Julia 中的作用域

Guangyao Zhao

2023-03-04

Contents

Julia 中的作用域分为：

- 全局作用域（globle scope）
- 局部作用域（local scope）
- 软作用域（soft scope）
- 硬作用域（hard scope）

相比于 Python，其中软作用域和硬作用域是两个新概念。需要指出的是，在 Julia 中的局部作用域指的是 `function`，`for`，`while`，`if`，而 Python 仅仅指的是 `function`。

在嵌套函数中，软作用域指的是内部函数可以访问外部函数的变量：

```julia
# 软作用域
function foo()
    x = 1
    function bar()
        println(x) # 1
        x = 2
        println(x) # 2
    end
    bar()
    println(x)
```

```
end

foo() # 输出 2 2
```

硬作用域指的是不可以访问兄弟函数的变量：

```
# 硬作用域
function foo()
    x = 1
    function bar()
        y = 2
        println(x)
    end
    function baz()
        x = 2
        bar()
        # println(y) # 报错：y 未定义，即不可以调用兄弟函数
    end
    baz()
end

foo() # 输出 1
```

局部作用域和软作用域

在 Julia 中，局部变量是定义在一个函数、循环、条件语句等局部作用域中的变量，只在该作用域内可见。而软作用域是指嵌套的函数可以访问其包含函数的局部变量。

区别在于，局部变量是在定义它们的作用域中可见，而软作用域是在函数的整个定义范围内可见。局部变量的作用域仅限于定义它们的代码块，而软作用域中的函数可以在任何地方调用，因此可以访问包含函数的所有变量。

以下是一个示例，说明局部变量和软作用域的区别：

```
function outer()
    x = 1
    function inner()
```

```
        println(x)
    end
    return inner
end

f = outer()
f() # 输出 1
```

```
function outer2()
    function inner2()
        println(x)
    end
    x = 2
    return inner2
end

# g = outer2()
# g() # 报错，找不到 x 的值
```

在这个示例中，outer 函数定义了一个局部变量 x，并定义了一个内部函数 inner。内部函数可以访问 outer 函数的局部变量 x。当我们调用 outer 函数并将返回的 inner 函数赋给变量 f，然后调用 f 函数时，它会输出 1。

然而，当我们定义 outer2 函数时，内部函数 inner2 试图访问在它定义之后定义的局部变量 x。这会导致运行时错误，因为 inner2 的软作用域不包含 x 的定义。

Julia & Python

Python 没有软作用域这一概念：

```
tmp = "tmp1"
for i in range(2):
    print("tmp1:", tmp)
    for j in range(2):
        tmp = "tmp2"
        print("tmp2:", tmp)
```

```
tmp = "tmp1"
for i in 1:2
    println("tmp1:", tmp)
    for j in 10:11
        tmp = "tmp2"
        println("tmp2:", tmp)
    end
end
```