**Lua脚本说明**

目录

[Lua脚本说明 1](#_Toc28952)

[一 Lua脚本基本语法 1](#_Toc24519)

[1.1. Lua基本数据结构 1](#_Toc3046)

[1.2. Lua变量 2](#_Toc29617)

[1.3. Lua循环 2](#_Toc14794)

[1.4. Lua流程控制 4](#_Toc26948)

[1.5. Lua运算符 5](#_Toc9973)

[1.6. Lua数组 6](#_Toc21814)

[1.7. Lua迭代器 7](#_Toc29034)

[二 注意事项 8](#_Toc20756)

[2.1. 单位统一 8](#_Toc22964)

[2.2. 参数术语 9](#_Toc15276)

[2.3. 温馨提示 9](#_Toc2600)

[三 脚本接口 9](#_Toc18873)

[3.1. 数学模块 9](#_Toc12652)

[3.2. 运动模块 12](#_Toc20687)

[3.3. 内部模块 25](#_Toc21027)

[3.4. 扩展模块 27](#_Toc18147)

[3.5. TCP通信 29](#_Toc30943)

[3.6. 通用脚本接口 30](#_Toc742)

[四 应用案例 31](#_Toc21200)

[4.1. 基础案例 31](#_Toc28481)

[4.2. TCP通信案例 32](#_Toc13006)

[4.3. 调用接口案例 34](#_Toc9532)

# **Lua脚本基本语法**

## Lua基本数据结构

Lua是动态类型语言，变量不要类型定义，只需要为变量赋值。值可以存储在变量中，作为参数传递或结果返回。

Lua中有8个基本类型分别为:

nil,boolean,number,string,userdata,function,thread和table。

|  |  |
| --- | --- |
| 数据类型 | 描述 |
| boolean | 包含两个值：false和true |
| number | 表示双精度类型的实浮点数 |
| string | 字符串由一对双引号或单引号来表示 |
| function | 由C或Lua编写的函数 |
| userdata | 表示任意存储在变量中的C数据结构 |
| thread | 表示执行的独立线路，用于执行协同程序 |
| nil | 只有nil属于该类，表示一个无效值（在条件表达式中相当于false） |
| table | Lua中的表(table)其实就是一个“关联数组”(assoclative arrays),数组的索引可以是数字、字符串或表类型。在Lua里，table的创建是通过“构造表达式”来完成，最简单构造表是{}，用来创建一个空表。 |

示例：

print(type(true))               --> boolean

print(type(10.4\*3))             --> number

print(type("Hello world"))      --> string

print(type(print))              --> function

print(type(nil))                --> nil

print(type({})) --> table

## Lua变量

变量在使用前，需要在代码中进行说明，即创建该变量。

编译程序执行代码之前编译器需要知道如何给语句变量开辟存储区，用于存储变量的值。

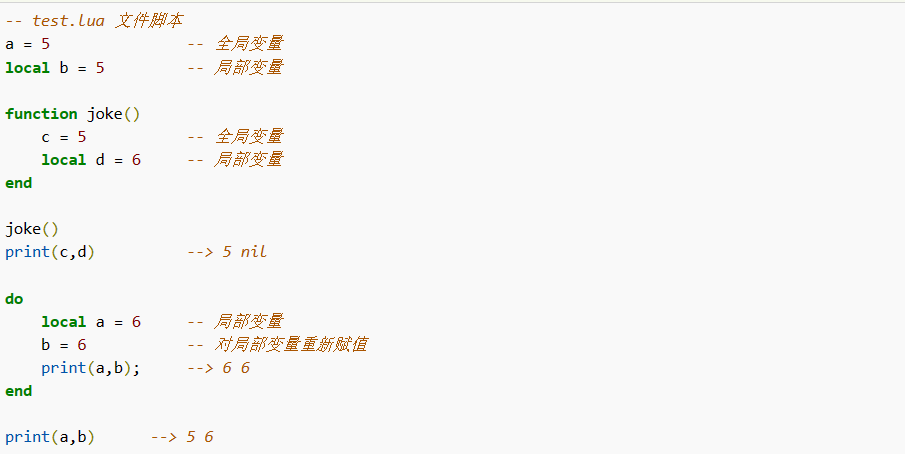
Lua变量有三种类型：全局变量，局部变量，表中的域。

Lua中的变量全是全局变量，哪怕是语句块或是函数里，除非用local显示声明为局部变量。

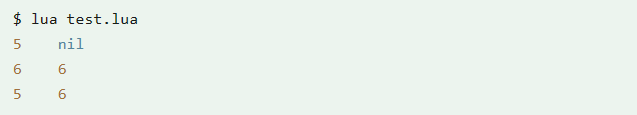
局部变量的作用域为从声明我实在开始到所在语句块结束。

变量的默认值均为nil

示例如下：



执行以上示例输出结果为：



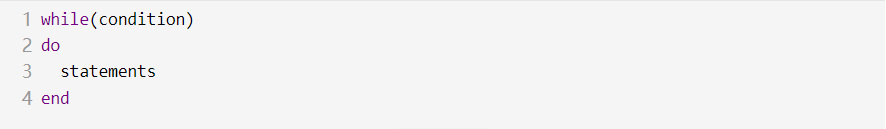
## Lua循环

Lua语言提供了以下几种循环处理方式：

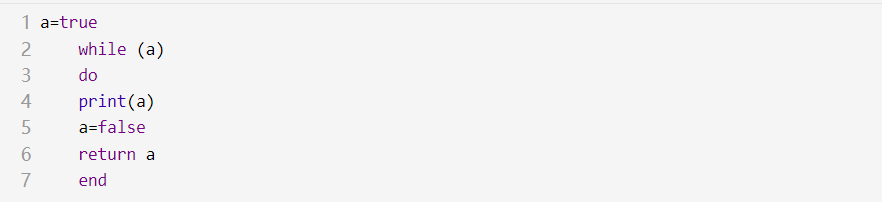
|  |  |
| --- | --- |
| 循环类型 | 描述 |
| while循环 | 在条件为true时，让程序重复地执行某些语句。执行语句前会先检查条件是否为true |
| for循环 | 重复执行指定语句，重复次数可在for语句中控制 |
| repeat...util | 重复执行循环，直到指定的条件为真时为止 |
| 嵌套循环 | 可以在循环内嵌套一个或多个循环语句(while do ... end;for ... do ... end;repeat ... until;) |

### while循环

Lua编程语言中while循环语句在判断条件为true时会重复执行循环体语句



示例如下：



执行以上示例输出结果为：

while循环结果

### for循环

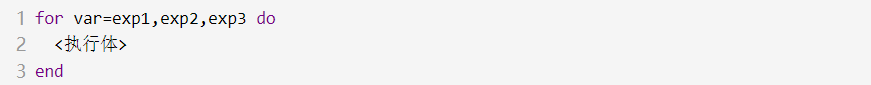
Lua编程语言中for循环语句可以重复执行指定语句，重复次数可在for语句中控制。

Lua编程语言中for语句有两大类：

* 数值for循环
* 泛型for循环

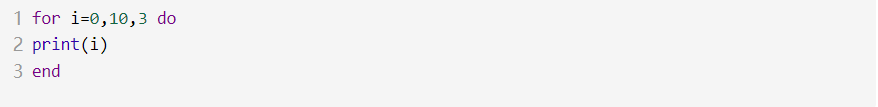
#### 数值for循环

Lua编程语言中数值for循环语法格式：

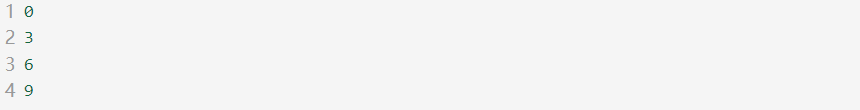


var从exp1变化到exp2，每次变化以exp3为步长递增var，并执行一次“执行体”。exp3是可选的，如果不指定，默认为1。exp2可以用函数来表示，灵活的对数进行加减乘除。

示例如下：



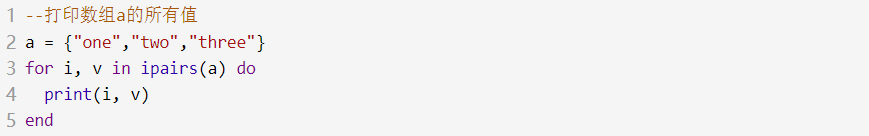
执行以上示例输出结果为：



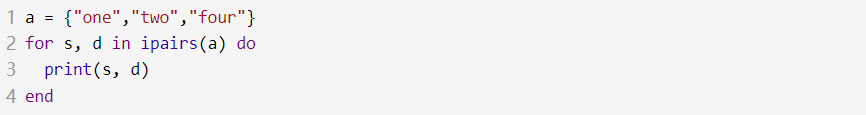
#### 泛型for循环

泛型for循环通过一个迭代器函数来遍历所有值，类似java中的foreach语句

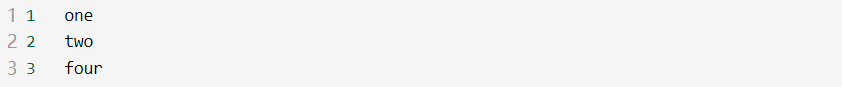
Lua编程语言中泛型for循环语法格式：



示例如下:

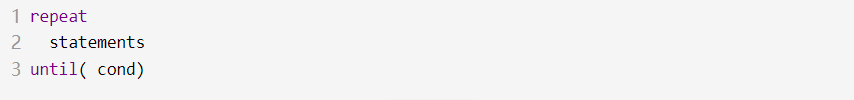


执行以上示例输出结果为：

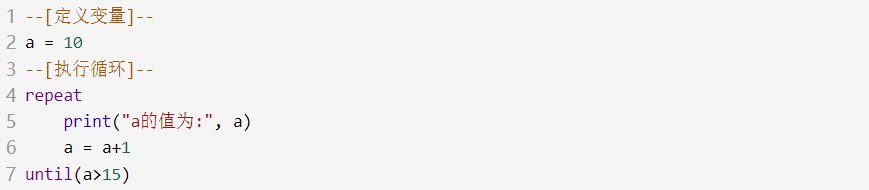


**3.3 repeat...util**

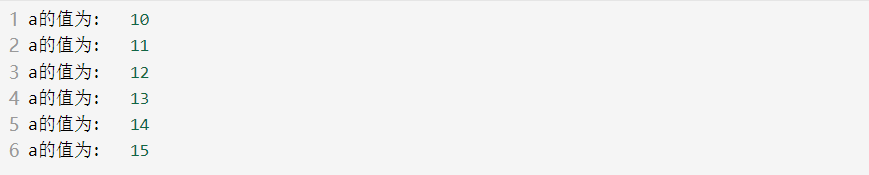
Lua编程语言中repeat...until循环语句不同于for和while循环，for和while循环的条件语句在当前循环执行开始时判断，而repeat...until循环的条件语句在当前循环结束后判断。

****

示例如下：

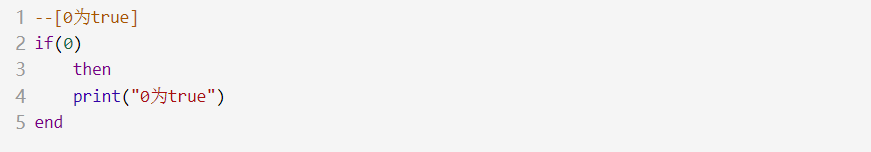
****

执行以上示例输出结果为：

****

## Lua流程控制

条件判断语句：Lua编程语言流程控制语句用过程序设定一个或多个条件语句来设定。在条件为true时执行指定程序代码，在条件为false时执行其他指定代码。



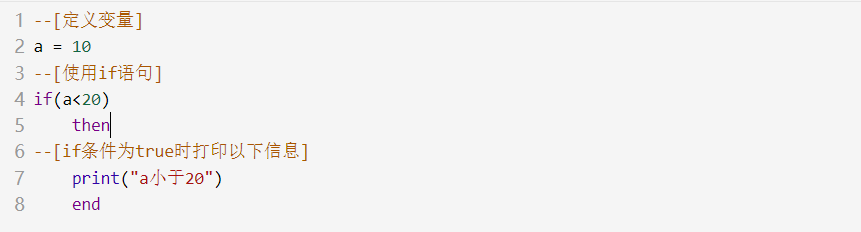
### Lua if语句

Lua if语句由一个布尔表达式作为条件判断，其后紧跟其他语句组成。

语法如下：



示例如下：



执行以上示例输出结果为：

if输出结果

### Lua if...else 语句

Lua if 语句可以与 else 语句搭配使用, 在 if 条件表达式为 false 时执行 else 语句代码块。

语法格式如下：



示例如下：

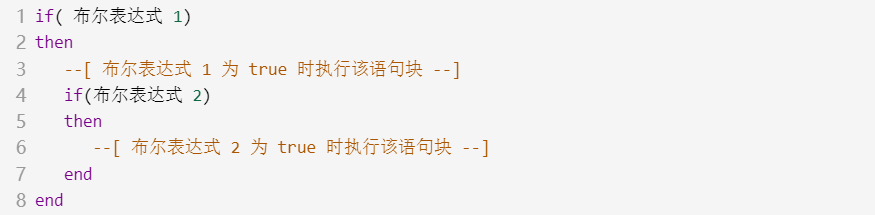


执行以上示例输出结果为：

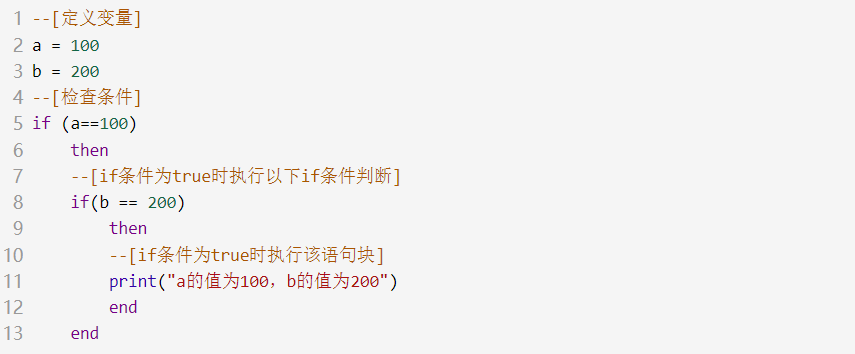
if else结果

### if嵌套语句

Lua if语句允许嵌套，这就意味着你可以在一个if或else if语句中插入其他的if或else if语句，语句如下：



示例如下:



执行以上实例输出结果为：

if嵌套结果

## Lua运算符

运算符是一个特殊的符号，用于告诉解释器执行特定的数学或逻辑运算。Lua提供了以下几种运算符类型:

* 算数运算符
* 关系运算符
* 逻辑运算符
* 其他运算符

### 算数运算符

下表列出了Lua语言中的常用运算符，设定A的值为10，B的值为20：

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 示例 |
| + | 加法 | A+B输出结果30 |
| - | 减法 | A-B输出结果-10 |
| \* | 乘法 | A\*B输出结果200 |
| / | 除法 | B/A输出结果2 |
| % | 取余 | B%A输出结果0 |
| ^ | 乘幂 | A^2输出结果100 |
| - | 负号 | -A输出结果-10 |
| // | 整除运算符(>=lua5.3) | 5//2输出结果2 |

### 关系运算符

下表列出了Lua语言中的常用运算符，设定A的值为10，B的值为20：

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 示例 |
| == | 等于，检测两个值是否相等，相等返回true，否则返回false | (A==B)为false |
| -= | 不等于，检测两个值是否相等，不相等返回true，否则返回false | (A-=B)为true |
| > | 大于，如果左边的值大于右边的值，返回true，否则返回false | (A>B)为false |
| < | 小于，如果左边的值小于右边的值，返回true，否则返回false | (A<B)为true |
| >= | 大于等于，左侧值大于等于右侧值，返回true，否则返回false | (A>=B)为false |
| <= | 小于等于，左侧值小于等于右侧值，返回true，否则返回false | (A<=B)为true |

### 逻辑运算符

下表列出来Lua语言中的常用逻辑运算符,设定A的值为true，B的值为false

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 示例 |
| and | 逻辑与操作符，若A为false，则返回A，否则返回B | (A and B)  为false |
| or | 逻辑或操作符，若A为true，则返回A，否则返回B | (A or B)  为true |
| not | 逻辑非操作符，与逻辑运算符结果相反，如果条件为true，逻辑非为false | not(A and B)  为true |

### 其他运算符

下表列出来Lua语言中的连接运算符与计算表或字符串长度的运算符：

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 示例 |
| .. | 连接两个字符串 | a..b，其中a为”Hello”,b为“World”，输出结果为“Hello World” |
| # | 一元运算符，返回字符串或表的长度 | #”Hello”返回5 |

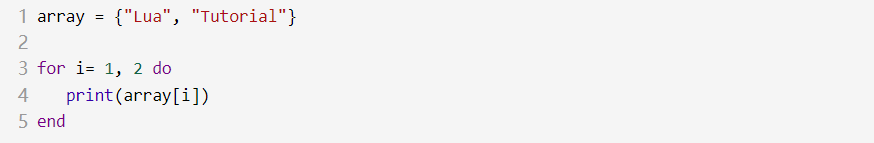
## Lua数组

数组，就是相同数据类型的元素按一定顺序排列的集合，可以是一维数组的多维数组。

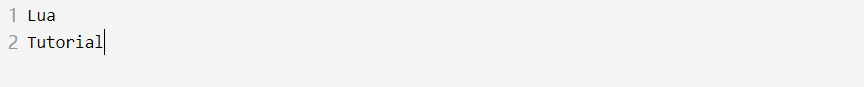
Lua数组的索引键值可以使用整数表示，数组的大小不是固定的。

### 一维数组

一维数组是最简单的数组，其逻辑结构是线性表。一维数组可以用for循环出数组中的元素。示例如下：



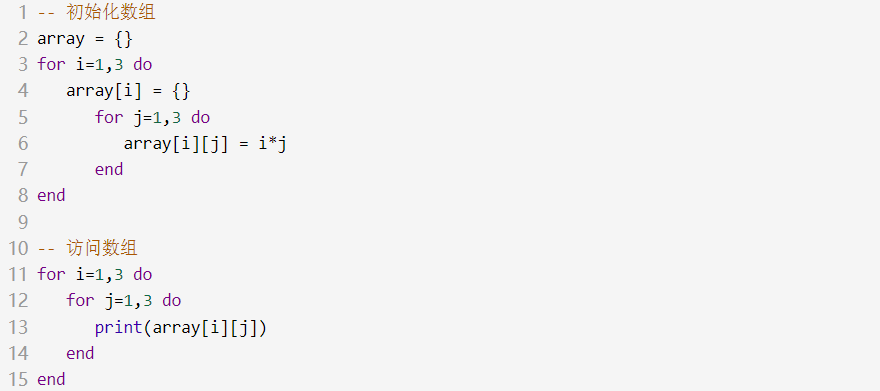
执行以上实例输出结果为：



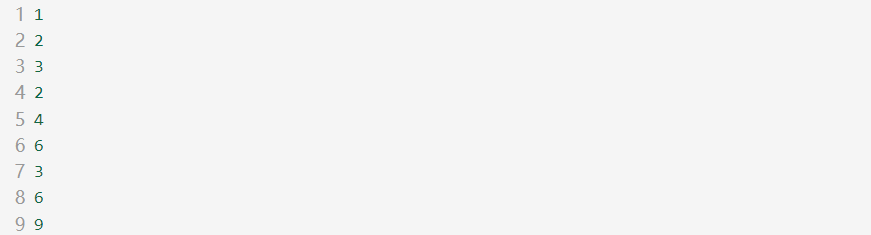
### 多维数组

多维数组即数组中包含数组或一维数组的索引键对应一个数组。

以下是一个三行三列的阵列多维数组：



执行以上实例输出结果为：



## Lua迭代器

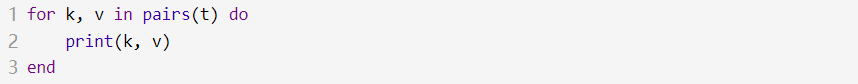
迭代器(iterator)是一种对象，它能够用来遍历标准模板库容器中的部分或全部元素，每个迭代器对象代表容器中的确定的地址。

在Lua中迭代器是一种支持指针类型的结构，它可以遍历集合的每一个元素。

### 泛型for迭代器

泛型for在自己内部保存迭代函数，实际上它保存三个值：迭代函数、状态常量、控制变量。

泛型for迭代器提供了集合的key/value对，语法格式如下：



以上示例中我们使用了Lua默认提供的迭代函数ipairs。

下面我们看看泛型for的执行过程：

* 首先，初始化，计算 in 后面表达式的值，表达式应该返回泛型 for 需要的三个值：迭代函数、状态常量、控制变量；与多值赋值一样，如果表达式返回的结果个数不足三个会自动用 nil 补足，多出部分会被忽略。
* 第二，将状态常量和控制变量作为参数调用迭代函数（注意：对于 for 结构来说，状态常量没有用处，仅仅在初始化时获取他的值并传递给迭代函数）。
* 第三，将迭代函数返回的值赋给变量列表。
* 第四，如果返回的第一个值为nil循环结束，否则执行循环体。
* 第五，回到第二步再次调用迭代函数。

在Lua中我们常常使用函数来描述迭代器，每次调用该函数就返回集合得下一个元素。Lua得迭代器包含以下两种类型

* 无状态的迭代器
* 多状态的迭代器

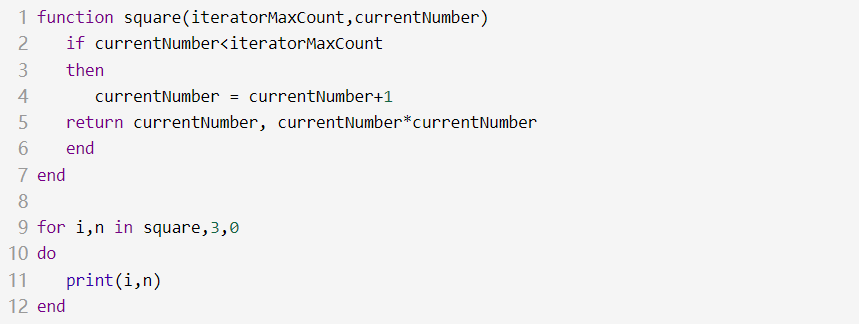
### 无状态的迭代器

无状态的迭代器是指不保留任何状态的迭代器，因此在循环中我们可以利用无状态迭代器避免创建闭包花费额外的代价。

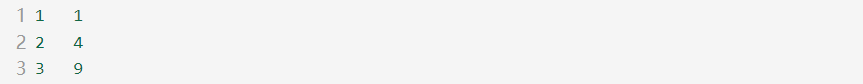
每一次迭代，迭代函数都是用两个变量（状态常量和控制变量）的值作为参数被调用，一个无状态的迭代器只利用这两个值可以获取下一个元素。

这种无状态迭代器的典型的简单的例子是 ipairs，它遍历数组的每一个元素，元素的索引需要是数值。

我们使用了一个简单的函数来实现迭代器，实现数字n的平方，示例如下：



执行以上实例输出结果为：



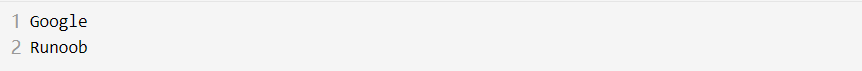
### 多状态的迭代器

很多情况下，迭代器需要保存多个状态信息而不是简单的状态常量和控制变量，最简单的方法是使用闭包，还有一种方法就是将所有的状态信息封装到 table 内，将 table 作为迭代器的状态常量，因为这种情况下可以将所有的信息存放在 table 内，所以迭代函数通常不需要第二个参数。

我们创建自己的迭代器,示例如下：



执行以上实例输出结果为：



# **注意事项**

## 2.1. 单位统一

|  |  |
| --- | --- |
| 名称 | 单位 |
| 关节角 | 弧度 |
| 距离 | m |
| 时间 | s |
| 关节加速度 | rad/ |
| 关节速度 | rad/s |
| 末端加速度 | m/ |
| 末端速度 | m/s |
| 姿态 | 四元数 |
| 重量 | kg |

**注：所有下文中接口均要求传递姿态参数为四元数形式，当使用欧拉角形式时，可以使用 rpy2quaternion 接口转换，欧拉角定义的旋转顺序为 Z,Y,X。**

## 2.2. 参数术语

必传参数：必须传递，不使用也要给出默认值，任何情况下绝对不能不传。

选传参数：根据函数的参数说明和业务逻辑选择是否需要传递。

捆绑参数：与一个参数进行捆绑关联，根据函数的参数说明选择传递方式。

必不传参数：一定不能传递，一般与捆绑参数配合使用，参照函数的参数说明。

**注：无特殊说明的，均为必传参数。**

工具参数：由工具运动学参数和工具动力学参数构成

工具运动学参数（又名：工具末端参数）：由工具末端位置参数和工具末端，姿态参数构成，该参数也可以描述工具坐标系。

工具动力学参数：由工具负载和重心参数构成。

参考坐标系参数：有三种不同类型，机械臂基坐标系参数，以下简称为基坐标系；

工具坐标系参数，基于机械臂法兰盘中心描述的工具坐标系，由工具末端位置参数和工具末端姿态参数构成；

**用户坐标系参数，基于机械臂基坐标系描述的用户坐标系，由用户坐标系标定方法，标定时使用的法兰盘中心基于基坐标系的三个路点，标定时使用的工具末端位置参数构成。**

**注：法兰盘中心是一个特殊的工具和工具坐标系。**

**当作为工具时，工具末端位置参数为(0,0,0),工具末端姿态参数为(1,0,0,0)，工具负载为0，重心为(0,0,0)；**

**当作为工具坐标系时，工具末端位置参数为(0,0,0), 工具末端姿态参数为(1,0,0,0)。**

**下文参数说明中所提到的工具和工具坐标系均包含法兰盘中心。**

**基坐标系是特殊的用户坐标系，可以理解为基坐标系是一个与机械臂基坐标系原点重合，三轴重合的用户坐标系。**

## 2.3. 温馨提示

**本文档中的全部接口顺序均不以示例中调用顺序为主，而是以功能类别为主，示例程序中如果出现任何上文中未提到的接口或者变量名，请自行搜索下文找到该接口或者变量名。**

**所有红字标注的内容都非常重要，请仔细阅读。**

# **脚本接口**

## 3.1. 数学模块

|  |  |
| --- | --- |
| **double acos(double f)** | |
| 功能 | 返回 f的反余弦主值（以弧度为单位）。如果 f在范围[- 1, 1]之外，则会  发生运行时错误 |
| 参数 | f |
| 返回 | f的反余弦,浮点值 |

|  |  |
| --- | --- |
| **double asin(double f)** | |
| 功能 | 返回 f的反正弦主值（以弧度为单位）。如果 f在范围[- 1, 1]之外，则会  发生运行时错误 |
| 参数 | f |
| 返回 | f的反正弦,浮点值 |

|  |  |
| --- | --- |
| **double atan(double f)** | |
| 功能 | 返回 f的反正切主值（以弧度为单位） |
| 参数 | f |
| 返回 | f 的反正切,浮点值 |

|  |  |
| --- | --- |
| **double atan2(double x,double y)** | |
| 功能 | 返参数回 x/y的反正切主值（以弧度为单位） |
| 参数 | x |
| y |
| 返回 | x/z的反正切,浮点值 |

|  |  |
| --- | --- |
| **double cos(double f)** | |
| 功能 | 返回 f弧度角的余弦 |
| 参数 | f |
| 返回 | f 的 y余弦,浮点值 |
| **double sin(double f)** | |
| 功能 | 返回 f弧度角的正弦 |
| 参数 | f |
| 返回 | f的正弦,浮点值 |

|  |  |
| --- | --- |
| **double tan(double f)** | |
| 功能 | 返回 f弧度角的正切 |
| 参数 | f |
| 返回 | f的正切,浮点值 |

|  |  |
| --- | --- |
| **double sqrt(double f)** | |
| 功能 | 返回 f的平方根。如果 f为负，则会发生运行时错误 |
| 参数 | f |
| 返回 | f的平方根,浮点值 |

|  |  |
| --- | --- |
| **double log(double b,double f)** | |
| 功能 | 返回 f~b基数的对数。如果 b或 f为负数，则会发生运行时错误 |
| 参数 | b |
| f |
| 返回 | f~b基数的对数，浮点值 |

|  |  |
| --- | --- |
| **double pow(double b,double e)** | |
| 功能 | 返回基数自乘指数幂的结果。如果基数为负，并且指数不是整数值，或者  如果基数为零，并且指数为负，则会发生运行时错误。 |
| 参数 | b |
| e |
| 返回 | 基数自乘指数幂，浮点值 |

|  |  |
| --- | --- |
| **int ceil(double x)** | |
| 功能 | 将浮点数四舍五入到不小于 f的最小整数。 |
| 参数 | f |
| 返回 | 四舍五入的整数 |

|  |  |
| --- | --- |
| **int floor(double x)** | |
| 功能 | 将浮点数四舍五入到不大于 f的最大整数 |
| 参数 | f |
| 返回 | 四舍五入的整数 |

|  |  |
| --- | --- |
| **double r2d(double r)** | |
| 功能 | 返回弧度 r转化为角度值 |
| 参数 | r |
| 返回 | 角度值，浮点值 |

|  |  |
| --- | --- |
| **double d2r(double d)** | |
| 功能 | 返回 d 度数的弧度值。实际上：(d/180)\*PI |
| 参数 | **d**角度值 |
| 返回 | 以弧度为单位的角度，浮点值 |

注：欧拉角顺序为 **ZYX**

|  |  |
| --- | --- |
| **{oriW,oriX,oriY,oriZ} rpy2quaternion({oriRX,oriRY,oriRZ})** | |
| 功能 | 欧拉角转换为四元数 |
| 参数 | 欧拉角（弧度） |
| 返回 | 根据参数欧拉角转换后的四元数 |

|  |  |
| --- | --- |
| **{oriRX,oriRY,oriRZ} quaternion2rpy({oriW,oriX,oriY,oriZ})** | |
| 功能 | 四元数转换为欧拉角 |
| 参数 | 四元数 |
| 返回 | 根据参数四元数转换后的欧拉角（弧度） |

## 3.2. 运动模块

|  |  |
| --- | --- |
| **void init\_global\_move\_profile (void)** | |
| 功能 | 初始化全局运动属性 |
| 参数 | 无 |
| 返回 | 无 |
| 说明 | 默认全局运动属性包括但不限于:  坐标系参数  工具参数  关节速度加速度阈值  末端速度加速度阈值  交融半径  全局路点  提前到位参数等 |

|  |  |
| --- | --- |
| **void set\_joint\_maxacc (**  **{double joint1MaxAcc, double joint2MaxAcc, double joint3MaxAcc, double joint4MaxAcc, double joint5MaxAcc, double joint6MaxAcc})** | |
| 功能 | 设置关节 1-6 的最大加速度,单位 rad/s^2. |
| 参数 | 类型为 table |
| 返回 | 无 |
| 示例 | set\_joint\_maxacc({1.0,1.0,1.0,1.0,1.0,1.0}) |

|  |  |
| --- | --- |
| **void set\_joint\_maxvelc (**  **{double joint1MaxVelc, double joint2MaxVelc, double joint3MaxVelc, double joint4MaxVelc, double joint5MaxVelc, double joint6MaxVelc})** | |
| 功能 | 设置关节 1-6 的最大速度,单位 rad/s. |
| 参数 | 类型为 table |
| 返回 | 无 |
| 示例 | set\_joint\_maxvelc({1.0,1.0,1.0,1.0,1.0,1.0}) |

|  |  |
| --- | --- |
| **void set\_end\_maxacc(double endMaxAcc)** | |
| 功能 | 设置末端最大加速度,单位 m/s^2. |
| 参数 | 末端最大加速度 |
| 返回 | 无 |
| 示例 | set\_end\_maxacc (1.0) |

|  |  |
| --- | --- |
| **void set\_end\_maxvelc(double endMaxVelc)** | |
| 功能 | 设置末端最大速度,单位 m/s. |
| 参数 | 末端最大速度 |
| 返回 | 无 |
| 示例 | set\_end\_maxvelc (1.0) |

|  |  |
| --- | --- |
| **void move\_joint(**  **{double joint1Angle, double joint2Angle, double joint3Angle,**  **double joint4Angle, double joint5Angle, double joint6Angle},bool isBlock)** | |
| 功能 | 轴动,单位弧度 |
| 参数 | {double joint1Angle, double joint2Angle, double joint3Angle,  double joint4Angle, double joint5Angle, double joint6Angle}  目标路点的六个关节角弧度值。 |
| IsBlock运动阻塞标志位。为 true时， 接口阻塞直到运动至目标路点；为 false，接口  立即返回。 |
| 返回 | 无 |
| 示例 | move\_joint({-0.000003, -0.127267, - 1.321122, 0.376934, - 1.570796, - 0.000008},true) |

|  |  |
| --- | --- |
| **void move\_line(**  **{double joint1Angle, double joint2Angle, double joint3Angle,**  **double joint4Angle, double joint5Angle, double joint6Angle},**  **bool isBlock)** | |
| 功能 | 直线运动,单位弧度. |
| 参数 | 与上文 move\_joint 函数参数完全一致。 |
| 返回 | 无 |
| 示例 | move\_line({-0.000003, -0.127267, - 1.321122, 0.376934, - 1.570796, - 0.000008},true) |

|  |  |
| --- | --- |
| **void set\_relative\_offset(**  **{double posOffsetX, double posOffsetY, double posOffsetZ},**  **{double oriOffsetW, double oriOffsetX, double oriOffsetY, double oriOffsetZ}, {double toolEndPosX, toolEndPosY, toolEndPosZ},**  **{double toolEndOriW, toolEndOriX, toolEndOriY, toolEndOriZ},**  **CoordCalibrateMethod coordCalibrateMethod,**  **{double point1Joint1, double point1Joint2, double point1Joint3,**  **double point1Joint4, double point1Joint5, double point1Joint6},**  **{double point2Joint1, double point2Joint2, double point2Joint3,**  **double point2Joint4, double point2Joint5, double point2Joint6},**  **{double point3Joint1, double point3Joint2, double point3Joint3,**  **double point3Joint4, double point3Joint5, double point3Joint6},**  **{double toolEndPosXForCalibUserCoord, toolEndPosYForCalibUserCoord, toolEndPosZForCalibUserCoord})** | |
| 功能 | 设置相对偏移属性 |
| 参数 | {double posOffsetX, double posOffsetY, double posOffsetZ}  基于参考坐标系的位置偏移量，必传参数。如果不需要位置偏移, 则传递为  {0, 0, 0}。 |
| {double oriOffsetW, double oriOffsetX, double oriOffsetY, double oriOffsetZ}  基于参考坐标系的姿态偏移量，选传参数。如果不需要姿态偏移, 则可以传  递为{1, 0, 0, 0}或者不传递。 |
| 工具坐标系参数，选传参数。包括:  工具末端位置参数{double toolEndPosX, toolEndPosY, toolEndPosZ}, 工具末  端姿态参数{double toolEndOriW, toolEndOriX, toolEndOriY, toolEndOriZ}。  注：当基于基坐标系或用户坐标系时，该参数为必不传参数，当基于工具坐  标系时，为必传参数 |
| 用户坐标系参数，选传参数。包括:  CoordCalibrateMethod 标定用户坐标系的方法枚举，参考枚举类型章节。  {double point1Joint1, double point1Joint2, double point1Joint3,  double point1Joint4, double point1Joint5, double point1Joint6},  {double point2Joint1, double point2Joint2, double point2Joint3,  double point2Joint4, double point2Joint5, double point2Joint6},  {double point3Joint1, double point3Joint2, double point3Joint3,  double point3Joint4, double point3Joint5, double point3Joint6} |

|  |  |
| --- | --- |
|  | 标定用户坐标系所使用的法兰盘中心基于基坐标系的三个点关节角。  {double toolEndPosXForCalibUserCoord, toolEndPosYForCalibUserCoord,  toolEndPosZForCalibUserCoord}  标定用户坐标系所使用的工具末端位置参数。选传参数， 当使用法兰盘中心 标定用户坐标系时可以不传递该参数或者传递为(0,0,0)；当使用工具标定用  户坐标系时为必传参数。  注：当基于基坐标系或工具坐标系时，该参数为必不传参数，当基于用户坐  标系时，为必传参数。 |
| 返回 | 无 |
| 示例 | set\_relative\_offset (  {0.2,0.2,0.2},  CoordCalibrateMethod.zOzy,  {-0.000003, -0.127267, - 1.321122, 0.376934, - 1.570796, -0.000008}, {-0.186826, -0.164422, - 1.351967, 0.383250, - 1.570795, -0.186831}, {-0.157896, 0.011212, - 1.191991, 0.367593, - 1.570795, -0.157901}, {0.1, 0.2, 0.3}) |

|  |  |
| --- | --- |
| **{**  **"pos" = {**  **"x" = posX**  **"y" = posY**  **"z" = posZ**  **}**  **"ori" = {**  **"w" = oriW**  **"x" = oriX**  **"y" = oriY**  **"z" = oriZ**  **}**  **"joint" = {**  **"j1" = joint1Angle**  **"j2" = joint2Angle**  **"j3" = joint3Angle**  **"j4" = joint4Angle**  **"j5" = joint5Angle**  **"j6" = joint6Angle**  **}**  **}**  **get\_current\_waypoint(void)** | |
| 功能 | 返回当前机械臂的实时路点位置，姿态，关节角度的嵌套 table |
| 参数 | 无 |
| 返回 | 类型为 table 的实时路点位置、姿态、关节角。 |
| 示例 | realPoint = get\_current\_waypoint()  print("posX:"..realPoint.pos.x,"poxY:"..realPoint.pos.y,"poxZ : "..realPoint.pos.z)  print("oriW:"..realPoint.ori.w,"oriX:"..realPoint.ori.x,"oriY:  "..realPoint.ori.y,"oriZ : "..realPoint.ori.z)  print("joint1:"..realPoint.joint.j1,"joint2:"..realPoint.joint.j2,"joint3: "..realPoint.joint.j3,"joint4:"..realPoint.joint.j4,"joint5:  "..realPoint.joint.j5,"joint6 : "..realPoint.joint.j6) |

|  |  |
| --- | --- |
| **{joint1Angle, joint2Angle, joint3Angle, joint4Angle, joint5Angle, joint6Angle} get\_target\_pose(**  **{ikRefPointJoint1Angle, ikRefPointJoint2Angle, ikRefPointJoint3Angle, ikRefPointJoint4Angle, ikRefPointJoint5Angle, ikRefPointJoint6Angle}, {double toolEndPosXOnRefCoord, toolEndPosYOnRefCoord,**  **toolEndPosZOnRefCoord},**  **{double toolEndOriWOnRefCoord, toolEndOriXOnRefCoord,**  **toolEndOriYOnRefCoord, toolEndOriZOnRefCoord},**  **bool enableEndRotate, double endRotateAngle,**  **{double toolEndPosX, toolEndPosY, toolEndPosZ},**  **{double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ}, CoordCalibrateMethod coordCalibrateMethod,**  **{double point1Joint1, double point1Joint2, double point1Joint3,**  **double point1Joint4, double point1Joint5, double point1Joint6},**  **{double point2Joint1, double point2Joint2, double point2Joint3,**  **double point2Joint4, double point2Joint5, double point2Joint6},**  **{double point3Joint1, double point3Joint2, double point3Joint3,**  **double point3Joint4, double point3Joint5, double point3Joint6},**  **{double toolEndPositionForCalibCoordX, toolEndPositionForCalibCoordY,**  **toolEndPositionForCalibCoordZ}**  **)** | |
| 功能 | 获取根据指定的逆解参考点、位置、姿态、工具、用户坐标系和末端旋转角度  参数，逆解后的关节角度 |
| 参数 | {ikRefPointJoint1Angle, ikRefPointJoint2Angle, ikRefPointJoint3Angle, ikRefPointJoint4Angle, ikRefPointJoint5Angle, ikRefPointJoint6Angle}  逆解参考点，选传参数。  传参时，以输入参数为逆解的参考点；不传参时，以当前机械臂的实时路点  为逆解的参考点。 |
| {double toolEndPosXOnRefCoord, toolEndPosYOnRefCoord,  toolEndPosZOnRefCoord}  基于参考坐标系的工具末端位置参数。必传参数。 |
| {double toolEndOriWOnRefCoord, toolEndOriXOnRefCoord,  toolEndOriYOnRefCoord, toolEndOriZOnRefCoord}  基于参考坐标系的工具末端姿态参数。  选传参数，如果不传递该参数，默认保持当前实时路点姿态。 |

|  |  |
| --- | --- |
|  | bool enableEndRotate  使能末端旋转参数，必传参数。  double endRotateAngle  末端轴动参数，捆绑参数。  若 enableEndRotate 参数为 true，则该参数为必传参数， 将逆解后第六关节的  结果替换为该参数的值。  若 enableEndRotate 参数为 false，则该参数为必不传参数。 |
| 工具末端参数，选传参数。包括：  **{double toolEndPosX, toolEndPosY, toolEndPosZ}**  工具末端位置参数。  **{double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ}**  工具末端姿态参数。  注：当基于法兰盘中心时，工具末端参数可以不传递或传递为**{0,0,0}**，  **{1,0,0,0}**。 |
| 用户坐标系参数，选传参数。包括:  CoordCalibrateMethod 标定用户坐标系的方法枚举，参考枚举类型章节。  {double point1Joint1, double point1Joint2, double point1Joint3,  double point1Joint4, double point1Joint5, double point1Joint6},  {double point2Joint1, double point2Joint2, double point2Joint3,  double point2Joint4, double point2Joint5, double point2Joint6},  {double point3Joint1, double point3Joint2, double point3Joint3,  double point3Joint4, double point3Joint5, double point3Joint6}  标定用户坐标系所使用的法兰盘中心基于基坐标系的三个点关节角。  {double toolEndPosXForCalibUserCoord, toolEndPosYForCalibUserCoord,  toolEndPosZForCalibUserCoord}  标定用户坐标系所使用的工具末端位置参数。选传参数， 当使用法兰盘中心 标定用户坐标系时可以不传递该参数或者传递为(0,0,0)；当使用工具标定用户  坐标系时为必传参数。  注： 当基于基坐标系逆解时， 该参数为必不传参数。 |
| 返回 | 如果逆解成功，返回逆解后的关节角  {joint1Angle, joint2Angle, joint3Angle, joint4Angle, joint5Angle, joint6Angle}；  如果逆解失败，返回 false。 |
| 示例 | while (true) do  sleep(0.001)  init\_global\_move\_profile()  set\_joint\_maxvelc({1.298089,1.298089,1.298089,1.555088,1.555088,1.555088}) set\_joint\_maxacc({8.654390,8.654390,8.654390,10.368128,10.368128,10.368128 })  set\_end\_maxvelc(1.000000)  set\_end\_maxacc(1.000000)  move\_joint(get\_target\_pose({0, 0, -0.131415}, {0.707114, 0.000014, 0.7071, 0.000007}, true, d2r(20), {0.1, 0.2, 0.3}, {1.0, 0.0, 0.0, 0.0},  CoordCalibrateMethod.zOzy, {-0.000003, -0.127267, - 1.321122, 0.376934, -  1.570796, -0.000008}, {-0.186826,-0.164422, - 1.351967, 0.383250, - 1.570795, - 0.186831},{-0.157896, 0.011212, - 1.191991, 0.367593, - 1.570795, -0.157901}, {0.1, 0.2, 0.3}), true)  move\_line(get\_target\_pose({0, 0, 0.131181}, {0.707112, 0.000013, 0.707101, 0.000008}, true, d2r(20), {0.1, 0.2, 0.3}, {1.0, 0.0, 0.0, 0.0},  CoordCalibrateMethod.zOzy, {-0.000003, -0.127267, - 1.321122, 0.376934, -  1.570796, -0.000008}, {-0.186826,-0.164422, - 1.351967, 0.383250, - 1.570795, - 0.186831},{-0.157896, 0.011212, - 1.191991, 0.367593, - 1.570795, -0.157901}, {0.1, 0.2, 0.3}), true)  end |

|  |  |
| --- | --- |
| **{posX, posY, posZ},**  **{oriW, oriX, oriY, oriZ}**  **base\_to\_user(**  **{double joint1Angle, double joint2Angle, double joint3Angle,**  **double joint4Angle, double joint5Angle, double joint6Angle},**  **{double toolEndPosX, toolEndPosY, toolEndPosZ},**  **{double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ}, CoordCalibrateMethod coordCalibrateMethod,**  **{double point1Joint1, double point1Joint2, double point1Joint3,**  **double point1Joint4, double point1Joint5, double point1Joint6},**  **{double point2Joint1, double point2Joint2, double point2Joint3,**  **double point2Joint4, double point2Joint5, double point2Joint6},**  **{double point3Joint1, double point3Joint2, double point3Joint3,**  **double point3Joint4, double point3Joint5, double point3Joint6},**  **{double toolEndPosXForCalibCoord, toolEndPosYForCalibCoord,**  **toolEndPosZForCalibCoord}**  **)** | |
| 功能 | 将法兰盘中心的位姿转换为工具末端基于参考坐标系的位姿 |
| 参数 | {double joint1Angle, double joint2Angle, double joint3Angle,  double joint4Angle, double joint5Angle, double joint6Angle}  六个关节弧度（唯一确定一个法兰盘中心的位姿）。 |
| 工具末端参数，选传参数。包括：  **{double toolEndPosX, toolEndPosY, toolEndPosZ}**  工具末端位置参数。  **{double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ}**  工具末端姿态参数。  注： 当基于法兰盘中心时，工具末端参数可以不传递或传递为**{0,0,0}**，  **{1,0,0,0}**。 |
| 用户坐标系参数，选传参数。包括:  CoordCalibrateMethod 标定用户坐标系的方法枚举，参考枚举类型章节。  {double point1Joint1, double point1Joint2, double point1Joint3,  double point1Joint4, double point1Joint5, double point1Joint6},  {double point2Joint1, double point2Joint2, double point2Joint3,  double point2Joint4, double point2Joint5, double point2Joint6},  {double point3Joint1, double point3Joint2, double point3Joint3, |

|  |  |
| --- | --- |
|  | double point3Joint4, double point3Joint5, double point3Joint6}  标定用户坐标系所使用的法兰盘中心基于基坐标系的三个点关节角。  {double toolEndPosXForCalibUserCoord, toolEndPosYForCalibUserCoord,  toolEndPosZForCalibUserCoord}  标定用户坐标系所使用的工具末端位置参数。选传参数， 当使用法兰盘中心 标定用户坐标系时可以不传递该参数或者传递为(0,0,0)；当使用工具标定用  户坐标系时为必传参数。  注： 当参考系为基坐标系时， 该参数为必不传参数。 |
| 返回 | {posX, posY, posZ},  {oriW, oriX, oriY, oriZ}  工具末端基于参考坐标系的位姿 |
| 示例 | pos,ori = base\_to\_user(get\_global\_variable("Realtime\_Waypoint"), {0.1,0.2,0.3},  {1,0,0,0}, get\_user\_coord\_param("coord\_name"))  print(string.format("%6.6f,%6.6f,%6.6f,%6.6f,%6.6f,%6.6f,%6.6f",pos[1],  pos[2], pos[3], ori[1], ori[2], ori[3], ori[4])) |

|  |  |
| --- | --- |
| **void add\_waypoint(**  **{double joint1Angle, double joint2Angle, double joint3Angle,**  **double joint4Angle, double joint5Angle, double joint6Angle}**  **)** | |
| 功能 | 向全局路点列表中添加路点， 服务于 move\_track 函数 |
| 参数 | 类型 table |
| 返回 | 无 |
| 示例 | add\_waypoint({-0.000003,-0.127267, - 1.321122, 0.376934, - 1.570796,-0.000008}) |

|  |  |
| --- | --- |
| **void clear\_global\_waypoint\_list(void)** | |
| 功能 | 清除全局路点列表 |
| 参数 | 无 |
| 返回 | 无 |
| 示例 | clear\_global\_waypoint\_list()  说明：在多次使用 move\_track 时，需要清除上一次的轨迹路点 |

|  |  |
| --- | --- |
| **void move\_track(MoveTrackType trackType, bool isBlock)** | |
| 功能 | 轨迹运动， 根据全局路点列表（通过 add\_waypoint 函数添加） |
| 参数 | trackType 轨迹类型。  [参考枚举类型章](#bookmark35)节包括圆弧、圆、 moveP 、B 样条等。 |
| isBlock 运动阻塞标志位。  为 true 时，接口阻塞直到运动至目标路点；为 false，接口立即返回。 |
| 返回 | 无 |
| 说明 | 与 add\_waypoint 合用，例如轨迹有 3 个点， 先执行 add\_waypoint 加入 3 个  点，再执行  目前轨迹运动类型支持圆弧/圆， moveP ，参见枚举类型 |
| 示例 | --set tool parms  set\_tool\_kinematics\_param({0.111100, 0.222000, 0.333000}, {1.000000, 0.000000, 0.000000, 0.000000})  set\_tool\_dynamics\_param(0, {0, 0, 0}, {0, 0, 0, 0, 0, 0})  --init var  offset = 0  direction = 1 -- 1:Forward - 1:Reverse  --Move to ready point  init\_global\_move\_profile()  set\_joint\_maxvelc({1.298089,1.298089,1.298089,1.555088,1.555088,1.555088}) set\_joint\_maxacc({8.654390,8.654390,8.654390,10.368128,10.368128,10.36812 8})  move\_joint({0.208890, -0.044775, - 1.246891, 0.368688, - 1.570800, 0.208869}, true)  while (true) do  --Move to the first track point  init\_global\_move\_profile()  set\_end\_maxvelc(1.000000)  set\_end\_maxacc(1.000000)  set\_relative\_offset({offset \* 0.05, 0, 0}, CoordCalibrateMethod.zOzy, {-  0.000003, -0.127267, - 1.321122, 0.376934, - 1.570796, -0.000008}, {-0.244530, - 0.169460, - 1.356026, 0.384230, - 1.570794, -0.244535}, {-0.196001, 0.070752, - 1.129614, 0.370431, - 1.570795, -0.196006}, {0.111100, 0.222000, 0.333000})  move\_line({0.208890, -0.044775, - 1.246891, 0.368688, - 1.570800, 0.208869},  true) |

|  |  |
| --- | --- |
|  | --Move cir  init\_global\_move\_profile()  set\_end\_maxvelc(1.000000)  set\_end\_maxacc(1.000000)  set\_relative\_offset({offset \* 0.05, 0, 0}, CoordCalibrateMethod.zOzy, {-  0.000003, -0.127267, - 1.321122, 0.376934, - 1.570796, -0.000008}, {-0.244530, - 0.169460, - 1.356026, 0.384230, - 1.570794, -0.244535}, {-0.196001, 0.070752, - 1.129614, 0.370431, - 1.570795, -0.196006}, {0.111100, 0.222000, 0.333000})  add\_waypoint({0.208890, -0.044775, - 1.246891, 0.368688, - 1.570800, 0.208869})  add\_waypoint({-0.237646, -0.169014, - 1.355669, 0.384145, - 1.570793, - 0.237655})  add\_waypoint({-0.000009, 0.087939, - 1.110852, 0.372015, - 1.570793, - 0.000007})  set\_circular\_loop\_times(0)  move\_track(MoveTrackType.ARC\_CIR, true)  if (offset >= 2) then  direction = - 1  elseif (offset <= 0) then  direction = 1  end  offset = offset + direction;  end |

|  |  |
| --- | --- |
| **void set\_circular\_loop\_times(int times)** | |
| 参数 | times 圆运动的圈数，与 move\_track 一起使用.  times=0 时，为圆弧运动， times>0 时，为圆运动。 |
| 返回 | 无 |
| 示例 | 详见 move\_track 示例 |

|  |  |
| --- | --- |
| **void set\_blend\_radius(double blendRadius)** | |
| 功能 | 设置交融半径 |
| 参数 | blendRadius 为交融半径，单位米 |
| 返回 | 无 |
| 示例 | set\_blend\_radius(0.01) |

|  |  |
| --- | --- |
| **void set\_arrival\_ahead\_distance\_mode(double distance)** | |
| 功能 | 设置提前到位距离模式 |
| 参数 | distance 提前到位距离 |
| 返回 | 无 |
| 示例 | set\_arrival\_ahead\_distance\_mode (0.01) |

|  |  |
| --- | --- |
| **void set\_arrival\_ahead\_time\_mode(double time)** | |
| 功能 | 设置提前到位时间模式 |
| 参数 | time 提前到位时间 |
| 返回 | 无 |
| 示例 | set\_arrival\_ahead\_time\_mode(0.01) |

|  |  |
| --- | --- |
| **void set\_arrival\_ahead\_blend\_mode(double blendRadius)** | |
| 功能 | 设置提前到位交融半径模式 |
| 参数 | blendRadius 交融半径 |
| 返回 | 无 |
| 示例 | set\_arrival\_ahead\_blend\_mode (0.01) |

|  |  |
| --- | --- |
| **void set\_robot\_collision\_class(int collisionClass)** | |
| 功能 | 设置碰撞等级 |
| 参数 | collisionClass 0~10 |
| 返回 | 无 |
| 示例 | set\_robot\_collision\_class(6) |

|  |  |
| --- | --- |
| **void set\_tool\_kinematics\_param(**  **{double posX, double posY, double posZ}, {double oriW = 1,double oriX = 0, double**  **oriX = 0, double oriX = 0}**  **)** | |
| 功能 | 设置工具运动学参数，姿态可以不传递 |
| 参数 | 两个 table，第一个 table 为位置必传，第二个 table 为姿态，可以不传。 |
| 返回 | 无 |
| 示例 | set\_tool\_kinematics\_param（{0.1,0.2,0.3}） |

|  |  |
| --- | --- |
| **{double toolEndPosX, toolEndPosY, toolEndPosZ},**  **{double toolEndOriW, toolEndOriX, toolEndOriY, toolEndOriZ}**  **get\_tool\_kinematics\_param(string toolName)** | |
| 功能 | 获取工具运动学参数 |
| 参数 | toolName 工具名称，需要与示教器工具标定界面中的工具名称一致 |
| 返回 | 返回工具末端位置参数，工具末端姿态参数 |
| 示例 | get\_tool\_kinematics\_param("toolName") |

|  |  |
| --- | --- |
| **void set\_tool\_dynamics\_param(**  **double payload,**  **{double gravityCenterX, double gravityCenterY, double gravityCenterZ},**  **{double inertiaXX = 0, double inertiaXY = 0, double inertiaXZ = 0, double inertiaYY = 0, double inertiaYZ = 0, double inertiaZZ = 0})** | |
| 功能 | 设置工具动力学参数， 负载和重心 xyz 必传，惯量可以不传 |
| 参数 | payload 负载 单位 kg  重心 table  惯量 table |
| 返回 | 无 |
| 示例 | set\_tool\_kinematics\_param（3 ，{0.1,0.2,0.3}） |

|  |  |
| --- | --- |
| **CoordCalibrateMethod coordCalibrateMethod,**  **{double point1Joint1, double point1Joint2, double point1Joint3,**  **double point1Joint4, double point1Joint5, double point1Joint6},**  **{double point2Joint1, double point2Joint2, double point2Joint3,**  **double point2Joint4, double point2Joint5, double point2Joint6},**  **{double point3Joint1, double point3Joint2, double point3Joint3,**  **double point3Joint4, double point3Joint5, double point3Joint6},**  **{double toolEndPosXForCalibCoord, toolEndPosYForCalibCoord,**  **toolEndPosZForCalibCoord}**  **get\_user\_coord\_param(string userCoordName)** | |
| 功能 | 获取用户坐标系参数 |
| 参数 | userCoordName 用户坐标系名称  需要与示教器用户坐标系标定界面中的用户坐标系名称一致 |
| 返回 | 返回用户坐标系参数（包括：  标定用户坐标系的方法枚举；  标定用户坐标系所使用的法兰盘中心基于基坐标系的三个点关节角；  标定用户坐标系所使用的工具末端位置参数。  ) |
| 示例 | get\_user\_coord\_param("userCoordName ") |

|  |  |
| --- | --- |
| **void robot\_pause(void)** | |
| 功能 | 机械臂暂停运动。当且仅当机械臂处于运动状态时，才可以调用。 |
| 参数 | 无 |
| 返回 | 无 |
| 示例 | robot\_pause() |

|  |  |
| --- | --- |
| **void robot\_continue(void)** | |
| 功能 | 机械臂恢复运动。当且仅当机械臂处于暂停状态时，才可以调用。 |
| 参数 | 无 |
| 返回 | 无 |
| 示例 | robot\_continue() |

|  |  |
| --- | --- |
| **void robot\_slow\_stop(void)** | |
| 功能 | 机械臂缓停。当且仅当机械臂处于运动状态时，才可以调用。 |
| 参数 | 无 |
| 返回 | 无 |
| 示例 | robot\_slow\_stop() |

|  |  |
| --- | --- |
| **void robot\_fast\_stop(void)** | |
| 功能 | 机械臂急停。当且仅当机械臂处于运动状态时，才可以调用。 |
| 参数 | 无 |
| 返回 | 无 |
| 示例 | robot\_fast\_stop() |

## 3.3. 内部模块

|  |  |
| --- | --- |
| **void sleep(double second);** | |
| 功能 | 睡眠等待， |
| 参数 | second ，等待时间，单位秒 |
| 返回 | 无 |
| 示例 | sleep(0.1) |

注： **IO** 名称参照“控制柜标配 **IO** 名称”章节

|  |  |
| --- | --- |
| **void set\_robot\_io\_status(RobotIOType ioType, string name, double value)** | |
| 功能 | 设置机械臂本体 IO 状态 |
| 参数 | ioType 表示 IO 类型, 枚举类型参考上文。 |
| name 表示 IO 名称, 字符串类型 |
| value IO 状态值，double 类型 |
| 返回 | 无 |
| 说明 | 机械臂本体标配 IO 名称请参考示教器 V4 版本的 IO 设置界面 |
| 示例 | set\_robot\_io\_status (RobotIOType.RobotBoardUserDI," U\_DO\_00",1) |

|  |  |
| --- | --- |
| **double get\_robot\_io\_status(RobotIOType ioType, string name)** | |
| 功能 | 获取机械臂本体 IO 状态 |
| 参数 | ioType 表示 IO 类型, 枚举类型参考上文。 |
| name 表示 IO 名称, 字符串类型 |
| 返回 | 对应 IO 的状态值， double 类型 |
| 示例 | a= get\_robot\_io\_status (RobotIOType.RobotBoardUserDI," U\_DI\_00")  print(a) |

|  |  |
| --- | --- |
| **void set\_tool\_power\_voltage(ToolPowerType toolPowerType)** | |
| 功能 | 设置工具电源电压 |
| 参数 | toolPowerType 工具电源电压枚举值，参考枚举类型章节。 |
| 返回 | 对应 IO 的状态值， double 类型 |
| 示例 | set\_tool\_power\_voltage(ToolPowerType.OUT\_ 12V) |

|  |  |
| --- | --- |
| **void init\_global\_variables(string varNameList)** | |
| 功能 | 初始化示教器全局变量值（变量配置界面中的变量值） |
| 参数 | varNameList 变量名称列表字符串， 以逗号为分隔符，例如： " varName1,  varName2 "。  如果该参数为空，则初始化全部的示教器变量值 |
| 返回 | 无 |
| 示例 | init\_global\_variables("varName1, varName2") |

|  |  |
| --- | --- |
| **variant get\_global\_variable(string varName)** | |
| 功能 | 获取示教器全局变量值 |
| 参数 | varName 变量名称 |
| 返回 | 对应变量名称的变量值，返回值类型取决于变量的类型 |
| 示例 | var= get\_global\_variable ("varName ")  print(var) |

**注：示教器 中 内 置 了 一 个 特 殊 的 实 时 路 点 变 量 , 名称为 :**

**"Realtime\_Waypoint".**

**该变量只允许获取信息，通过如下调用形式获取当前机械臂的实时路点信息， get\_global\_variable("Realtime\_Waypoint")**

**返回值是当前实时路点的关节角{joint1Angle, joint2Angle, joint3Angle,**

**joint4Angle, joint5Angle, joint6Angle}。**

|  |  |
| --- | --- |
| **void set\_global\_variable(string varName, variant varValue)** | |
| 功能 | 设置示教器全局变量值 |
| 参数 | varName 变量名称 |
| varValue 变量值 类型根据实际变量类型传参,支持 bool、int、double 三种类  型。 |
| 返回 | 无 |
| 示例 | set\_global\_variable("varName ", 1)  set\_global\_variable("varName ", 1.1)  set\_global\_variable("varName ", true) |

## 3.4. 扩展模块

### 3.4.1. Modbus

|  |  |
| --- | --- |
| **double get\_modbus\_io\_status(string ioName)** | |
| 功能 | 获取 modbus IO 状态 |
| 参数 | ioName  Modbus IO 名称（由示教器扩展→外设→Modbus →IO Config 定义） |
| 返回 | modbus IO 状态 |
| 示例 | get\_modbus\_io\_status("M\_DO\_0") |

|  |  |
| --- | --- |
| **void set\_modbus\_io\_status(string ioName, double ioValue)** | |
| 功能 | 设置 modbus IO 状态 |
| 参数 | ioName  Modbus IO 名称（由示教器扩展→外设→Modbus →IO Config 定义） |
| ioValue  Modbus IO 状态 |
| 返回 | 无 |
| 示例 | set\_modbus\_io\_status("M\_DO\_0", 1) |

### 3.4.1. PLC

|  |  |
| --- | --- |
| **double get\_plc\_io\_status(string ioName)** | |
| 功能 | 获取 PLC IO 状态 |
| 参数 | ioName  PLC IO 名称（由示教器扩展→外设→Plc →IO Config 定义） |
| 返回 | PLC IO 状态 |
| 示例 | get\_plc\_io\_status ("P\_DO\_0") |

|  |  |
| --- | --- |
| **void set\_plc\_io\_status(string ioName, double ioValue)** | |
| 功能 | 设置 PLC IO 状态 |
| 参数 | ioName  PLC IO 名称（由示教器扩展→外设→Plc →IO Config 定义） |
| ioValue  PLC IO 状态 |
| 返回 | 无 |
| 示例 | set\_plc\_io\_status ("P\_DO\_0", 1) |

## 3.5. TCP通信

### 3.5.1. TCP接口说明

TCP 接口使用闭包封装的方式向用户暴露有关TCP 服务器与客户端的接口， 其中 TCP 服务器接口统一使用包名"tcp.server" ；TCP 客户端接口统一使用包名 "tcp.client"。调用 TCP 的有关接口时，需要在接口函数之前添加包名，例如

tcp.server.func\_name(param), tcp.client.func\_name(param)。

### 3.5.2. TCP服务器

TCP 服务器接口均在"tcp.server"包下。

|  |  |
| --- | --- |
| **void listen(int port)** | |
| 功能 | TCP 服务器监听端口 port |
| 参数 | port：端口号 |
| 返回 | 无 |
| 示例 | tcp.server.listen(8888) |

|  |  |
| --- | --- |
| **bool is\_connected(string IP)** | |
| 功能 | 判断地址为 IP 的客户端是否与本地服务器建立连接 |
| 参数 | IP：IP 地址 |
| 返回 | 如果参数中的 IP 已经连接了 TCP 服务器，则返回 true；否则返回 false。 |
| 示例 | while (tcp.server.is\_connected(ip) ~= true) do  sleep(1)  end  print("connection succeeded") |

|  |  |
| --- | --- |
| **string recv\_str\_data(string IP)** | |
| 功能 | 本地服务器以字符串形式接收来自地址为 IP 的客户端的数据 |
| 参数 | IP：IP 地址 |
| 返回 | 返回接收到的数据 |
| 示例 | recv=tcp.server.recv\_str\_data("<127.0.0.1>")  print(recv) |

|  |  |
| --- | --- |
| **table recv\_asc\_data(string IP)** | |
| 功能 | 本地服务器以 **ASCII** 码形式接收来自地址为 IP 的客户端的数据 |
| 参数 | IP：IP 地址 |
| 返回 | 返回接收到的数据，返回值格式为一维的 table ，key 取默认值（从 1 开始） |
| 示例 | recv=tcp.server.recv\_asc\_data("<127.0.0.1>") |

|  |  |
| --- | --- |
| **void send\_str\_data(string IP, string msg)** | |
| 功能 | 本地服务器以字符串形式向地址为 IP 的客户端发送消息 msg |
| 参数 | IP：IP 地址； msg：发送的消息 |
| 返回 | 无 |
| 示例 | tcp.server.send\_str\_data("<127.0.0.1>", "Helloworld") |

|  |  |
| --- | --- |
| **void send\_asc\_data(string IP, table msg)** | |
| 功能 | 本地服务器以 **ASCII** 码形式向地址为 IP 的客户端发送消息 msg |
| 参数 | IP：IP 地址 |
| msg：发送的消息，格式为一维的 table ，key 取默认值（从 1 开始）。 |
| 返回 | 无 |
| 示例 | tcp.server.send\_str\_data("<127.0.0.1>", "Hello ")  world = {string.byte("world",1), string.byte("world",2), string.byte("world",3),  string.byte("world",4), string.byte("world",5)}  tcp.server.send\_asc\_data("<127.0.0.1>", world) |

|  |  |
| --- | --- |
| **void close(void)** | |
| 功能 | 本地服务器停止监听并断开所有已经建立的连接 |
| 参数 | 无 |
| 返回 | 无 |
| 示例 | tcp.server.close() |

## 3.6. 通用脚本接口

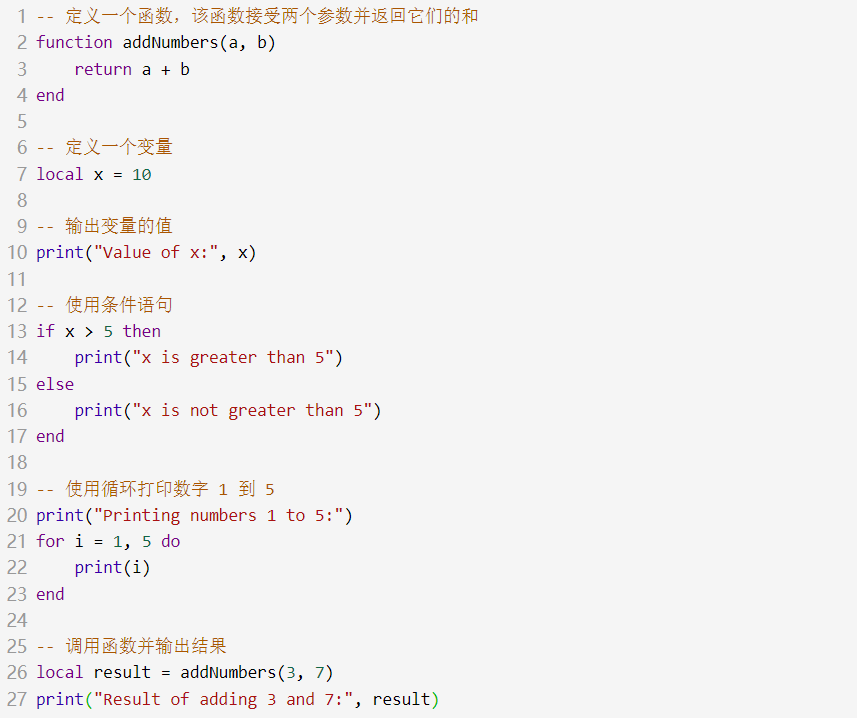
|  |  |
| --- | --- |
| **variable script\_common\_interface(string pluginName, string data)** | |
| 功能 | 参见《脚本&插件 通用接口文档》 |
| 参数 | pluginName 插件的名称，由插件依赖关系文件中的 TP\_PLUGIN\_NAME 定  义 |
| data 输入参数， 函数不对 data 做任何解析， 仅根据 pluginType 参数调用相对  应插件的插件通用接口，并将 data 作为输入参数。 |
| 返回 | 任意类型 |

# **应用案例**

## 4.1. 基础案例

功能描述：

这个示例展示了一些基本的Lua语法，包括变量、条件语句、循环、以及函数定义和调用。



## 4.2. TCP通信案例

功能描述：

该例子通过ASCII 码形式接收来自客户端的数据，如果接收到的数据为"end"则退出程序。





## 4.3. 调用接口案例

功能描述：

参考坐标系为法兰盘中心，工具末端位置参数为(0,0,0.2)，通过位置与姿态的相对偏移画一个五角星图案。

