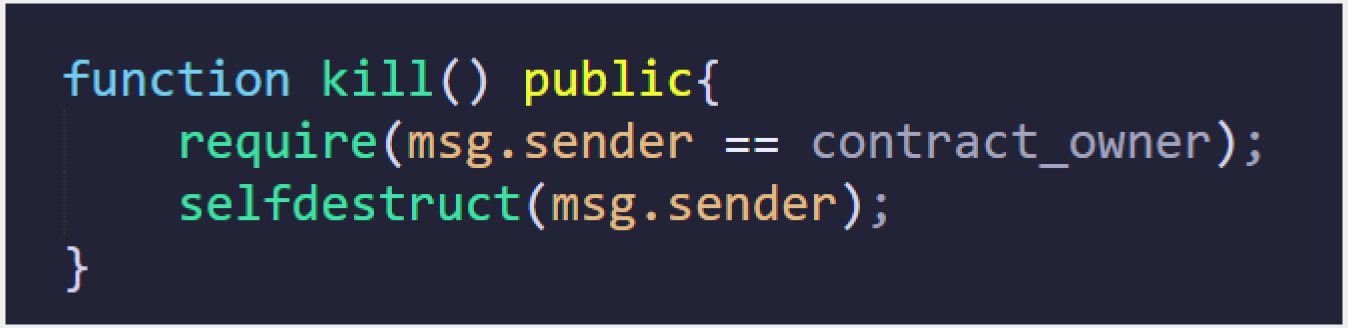


```
//最後買家
function last_buyer()public view returns(address){
    if(total_key == 0)
        return 0;
    return keyOwner[total_key-1];
}
```



```
//市價
function market_price()public view returns(uint){
    return avg_price;
}
```

合約自毀



```
function kill() public{
    require(msg.sender == contract_owner);
    selfdestruct(msg.sender);
}
```

網頁Demo

THANK YOU