



智能系统安全实践：环境配置

复旦大学系统软件与安全实验室

[\(secsys.fudan.edu.cn\)](http://secsys.fudan.edu.cn)



Outline

- Python入门
 - 在命令行中运行Python
 - 在Notebook中运行Python
- 机器学习常用Python库
 - NumPy的使用
- 实验环境配置

复旦大学系统软件与安全实验室

Python入门



- Python是一种脚本式语言
 - 跟C/C++/JAVA等语言，不需要经历编译->运行的步骤
 - 每个语句都可以实时运行
 - 变量不用先声明类型再赋值
- 熟悉Python：
 - 条件判断 / 循环
 - 函数 / 类
 - 内置数据结构（字典、列表等）

Python入门：条件判断

1. not 逻辑非运算，形如not exp。如果exp取值为真，则表达式取值为假，否则表达式取值为真
2. and，逻辑与运算，形如exp1 and exp2。如果exp1与exp2取值均为真，则表达式取值为真，否则表达式取值为假
3. or，逻辑或运算，形如exp1 or exp2。如果exp1与exp2取值均为假，则表达式取值为假，否则表达式取值为真
4. if - elif - else，条件语句，按顺序检查if、elif后接表达式的取值，若某一表达式取值为真则执行相应语句块并跳过后续所有判断，若全部表达式取值为假则执行else后续语句块，若无else则结束
5. 如右图所示，python用缩进代替c中的{}实现标识子语句块的功能

```
[ ]: if not x > 0:
    print("Case 1")
    elif x > 0 and x < 5:
    print("Case 2")
    elif x < 10 or x > 20:
    print("Case 3")
    else:
    if y < 0:
    print("Case 4")
    else:
    print("Case 5")
```

上述代码将二维空间划分成了5个子区域，试着分析每个子区域对应的空间范围

Python入门：循环

1. for循环

1. 一般会使用`range()`函数作为整数的枚举，而`range()`有3种

常见形式：

1. `range(end)`：从0开始枚举
2. `range(start, end)`
3. `range(start, end, step)`，默认`step=1`
4. 注意：`range()`枚举的取值中不包括`end`

2. `while`循环，形如`while exp`。若`exp`为真，则执行后续语句块并重新检查`exp`取值，否则结束循环。

```
[3]: count = 0
      for i in range(10):
          count += i
      print(count)
```

45

```
[4]: count = 0
      i = 0
      while i < 10:
          count += i
          i += 1
      print(count)
```

45

Python入门：函数

函数定义——`def funcName (var1, var2, ...)`

由若干语句组成的语句块，与Notebook中的一个代码模块类似

```
[3]: def addDef(a, b):  
      return a + b
```

```
[4]: a = 2.52  
      b0 = 3.52  
      b1 = 3  
      c0 = addDef(a, b0)  
      c1 = addDef(a, b1)  
      print(c0)  
      print(c1)
```

```
6.04  
5.52
```

```
[16]: def chooseChar(s, k):  
       return s[k]
```

```
[20]: s = "abcde"  
      print(chooseChar(s, 2))  
      print(chooseChar(s, 4))  
      a = [1, 2, 3, 4, 5]  
      print(chooseChar(a, 3))
```

```
c  
e  
4
```

通过函数调用消除功能相同的冗余代码，提升编码效率

函数不需要声明输入输出的数据类型

Python入门：类

类定义——`class className(superClass)`

由若干属性(数据变量)和若干方法(函数)集成的一个实例模板

1. `__init__()`，类默认的初始化方法，每次调用类生成一个实例时就会自动执行的函数
2. `self`，表示类本身的变量，在类内部会用`self`来区分类自带的变量和方法 and 外部定义或临时定义的变量和方法

```
[10]: class addClass:
      def __init__(self):
          self.a = 0
          self.b = 0

      def init(self, a, b):
          self.a = a
          self.b = b

      def calc(self):
          return self.a + self.b
```

```
[11]: C = addClass()
      a = 2.52
      b = 3.52
      print(C.calc())
      C.init(a, b)
      print(C.calc())
      0
      6.04
```

使用模板生成实例C，C就具有了模板中所定义的属性`a`, `b`，以及模板中所定义的方法`__init__`, `init`, `calc`

试着分析在执行完第二个代码模块后，类自带的属性变量有哪些，取值发生了什么变化

Python入门: List列表

1. 右图中, a, b, c代表2种初始化一维列表的方法, d, e代表2种初始化二维列表的方法
 - e的初始化其实存在一些问题, 尝试修改e中元素看看会发生什么
2. a. `append(var)`, 在列表a尾部添加一个新元素var
3. 通过`a[i]`访问第i个元素

```
[1]: a = [0, 0, 0]
      b = [0] * 3
      c = [0 for i in range(3)]
      d = [[0] * 3] * 3
      e = [c for i in range(3)]
      print('a', a)
      print('b', b)
      print('c', c)
      print('d', d)
      print('e', e)
      a.append(1)
      print('a', a)
      a.pop(3)
      print('a', a)
```

```
a [0, 0, 0]
b [0, 0, 0]
c [0, 0, 0]
d [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
e [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
a [0, 0, 0, 1]
a [0, 0, 0]
```


机器学习常用Python库：NumPy

Numpy支持大规模的向量、矩阵、高维数组的各种基础运算和函数处理，一些常见的机器学习数据为方便处理会使用Numpy进行读写

```
[3]: a = [[1, 2, 3], [1, 2, 3]]
      b = [[3, 2, 1], [3, 2, 1]]
      c = [[0, 0, 0], [0, 0, 0]]

      for i in range(2):
          for j in range(3):
              c[i][j] = a[i][j] + b[i][j]
      print(c)

[[4, 4, 4], [4, 4, 4]]
```

```
[4]: import numpy as np

      a = np.array([[1, 2, 3], [1, 2, 3]])
      b = np.array([[3, 2, 1], [3, 2, 1]])

      c = a + b
      print(c)

[[4 4 4]
 [4 4 4]]
```

```
[6]: a = [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
      b = [[3, 2, 1], [3, 2, 1], [3, 2, 1]]
      c = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

      for i in range(3):
          for j in range(3):
              for k in range(3):
                  c[i][j] += a[i][k] * b[k][j]
      print(c)

[[18, 12, 6], [18, 12, 6], [18, 12, 6]]
```

```
[8]: import numpy as np

      a = np.array([[1, 2, 3], [1, 2, 3], [1, 2, 3]])
      b = np.array([[3, 2, 1], [3, 2, 1], [3, 2, 1]])

      c = np.matmul(a, b)
      print(c)

[[18 12 6]
 [18 12 6]
 [18 12 6]]
```

如果使用+计算列表数据，能正常运行吗？得到的结果会一样吗？如果使用*计算Numpy数据，能正常运行吗？得到的结果会一样吗？

更多使用方法可以参考文档<https://www.numpy.org.cn/reference/>中的介绍

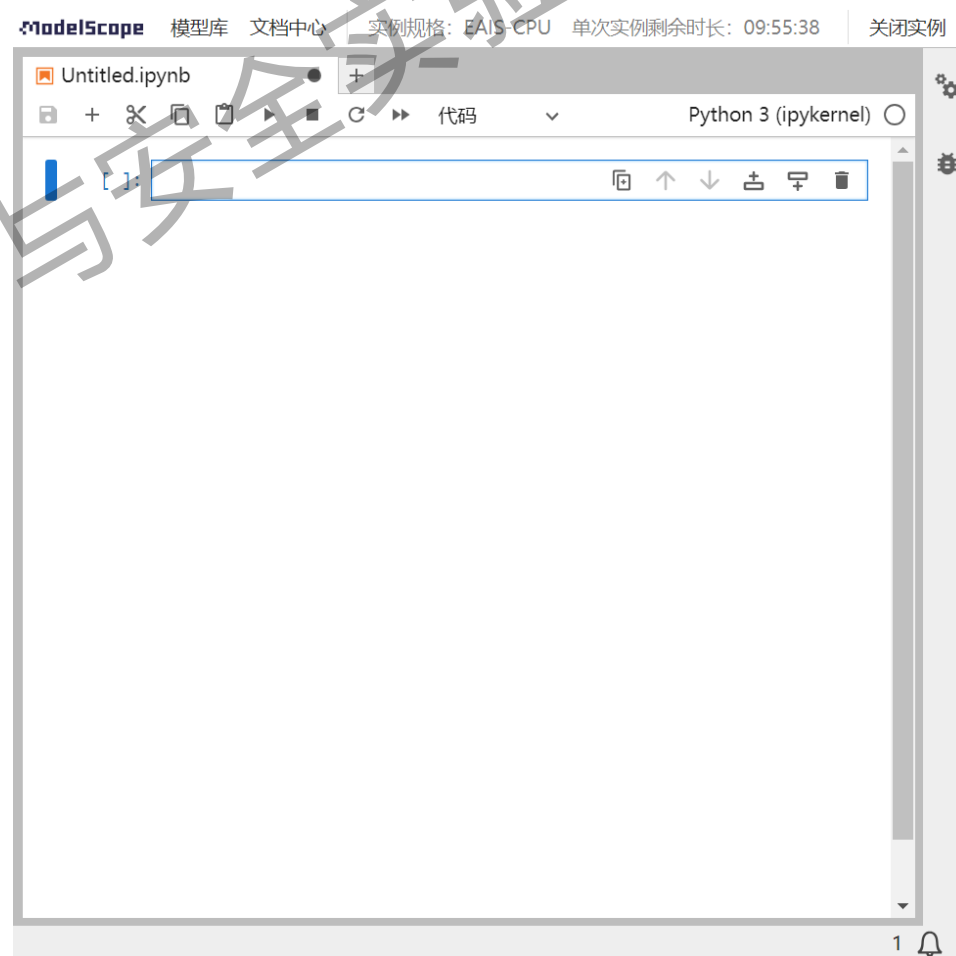
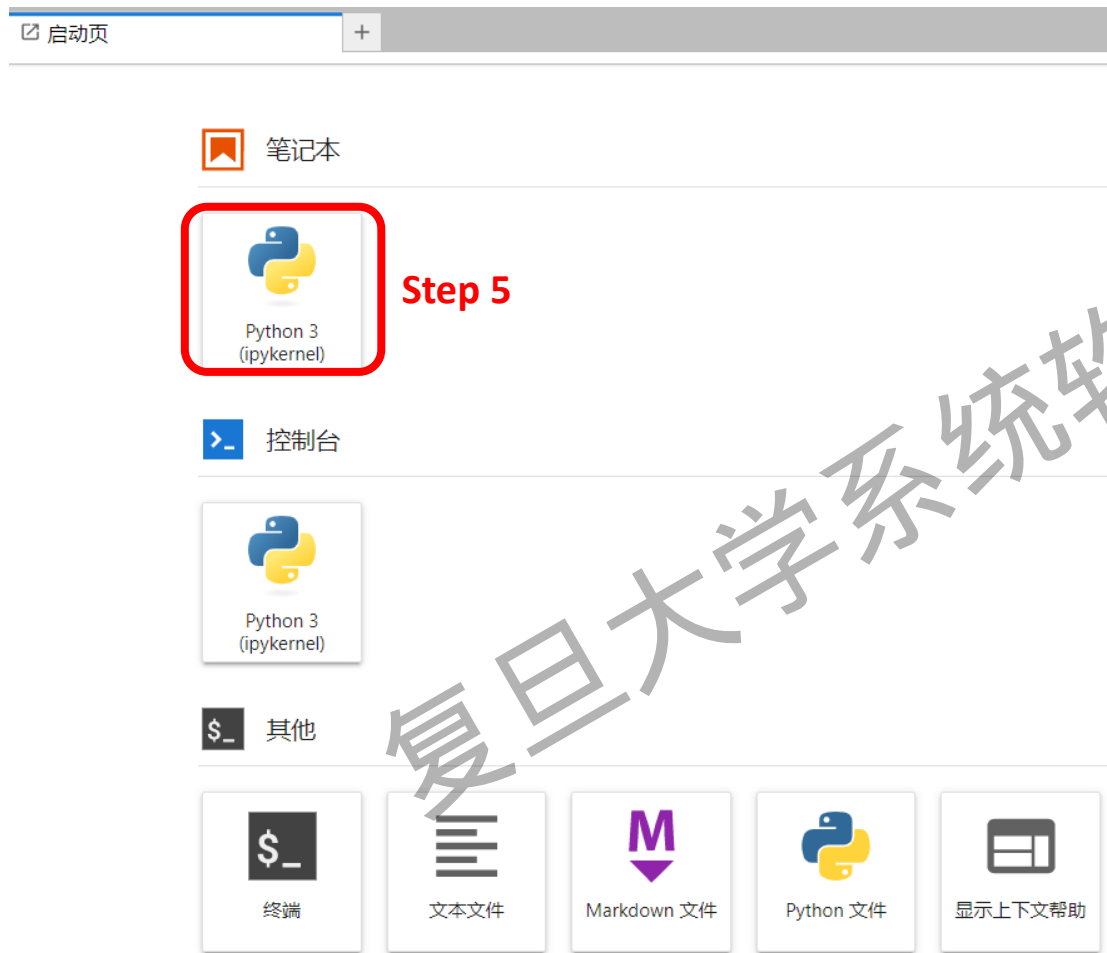
环境配置：使用服务器

1. 进入www.modelscope.cn/my/mynotebook，注册账号并登录
2. 关联阿里云账号，获取免费计算资源
如无，则再注册阿里云账号
3. 选择阿里云弹性加速计算EAS，并启动
注意服务器是临时资源，不保证会保存文件



环境配置：使用服务器

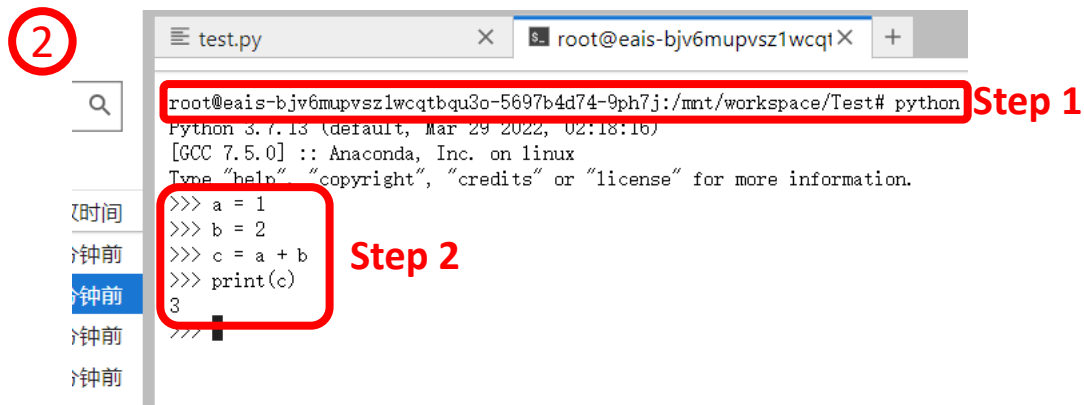
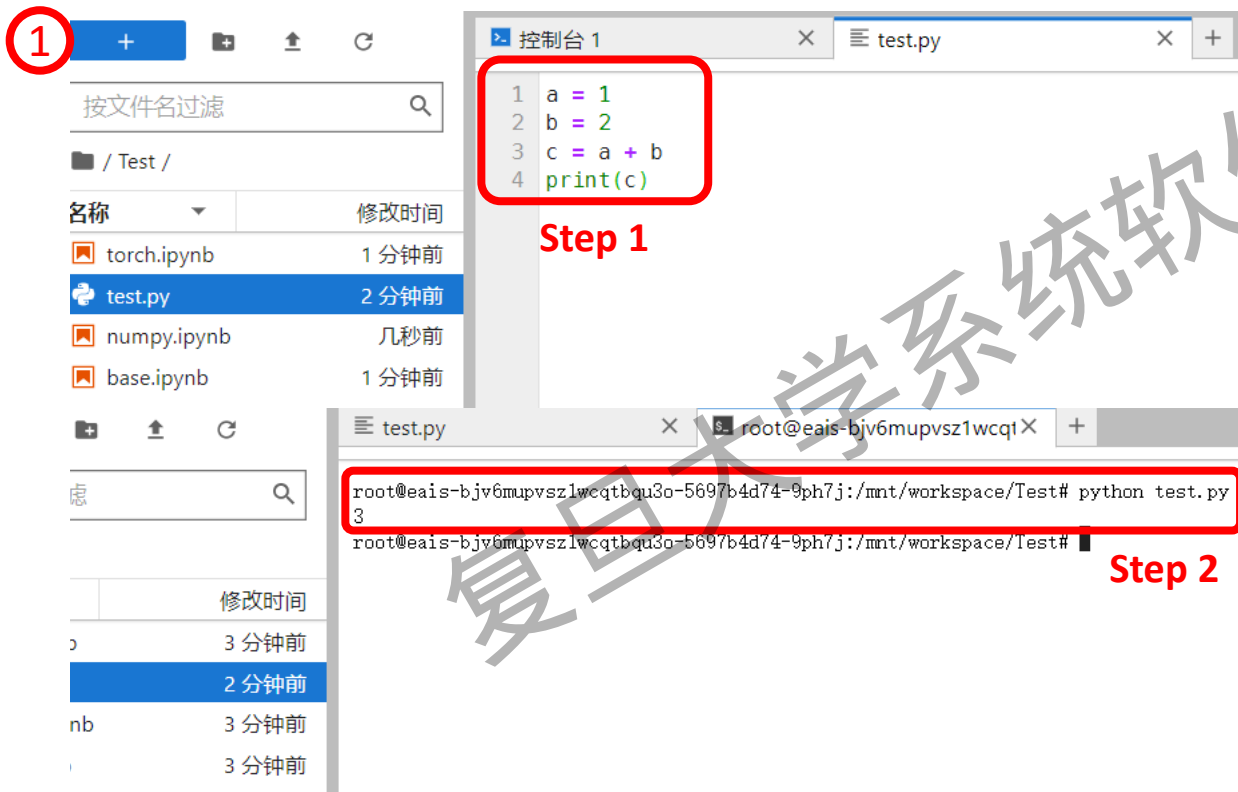
4. 等待模型加载完成点击“查看Notebook”进入开发界面，即可打开代码编辑界面



Python入门：命令行使用

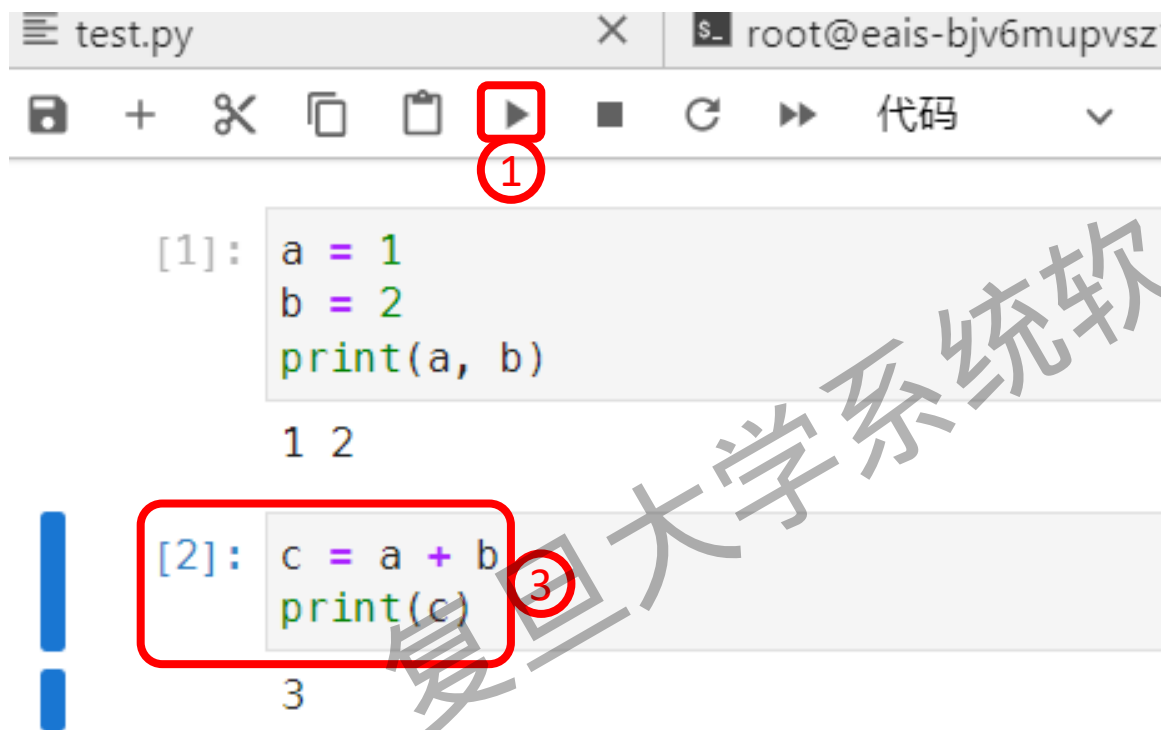
以最简单的a+b问题为例

1. 新建.py文件编写完整代码，然后在命令行中通过python命令加载运行
2. 直接在命令行中运行python命令，逐指令编写代码执行



Python入门：Notebook使用

Notebook的使用方式相当于命令行的2种使用方式的折衷版：每次可以只编写一个模块的代码，同时各个模块之间的执行顺序可以人工设定



```
[1]: a = 1
     b = 2
     print(a, b)
1 2

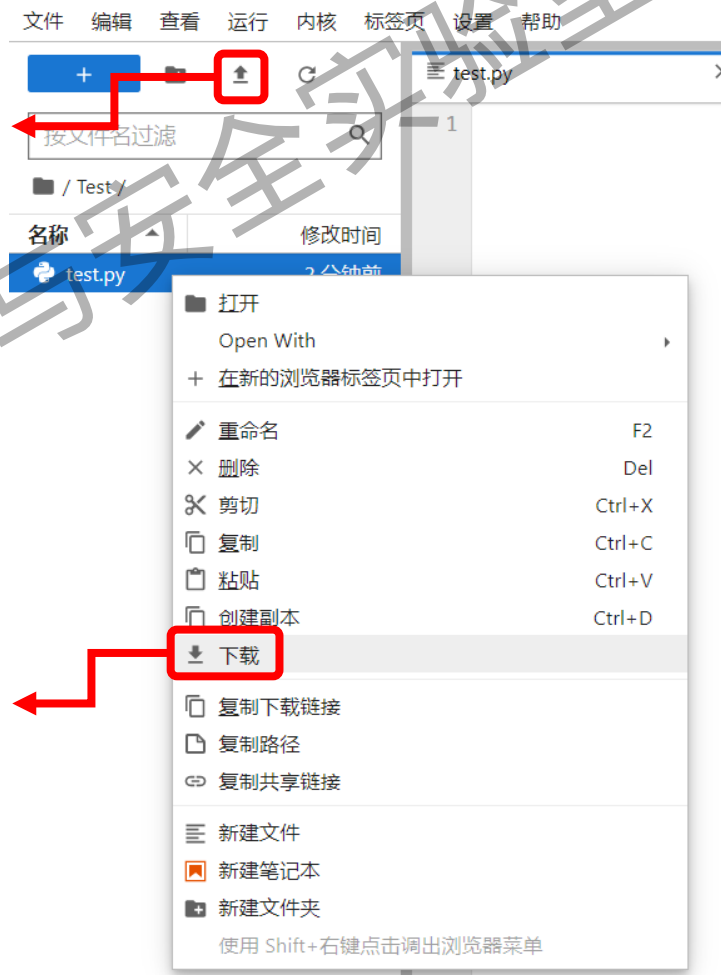
[2]: c = a + b
     print(c)
3
```

1. 选定模块，按下Ctrl+Enter或图示的按键即可运行对应模块
2. 新模块可以使用已执行模块的计算结果（模块左侧的数字代表已执行模块的执行顺序）

环境配置：使用服务器

- 文件的上传与下载

点击按键，从本地机器选择上传的文件



右键目标文件，选择“下载”将文件下载到本地机器

任务1：Python入门 - 组合数计算

- 尝试用Python实现组合数计算：

- 输入 n, m ，输出 $C_n^m = \frac{n!}{m!(n-m)!}$

- 实现方式：

- 在给定的Jupyter Notebook中填充合适的实现

```
[ ]: n = 5 # input  
     m = 3 # input
```

```
[ ]: ans = 1  
     # TODO: 直接计算组合数C(n,m)  
     print(ans)
```

```
[ ]: c = [[0 for j in range(n + 1)] for i in range(n + 1)]  
     for i in range(n + 1):  
         # TODO: 迭代推导组合数C(n,m)  
     print(c[n][m])
```

任务2: NumPy入门 - 矩阵乘法

- 尝试使用NumPy计算矩阵乘法:

- 初始化两个矩阵 a, b
- 用Python的for循环计算矩阵乘法结果
- 将 a, b 转换成numpy.array表示并计算两者矩阵乘的结果
- 将计算结果转换成Python的list表示并输出

- 实现方式:

- Python文件运行或者Jupyter都可以



Q&A

复旦大学系统软件与安全实验室

[\(secsys.fudan.edu.cn\)](http://secsys.fudan.edu.cn)

