Crash Course with MoonBit

- 基础类型
- 自定义数据类型
- 常用数据类型
- 项目与模块
- 测试与调试

基础类型

Primitive

- `Bool` : `true` `false`
- `Int` / `UInt` / `Int64` / `UInt64` / `BigInt`
  - `(1 : Int64)`, `1L`, `1UL`, `1N`, …
- `Float` / `Double`
  - `(1.0 : Float)`, `1.0`
- Byte / Bytes
  - `b'\x01'`, `b"Hello World!"`
- Char / String
  - `'a'`, `"asdf"`

Tuple

- `let tuple : (A, B, C) = (a, b, c)` with `a : A`, `b : B`, `c : C`
- `(a, b, c).0 == a`
- `(a, b, c).1 == b`
- `(a, b, c).2 == c`
- `(2024, "九月", 1725434006138)`
- `// (a, b, c).2 = 10 <-- Nope`

Array

```
 1. test {
 2.   let array = [1, 2, 3]
 3.   inspect!(array[0], content="1")
 4.
 5.   array[0] = 100
 6.   inspect!(array, content="[100, 2, 3]")
 7.
 8.   array.push(1000)
 9.   inspect!(array, content="[100, 2, 3, 1000]")
10.
11.   let array = Array::make(3, 100)
12.   inspect!(array, content="[100, 100, 100]")
13. }
```

ArrayView

```
 1. test {
 2.   let array = [1, 2, 3]
 3.
 4.   let slice = array[1:]
 5.   inspect!(slice, content="[2, 3]")
 6.
 7.   let slice = array[:2]
 8.   inspect!(slice, content="[1, 2]")
 9.
10.   let slice = array[:]
11.   inspect!(slice, content="[1, 2, 3]")
12.
13.   slice[0] = 100
14.   inspect!(array, content="[100, 2, 3]")
15. }
```

if

```
1. test {
2.   let i = 0
3.   if i == 0 {
4.     ()
5.   }
6.
7.   let _ = if i == 1 {
8.     true
9.   } else {
10.    false
11.  }
12.
13.  inspect!(1 + (if true { 2 } else { 3 }) + 5, content="8")
14. }
```

for / for..in

```
 1. fn sum1(array : Array[Int]) -> Int {
 2.   let mut sum = 0
 3.   for i = 0; i < array.length(); i = i + 1 {
 4.     sum += i
 5.   }
 6.   sum
 7. }
 8.
 9. fn sum2(array : Array[Int]) -> Int {
10.   let mut sum = 0
11.   for i in array {
12.     sum += i
13.   }
14.   sum
15. }
```

for / for..in

```
1. fn sum3(array : Array[Int]) -> Int {
2.   for i = 0, sum = 0;
3.       i < array.length();
4.       i = i + 1, sum = sum + i {
5.     // sum = sum + 1 <- invalid
6.   } else {
7.     sum
8.   }
9. }
```

Function / |> / . / ..

```
 1. fn incr(i : Int, ~step : Int = 1) -> Int {
 2.   i + step
 3. }
 4.
 5. test {
 6.   inspect!(incr(1), content="2")
 7.   inspect!(incr(1, step=10), content="11")
 8.   inspect!(1 |> incr(step=10), content="11")
 9.   [].push(0)
10.   []..push(1)
11.     ..push(2)
12.     ..push(3) |> inspect!(content="[1, 2, 3]")
13. }
```

自定义数据类型

struct

```
 1. struct Point {
 2.   x : Int
 3.   y : Int
 4. }
 5.
 6. test {
 7.   let p = { x : 1, y : 2 } // <-- Point
 8.   let p2 = { ..p, y : 3 } // { x : 1, y : 3 }
 9.   // let p3 = { x : Int } <-- error
10.   let three = p2.y
11. }
```

struct

```
 1. struct A { x : Int }
 2. struct B { x : Int }
 3.
 4. fn acceptA(a : A) -> Unit {
 5.   // a.x = 100 <-- NOT OK
 6. }
 7.
 8. test {
 9.   acceptA({ x : 1 }) // OK : type inference works
10.   acceptA(A::{x : 1}) // OK : explicit construction
11.   // acceptA(({x : 1} : B)) <-- NOT OK
12. }
```

struct

```
 1. struct Ref[T] {
 2.   mut val : T
 3. }
 4.
 5. fn update(self : Ref[Int], y : Int) -> Unit {
 6.   self.val = y
 7. }
 8.
 9. test {
10.   ({ val: 100 } : Ref[Int])..update(2)..update(3)..update(4).val |> inspect!(content="4")
11. }
```

enum

```
enum Option[T] {
  Some(T)
  None
}

enum Result[T, Err] {
  Ok(T)
  Err(Err)
}
```

match

```
 1. fn or[T](option : Option[T], default : T) -> T {
 2.   match option {
 3.     Some(t) => t
 4.     None => default
 5.   }
 6. }
 7.
 8. fn head[T](array : ArrayView[T]) -> (T?, ArrayView[T]) {
 9.   match array {
10.     [] => (None, array)
11.     [hd, .. as tail] => (Some(hd), tail)
12.   }
13. }
```

loop

```
 1. fn fib(n : Int) -> Int {
 2.   fn aux(i, j, n) {
 3.     match n {
 4.       0 => j
 5.       n => aux(j, i + j, n - 1)
 6.     }
 7.   }
 8.   aux(0, 1, n)
 9. }
10.
11. fn fib2(n : Int) -> I nt {
12.   loop 0, 1, n {
13.     _i, j, 0 => j
14.     i, j, n => continue j, i + j, n - 1
15.   }
16. }
```

常用数据类型

Iter

```
1. test {
2.   [1, 2, 3].iter().map(fn { i => i + 1 }).collect()
3.   |> inspect!(content="[2, 3, 4]")
4. }
```

Map

```
 1. test {
 2.   let a = {} // Map[?, ?]
 3.   let b = {
 4.     "a" : 10,
 5.     "b" : 20
 6.   } // Map[String, Int]
 7.   b["c"] = 100
 8.   inspect!(b["a"], content=
 9.     "Some(10)"
10.   )
11. }
```

Map

```
 1. fn get(a : Map[String, Int]) -> Int {
 2.   match a {
 3.     {
 4.       "key1"  : i,
 5.       "key2"? : j // j : Int?
 6.     } => {
 7.       i + j.or(0)
 8.     }
 9.     _ => 0
10.   }
11. }
```

Map

```
1. fn visit(a : Map[String, Int]) -> Unit {
2.   for k, v in a {
3.     ()
4.   }
5. }
```

JSON

```
1. let b : Json = { "a": 10, "b": ["hello", "world", true] }
```

JSON

```
fn parse_payload(payload : Json) -> Unit {
  match payload {
    {
      "action": String("opened" | "synchronize" as action),
      "repository": {
        "name": String(repository),
        "owner": { "login": String(owner) },
      },
      "installation": { "id": Number(installation) },
      "pull_request": { "number": Number(pull_request) },
    } => ()
    _ => ()
  }
}
```

Newtype

```
type MyInt Int derive(Show)

pub fn op_add(self : MyInt, other : MyInt) -> MyInt {
  self._ - other._
}

test {
  inspect!(MyInt(5) + MyInt(5), content="MyInt(0)")
}
```

项目与模块

- MoonBit的一个模块由多个包组成
- 定义模块：`moon.mod.json`
- 定义包：`moon.pkg.json`

moon.mod.json

```
{
  "name": "user/mod",
  "version": "0.1.0", // Semantic Versioning
  "license": "Apache-2.0", // SPDX Identifier
  "deps": {
    "moonbitlang/x": "0.4.6"
  }
}
```

moon.pkg.json

```
{
  // 定义依赖
  "import" : [
    "user/mod/x", // module name + package name, @x
    { "path": "moonbitlang/x", "alias" : "package" } // @package
  ],
  "test-import": [
    // 测试依赖
  ]
}
```

Visibility

- 函数与数据
  - `pub fn` / `pub let`：可见，用 `@包名.` 访问
  - `fn` / `let`：不可见
- 数据结构

修饰符	类型可见	类型可读	类型可构造
`pub struct`	✅	✅	✅
`pub(readonly) struct`	✅	✅	❌
`struct`	✅	❌	❌
`priv struct`	❌	❌	❌

测试

- 白盒测试：开发文件或`*_wbtest.mbt`，开发者视角
- 黑盒测试：`*_test.mbt`，用户视角

Snapshot test

- 文件内：`inspect!(1)` -> `inspect!(1, content="1")`
- 文件外：

```
1. test (it : @test.T) {
2.   it.write(".")
3.   it.writeln("..")
4.   it.snapshot!(filename="001.txt") // __snapshot__/001.txt
5. }
```

- `moon test --update`

调试