固体力学实验-动力机器人

Python入门与机器人描述文件

主讲老师：王宏涛 教授

助教：邵烨程 博士研究生

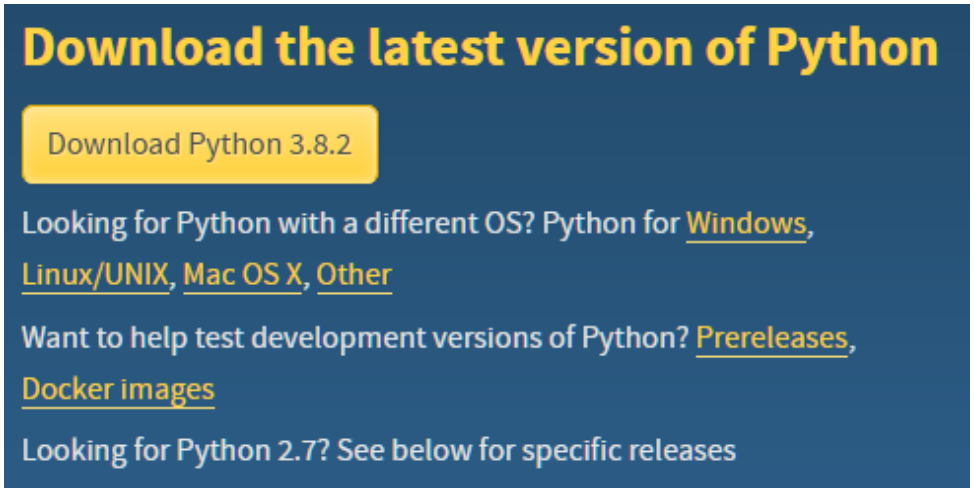
主要内容

- Python语言基础
  - 环境配置
  - 语言基础
- 机器人物理建模
  - 使用URDF格式建立机器人模型
  - URDF详解
- 使用pybullet进行动力学仿真
  - Pybullet的安装与测试
  - Pybullet常用API解读
- 推荐学习资料

安装python

从python官网下载最新版python，并双击文件进行安装。

Windows系统建议右键-以管理员身份运行。**注意勾选添加python到路径。**



Life is short
you need Python

从matlab到python

MATLAB命令行

```
命令行窗口
>> 1+1

ans =

     2

>> 2*3

ans =

     6

>> a='hello';
>> disp(a)
hello
>>
```

Python交互模式

```
Python 3.7 (64-bit)
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:
30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
 information.
>>> 1+1
2
>>> 2*3
6
>>> a='hello'
>>> print(a)
hello
>>>
```

从matlab到python

MATLAB运行.m文件

编辑器 - C:\data\temp\hello.m

hello.m

```
% This is a comment.
a = 1;
b = 2;
c = a + b;
disp(c)
```

```
>> hello
     3
```

Python运行.py文件

hello.py - C:\data\temp\hello.py (3.7.6)

File Edit Format Run Options Window Help

```
a = 1
b = 2
c = a + b
print(c)
```

Ln: 5 Col: 0

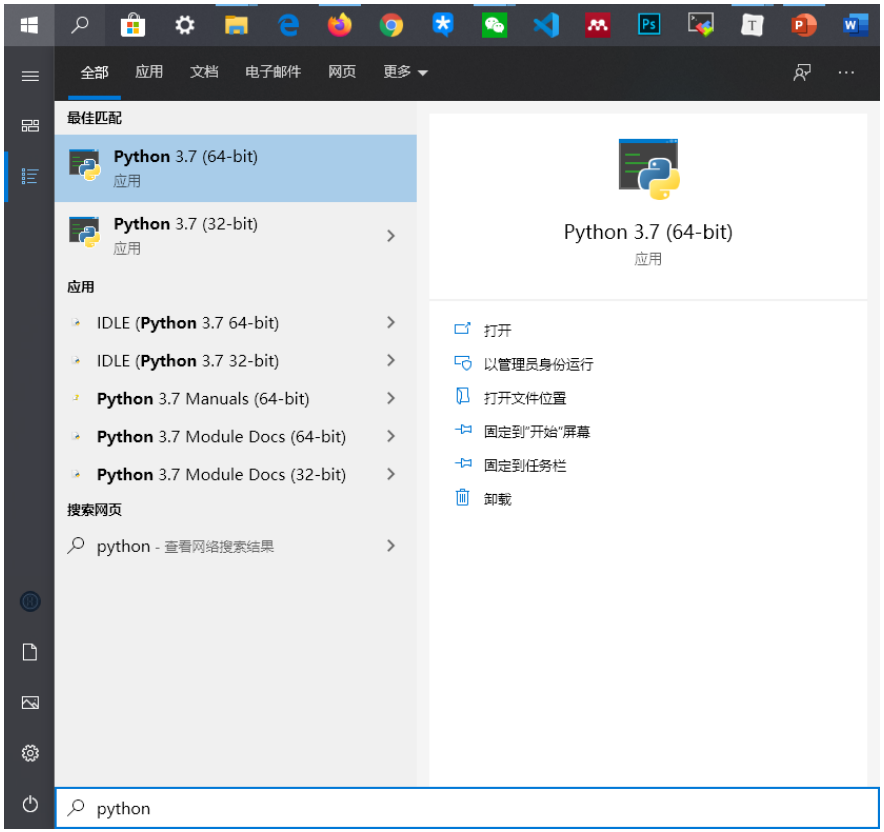
C:\Windows\System32\cmd.exe

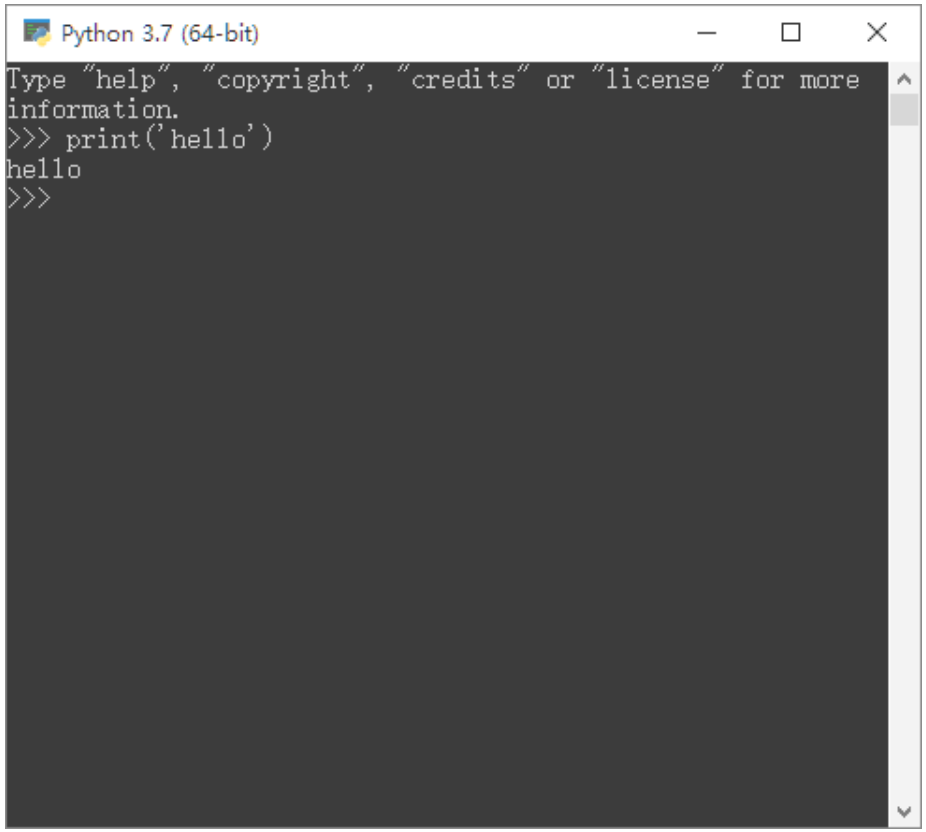
```
C:\data\temp>python hello.py
3
```

运行脚本文件时，基本等效于逐行执行文件。

使用python交互环境(Windows)

从windows打开python(使用win+s搜索)





使用python交互环境(macOS)

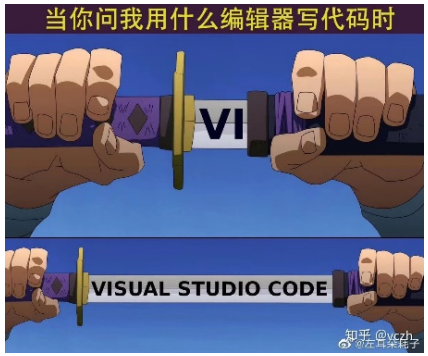
打开终端(command+空格，搜索terminal.app)

运行python

使用VS Code编写并运行python程序

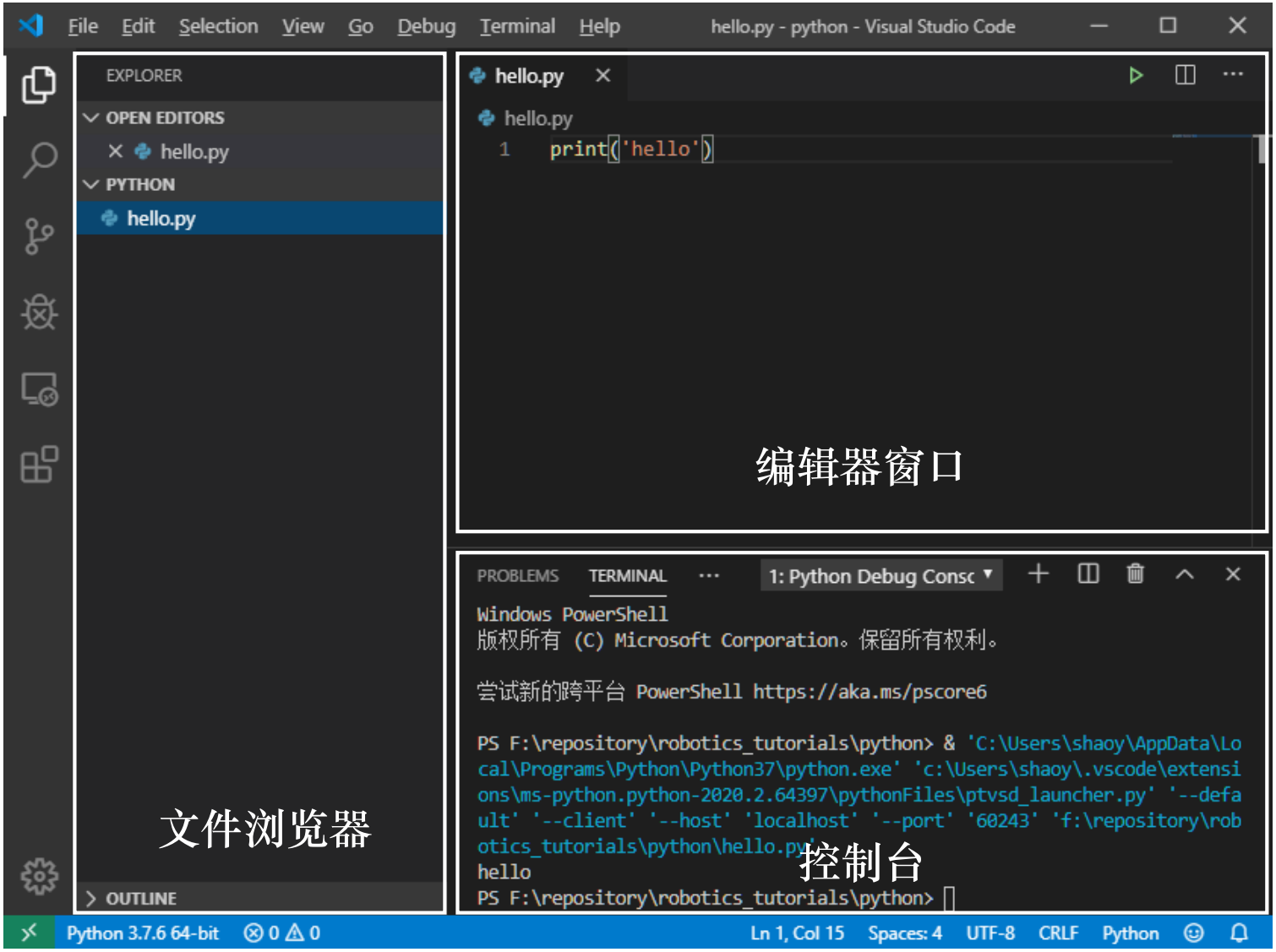
安装VS Code (https://code.visualstudio.com/)

安装python插件



使用VS Code编写并运行python程序

编写文件并使用F5运行



变量与数据类型

与MATLAB一样，python为动态类型。

创建变量时不需要声明类型，任何变量都可以print输出。

Python 3.7 (64-bit)

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 1
>>> print(a)
1
>>> a = 'name'
>>> print(a)
name
>>> a = [1, 2, 3]
>>> print(a)
[1, 2, 3]
>>> b = 1
>>> c = 2
>>> a = b + c
>>> print(a)
3
>>>
```

下载本部分代码 https://gitee.com/ycshao/robotics_tutorial

字符串与输出

Python单引号、双引号三引号都可以用于字符串。

使用单引号时，双引号会被识别成内容，反之亦然。

语法1：python一行为一句，没有分号。

```
a = 'a string'
b = "string"
c = '''line 1
line 2
line 3'''
d = '这是"双引号"'
e = "这是'单引号'"
f = '我print的时候没有加换行符'

变量名可以用中文 = '但是print自动换行了'

print(a)
print(b)
print(c)
print(d)
print(e)
print(a[0])
print(f)
print(变量名可以用中文)
```

输出

```
a string
string
line 1
line 2
line 3
这是"双引号"
这是'单引号'
a
我print的时候没有加换行符
但是print自动换行了
```

字符串与输出

对字符串进行+运算，会自动拼接两个字符串。

print()函数中使用逗号，会在同一行隔开一个空格进行输出。

''.format()格式化类似于C中的%d，%s等方式格式化。

f'{}'格式化时，会运算{}中的表达式。

```python
# 这个符号代表注释
# 在print中使用逗号时，会在同一行隔开一个空格输出
# 对字符串使用加法，会自动拼接字符串
a = 'str'
b = 'ing'
c = a + b
print(c)
print(a, b)

c = 233
print(a, c)

# format格式化
d = '{} is {}'.format(a, c)
print(d)
print('{} is {}'.format(a, c))
print('保留两位小数 {:.2f}'.format(c))
print('按照c a c的顺序输出\n{1} {0} {1}'.format(a, c))

# f-string格式化
d = f'这是更简单的方法：{a} {c} {c/10} {1+2}'
print(d)
```

输出

```
string
str ing
str 233
str is 233
str is 233
保留两位小数 233.00
按照c a c的顺序输出
233 str 233
这是更简单的方法：str 233 23.3 3
```

list

list为有序列表，可以存放任何相同或不同的内容
可以使用类似于MATLAB的方法切片读取list
对于可比较的内容，可以进行sort

```python
list1 = [1, 2, 4, 3]
list2 = ['name', 'score', 100]
print(list1)
print(list1[0])
print(list1[1:3]) # 左闭右开区间
print(list1[:2]) # 相当于MATLAB的start:1
print(list1[1:]) # 相当于MATLAB的1:end
print(list1[-1])

list1.sort()
print(list1)

list2.insert(1, 'C++')
print(list2)
list2.remove('score')
print(list2)
idx = list2.index('name')
print(idx)
list2.pop(0)
print(list2)

list1.append(5)
print(list1)
list1.extend(list2)
print(list1)
list1.append(list2)
print(list1)
```

输出

```
[1, 2, 4, 3]
1
[2, 4]
[1, 2]
[2, 4, 3]
3
[1, 2, 3, 4]
['name', 'C++', 'score', 100]
['name', 'C++', 100]
0
['C++', 100]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 'C++', 100]
[1, 2, 3, 4, 5, 'C++', 100, ['C++', 100]]
```

dict

dict为字典，存放**无序**的key-value对。

```
a = {'微积分': 93, 'Linear Algebra': 61, 'Mechanics': 100, '英语水平测试': 'pass'}
print(a)
print(a['微积分'])
a.pop('Linear Algebra')
print(a)
print('Linear Algebra' in a)
print(a['dota2'])
```

输出

```
{'微积分': 93, 'Linear Algebra': 61, 'Mechanics': 100, '英语水平测试': 'pass'}
93
{'微积分': 93, 'Mechanics': 100, '英语水平测试': 'pass'}
False
Exception has occurred: KeyError 'dota2'
```

tuple与set

tuple为不可变的list

set为只有key没有value的dict

```
tpl = (1, 2, 'hello')
print(tpl)
set1 = set([1, 3, 2])
set2 = set(['dota', 'dota', 'dota', 'lol'])
print(set1)
print(set2)
```

输出

```
(1, 2, 'hello')
{1, 2, 3}
{'lol', 'dota'}
```

条件判断与bool

语法2：python中没有end，没有大括号，使用冒号和缩进表示上下文。

语法3：可以使用空格或者tab来缩进，但是混用时解释器会报错。

语法4：pass为不执行任务的空语句，用来填充内容，避免语法错误。类似与C中的单独分号。

bool的关键字为True和False（注意大小写）

```python
a = 10
if a > 5:
    print(a)
else:
    print('a <= 10')

b = True
if not b:
    pass
else:
    print('False')

if a > 100:
    print('a > 100')
elif a > 5:
    print('a > 5')
elif a > 1:
    print('a > 1')
else:
    print('else')
```

输出

```
10
False
a > 5
```

多级缩进示例

```python
a = 10
if a > 5:
print(a)                # WRONG!!!!!!
    print(a)            # CORRECT
else:
    print('a <= 10')    # CORRECT
    if a < 10:
        print('2333')   # CORRECT
```

循环

Python中的for循环有很多高级功能。

```python
# range
for i in range(10):
    print(i, end=' ') # 取消换行

for i in range(5, 10):
    print(i, end=' ')

for i in range(5, 10, 2):
    print(i, end=' ')

# loop over a list
courses = ['Calculus', 'Linear Algebra', 'Mechanics']
# low-level
for i in range(3):
    print(courses[i])
# advanced
for course in courses:
    print(course)
# advanced
for i, course in enumerate(courses):
    print(f'The No.{i+1} course is {course}')

# list a range
mylist = list(range(1, 101))
print(mylist)
sum = 0
for i in mylist:
    sum += i
print(sum)
```

输出

```
0 1 2 3 4 5 6 7 8 9
5 6 7 8 9
5 7 9
Calculus
Linear Algebra
Mechanics
Calculus
Linear Algebra
Mechanics
The No.1 course is Calculus
The No.2 course is Linear Algebra
The No.3 course is Mechanics
[1, 2, 3, ..., 100]
5050
```

循环

Python中的while，break，continue等功能与其他语言类似

```python
sum = 0
n = 99
while n > 0:
    sum += n
    n -= 2
print(sum)

sum = 0
n = 99
while True:
    sum += n
    n -= 2
    if n < 0:
        break
print(sum)

sum = 0
n = 99
while n > 0:
    n -= 2
    if n > 50:
        continue
    sum += n
print(sum)
```

输出

```
2500
2500
624
```

函数

Python定义函数时不需要声明返还值的信息，可以返还多个值，为输入设置默认值，像MATLAB的隐式声明一样使用lambda表达式。

```python
def my_function(a):
    return a + a

print(my_function(100))
print(my_function('Hello'))

def my_second_function(a, c):
    b = a + c
    return a, b

c, d = my_second_function(1, 3)
print(c, d)

def my_third_function(name, age, major='unknown', language='python'):
    print(f"{name}'s age is {age}")
    print(f"{name}'s major is {major}")
    print(f'{name} codes in {language}')
    print('------------------------------')

my_third_function('Adam', 18)
my_third_function('Adam', 18, 'Mechanics')
my_third_function('Adam', 18, language='C++')
my_third_function(name='Adam', age=18, language='Rust', major='CS')

my_fourth_function = lambda x: x*x
print(my_fourth_function(11))
```

输出

```
200
HelloHello
1 4
Adam's age is 18
Adam's major is unknown
Adam codes in python
------------------------------
Adam's age is 18
Adam's major is Mechanics
Adam codes in python
------------------------------
Adam's age is 18
Adam's major is unknown
Adam codes in C++
------------------------------
Adam's age is 18
Adam's major is CS
Adam codes in Rust
------------------------------
121
```

模块的使用与常用模块

Python使用import语句调用其他模块，类似于C中的include语句。

可以对模块进行重命名，或者只调用模块中的某些内容

```python
import math
print(math.log10(100))

import math as m
print(m.log10(100))

from math import log10
print(log10(100))

from  myfunc import my_func
my_func(100)
```

输出

```
2.0
2.0
2.0
This is 100
```

myfunc.py

```python
def my_func(a):
    print(f'This is {a}')
```

numpy的安装与使用

pip-Python包管理工具

Python大部分模块都被托管在PyPI(Python Package Index)上

pip的操作主要在命令行中完成
(win+x搜索cmd/command+space搜索terminal.app)

建议使用清华大学开源镜像站的pip源

pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple

安装numpy（主流python矩阵计算库）

pip install --user numpy

使用numpy

```python
import numpy as np

a = np.array([[1, 2], [3, 4]])
print(a)
print(a[0, 0])

a = np.arange(6).reshape([2,3])
print(a)
print(a[:, 1])
```

输出

```
[[1 2]
 [3 4]]
1
[[0 1 2]
 [3 4 5]]
[1 4]
```

numpy for matlab users

numpy

```python
import numpy as np

a = np.arange(12).reshape([3, 4])

b = np.arange(8).reshape([4, 2])
# martix multiplication
c = np.matmul(a, b)
# element-wise multiplication
d = a*a
# bool index
d[a > 6] = 0
# sin
np.sin(a)
# shape
a.shape
# concatenate
np.hstack((a, b))
np.vstack((a, b))
```

matlab

```
a = 0:11;
a = reshape(a, [3, 4]);
b = 0:7;
b = reshape(b, [4, 2]);
% martix multiplication
c = a*b;
% element-wise multiplication
d = a.*a;
% bool index
d(d > 6) = 0;
% sin
sin(a);
% shape
size(a);
% concatenate
[a, b]
[a; b]
```

numpy指标从[0]开始，matlab指标从(1)开始。
numpy中乘法默认为元素乘法，matlab中乘法默认为矩阵乘法。

类和实例

类(class)和实例(instance)

类是抽象的概念，例如“学生”；实例是具体的对象，例如“小明同学”。

```
1   class Student(object):
2       pass
3
4   xiao_ming = Student()
5   xiao_ming.name = 'Xiao Ming'
6   xiao_ming.score = 99
7   print(f"{xiao_ming.name}'s score is {xiao_ming.score}")
8
9   xiao_ming.score = 60
10  print(f"{xiao_ming.name}'s score is {xiao_ming.score}")
```

1-2 定义Student类

4 创建实例xiao_ming

5-6 为实例添加属性

7 访问属性

9 修改属性

10 访问属性

输出

```
Xiao Ming's score is 99
Xiao Ming's score is 60
```

可以通过属性，将各种数据绑定到类。类似于字典或者C/MATLAB的结构体。

面向对象编程

类除了存储数据，还可以对那些数据进行高级操作，即有“功能”的结构体。

```
1   class Student(object):
2       def __init__(self, name, score_, age):
3           self.name = name
4           self.score = score_
5           self.age = age
6
7       def print_score(self):
8           print(f"{self.name}'s score is {self.score}")
9
10      def print_age(self):
11          print(f"{self.name}'s age is {self.age}")
12
13  xiao_ming = Student('Xiao Ming', 99, 18)
14  print(xiao_ming.name)
15  print(xiao_ming.score)
16  xiao_ming.print_score()
17  xiao_ming.print_age()
```

输出

```
Xiao Ming
99
Xiao Ming's score is 99
Xiao Ming's age is 18
```

2-5　构造函数，构建实例时执行。

在类的内部，用self指代类本身，程序第3行为类本身添加一个name属性，并且将函数传入的name赋值给类本身的name属性。

7-8 & 9-10 定义类的方法

类中所有函数的第一个传入值均为self，调用时不需要显式传入。

13　构造实例

构造xiao_ming。调用__init__函数，传入三个参数。

14-15访问xiao_ming的属性。

16-17使用xiao_ming的方法。

访问限制

双下划线开头的属性为私有属性，无法从类的外部访问。只能在类的内部进行操作。

```
1   class Student(object):
2       def __init__(self, name, score):
3           self.name = name
4           self.__score = score
5
6       def get_score(self):
7           return self.__score
8
9       def print_score(self):
10          print(f"{self.name}'s score is {self.__score}")
11
12  xiao_ming = Student('Xiao Ming', 99)
13  print(xiao_ming.name)
14  xiao_ming.print_score()
15
16  print(xiao_ming.__score)
17  print(xiao_ming.get_score())
```

4 创建私有属性__score

6-7 在类的内部访问私有属性

9-10 在类的内部访问私有属性

16 试图从类的外部访问私有属性

输出

```
Xiao Ming
Xiao Ming's score is 99
Exception has occurred: AttributeError
'Student' object has no attribute '__score'
99
```

君子之约
一般约定用单下划线开头表示名义上的私有成员，实际不私有。

继承

```python
class Student(object):
    def __init__(self, name, score):
        self.name = name
        self.__score = score

    def learn(self):
        print('Learn something')

    def print_score(self):
        print(f"{self.name}'s score is {self.__score}")

class MechanicsStudent(Student):
    def learn(self):
        print('Learn mechanics')

class PhysicsStudent(Student):
    def learn(self):
        print('Learn physics')

    def play(self, game):
        print(f'Play {game}')

xiao_ming = Student('Xiao Ming', 99)
xiao_qiang = MechanicsStudent('Xiao Qiang', 100)
xiao_huang = PhysicsStudent('Xiao Huang', 70)

xiao_qiang.print_score()
xiao_huang.print_score()
xiao_ming.learn()
xiao_qiang.learn()
xiao_huang.learn()
xiao_huang.play('Warcraft III')
```

类之间可以继承。

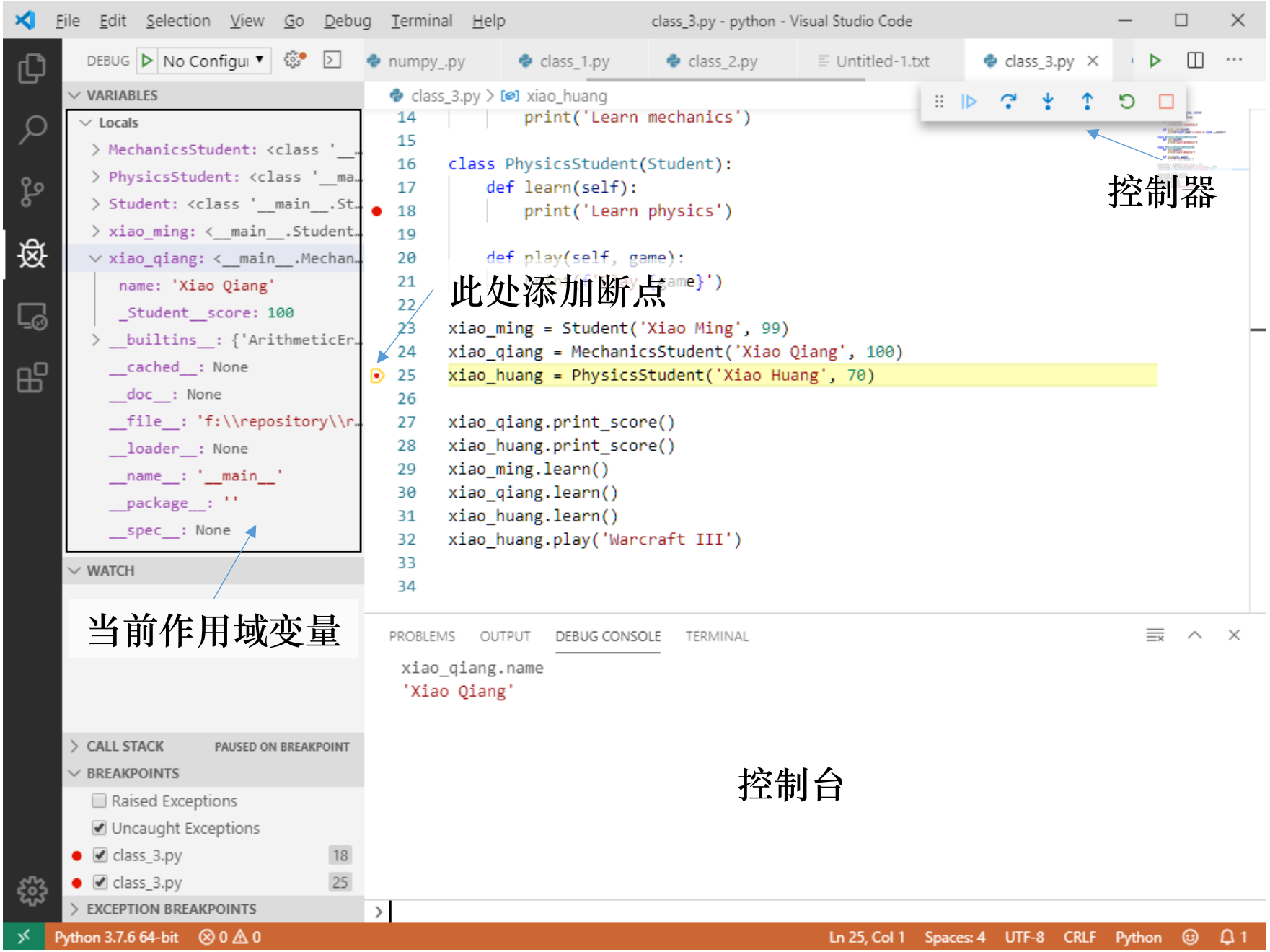
子类无需声明即可继承父类所有成员和方法。

为子类添加新成员时，如果与父类重名，可自动覆盖。

输出

```
Xiao Qiang's score is 100
Xiao Huang's score is 70
Learn something
Learn mechanics
Learn physics
Play Warcraft III
```

使用VS Code调试


File Edit Selection View Go Debug Terminal Help
class_3.py - python - Visual Studio Code
DEBUG No Configu
numpy_.py
class_1.py
class_2.py
Untitled-1.txt
class_3.py
class_3.py > xiao_huang
VARIABLES
Locals
MechanicsStudent: <class '__
PhysicsStudent: <class '__ma
Student: <class '__main__.St
xiao_ming: <__main__.Student
xiao_qiang: <__main__.Mechan
name: 'Xiao Qiang'
_Student__score: 100
__builtins__: {'ArithmeticEr
__cached__: None
__doc__: None
__file__: 'f:\\repository\\r
__loader__: None
__name__: '__main__'
__package__: ''
__spec__: None
WATCH
CALL STACK PAUSED ON BREAKPOINT
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
class_3.py 18
class_3.py 25
EXCEPTION BREAKPOINTS
14 print('Learn mechanics')
15
16 class PhysicsStudent(Student):
17 def learn(self):
18 print('Learn physics')
19
20 def play(self, game):
21
22
23 xiao_ming = Student('Xiao Ming', 99)
24 xiao_qiang = MechanicsStudent('Xiao Qiang', 100)
25 xiao_huang = PhysicsStudent('Xiao Huang', 70)
26
27 xiao_qiang.print_score()
28 xiao_huang.print_score()
29 xiao_ming.learn()
30 xiao_qiang.learn()
31 xiao_huang.learn()
32 xiao_huang.play('Warcraft III')
33
34
控制器
此处添加断点
当前作用域变量
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
xiao_qiang.name
'Xiao Qiang'
控制台
Python 3.7.6 64-bit
Ln 25, Col 1 Spaces: 4 UTF-8 CRLF Python

统一机器人描述格式(URDF)

URDF是使用xml语言描述机器人结构的一种文件格式。

```xml
<?xml version="1.0"?>
<robot name="physics">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue">
        <color rgba="0 0 .8 1"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <cylinder length="0.6" radius="0.17"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="10"/>
      <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
  </link>

  <!-- <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0.0 0 .25"/>
  </joint> -->

</robot>
```

xml版本。不用动。

单行元素

<name param=xx … />

多行元素

<name>

……

</name>

多行注释

<!--…

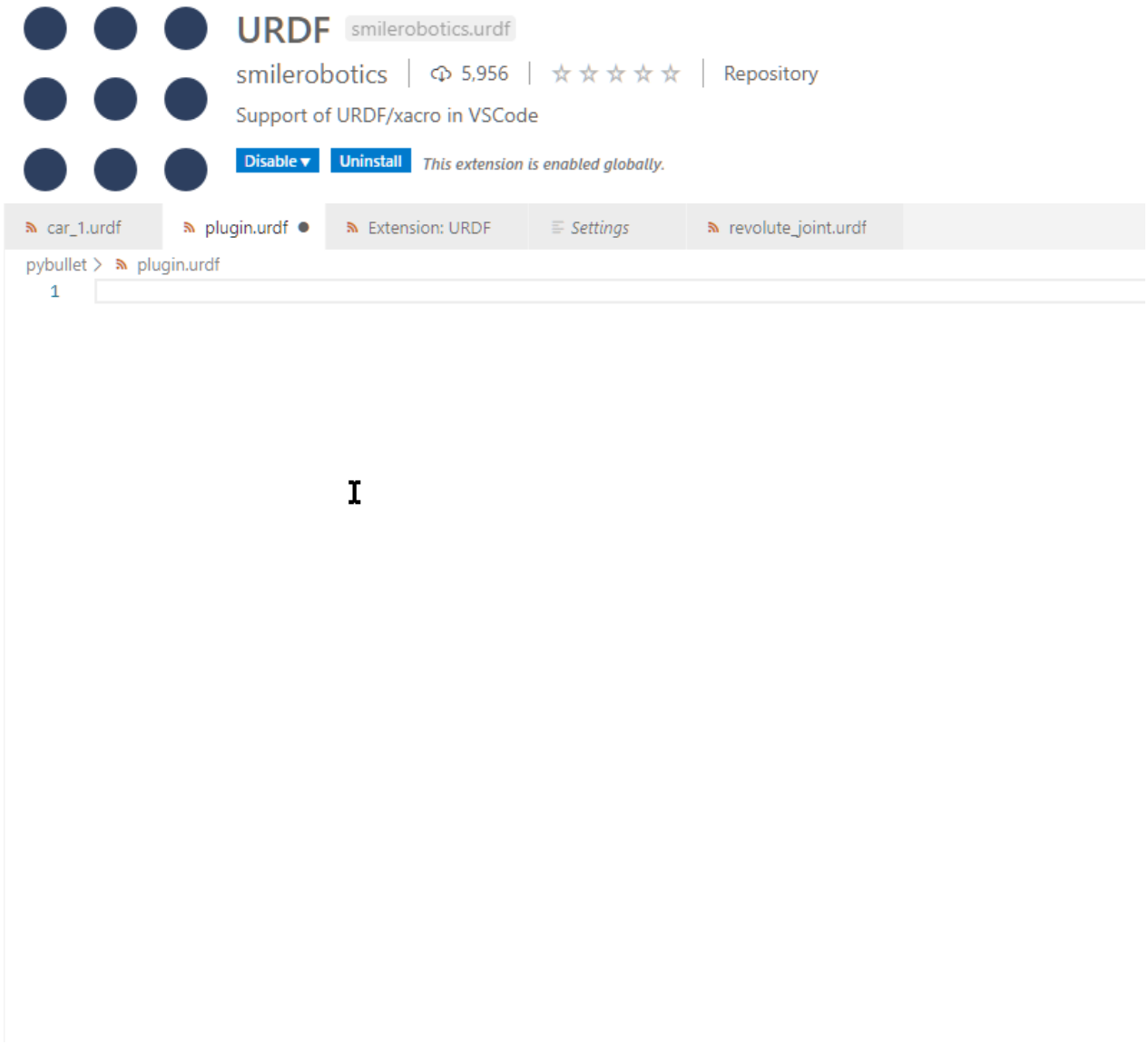
……

>

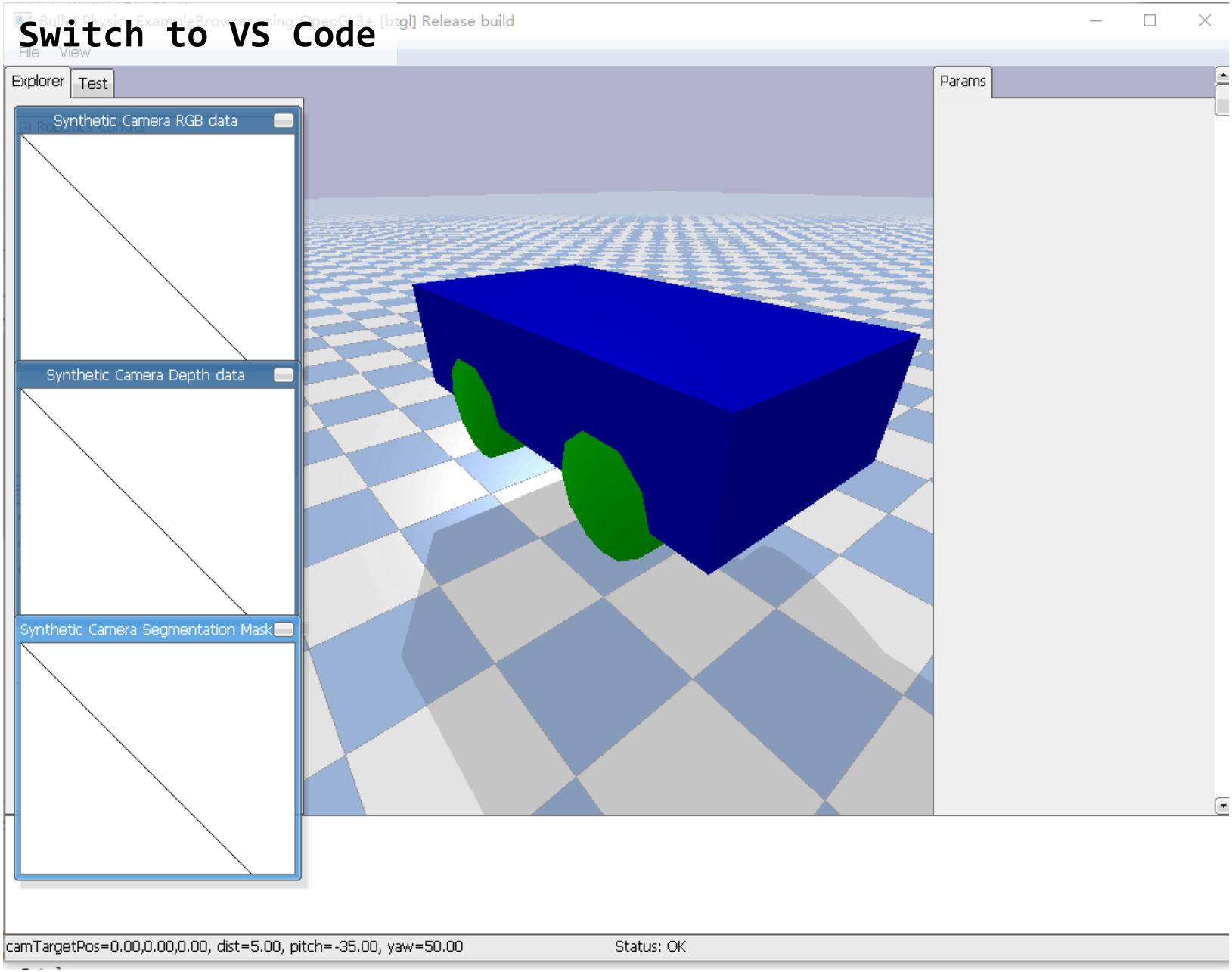
HINTS:使用VS Code中ctrl+/组合键快速注释与取消注释

使用VS Code的URDF插件

使用插件可以对文件内容进行自动补全



演示：构建机器人

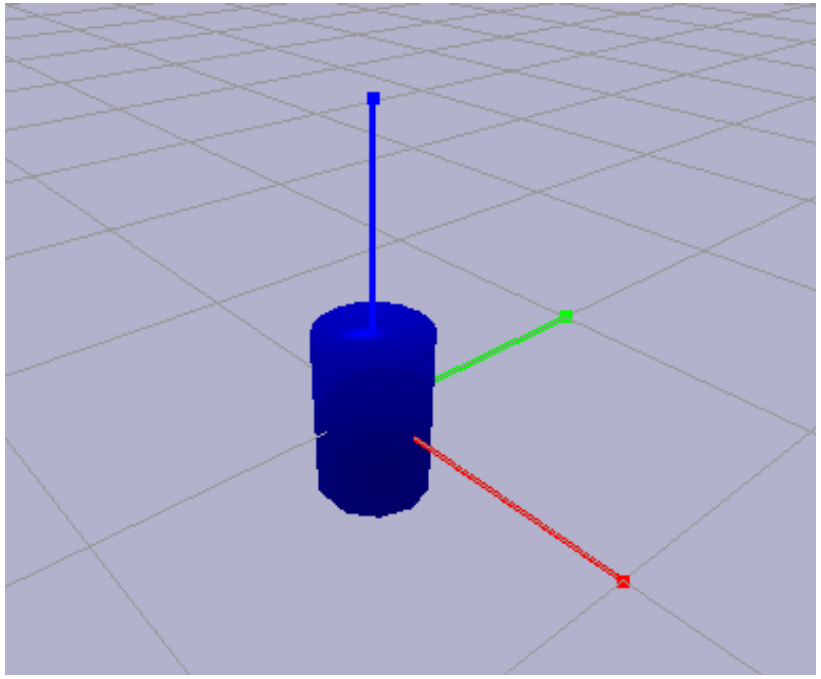

Switch to VS Code
[btgl] Release build
File View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
camTargetPos=0.00,0.00,0.00, dist=5.00, pitch=-35.00, yaw=50.00
Status: OK

构造可视的机器人

构建一个单刚体的机器人

```
1   <?xml version="1.0"?>
2   <robot name="physics">
3     <link name="base_link">
4       <visual>
5         <geometry>
6           <cylinder length="0.6" radius="0.2"/>
7         </geometry>
8         <material name="blue">
9           <color rgba="0 0 .8 1"/>
10        </material>
11      </visual>
12    </link>
13
14  </robot>
```

2	机器人及其名称
3-12	定义一个link，名叫base_link
4-11	定义所属link的visual属性
5-7	定义所属visual属性的geometry
8-10	定义所属visual属性的material



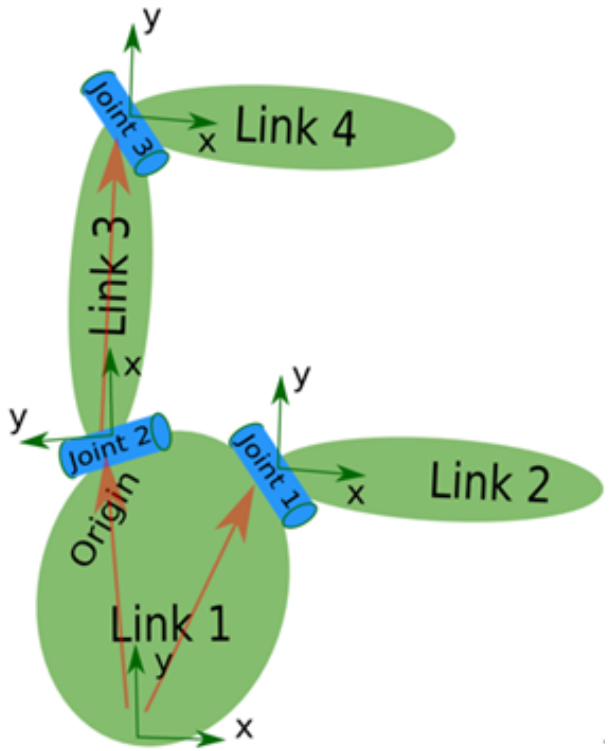
构建可视的机器人

构建多刚体机器人

```xml
<?xml version="1.0"?>
<robot name="physics">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0.22 0 .25"/>
  </joint>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 .2 .1"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
    </visual>
  </link>

</robot>
```

urdf文件所定义的机器人为树状多连杆机构，由link和joint构成。每一个link/joint内部均使用局部坐标系。

`<origin rpy="…" xyz="…"/>`为局部坐标系相对于上一级坐标系的相对位置

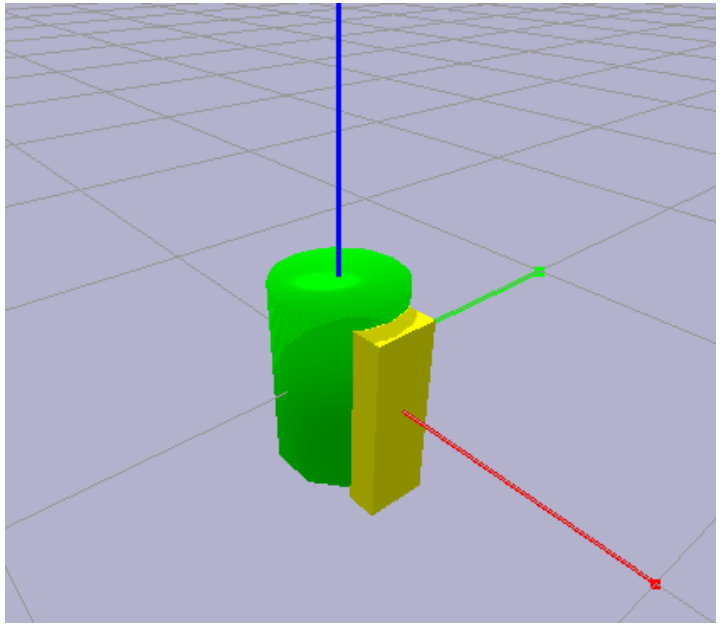


构建可视的机器人

构建多刚体机器人

```xml
<?xml version="1.0"?>
<robot name="physics">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
    <origin xyz="0.22 0 .25"/>
  </joint>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 .2 .1"/>
      </geometry>
      <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
    </visual>
  </link>

</robot>
```

urdf文件所定义的机器人为树状多连杆机构，由link和joint构成。



joint属性

continuous joint

```
<joint name="head_swivel" type="continuous">
    <parent link="base_link"/>
    <child link="head"/>
    <axis xyz="0 0 1"/>
    <origin xyz="0 0 0.3"/>
</joint>
```

revolute joint

```
<joint name="left_gripper_joint" type="revolute">
    <axis xyz="0 0 1"/>
    <limit effort="1000.0" lower="0.0" upper="0.548" velocity="0.5"/>
    <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
    <parent link="gripper_pole"/>
    <child link="left_gripper"/>
</joint>
```

Prismatic joint

```
<joint name="gripper_extension" type="prismatic">
    <parent link="base_link"/>
    <child link="gripper_pole"/>
    <limit effort="1000.0" lower="-0.38" upper="0" velocity="0.5"/>
    <origin rpy="0 0 0" xyz="0.19 0 0.2"/>
</joint>
```

link属性

link的属性包括inertial(质量与转动惯量),visual(可视化)和collision(碰撞).

```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
  </inertial>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
      <cylinder radius="0.5" length="2.0"/>
      <sphere radius="0.5/>
      <mesh filename="xxx.dae" scale="1.0 1.0 1.0"/>
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>
```

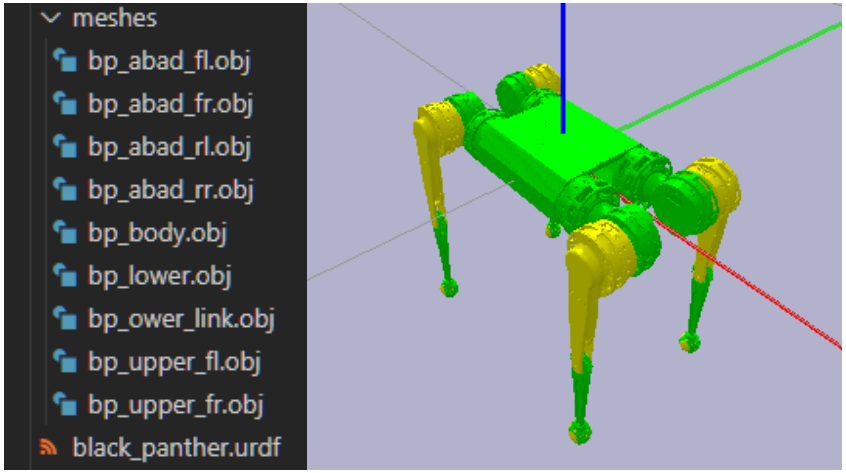
使用mesh文件

使用mesh文件可以定义更加复杂的形状，使模拟计算与真实机器人更加相符。

利用三维建模软件(SolidWorks，UG，Inventor等)绘制三维模型后，可以将零件导出为dae或stl格式，即可在URDF文件中以相对路径引用。

```xml
<?xml version="1.0" ?>
<robot name="blank_panther" xmlns:xacro="http://ros.org/wiki/xacro">

    <link name="body">
        <visual>
            <geometry>
                <mesh filename="meshes/bp_body.obj"/>
            </geometry>
            <origin rpy="0.0 0.0 0.0" xyz="0.0 0.0 0.0"/>
        </visual>
        <inertial>
            <mass value="2.955"/>
            <!-- <mass value="8.3"/> -->
            <origin xyz="0.0 0.0 -0.003"/>
            <inertia ixx="0.013355" ixy="0" ixz="0" iyy="0.040234" iyz="0" izz="0.048945"/>
        </inertial>
    </link>
```

注意斜杠方向！！！



bullet与pybullet

bullet是一个开源物理引擎。

用于计算运动和碰撞等动力学过程。

Github: https://github.com/bulletphysics/bullet3

Website: https://pybullet.org/wordpress/

Bullet底层通过C/C++实现，pybullet为bullet的官方python接口。

Pybullet QuickStart:

Official Google Doc, Local pdf Version

pybullet的安装与使用

使用pip安装pybullet

```
pip install --user pybullet
```

测试pybullet

```python
import pybullet as p
import time
import pybullet_data


physicsClient = p.connect(p.GUI)
p.setAdditionalSearchPath(pybullet_data.getDataPath())
p.setGravity(0,0,-10)

planeId = p.loadURDF("plane.urdf")

cubeStartPos = [0,0,1]
cubeStartOrientation = p.getQuaternionFromEuler([0,0,0])
boxId = p.loadURDF("r2d2.urdf",cubeStartPos, cubeStartOrientation)

for i in range (1000):
    p.stepSimulation()
    time.sleep(1./240.)

cubePos, cubeOrn = p.getBasePositionAndOrientation(boxId)
print(cubePos, cubeOrn)
p.disconnect()
```

小车控制代码解读

Switch to VS Code

old_driver.py 小车的控制

awesome_old_driver.py 镜头控制与调试功能

一般设置

```python
import pybullet as p

# 连接物理引擎
physicsClient = p.connect(p.GUI)
# p.setGravity(0, 0, 0)
p.setGravity(0, 0, -10)

# 载入机器人
robotID = p.loadURDF("r2d2.urdf", [0, 0, 0], [0, 0, 1, 0])

# 实时模拟
p.setRealTimeSimulation(0)
# p.setRealTimeSimulation(1)

# 时间步长
p.setTimeStep(1/240) # default

# 模拟一步
p.stepSimulation()
```

pyBullet仅仅是一个操作接口，并不能直接访问底层信息。

physicsClient并非一个Client对象，仅仅是一个整数。robotID同理。

Pybullet中，所有机器人、连杆、关节等均通过整数表示。

Transform（坐标变换）

Pybullet内置了欧拉角、四元数以及变换矩阵之间的转换函数

```
import pybullet as p

row, pitch, yaw = 1.57, 0, 0
quat = p.getQuaternionFromEuler([row, pitch, yaw])

x, y, z, w = 1, 0, 0, 0
rpy = p.getEulerFromQuaternion([x, y, z, w])

mat = p.getMatrixFromQuaternion([x, y, z, w])
```

机器人控制-joints

```python
import pybullet as p
import pybullet_data

physicsClient = p.connect(p.GUI)
p.setGravity(0, 0, 0)
p.setAdditionalSearchPath(pybullet_data.getDataPath())

robotID = p.loadURDF("r2d2.urdf", [0, 0, 0], [0, 0, 1, 0])

# joints
numJoint = p.getNumJoints(robotID)
jointStateList = []
jointInfoList = []
mode = p.POSITION_CONTROL
maxForce = 500
for jointID in range(p.getNumJoints(robotID)):
    p.enableJointForceTorqueSensor(robotID, jointID, 1)
    jointInfoList.append(p.getJointInfo(robotID, jointID))
    jointStateList.append(p.getJointState(robotID, jointID))
    p.setJointMotorControl2(robotID, jointID, controlMode=mode, force=maxForce)

jointStateList_ = p.getJointStates(robotID, list(range(numJoint)))
```

获取指定机器人的joint数量

遍历所有joint

打开joint的力传感器

获取joint信息

获取joint状态

设置joint控制模式

获取joints状态

JointInfo

以list的形式保存joint的所有静态信息。

jointIndex	int	the same joint index as the input parameter
jointName	string	the name of the joint, as specified in the URDF (or SDF etc) file
jointType	int	type of the joint, this also implies the number of position and velocity variables. JOINT_REVOLUTE, JOINT_PRISMATIC, JOINT_SPHERICAL, JOINT_PLANAR, JOINT_FIXED. See the section on Base, Joint and Links for more details.
qIndex	int	the first position index in the positional state variables for this body
uIndex	int	the first velocity index in the velocity state variables for this body
flags	int	reserved
jointDamping	float	the joint damping value, as specified in the URDF file
jointFriction	float	the joint friction value, as specified in the URDF file
jointLowerLimit	float	Positional lower limit for slider and revolute (hinge) joints.
jointUpperLimit	float	Positional upper limit for slider and revolute joints. Values ignored in case upper limit <lower limit.
jointMaxForce	float	Maximum force specified in URDF (possibly other file formats) Note that this value is not automatically used. You can use maxForce in 'setJointMotorControl2'.
jointMaxVelocity	float	Maximum velocity specified in URDF. Note that the maximum velocity is not used in actual motor control commands at the moment.
linkName	string	the name of the link, as specified in the URDF (or SDF etc.) file
jointAxis	vec3	joint axis in local frame (ignored for JOINT_FIXED)
parentFramePos	vec3	joint position in parent frame
parentFrameOrn	vec4	joint orientation in parent frame (quaternion x,y,z,w)
parentIndex	int	parent link index, -1 for base

JointState

以list的形式保存joint的所有动态信息。

jointPosition	float	The position value of this joint.
jointVelocity	float	The velocity value of this joint.
jointReactionForces	list of 6 floats	These are the joint reaction forces, if a torque sensor is enabled for this joint it is [Fx, Fy, Fz, Mx, My, Mz]. Without torque sensor, it is [0,0,0,0,0,0].
appliedJointMotorTorque	float	This is the motor torque applied during the last stepSimulation.

Motor Control

```
p.setJointMotorControl2(robotID,
                        jointID,
                        controlMode,
                        targetPosition,
                        targetVelocity,
                        force,
                        positionGain,
                        velocityGain,
                        maxVelocity,
                        physicsClientId)
```

	POSITION_CONTROL	VELOCITY_CONTROL	TORQUE_CONTROL
targetPosition	1	0	0
targetVelocity	1	1	0
force	max force	max force	external force
positionGain	kp	kp	
velocityGain	kd	kd	
maxVelocity	max velocity		

机器人控制-links

```python
import pybullet as p
import pybullet_data

physicsClient = p.connect(p.GUI)
p.setGravity(0, 0, 0)
p.setAdditionalSearchPath(pybullet_data.getDataPath())

robotID = p.loadURDF("r2d2.urdf", [0, 0, 0], [0, 0, 1, 0])

# joints
numJoint = p.getNumJoints(robotID)
linkStates = []
for linkID in range(numJoint):
    linkStates.append(p.getLinkState(robotID, linkID))

linkStates_ = p.getLinkStates(robotID, list(range(numJoint)))
baseVelocity = p.getBaseVelocity(robotID)
cubePos, cubeOrn = p.getBasePositionAndOrientation(robotID)

forceVec = [0, 0, 1]
forcePos = [0, 0, 0]
flag = p.LINK_FRAME
linkID = 1
p.applyExternalForce(robotID, linkID, forceVec, forcePos, flag)
```

遍历所有link

获取link状态

获取link状态

获取base速度

获取base位置与姿态

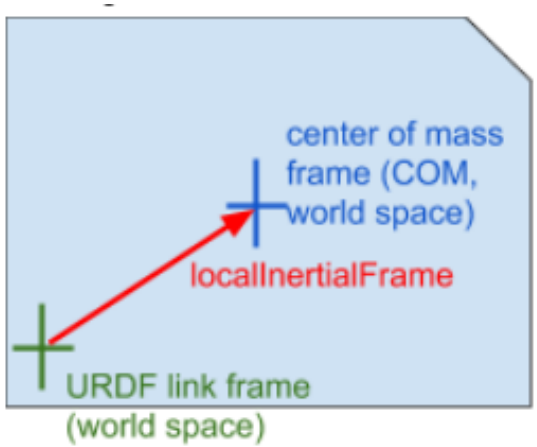
施加外力(仅适用于非实时)

base以外的link数量等于joint数量

LinkState

以list的形式保存link的所有动态信息。

linkWorldPosition	vec3, list of 3 floats	Cartesian position of center of mass
linkWorldOrientation	vec4, list of 4 floats	Cartesian orientation of center of mass, in quaternion [x,y,z,w]
localInertialFramePosition	vec3, list of 3 floats	local position offset of inertial frame (center of mass) expressed in the URDF link frame
localInertialFrameOrientation	vec4, list of 4 floats	local orientation (quaternion [x,y,z,w]) offset of the inertial frame expressed in URDF link frame.
worldLinkFramePosition	vec3, list of 3 floats	world position of the URDF link frame
worldLinkFrameOrientation	vec4, list of 4 floats	world orientation of the URDF link frame
worldLinkLinearVelocity	vec3, list of 3 floats	Cartesian world velocity. Only returned if computeLinkVelocity non-zero.
worldLinkAngularVelocity	vec3, list of 3 floats	Cartesian world velocity. Only returned if computeLinkVelocity non-zero.



镜头控制与调试功能

```python
while t < 1200:
    p.stepSimulation()
    time.sleep(1/240)
    wheelState0 = p.getJointState(carId, 0)
    wheelState1 = p.getJointState(carId, 1)
    desiredWheelState0 = wheelState0[0] - 0.3
    desiredWheelState1 = wheelState1[0] - 0.3
    p.setJointMotorControl2(carId, jointIndex=0, controlMode=p.POSITION_CONTROL, targetPosition=desiredWheelState0)
    p.setJointMotorControl2(carId, jointIndex=1, controlMode=p.POSITION_CONTROL, targetPosition=desiredWheelState1)

    baseOrnLineId = p.addUserDebugLine([0, 0, 0], [3, 0, 0], lineColorRGB=[1, 0, 0], parentObjectUniqueId=carId)
    WheelLineId = p.addUserDebugLine([0, 0, 1], [3, 0, 2], lineColorRGB=[0, 0, 0], parentObjectUniqueId=carId, parentLinkIndex=0)

    basePos, baseOrn = p.getBasePositionAndOrientation(carId)
    rpy = p.getEulerFromQuaternion(baseOrn)
    yaw += 0.3
    cameraYaw = rpy[2] + yaw
    cameraPitch = rpy[1] - 45
    p.resetDebugVisualizerCamera(cameraDistance=10.0, cameraYaw=cameraYaw, cameraPitch=cameraPitch, cameraTargetPosition=basePos)

    t += 1
```

addUserDebugLine 显示参考线

resetDebugVisualizerCamera 设置相机位置

推荐学习资料

Python基础与知识速查

廖雪峰 https://www.liaoxuefeng.com/wiki/1016959663602400

Python官方文档 https://docs.python.org/3/

Numpy官方中文文档 https://www.numpy.org.cn/

URDF相关资料

ROS官方文档 http://wiki.ros.org/urdf/

官方教程 http://wiki.ros.org/urdf/Tutorials

Pybullet

官方网站 https://pybullet.org/wordpress/

官方教程 Official Google Doc，Local pdf Version

官方实例 github

本讲义有关代码 https://gitee.com/ycshao/robotics_tutorial


Your resume is impressive, but how do you troubleshoot problems?
1
Search engine.
2
3
Welcome aboard!
4
Any IT Job
Cyanide and Happiness © Explosm.net