UARM 创意百宝箱

(uArm Creator Studio, UCS)

用户手册

2016.11

腾讯视频 UCS 视频教程

可点击目录直接跳到对应章节

目录

Introduction

uArm 创意百宝箱（UCS）是什么？

uArm 创意百宝箱（英文名：uArm Creator Studio，简称 UCS）是服务于每位 uArm 用户的图形化操作软件。无论你是机器人小白，还是技术超群的极客，都可以使用它来轻松操作机械臂，分享你的开发成果，甚至让机械臂“看见”。

从“小白”到“大神”，UCS 全程陪伴

uArm 既是一种可视化编程语言（VPL），让你绕过繁琐的代码，又是一个支持 Python 编程的集成开发环境（IDE）。 通过这个软件，你可以：

- “手把手”拖拽示教，零门槛操作 uArm
- 拼接功能模块，快速让 uArm 做酷炫的事
- 直接用 Python 编程，用技术探索无限可能

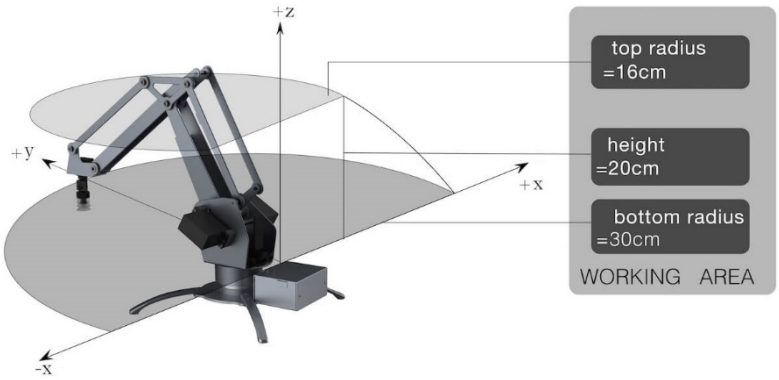
无需编程，让机械臂“看见”

UCS 内置视觉功能，你只需连接摄像头即可让机械臂“看见”，在视觉的辅助下更灵活地适应环境。

入门

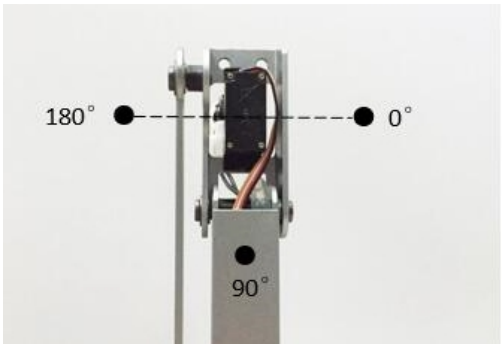
基本概念

1) uArm 的三维坐标



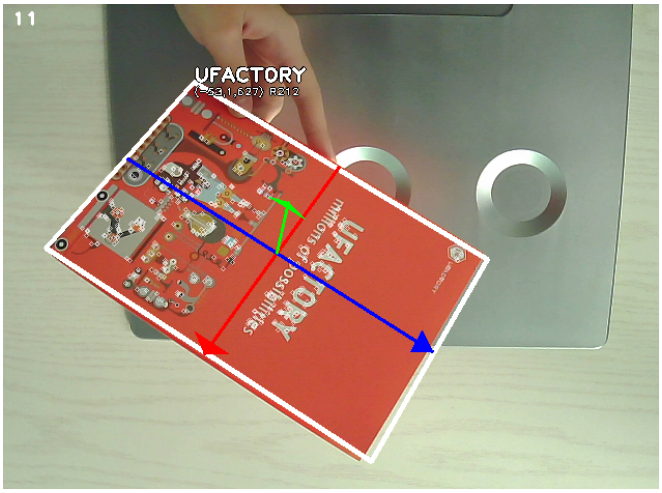
2) 吸盘的旋转角度

90°



你可将吸盘旋转到 0° - 180° 之间

3) 可识别物体的三维坐标



蓝箭头: X 轴 / 红箭头: Y 轴 / 绿箭头: Z 轴

连接 uArm

1. 在使用 UCS 之前，请确保:

1) 你的 uArm 已通电且通过 USB 连接到电脑。
2) 如果你是 Windows 用户，请确保你的电脑已安装 uArm 驱动。
   请参考 uArm 软件安装说明
3) 你的 uArm 固件已是最新版本。

注：
第一次使用 UCS 前，请先双击安装包中的固件程序。
该程序会自动刷新 uArm 固件，让 UCS 顺利识别你的 uArm。

2. 连接 uArm:

1）点击工具栏的“设备”—“查找 uArm”

2）选择 uArm 对应的端口，点击“应用”，然后等待 10 秒。
若页面出现多个端口，这属于正常现象，是由于你的电脑连接了其它硬件所致。试着连接各端口以找到 uArm 在你电脑上的专属端口。

“设备”按钮处会自动显示连接状态。
上面为 uArm 连接，下面为摄像头连接。

寻求帮助：

- 点击 UCS 菜单栏“Bug 提交”，提交你的问题
- 在 UFACTORY 官方论坛上提问

图形化编程

基本介绍

“任务”、“事件”、“命令”

1）命令：单个指令/动作

如：移动、吸取等等。后面的章节会详细解释每个命令的内涵。

2）事件：触发命令的动作。

举例：

当按下键盘上的 A（事件：按键 A）时，

让 uArm 从位置 A 移动到位置 B（“按键 A 命令列表”中的命令）。

你可以在不同的事件中加入不同的命令，
这样就可以同时用多种方式控制 uArm 了！

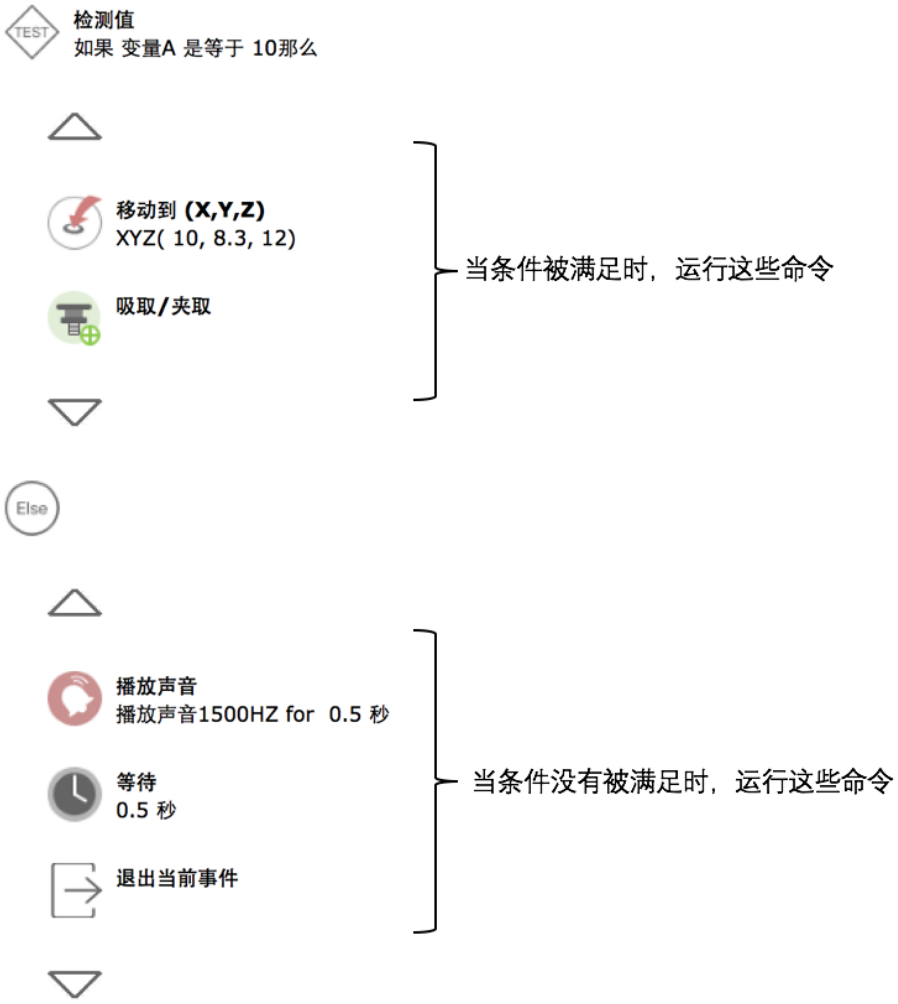
3）任务：所有事件（包括事件对应的命令）的集合。

你用 UCS 做出来的每个成果（包括所有事件和命令）都是一个“任务”。

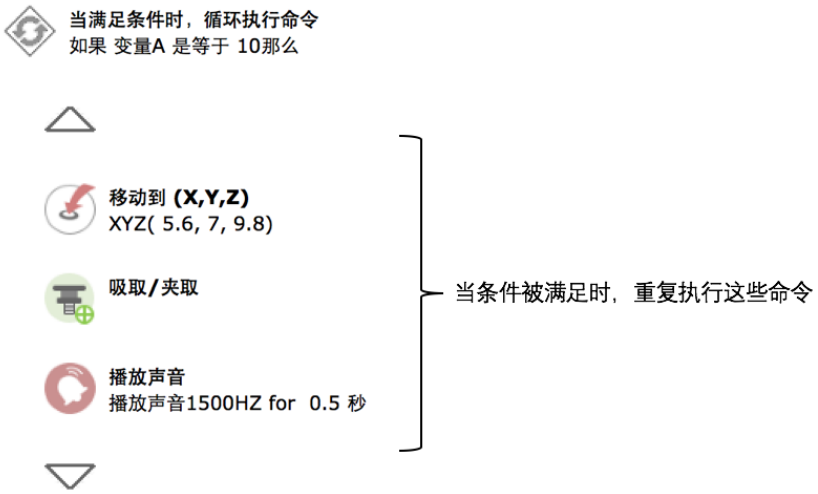
当你点击“保存”后，所有任务都会以“.task”文件保存在你的文件夹内。

编程逻辑（条件、循环）

1. 条件命令（if...else...）



2. 循环命令



命令

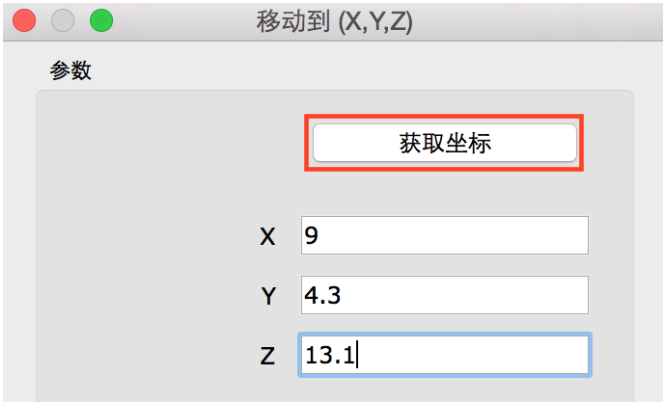
基本命令

移动到位置 (x,y,z)

让 uArm 末端移动到指定位置。

设置“目的地”的方法：

1. 用手直接把 uArm 末端拖动到指定位置，然后点击“获取坐标”。



2. 手动填写 x, y, z 的值（单位：厘米）。

X Y Z	1. 参数值可以是数字、自定义的变量，或 Python 表达式。 2. 若不想改变现有的坐标值，可选择不填。 （如：不填写 x 值，则默认保持 x 轴位置不变）
相对位置	勾选“相对位置”模式后，uArm 以现有位置为起点，移动(x,y,z) **举例：** 相对位置模式下，设置(x,y,z)为（ ， ，5） （不填写 x 和 y 的值，只将 z 设置为 5） uArm 末端会保持现有的 x, y 不变，在原有位置上上升(z 轴增加)5。

Python 表达法：

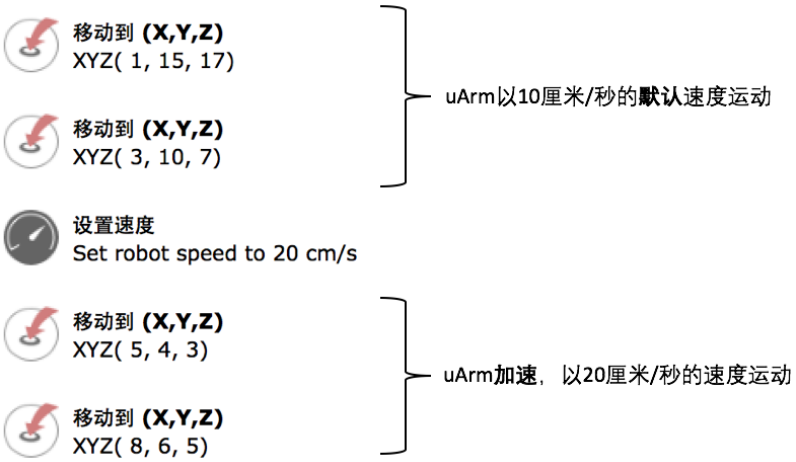
```
robot.setPos(x=0, y=15, z=15, relative=False)

#####   OR   #####

params = {"x":"0", "y":"15", "z":"15","relative": False}
MoveXYZCommand(env, interpreter, params).run()
```

（“相对位置”模式：把“relative”后面的“False”改为“True”）

设置速度

调整 uArm 运动的速度。

默认速度：10 厘米/秒
你可以通过填写其他数值，让 uArm 加速/减速。



Python 表达法

```
robot.setSpeed(10)

#####   OR   #####

SpeedCommand(env, interpreter, {"speed": "10"}).run()
```

设置吸盘角度

让 uArm 末端吸盘旋转到指定角度。

角度	可以填写数字、变量或表达式 范围：0 度－180 度
相对角度	勾选后，uArm 末端吸盘会在现有角度基础上继续旋转所填角度

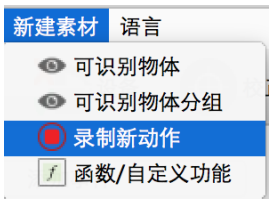
Python 表达法

```
robot.setServoAngles(servo3=90)

#####   OR   #####

params = {"angle": "90", "relative": False}
MoveWristCommand(env, interpreter, params).run()
```

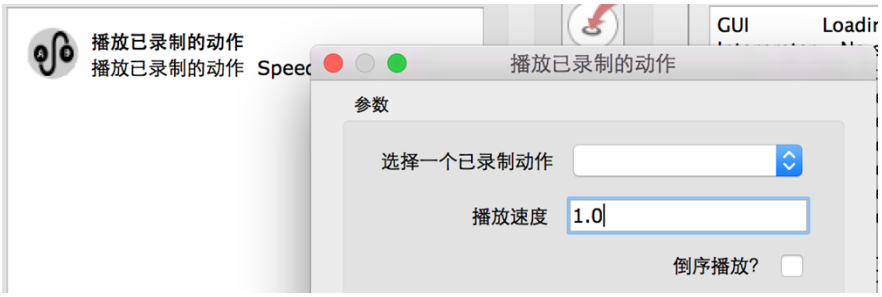
播放已录制的动作

录制/示教模式的使用方法：

1. 录制新动作



2. 播放已录制的动作
   将此图标拖进命令列表，选择你想要播放的动作，并设置播放速度。



选择一个已录制动作	选择你想播放的动作名称
播放速度	1.0：原始速度。 2：原始速度的两倍 0.5：原来速度的 1/2 以此类推
倒序播放	从后往前播放

Python 表达法

```
params = {"objectID":"NAME","speed": "1.0","reversed":False}
MotionRecordingCommand(env, interpreter, params).run()
```

锁定电机

锁定后，无法手动移动该电机控制的机械臂关节

底座电机	控制底座转动
后臂电机	面向机械臂背面，左侧电机
前臂电机	面向机械臂背面，右侧电机
末端电机	控制末端旋转

Python Equivalent

```
robot.setActiveServos(servo0=True,
    servo1=True,
    servo2=True,
    servo3=True)

#####   OR   #####

params = {"servo0":True,"servo1":True,"servo2":True,"servo3": True}
AttachCommand(env, interpreter, params).run()
```

解锁电机 Detach Servos

解锁后，可以手动移动该电机控制的机械臂关节

Python Equivalent

```
robot.setActiveServos(servo0=False, servo1= False, servo2= False,
servo3= False)

#####   OR   #####

params = {"servo0": True,"servo1":True,"servo2":True,"servo3": True}
DetachCommand(env, interpreter, params).run()
```

开启吸盘

开启 uArm 末端的吸盘

Python 表达法

```
robot.setGripper(True)

#####   OR   #####

GripCommand(env, interpreter).run()
```

关闭吸盘

关闭 uArm 末端的吸盘

Python 表达法

```
robot.setGripper(False)

#####   OR   #####

DropCommand(env, interpreter).run()
```

等待

在命令间设置等待时间（单位：秒）。

Python 表达法

```
sleep(TIME)  # Sleeps for three seconds, as an example

#####   OR   #####

WaitCommand(env, interpreter, {"time": "TIME"}).run()
```

播放声音

让 uArm（主控板上的蜂鸣器）发声。

频率	蜂鸣器发声的频率。数字越大，声音越大
发声时长	声音持续的时间（单位：秒）
Wait	If this is unchecked, the tone will start playing, but the panel will continue.

Python 表达法

```
robot.setBuzzer(1500, 1)  # Turns buzzer on at 1500 Hz for 1 second

#####   OR   #####

params = {"frequency": "1500", "time": "1", "waitForBuzzer": True}
BuzzerCommand(env, interpreter, params).run()
```

视觉命令

“摄像头－uArm”相对位置校正

在菜单栏点击“校正”，选择“摄像头－uArm”相对位置校正，根据页面指引完成校正。

请校正后再使用以下命令：

 移动到物体

 随物体旋转吸盘

 捡起物体

 条件：若物体的摆放角度为...

不需要校正就可以使用以下命令：

 条件：若看见某物体

 条件：若物体在指定区域内

移动到物体

让 uArm“看见”某物体后直接移动到该物体。

使用此功能前，请确保：

1） 已连接摄像头
2） 已完成“摄像头－uArm”相对位置校正
3） 已在素材库中创建可识别物体

选择一个物体	选择后，摄像头会实时追踪此物体，并识别物体的三维坐标
X	1. 可以填写数字、变量或表达式。 也可不填（相当于填“0”）。 2. uArm 会以该物体为原点，移动(x,y,z)。
Y	
Z	

举例：

(x,y,z) = (0,0,0)　　uArm 移动到物体正上方
(x,y,z) = (0,0,5)　　uArm 移动到物体上方 5 厘米处
(x,y,z) = (3,0,5)　　uArm 移动到物体右边 3 厘米，上方 5 厘米处

Python 表达法

```
params = {"objectID": "OBJECT NAME", "x": "0", "y": "0", "z": "0"}
MoveRelativeToObjectCommand(env, interpreter, params).run()
```

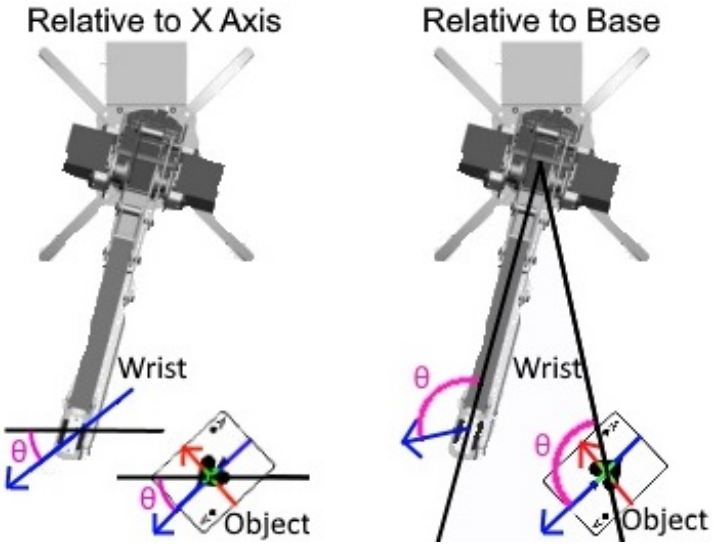
随物体旋转吸盘

选择你想让 uArm“看见”的物体，让 uArm 随物体的旋转来转动吸盘。
通过此功能，你可以让 uArm 将多个物体整齐叠放。
不熟悉“角度”概念？试着旋转物体，看看吸盘是怎么跟着动的，你很快就会摸索出此功能的使用方法！

使用此功能前，请确保：

1） 已连接摄像头
2） 已完成“摄像头－uArm”相对位置校正
3） 已在素材库中创建可识别物体

问：物体的“角度”是什么？
答：打开素材库，你会发现每个 uArm 能“看见”的物体都会有一个三维坐标（蓝－x 轴；红－y 轴；绿－z 轴）。



1）相对于 uArm X 轴的角度（左图）：
蓝色箭头（正方向）与 uArmX 轴的夹角。
2）相对于 uArm 底座的角度（右图）：
在 uArm 底座中点和该物体中点间连一条线，这条线与蓝色箭头（正方向）的夹角。

Python 表达法

```
params = {"objectID":"OBJECT NAME","angle":"0", "relToBase": False}
MoveWristRelativeToObjectCommand(env, interpreter, params).run()
```

捡起物体

让 uArm“看见”某物体后捡起物体。

使用此功能前，请确保：

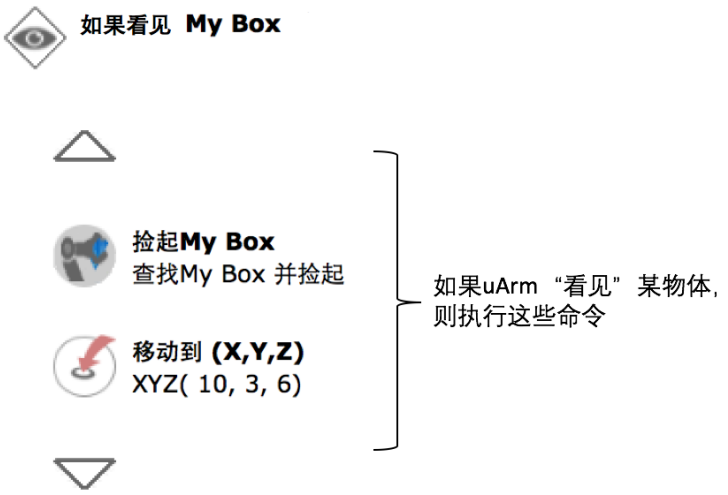
1） 已连接摄像头
2） 已完成“摄像头－uArm”相对位置校正
3） 已在素材库中创建可识别物体

Python 表达法

```
params = {"objectID":"OBJECT NAME"}
PickUpObjectCommand(env, interpreter, params).run()
```

给开发者的小提示：
“PickUpObjectCommand”是一个条件命令。
当 uArm 成功捡起物体时，返回“True”，否则返回“False”。
你可以设置让 uArm 在成功捡起时执行某些代码，否则执行其它代码

条件：若“看见”物体



使用此功能前，请确保：

1） 已连接摄像头

2） 已在素材库中创建可识别物体

如果“看见”：	选择一个需要识别的物体
可信度	建议选择“低”，即当摄像头模糊捕捉到该物体时就判断为“已看见”。 若选择 “高”，则当摄像头非常清晰地识别该物体时，才判断为“已看见”。
当	建议不调整，除非你想让摄像头记住它 1-2 秒前看到的图像。
如果没有“看见”	逆转条件，如果 uArm 没有识别到物体，就执行命令

Python 表达法

```
params = {"objectID": "OBJECT NAME", "age": 0, "confidence": 0,
"not": False}
test = TestObjectSeenCommand(env, interpreter, params)
if test.run():
    print("The Test Returned True!")

#####   OR   #####

trackable = resources.getObject("Object Name")
vision.addTarget(trackable)

# Somewhere later on in the script put this, so that vision has time
to recognize the object
tracked = vision.searchTrackedHistory(trackable=trackable, maxAge=0,
```

```
minPoints=30)
if tracked is not None:
    print("The Object Was Seen!")
```

条件：若物体在指定区域内

选择要识别的物体，用鼠标拖拽出摄像头视野范围内的区域。
当 uArm 看见物体在指定区域内时，执行命令。

使用此功能前，请确保：

1） 已连接摄像头
2） 已在素材库中创建可识别物体

Python 表达法

```
params = {"objectID": "NAME",
          "location":[[0, 0],[100,100]],
          "part": "any",
          "not": False}

test = TestObjectLocationCommand(env, interpreter, params)
if test.run():
    print("The Test Returned True!")
```

条件：若物体的摆放角度为.....

让 uArm “看见”某物体，若该物体的摆放角度在某个范围内（如：30－60 度之间），则运行“上下箭头”之间的命令块。

问：什么是物体的角度？
答：以 uArm X 轴正方向为起点，即 0 度，逆时针旋转至物体坐标的蓝箭头（正方向）所得到的角度。
如，当 uArm X 轴指向 3 点钟方向，物体的蓝箭头指向 12 点钟方向，则物体摆放角度为 90 度。

使用此功能前，请确保：

1） 已连接摄像头
2） 已完成“摄像头－uArm”相对位置校正
3） 已在素材库中创建可识别物体

Python 表达法

```
params = {"objectID": "NAME","start": "0", "end":"90", "not": False}

test = TestObjectAngleCommand(env, interpreter, params)
if test.run():
    print("Objects rotation is between 0, 90degrees from X axis!")
```

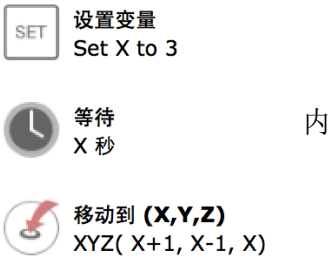
逻辑命令

设置变量

自定义一个变量。使用变量可以让你的程序变得更简单。

1. 变量可以在 UCS 中的所有文本框中使用。

2. 你可以为变量赋值，也可以在一个变量内包含其它变量。



3. 给开发者的提示：

1） 你在 UCS 图形化界面设置的变量可直接用于 UCS 内的 Python 编写窗口。反之亦然。

如：右图所示，Python 代码内设置了 X 变量，图形化命令可直接使用该变量。）

2） 变量内容可以是 Python 代码。

如：右图所示，设置 X 为一个 Python list，包含三个不同的值，然后在命令中分别调用不同的值。

变量名称	以字母开头，可由字母和数字组成，且不能有空格。
变量内容	可以是数字、字母、Python 代码（如：Python list）。也可包含其它变量。

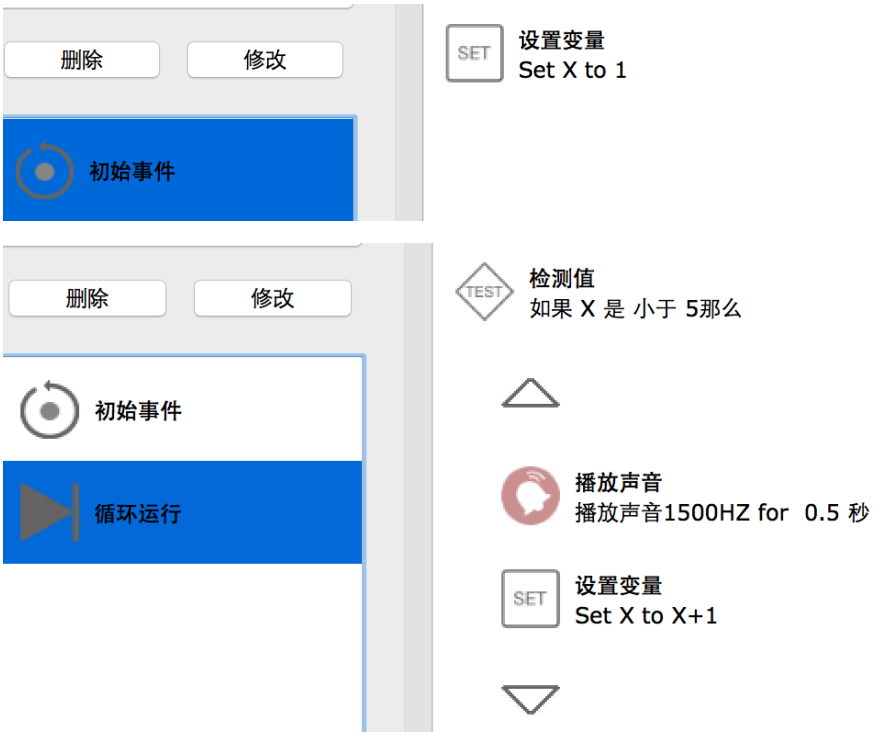
Python 表达法

```
VARIABLENAME = EXPRESSION
```

Test: 检测变量值

条件命令：检测变量的值，若满足条件，则执行接下来的命令。

举例：
如下图所示，设置 X 的初始值为 1。
当 X<5 时，uArm 发出 beep 声，且变量 X 的值＋1。
X=1: beep, X=1+1=2<5
…
X=4: beep, X=4+1=5，不符合“X<5”的条件，程序停止。
uArm 一共”beep”4 次。



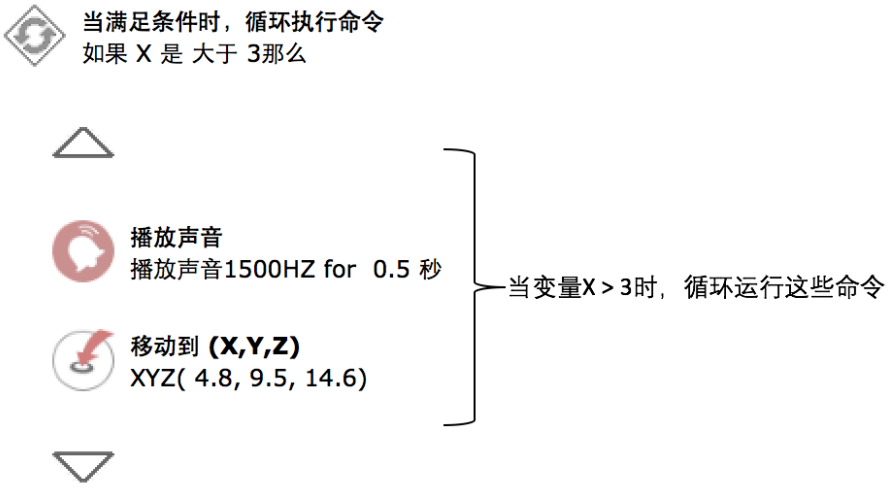
变量内容	数字、变量或 Python 代码
检测	检测的方式，包括：等于、不等于、大于、小于

Python 表达法

```
if EXPRESSION == EXPRESSION:
    print("Code runs here!")

# Other operators: >, <, !=
```

Loop: 当条件符合时，循环运行

当满足条件时，循环执行部分命令。用法和“检测变量值”类似。
但 Test 下的命令块只运行一次，Loop 下的命令块循环运行。



Python 表达法

```
while EXPRESSION:
    print("Running Code!")
```

Else：条件不符合时运行

通常和在条件命令（“看见/没看见”物体；物体在/不在指定区域内；检测变量值）之后使用。当条件不符合时，则运行之后的命令块。

结束事件

退出当前事件。其它事件继续运行。

结束任务

退出当前任务，该任务内的所有事件都停止运行。

命令块开始

命令块结束

自定义

运行 Python 代码

运行你在该窗口内编写的代码。

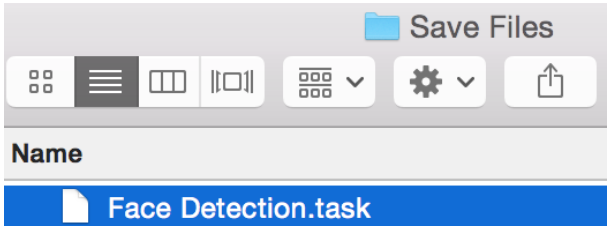
提示：

1. 无需另外安装 Python 环境，即可在该窗口直接编程。
2. UCS 为 Python 开发者提供了一些内置变量，点击编程窗口上方的“Python 编程指引”，可看到相关说明。
3. 你可以直接写代码，也可以直接导入任何标准的 Python 库。
4. 当你用 Python 设置过变量后，你可在软件的任何地方使用该变量

执行任务

在正在运行的任务 A 中调用任务 B（每个任务都可被保存为带有“.task”后缀的文件）。
即，在现有任务中嵌入以前保存过的任务(.task 文件)，别人开发的任务，等等。

使用次功能前，请确保：你的 UCS 主目录中已经有”.task”文件。
打开主目录 —— 打开“Save Files”文件夹 —— 查看已有 task 文件



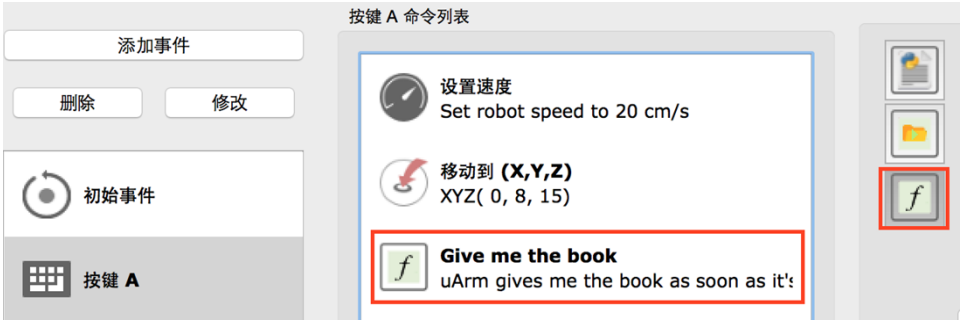
选择任务	选择要导入的任务文件
共享当前任务的变量	勾选后，现有任务 A 和被导入的任务 B 共享变量。 “共享变量”举例： 在任务 A 中执行任务 B， • 当任务 B 中已设置变量 X = 5，任务 A 可直接使用此变量，无需重新设置。 • 在任务 A 改变 X 的值(X = 10)，任务 B 中变量 X 的值也跟着改变。 若不想两个任务之间互相影响，请不要勾选。

Python 表达法

```
params = {"filename": "VALID FILENAME", "shareScope": False}
RunTaskCommand(env, interpreter, params).run()

'''
 Be careful when setting the filename. If the name has \t or \n
inside it, it will be interpreted # incorrectly. Use two slashes \\
on every slash in the filename to avoid this problem.
'''
```

执行自定义功能（Function）

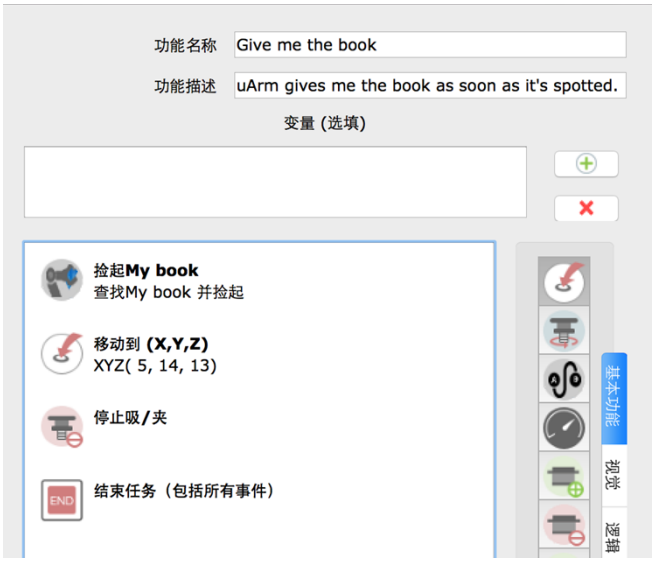
在现有任务中调用自定义功能。



问：这功能和“执行其它任务”有什么区别？

答：一个任务可以包含多个事件，而一个自定义功能只相当于一个事件。

1. 使用此功能前，请确保：
   你已在素材库中创建过自定义功能。



2. 自定义功能中的“变量”：
   你可在创建自定义功能时添加变量，每次调用时改动变量值。

举例：

1） 创建自定义功能“uArm 发声 X 次”（左图）

2） 每次调用该功能时，输入想要 uArm 发声的次数（下图）

Python 表达法

```
params ={"objectID":"NAME","arguments":{"arg1":"val", "arg2":"val"}}
RunFunctionCommand(env, interpreter, params).run()

# If the function has no arguments, then set "arguments" to {}
```