

python语言入门与实践

列表及其操作

主讲人：吴陈炜 博士

复旦大学智能机器人研究院 工程博士

北京大学信息科学与技术学院 理学硕士

浙江大学信息与电子工程学系 工学学士

网易（杭州）网络有限公司 项目经理

美国项目管理协会 PMP（项目管理专业人士）

之江实验室 科研主管

2020 年 5 月 14 日

今日知识要点

- 列表基础知识
- 字符串与列表的关系
- for循环遍历
- if条件语句

python的四大容器v.s.三大控制

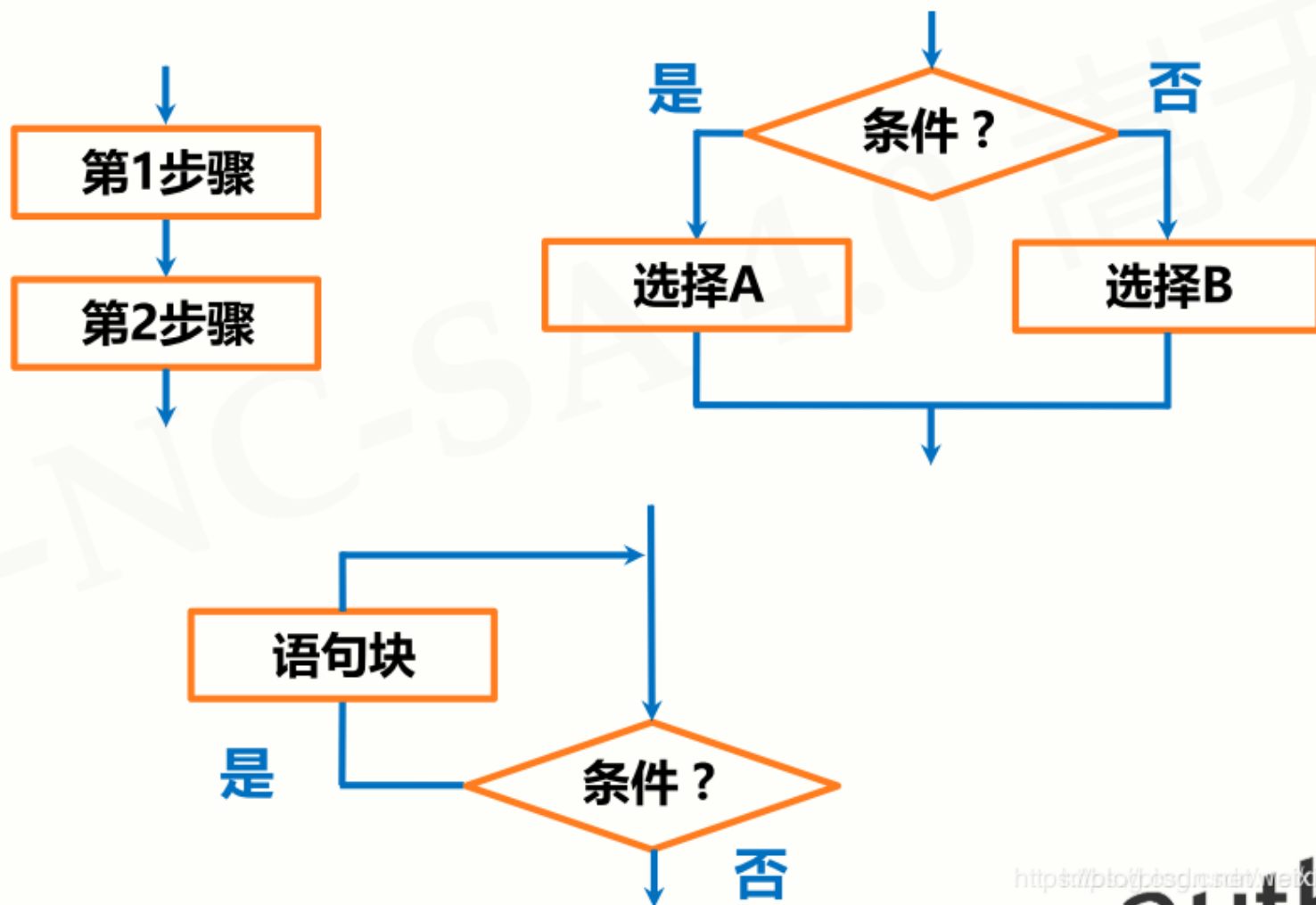


“程序的控制结构”

- 顺序结构

- 分支结构

- 循环结构



列表的基础知识

列表结构

- 利用**中括号**表示列表
- 列表内的元素用**逗号**隔开
- 注意是**英文输入法**下的逗号

```
student1 = ['lilei', 18, 'class01', 201901]  
student2 = ['hanmeimei', 19, 'class02', 201902]
```

列表的**可变性**：可以修改列表里的内容。

```
>>> list('Hello')  
['H', 'e', 'l', 'l', 'o']
```

获取列表中的某个元素

编程语言中通常第一个位置的编号是0

```
grade = [98, 99, 95, 80]
```

↑ ↑ ↑ ↑
编号: 0 1 2 3

- 中括号内数字指定元素位置

```
print(grade[0])  
print(grade[0] + grade[3])
```

```
98  
178
```

获取列表中连续的几个元素

中括号内用起始位置:结束位置描述

注意：不包括结束位置那个元素

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
print(numbers[2:6])
```

从几号开始

到几号之前结束

```
[2, 3, 4, 5]
```


列表的分片赋值

```
name = list('Python')  
print(name)  
  
name[2:] = list('abc')  
print(name)
```

```
['P', 'y', 't', 'h', 'o', 'n']  
['P', 'y', 'a', 'b', 'c']
```

```
numbers = [1, 5]  
numbers[1:1] = [2, 3, 4]  
print(numbers)
```

```
numbers[1:4] = []  
print(numbers)
```

获取列表长度

用len(列表)来获取

例：


```
student_list = ['李雷', '韩梅梅', '马冬梅']  
print(len(student_list))
```

修改列表中的元素

找到需要修改的元素编号

列表名[编号]=新值

```
name = ['lilei', 'hanmeimei']  
print(name)
```



```
name[0] = 'madongmei'  
print(name)
```

```
['lilei', 'hanmeimei']  
['madongmei', 'hanmeimei']
```

向列表添加元素

在列表变量后加

`.append(要添加的元素)`

这是列表型变量通用的方法

```
inventory = ['钥匙', '毒药']  
print(inventory)  
inventory.append('解药')  
print(inventory)
```

```
['钥匙', '毒药']  
['钥匙', '毒药', '解药']
```

删除列表元素

用`del`+列表元素来删除

例：

```
student_list = ['李雷', '韩梅梅', '马冬梅']  
del student_list[0]  
print(student_list)
```

```
['韩梅梅', '马冬梅']
```

两个列表相加

直接用加号

```
numbers1 = [0, 1, 2, 3, 4]  
numbers2 = [5, 6, 7, 8, 9]  
print(numbers1+numbers2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

判断某个元素是否存在于列表中

利用in来判断

if 要判断的元素 in 列表:

```
inventory = ['钥匙', '毒药', '解药']  
if '解药' in inventory:  
    print('Yes')  
else:  
    print('No')
```

Yes

获取列表中某个元素的重复次数

用列表.count(元素)来获取

例：

```
numbers1 = [0, 1, 1, 2, 3, 4, 1]  
print(numbers1.count(1))
```


获取列表中某个元素第一次出现的位置

用列表.index(元素)来获取

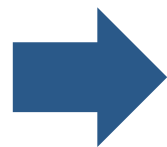
例：

```
numbers1 = [0, 1, 1, 2, 3, 4, 1]  
print(numbers1.index(1))
```

排序

- 使用.sort()方法对列表元素进行永久排序

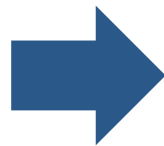
```
test_list = [3,2,6,5,4,1]
test_list.sort(reverse = True)
print(test_list)
```



```
[6, 5, 4, 3, 2, 1]
[Finished in 0.2s]
```

- 使用sorted()函数对列表对象进行临时排序

```
test_list = [3,2,6,5,4,1]
print(sorted(test_list,reverse=True))
print(test_list)
```



```
[6, 5, 4, 3, 2, 1]
[3, 2, 6, 5, 4, 1]
[Finished in 0.2s]
```

Test

- 编写程序，完成以下要求：
- 将一串字符串'132569874'转换成列表并将其输出；
- 对其中偶数下标的元素进行降序排列，奇数下标的元素不变。

Answer

```
30 string = '132569874'
31 str_list = list(string)
32 list_tmp = str_list[::-2]
33 list_tmp.sort(reverse=True)
34 str_list[::-2] = list_tmp
35 print(str_list)
```

```
['8', '3', '6', '5', '4', '9', '2', '7', '1']
[Finished in 0.1s]
```

元组

列表和元组的对比

```
a = [1, 'hanmeimei', 18]
b = (2, 'lilei', 19)
```

列表 (list)

元组 (tuple)

列表和元组的区别：

- 列表中，元素用方括号[]包裹；在元组中，元素用圆括号()包裹。
- 列表中的元素可以被修改、添加、删除，即列表是可变的数据类型，元组是不可变的数据类型。

- 元组是不可变的

```
b = (2, 'lilei', 19)
b[0]=1
```

```
b[0]=1
TypeError: 'tuple' object does not support item assignment
```

- 对元组的取值与分片操作

```
b = (2, 'lilei', 19)
print(b)
print(b[2])
print(b[0:2])
```

```
(2, 'lilei', 19)
19
(2, 'lilei')
```

使用for循环操作列表

for循环

```
students_list = ["李雷", "韩梅梅", "马冬梅"]  
for student in students_list:  
    print(student)
```

```
李雷  
韩梅梅  
马冬梅
```

按顺序选出来students_list列表中的一个值，
赋予到student变量，然后执行循环内的语句

循环数字

```
for i in range(10):  
    print(i)
```

0
1
2
3
4
5
6
7
8
9

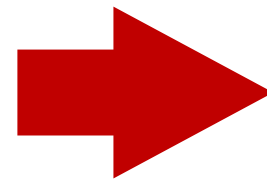
用range()函数可以得到一个整数序列对象

数值列表range()的使用

```
range(0, 10, 2)
```

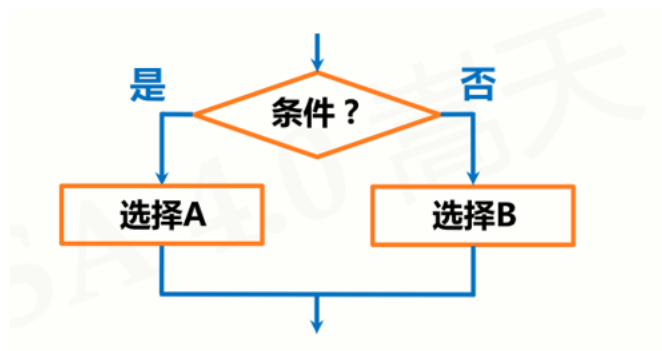
(起始, 结束, 间隔)

```
for i in range(0, 10, 2)  
    print(i)
```



0
2
4
6
8

if条件判断

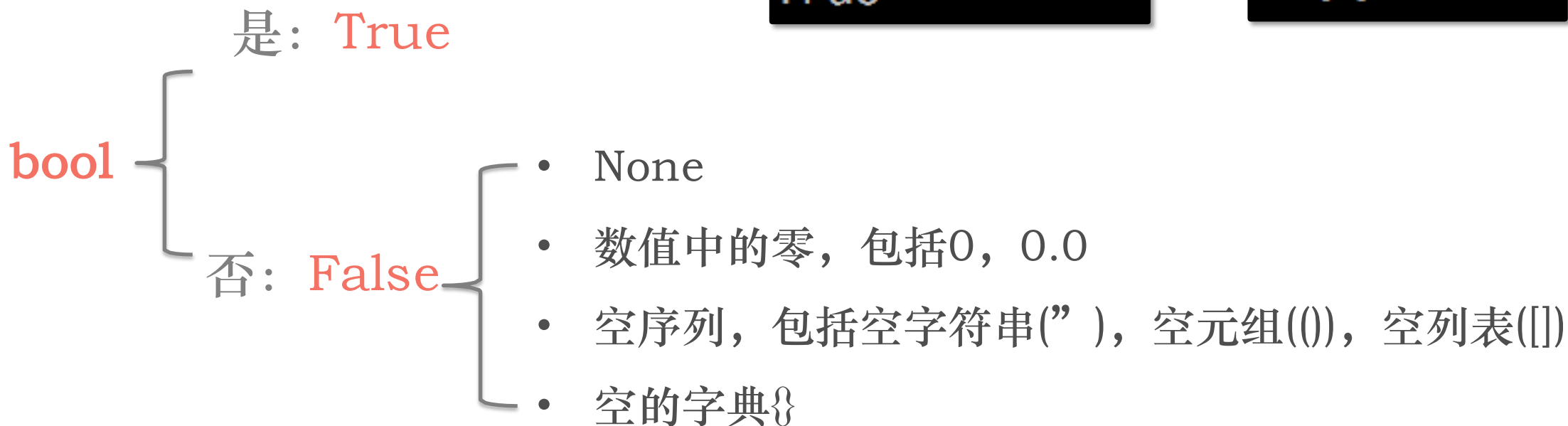


知识点复习：布尔值

意义：表示判断中的是与否。
一般用于条件测试中。

```
>>> a = True
>>> print(a)
True
```

```
>>> 10<5
False
>>> 10>8
True
```



逻辑运算

逻辑运算符：用于检测两个或两个以上的条件是否满足。

逻辑运算只存在与布尔类型中。

运算符		描述
and,	逻辑“与”	当运算符两边的两个运算对象都为true时，结果为true
or,	逻辑“或”	当运算符两边的两个运算对象其中有一个运算对象为true时， 结果 即为true
not,	逻辑“非”	用于反转运算对象的状态

布尔表达式

- 3 and 5 True and True True 5
- 3 or 5 True or True True 3
- 0 or 5 False or True True 5
- 3 and not 5 True and False False False

表达式的应用——条件测试

- 检查当前变量是否与一个特定值相等/不相等。

- 比较数字的大小。

- 检查特定值是否在某序列里。

表达式的应用——多条件检查

- 使用`and`检查多个条件

```
>>> age_lilei = 17
>>> age_hanmeimei = 18
>>> age_lilei >= 18 and age_hanmeimei >= 18
False
>>> age_lilei >= 15 and age_hanmeimei >= 15
True
```

- 使用`or`检查多个条件

```
>>> age_lilei >= 18 or age_hanmeimei >= 15
True
>>> age_lilei >= 20 or age_hanmeimei >= 20
False
```

知识点复习：Python代码缩进问题

用四个空格或者一个Tab来表示缩进都可以，但不要混用

相同缩进位置的代码表示他们是同一个代码块

```
China:
    Tsinghua University:
        lilei
        hanmeimei
    Peking University:
        lilei
        madongmei
America:
    MIT:
        xiaoming
        Jane
    Stanford:
        xiaohong
        David
```

```
import pyodbc

conn = pyodbc.connect("DSN=xxx;PWD=password")
cursor = conn.cursor()
cursor.execute("SELECT field1 FROM table1")

while True :
    row = cursor.fetchone()
    if not row :
        break
    print(row)

cursor.close()
conn.close()
```

一、单分支结构

else 可以不出现!

1.1 基本语句结构

```
1 | if <条件> :  
2 |     <执行语句块>
```

1.2 示例代码

```
1 | input_number = 12  
2 | if input_number%2 == 0:  
3 |     print("This is an even number")
```

输出结果:

```
1 | This is an even number  
2 | [Finished in 0.8s]
```

我们看到，程序成功的判断除了这是一个偶数

二、二分支结构

2.1 基本语句结构

```
1 | if <条件> :  
2 |     <语句块1>  
3 | else :  
4 |     <语句块2>
```

2.2 示例

```
1 | input_number = 12  
2 | if input_number%2 == 0:  
3 |     print("This is an even number")  
4 | else:  
5 |     print("This is an odd number")
```

输出结果：

```
1 | This is an even number  
2 | [Finished in 0.4s]
```

三、多分支结构

else 也可以不出现!

3.1 基本结构语句

```
1 if <条件1> :  
2     <语句块1>  
3 elif <条件2> :  
4     <语句块2> .....  
5 else : elif  
6     <语句块N>
```

3.2 示例程序

```
1 score = 86  
2 if score >= 90:  
3     grade = "A"  
4 elif score >= 80:  
5     grade = "B"  
6 elif score >= 70:  
7     grade = "C"  
8 elif score >= 60:  
9     grade = "D"  
10 print("输入成绩属于级别{}".format(grade))  
11
```

输出的结果是:

```
1 输入成绩属于级别B  
2 [Finished in 0.4s]
```

Example:条件判断与列表综合

5.1 一个简单示例

下面是一个简短的示例，演示了如何使用if语句来正确地处理特殊情形。假设你有一个汽车列表，并想将其中每辆汽车的名称打印出来。对于大多数汽车，都应以首字母大写的方式打印其名称，但对于汽车名'bmw'，应以全大写的方式打印。下面的代码遍历一个列表，并以首字母大写的方式打印其中的汽车名，但对于汽车名'bmw'，以全大写的方式打印：

`cars.py`

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
```

```
for car in cars:
```

```
❶   if car == 'bmw':  
       print(car.upper())  
   else:  
       print(car.title())
```

```
Audi  
BMW  
Subaru  
Toyota
```

Example:列表、for循环、if语句的综合使用

★ 来看看在制作比萨前如何拒绝怪异的配料要求。下面的示例定义了两个列表，其中第一个列表包含比萨店供应的配料，而第二个列表包含顾客点的配料。这次对于requested_toppings中的每个元素，都检查它是否是比萨店供应的配料，再决定是否在比萨中添加它：

```
❶ available_toppings = ['mushrooms', 'olives', 'green peppers',  
                        'pepperoni', 'pineapple', 'extra cheese']  
  
❷ requested_toppings = ['mushrooms', 'french fries', 'extra cheese']  
  
❸ for requested_topping in requested_toppings:  
❹     if requested_topping in available_toppings:  
        print("Adding " + requested_topping + ".")  
❺     else:  
        print("Sorry, we don't have " + requested_topping + ".")  
  
print("\nFinished making your pizza!")
```

思考：

这段代码如何打印？

公布答案!

```
1 available_toppings = ['mushrooms', 'olives', 'green peppers',  
2                       'pepperoni', 'pineapple', 'extra cheese']  
3 requested_toppings = ['mushrooms', 'fresh fries', 'extra cheese']  
4  
5 for requested_topping in requested_toppings:  
6     if requested_topping in available_toppings:  
7         print("Adding " + requested_topping + ".")  
8     else:  
9         print("Sorry, we don't have " + requested_topping + ".")  
10 print("\nFinished make your pizza!")
```

```
Adding mushrooms.  
Sorry, we don't have fresh fries.  
Adding extra cheese.
```

```
Finished make your pizza!  
[Finished in 0.1s]
```


使用for循环和if语句判断一个数是否为质数

```
1 import math
2 def isprime(num):
3     for i in range(2,int(math.sqrt(num))+1):
4         if num%i == 0:
5             return False
6     return True
7
8 for i in range(2,20):
9     if isprime(i):
10        print(f'{i} is a prime number.')
11    else:
12        print(f'{i} is not a prime number.')
```

```
2 is a prime number.
3 is a prime number.
4 is not a prime number.
5 is a prime number.
6 is not a prime number.
7 is a prime number.
8 is not a prime number.
9 is not a prime number.
10 is not a prime number.
11 is a prime number.
12 is not a prime number.
13 is a prime number.
14 is not a prime number.
15 is not a prime number.
16 is not a prime number.
17 is a prime number.
18 is not a prime number.
19 is a prime number.
```

作业

- 阅读《Python Crash Course》CH3-CH5，完成各自章节练习
- 使用for循环打印九九乘法表
- 根据泰勒级数计算圆周率 π

Chapter 3: Introducing Lists

Chapter 4: Working with Lists

Chapter 5: if Statements

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots$$

