字符串操作，猜测和检查，近似值，平分

（下载幻灯片和 .py 文件 Ă Ě 跟随！）

6.0001 讲座 3

上次

字符串

分支 if/elif/else

while循环

for 循环

今天

字符串操作

字符串

认为是区分大小写的字符序列

可以用 ==、>、< 等比较字符串。

len() 是用于检索括号中字符串长度的函数

s = abc

len(s) 计算结果为 3

字符串

方括号用于对字符串执行索引以获取某个索引/位置的值

s = abc

指数: 0 1 2 索引总是从 0 开始

指数: -3 -2 -1 最后一个元素总是在索引 -1

s[0] 计算结果为“a”

s[1] 计算结果为“b”

s[2] 计算结果为“c”

s[3] 试图索引越界,错误

s[-1] 计算结果为“c”

s[-2] 计算结果为“b”

s[-3] 计算结果为“a”

字符串

可以使用 [start:stop:step] 对字符串进行切片

如果给出两个数字,[start:stop],默认 step=1

你也可以省略数字,只留下冒号

s = abcdefgh

s[3:6] 计算结果为“def”,与 s[3:6:1] 相同

s[3:6:2] 计算结果为“df”

s[::] 计算结果为“abcdefgh”,与 s[0:len(s):1] 相同

s[::-1] 计算结果为“hgfedbca”,与s[-1:-(len(s)+1):-1]相同

s[4:1:-2] 计算结果为“ec”

If unsure what some command does, try it out in your console!

字符串

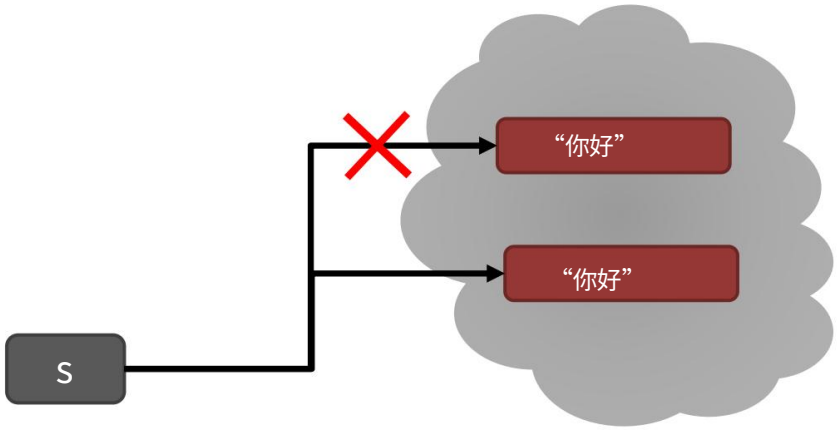
字符串是“不可变的”　　不能修改

s = 你好

s[0] = y s　　给出错误

= y +s[1:len(s)]　　是允许的，

s 绑定到新对象



循环回顾

for 循环有一个循环变量,它遍历一组值

for var in range(4): var 迭代值 0,1,2,3
与 var 的每个值 循环内的表达式执行<expressions>

for var in range(4,6): var 迭代值 4,5
<表达式>

range 是一种迭代数字的方法,但是 for 循环变量可以迭代任何一组值,而不仅仅是数字!

字符串和循环

这两个代码片段做同样的事情底部一个更“pythonic”

```
s =  abcdefgh
对于范围内的索引（len（s））：
    如果 s[index] ==  i  或 s[index] ==  u  ：
        print(  有一个 i 或 u  )

对于 s 中的字符：
    如果 char ==  i  或 char ==  u  ：
        print(  有一个 i 或 u  )
```

代码示例：

机器人啦啦队

an_letters = aefhilmnorsxAEFHILMNORSX

对于单词中的字符：

锻炼

```
s1 = “我的摇滚”
s2 =    我统治 mit
如果 len(s1) == len(s2)：
        对于 s1 中的 char1：
            对于 s2 中的 char2：
                如果 char1 == char2：
                    print(    普通字母    )
                    休息
```

猜测和检查

下面的过程也称为穷举

GUESS-AND-CHECK - 立方根

```
立方体 = 8

对于范围内的猜测（立方体+1）：

    如果猜测**3 == 立方体：

        打印（“立方根”,立方，“是”,猜测）
```

GUESS-AND-CHECK - 立方根

```
立方体 = 8

对于范围内的猜测（abs（cube)+1）：

        如果猜测**3 >= abs(cube)：

                休息

如果猜测**3 != abs(cube):

        print(cube,    不是一个完美的立方体    )

别的：

        如果立方体 < 0：

                猜测 = - 猜测

        print(      +str(cube)+    的立方根是    +str(guess))
```

近似解决方案

足够好的解决方案

从猜测开始,然后增加一些小值

继续猜测|guess3-cube| >= epsilon 表示一些小的 epsilon

减小增量大小　　较慢的程序

增加ε　　不太准确的答案

近似解 立方根

立方体 = 27

epsilon = 0.01 猜测 = 0.0

增量 = 0.0001

num_guesses = 0 而

abs(guess**3 - cube) >= epsilon:和 guess <= cube:

猜测 += 增量

num_guesses += 1

print(num_guesses = , num_guesses) if

abs(guess**3 - cube) >= epsilon:

print(立方根失败 , cube)

别的:

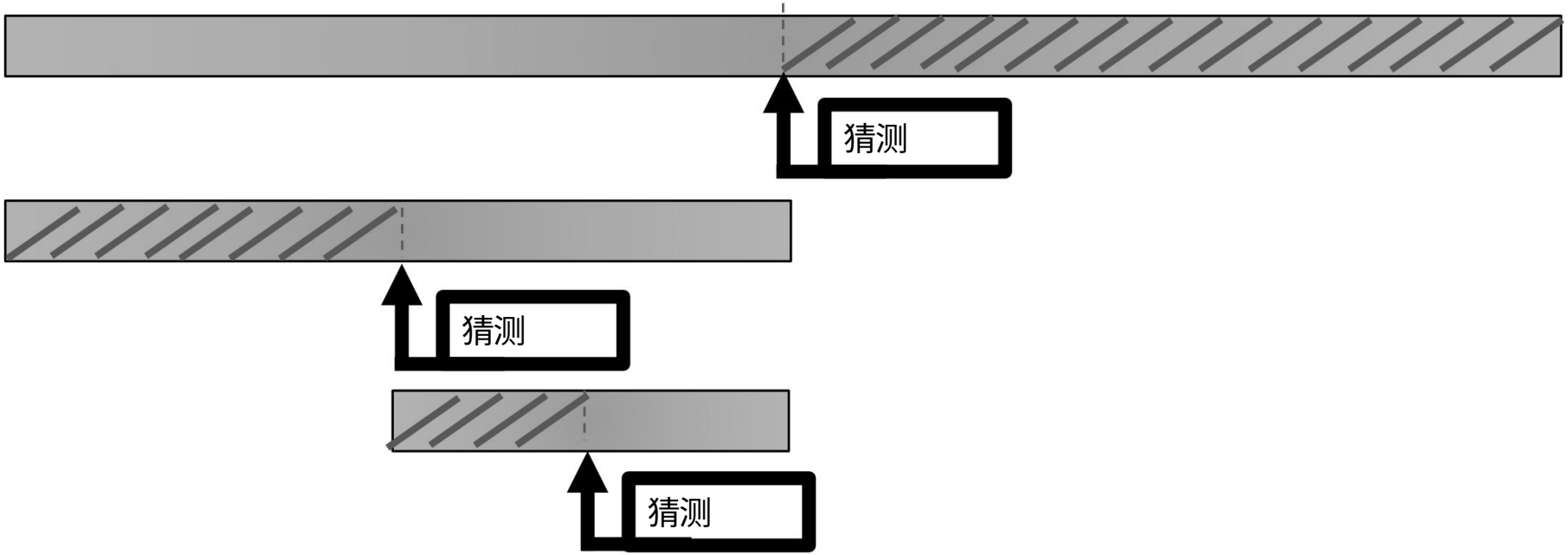
print(guess, 接近立方根 , cube)

平分搜索

每次迭代的半间隔

新的猜测介于两者之间

说明,让我们玩一个游戏!



BISECTION 搜索- 立方根

立方体 = 27

epsilon = 0.01 num_guesses

= 0 低 = 0

高 = 立方猜测 = (高 +

低)/2.0 而 abs(猜测**3 - 立方) >= epsilon: 如果猜

测**3 < 立方: 低 = 猜测

别的：

high =guessguess =

(high + low)/2.0 num_guesses += 1 print

num_guesses = , num_guesses print

guess, 接近立方根 , cube

平分搜索

收敛

搜索空间

- 第一个猜测： N/2
- 第二个猜测： N/4

第 k 个猜测： N/2k

x < 1

如果 x < 1,搜索空间为 0 到 x 但立方根大于x 且小于 1

修改代码以根据 x 的值选择搜索空间

6.0001 计算机科学和 Python 编程简介

2016 年秋季