上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

Python基础：选择和循环

课程组：叶南阳，郑冠杰，温颖

布尔类型

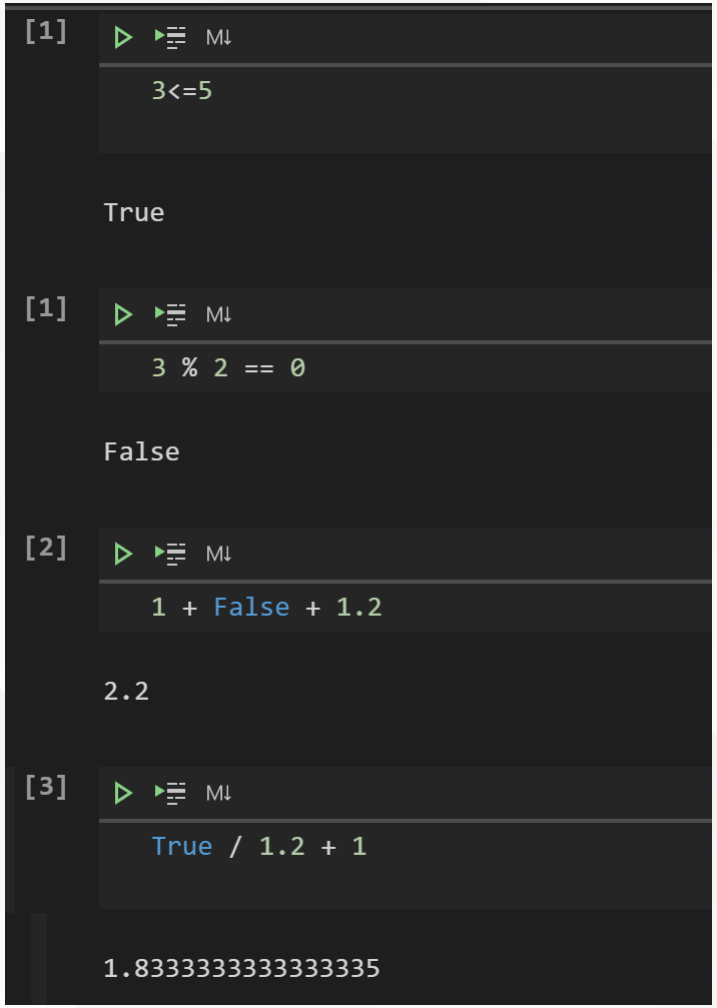
- 现实生活中我们如何表达真还是假？如何表达判断和对错？英语中我们用True和False对一个观点或者论断进行判断对错。“太阳东升西落”这句话是True。而“学这门课的同学们最后拿到的成绩都是D”这句话是False。
- 布尔类型就是Python中判断True还是False的一种基本数据类型。在Python中，代表布尔类型的基本变量只有两个，就是我们熟悉的True和False。True代表恒真，False代表恒假。其他的布尔类型通常出现在条件判断语句中，它在Python解释器下会被运算最后得到布尔类型的值，这种值一般用于if条件语句或者for、while等循环判断语句中。
- 布尔类型在适当的情况下可以被当成整数类型进行计算。True会在参与数值计算的时候自动解释成1，而False会被解释成0。

布尔类型的notebook实践

- 在测试简单的程序的时候，输入一个表达式按下shift+Enter就可以快速得到结果并换行

- In [1]: 3<=5
- Out[1]: True
- In [2]: 3 % 2 == 0
- Out[2]: False

- In [3]: 1 + False + 1.2
- Out[3]: 2.2
- In [4]: True / 1.2 + 1
- Out[4]: 1.8333333333333335

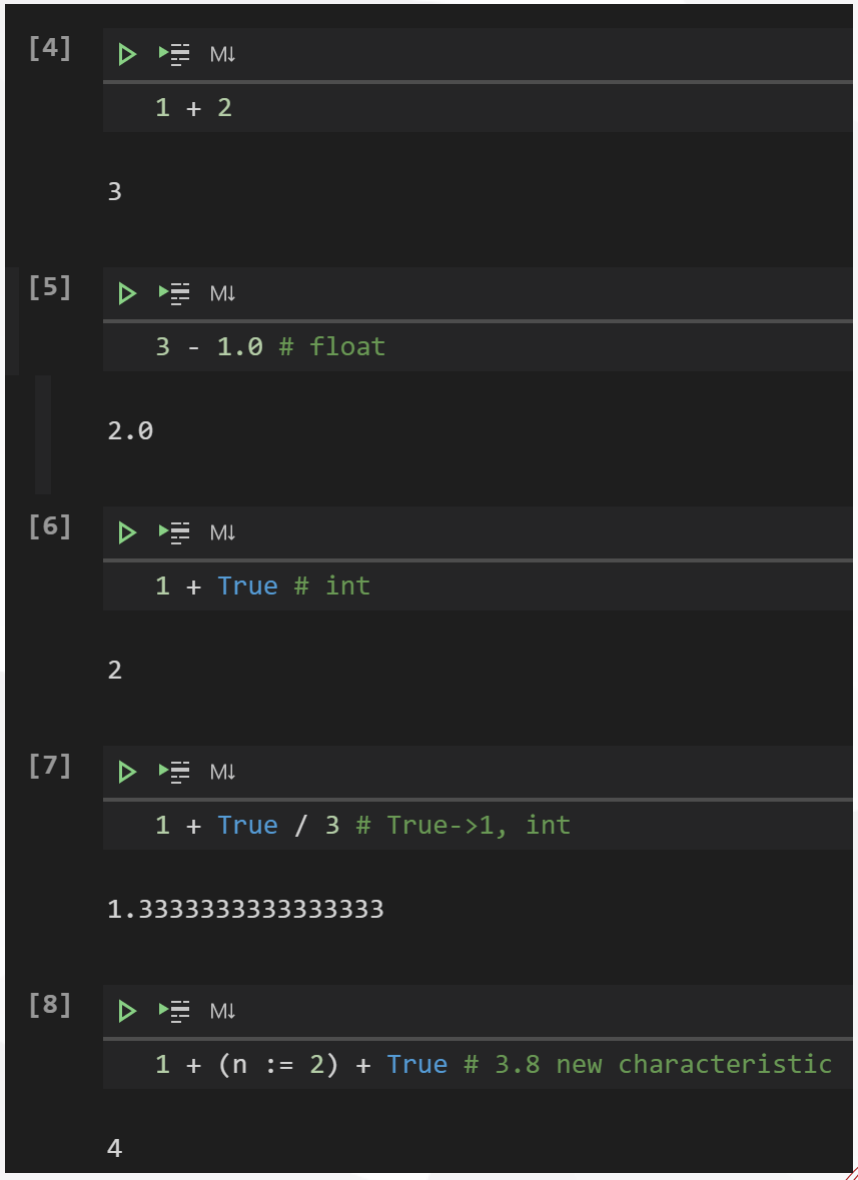


表达式和数值

- 到目前为止，我们已经知晓了Python中一些基本数据类型：整数、浮点数以及布尔类型。那么当我们需要写一些复杂的表达式的时候，我们就可以通过我们对基本数据类型的组合，得到一个复杂，同时具有更多功能或者信息的表达式。
- 表达式可以分成算术表达式、逻辑表达式、赋值表达式以及混合表达式等类型。算术表达式就是指只有数值参与运算的表达式，例如加减乘除乘方等运算；逻辑表达式大部分是相等，小于等比较运算；赋值表达式就是对一个变量进行赋值；混合表达式就是一个表达式中混合有多种表达式类型。
- 每个表达式都可以计算得到值，因为本质上表达式都能被Python解释器解释成值。但是如果表达式的数据类型不一致的时候，Python就会将表达式的数据类型统一，也就是进行了隐式的数据类型转换。转换的规则如下：
- 如果是逻辑表达式，那么表达式的数值类型就是布尔类型
- 如果有整数参与运算，那么表达式的数值类型就是整数类型（布尔类型会隐式转换成整数）
- 如果有浮点数参与运算，那么表达式的数值类型就是浮点数类型

表达式和数值notebook实践

```
In [5]: 1 + 2

Out[5]: 3

In [6]: 3 - 1.0 # float

Out[6]: 2.0

In [7]: 1 + True # int

Out[7]: 2

In [8]: 1 + True / 3 # True->1,

Out[8]: 1.3333333333333333

In [9]: 1 + (n := 2) + True # 3.8

Out[9]: 4
```



条件控制语句

- 接下来我们考虑另外一种情况。我们不仅想要按照顺序执行一段程序，我们还想让程序在一些情况下做一件事，另一些情况下做另一件事。程序对于条件语句的理解和我们日常生活对于多情况的理解略有不同，请看下面的例子：
- 例如：妻子对程序员丈夫说，下班后买点海鲜，如果下班太晚了，就买点肉。结果丈夫21点回到家后，带回来了一条鱼和一斤牛肉。
- 原因就在于丈夫对于买海鲜这件事的理解是一定会发生，而下班太晚这个分支情况是否满足对应着肉是否需要买这个事件。这是符合程序的理解方式的，但是不符号日常生活的理解方式。因此当我们使用条件控制语句的时候，一定要注意程序逻辑之间的关系，把握好if语句在处理分支情况下时是否和我们脑中想要程序做的事一致，否则很有可能出现程序运行结果错误但是当我们想debug的时候却怎么也找不出错误的情况。

条件控制语句（cont'd）

- 在Python中，我们使用if语句进行条件判断。主要的语法格式如下：
- if (bool expression):
- 　　# do something
- 整个if下的代码段使用tab键进行缩进，表示如果括号内的布尔表达式的求值结果为True，那么就执行if冒号下的代码片段，如果布尔表达式求值结果为False，那么就不执行if冒号下的代码段。需要注意的是，布尔表达式外的括号不是必须的，但是有的时候为了美观可以加上括号。
- 刚才丈夫买鱼的例子就是丈夫理解成了

```
买点海鲜
If（下班太晚）：
    买点肉
```

条件控制语句（cont'd）

- 如果我们不仅想要让程序在条件判断为真的时候做一些操作，还想让程序在条件判断为假的时候做另一些操作，就需要双向的条件判断语句：if-else。主要的语法格式如下：
- if (bool expression):
- 　　# do something
- else:
- 　　# do something else
- 那么当布尔表达式的求值结果为True的时候执行if下的代码段；而求值结果为False时执行else下的代码段。
- 在上面的例子中，妻子对于丈夫的期望实际上应该是：

```
if (下班太晚）:
    买点肉
else:
    买点海鲜
```

条件控制语句（cont'd）

- 有的时候当我们的程序有多个分支情况出现时，就需要嵌套if语句。语法如下：

```
if (bool expression1):
    # do something
    if (bool expression2):
        # do something else
    # next program
```

- 嵌套if语句主要用于帮助我们快速完成简单的分支，复杂的分支留在#next program 部分进行设计。理论上嵌套的if数量不限，但是为了程序的逻辑清晰和美观，一般不超过两个嵌套。

- 除了使用嵌套if语句，我们还使用if-elif-else语句进行多分支程序设计。语法如下：

```
if (expression1):
    # do a
elif (expression2):
    # do b
else:
    # do d
```

- 如果表达式1求值为True，执行a语句；如果为False，就对表达式2求值，如果为True执行b语句，反之执行d语句。用于应对多分支情况，elif数量上不限。

条件控制语句的notebook实践

- 例如：给定一个学生的学积分，判断这个学生的绩点。（只考虑3.0以上的情况）
- 思路：按照学积分和绩点的关系进行条件判断即可：
- [95, 100]: 4.3
- [90, 94]: 4.0
- [85, 89]: 3.7
- [80, 84]: 3.3
- [75, 79]: 3.0
- 思考1：用嵌套if语句好吗？
- 思考2：调换elif的顺序可行吗？

```
[9]
grade = int(input("Enter a student's grade:"))
if grade >= 95:
    print("GPA = 4.3")
elif grade >= 90:
    print("GPA = 4.0")
elif grade >= 85:
    print("GPA = 3.7")
elif grade >= 80:
    print("GPA = 3.3")
else:
    print("GPA = 3.0")

GPA = 3.0
```

While循环语句

- 在程序中，我们希望能够执行重复性的工作，例如统计1-10000里面的质数个数并且把它们打印出来。传统的if语句当然可以实现，但是10000个if判断实在对编程人员太不友好。因此我们想通过一些手段编写程序，使得我们可以用很短的程序块表达重复性的工作内容，那么这个时候就需要循环语句帮助。我们首先介绍While循环，语法如下：

```
while expression1:
    # do something
```

- 如果表达式1求值为True，那么会不断执行冒号下的语句，这段语句也就做循环体，执行到表达式1求值为False。因此，通常情况下表达式1中的变量必须在while循环体中不断变化，否则会陷入死循环中，程序将一直执行下去。
- 可以看到，while循环的特点是可以不给出循环终止条件，也没有循环范围的限定，我们可以让表达式恒真使得循环体一直执行下去，例如桌面窗口应用程序中，窗口页面运行过程就是一个恒真表达式，当点击 ✕ 关闭程序的时候，该循环语句才会终止。这个时候窗口消失，程序运行结束。

While循环notebook实践

- 例：如何编写求任何底数任何次幂的幂运算（复现Python “**” 运算符的功能）？
- 思路：考虑这样的一个情况：$a^n = a^{\{\frac{n}{2}\}} \times a^{\{\frac{n}{2}\}}, if\ n\ is\ even$。
- 再考虑另一种情况：$a^n = a^{(n-1)/2} \times a^{(n-1)/2} \times a, if\ n\ is\ odd$。
- 根据这两种情况，我们就可以考虑这样一种思路：随着n的每次整除2，a都变为原来的a倍（为什么？），那么只要当n为奇数的时候，我们乘上当前的a即可得到技术情况下的$a^{\{\frac{n-1}{2}\}} \times a$的结果，随后循环往复即可。（提示：逆向思考这个过程，这个过程模拟的是人脑中计算幂的过程）
- 代码如下：

[4]

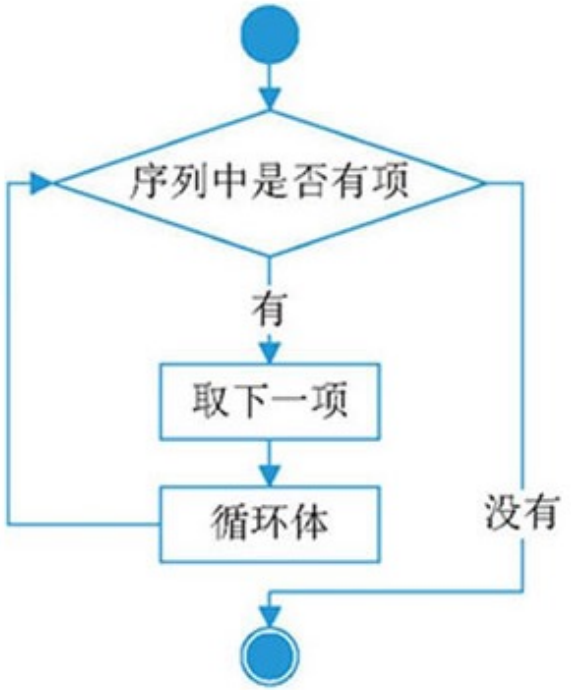
```
# 计算a和n次幂
a = int(input())
n = int(input())
ans = 1
while n > 0:
    if n & 1:
        ans *= a
    a *= a
    n >>= 1
print(ans)
```

for循环语句

- Python 中的循环语句有 2 种，分别是 while 循环和 for 循环，前面已经对 while 做了详细的讲解，本节给大家介绍 for 循环，它常用于遍历字符串、列表、元组、字典、集合等序列类型，逐个获取序列中的各个元素。语法如下：

```
for iterating_var in sequence:
    # do something
```

- 格式中，迭代变量用于存放从序列类型变量中读取出来的元素，所以一般不会在循环中对迭代变量手动赋值；代码块指的是具有相同缩进格式的多行代码（和 while 一样），由于和循环结构联用，因此代码块又称为循环体。



for循环notebook实践

- 例：如何编写想要实现从 1 到 100 的累加？
- 思路：使用for循环进行累加操作即可：
- sum = 0
- sum = 1
- …
- sum = 1 + 2 + 3… + 100
- 代码如下：

```
# 保存累加结果的变量
sum = 0
# 逐个获取从 1 到 100 这些值，并做累加操作
for i in range(101):
    sum += i

print("计算 1+2+...+100 的结果为: ")
print(sum)
```

运行结果：

```
计算 1+2+...+100 的结果为:
5050

Process finished with exit code 0
```

嵌套循环

- Python 语言允许在一个循环体里面嵌入另一个循环。
- 你可以在循环体内嵌入其他的循环体，如在while循环中可以嵌入for循环， 反之，你可以在for循环中嵌入while循环。
- Python for 循环嵌套语法：

```
for iterating_var in sequence :
    for iterating_var in sequence :
        statements(s)
    statements(s)
```

- Python while 循环嵌套语法：

```
while expression :
    while expression :
        statements(s)
    statements(s)
```

嵌套循环notebook实践

- 例：如何使用嵌套循环输出2~50之间的素数？
- 代码如下：

```
i = 2
while i < 50:
    j = 2
    while j <= (i/j):
        if not (i % j): break
        j = j + 1
    if j > i/j:
        print(i, " 是素数")
    i = i + 1
```

- 运行结果：

```
2  是素数
3  是素数
5  是素数
7  是素数
11  是素数
13  是素数
17  是素数
19  是素数
23  是素数
29  是素数
31  是素数
37  是素数
41  是素数
43  是素数
47  是素数

Process finished with exit code 0
```

Python程序实例

- 实例展示一：求两数最大公约数和最小公倍数

Python程序实例

- 实例展示二：判断输入年份是否为闰年

感谢聆听！

饮 水 思 源　爱 国 荣 校