

章节六:条件语句

6.0.1 教学目标

学生一般认为条件语句(也称为选择语句或决定语句)相比于其他的结构更容易理解。关于条件语句的主要困惑发生在学生开始用布尔组合时,例如:英语中随便用的单词“and”与布尔逻辑中的形式词“and”有一些不同。在这个单元的最后,学生应该能运用条件语句,创建具有比较的布尔表达式,他们也应该能正确的推出布尔操作符“and”、“or”和“`not`”的用法。

6.0.2 主题大纲

6.0 章节介绍
6.0.1 教学目标
6.0.2 主题大纲
6.0.3 关键术语
6.0.4 关键概念
6.1 课程计划
6.1.1 建议的时间表
6.1.2 CSTA标准
6.1.3 课程计划一 使用控制块-If,If...Else
6.1.4 课程计划二 修改问答机器人的循环赋值
6.1.5 课程计划三 使用布尔表达式
6.1.6 课程计划四 设计一个猜大小游戏和赛车游戏
6.2 资源
6.2.1 附加练习

6.0.3 关键术语

布尔值:true/false	If then else,If then else if else
简单与复杂的表达式	匹配功能
AND/OR-操作符	
is(比较),isnt,<,>;,	
数值比较	
字符串比较	

6.0.4 关键概念

使用条件控制代码

单词if能够用来使一块代码在一个条件的控制下,这块代码仅仅在条件为真时运行。

下面是一个用if通过检测键盘按下来控制乌龟的运动。

缩进的代码fd 2只在条件pressed('W')为真,也就是,当使用者正在按w键的时候运行。相似的,另外两行代码rt 2; dot blue, 5只在使用者按下d键时运行。

如果这两个键都没被按下,这两块缩进的代码都不会运行。如果这两个键都被按下,这两块代码都会运行。

```
forever ->
  if pressed('W')
    fd 2
  if pressed('D')
    rt 2
    dot blue, 5
```

用if检测键盘按下

对于两种选择中另一种使用“else”

当“if”没有发生的时候,关键词“else”允许你编写要使用的第二种行动。第二块代码将会在条件为假的时候发生。

```
forever ->
  if pressed('W')
    fd 2
  else
    rt 2
```

用if/else提供两种选择

这个程序在w被按下的时候将乌龟向前移动。当w没有按下的时候, 旋转乌龟自身。

对于多种选择使用“else if”链接

当存在三种甚至更多的行动时, if和else将被链接起来。

```
forever ->
  if pressed('W')
    fd 2
  else if pressed('S')
    bk 2
  else
    rt 2;
```

对于三种选择, 链接if, else if和else

这个代码中, 当w被按下时向前移动乌龟, 当s被按下时向后移动乌龟。如果什么都没有按下, 乌龟将向右自旋。链接的if/else只会选择为真的第一个条件, 因此, 当w与s都同时被按着时, 程序将只执行“fd 2”, 并不会向后移动 乌龟。

用“and”, “or”和“not”组成布尔表达式

词and, or和not是可以被用来组合条件的布尔操作符。例如, 下面的程序用了“and”和“not”。程序绘制了一个蓝色的环, 然后只有在w被按下并且乌龟没有接触到蓝色的环的时候向前移动乌龟。

```
dot blue, 500
dot white, 400
forever ->
  if pressed('W') and not touches('blue')
    fd 2
  else
    rt 2
```

用and和or的组合测试

尽管布尔操作符通常与在英语单词中读到它们有同样的意思, 但是理解它们作为数学操作符究竟如何发挥作用很重要。因为很容易产生意外的影响。

搞混“and”和“or”

考虑一个程序, up键和w键需要起同样的作用, 用同样的方法向前移动乌龟。我们可能会想着用and组合来用一个“if”来同时捕获这两种情况, 就像这样:

```
WRONG:

forever ->
  if pressed('up') and press('W')
    fd 2
```

错误的使用and组合两种情况

可是, 这段代码不会产生想要的结果! 为了理解原因, 我们需要理解and和or是如何操作真值的。

布尔值和布尔表

词and, or和not是组合“true”和“false”的布尔操作符(与你使用“+”, “x”组合常规数字的算术规则相似)。就像我们能够通过建立加法表和乘法表学习加法和乘法一样, 我们可以通过写下真值表理解and和or。这里有两个关于上面的程序的真值表:

pressed('up') and pressed('W')	pressed 'up' = false	pressed 'up' = true
pressed 'W' = false	false	false
pressed 'W' = true	false	true

pressed('up') or pressed('W')	pressed 'up' = false	pressed 'up' = true
pressed 'W' = false	false	true
pressed 'W' = true	true	true

逻辑与and组合两个布尔值, 只有在两个值都为true的时候“true”。例如, pressed('up') **and** pressed('W'), 只有在up键和w键同时都被按下时为true。

逻辑或or组合两个布尔值, 只要两个值之一为true就产生true。例如, pressed('up') **or** pressed('W'), 在只有w被按下, 或者只有up被按下, 或者两者都被按下时为“true”。这是我们想在程序里实现的结果。

把and改成or以修复程序。

用比较操作符测试数字

布尔表达式能够用来检测数字的属性。你在数学课堂上见到的大多数比较操作符能够在一个编程语言中使用, 但是它们可能被写做稍微不同的符号。例如, “小于或等于”被写做<=。这里有一个一些常见的数字布尔测试的总结:

表达式	描述	x = 0时的值	x = 3时的值	x = 6时的值
x is 3	x等于3	false	true	false
x isnt 3	x不等于3	true	false	true
x < 3	x小于3	true	false	false
x <= 3	x小于或等于3	true	true	false
x > 3	x大于3	false	false	true
x >= 3	x大于或等于3	false	true	true
0 < x <= 6	x大于0且小于或等于6	false	true	true
x % 2 is 1	x为奇数 (因为x除以2的余数为1)	false	true	false
x % 3 is 0	x被3整除	false	true	true

搞混“or”和比较

数值比较能够用布尔操作符组合。例如, (x > 6 and x isnt 9)意味着x是一个大于6且不等于9的数, (x is 5 or x is 11)意味着x或者是5或者是11。重要的是, 记住词or是操作真值而非操作数字的。因此左边版本的程序不会产生需要的结果。

WRONG: await readnum 'How many items?', defer n if n is 1 or 2 write 'Come to the speedy checkout.'	RIGHT: await readnum 'How many items?', defer n if n is 1 or n is 2 write 'Come to the speedy checkout.'
---	--

左边的程序无论你输入什么都会错误的导致快速检验行。为了理解原因, 记得or操作的是真值, 因此, 当你说“or 2”, 它将判断“2是true还是false?”这个问题。按照惯例, 任意非零数字都当作“true'”, 因此“or 2”使得不管num的值为多少这个表达式总是真。另一方面, 右边的程序将产生需要的结果: 只有number是2的时候使表达式“or n is 2”为真。

要考虑的另一个区别是操作符的优先级。词or比is的优先级低, 因此左边的表达式这样读: ((n is 1) or 2), 右边的表达式这样读: ((n is 1) or (n is 2))。

使用模式匹配测试字符串

测试文本字符串也能用来创建布尔值字符串的方法有准确地比较它们(查看它们的长度)或者用“match”方法检测字符串是否匹配一个模式。模式匹配可以用来确定一个字符串里面是否包含一个特定的字母模式。

下面的表格展示了一些例子。

Expression	Description	"appear"	"pear"	"peachy"
x is "pear"	x就是等于字符串"pear"	false	true	false
x.length is 6	x刚好有6个字符	true	false	true
x.match(/pp/)	x包含子字符串"pp"	true	false	false
x.match(/pea/)	x包含子字符串"pea"	true	true	true
x.match(/PEA/)	x包含子字符串"PEA"	false	false	false
x.match(/Pea/i)	x包含子字符串"Pea", 忽略大小写	true	true	true
x.match(/^pea/)	x以"pea"为字符串开头	false	true	true
x.match(/ear\$/)	x以"ear"为字符串结尾	true	true	false
x.match(/a(p ch)/)	x包含"a"且"a"后跟"p"或"ch"	true	false	true
x.match(/ap*e/)	x包含"a", 后跟0个或更多"p", 后跟"e"	true	false	false

在两个符号"/"中间用 的模式被叫做正则表达式。

一个正则表达式能够用来检测一个字符串是否包含一个固定的模式, 例如字符串是否包含字母"pp"。一般地, 正则表达式是是区分大小写的, 所以"PEA"不能匹配"pea", 但是在正则表达式后面加一个"i"能使其不区分大小写。

正则表达式模式有一些强大的功能。例如, 在一个正则表达式中, "^"匹配字符串的开头, "\$"匹配字符串的结尾, "(one|other)"用来匹配备选项, "*"允许一个子样式重复0次或更多次。

尽管上面展示的样式类型对于大多数情况已经足够, 正则表达式还有更多功能。如果你搜索“regular expression lessons”, 会发现互联网上还有很多关于正则表达式的优秀的资源。当你探索时, 你应该知道正则表达式中的符号是标准化的, 在JavaScript, CoffeeScript, Python, Perl, Java, C#与其他语言中使用的是同一种样式。

6.1.1 建议的时间表

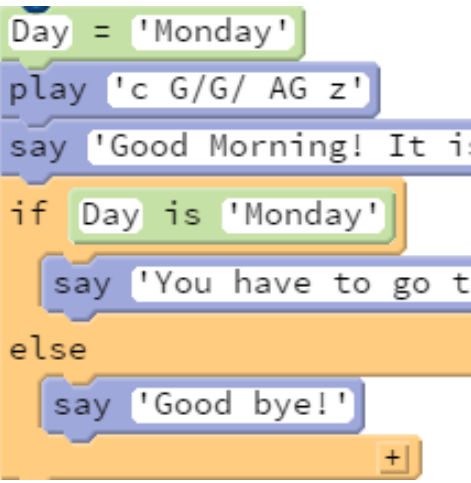
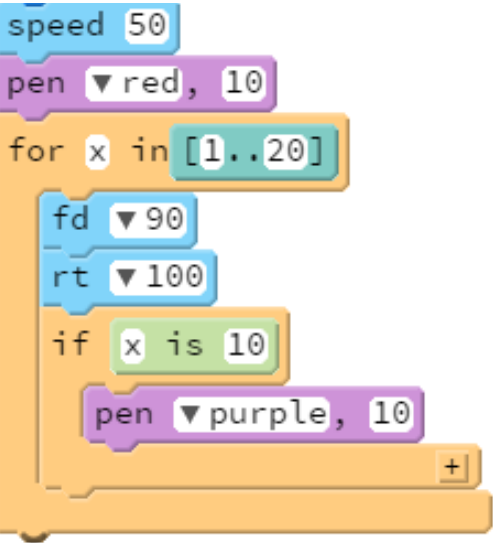
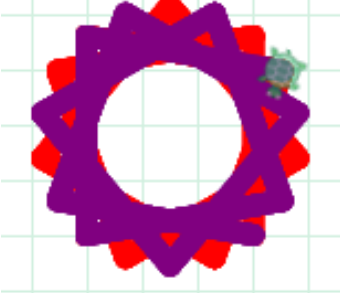
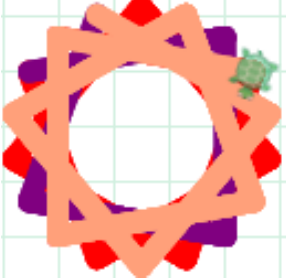
教学日	主题
2天	课程计划一: If, If Then Else语句使用有趣的可视化元素
1天	课程计划二: 问答机器人 与 课程计划三: 复杂`If`语句
1天	课程计划四: 结对编程猜大小游戏
1天	课程计划五: 赛车游戏

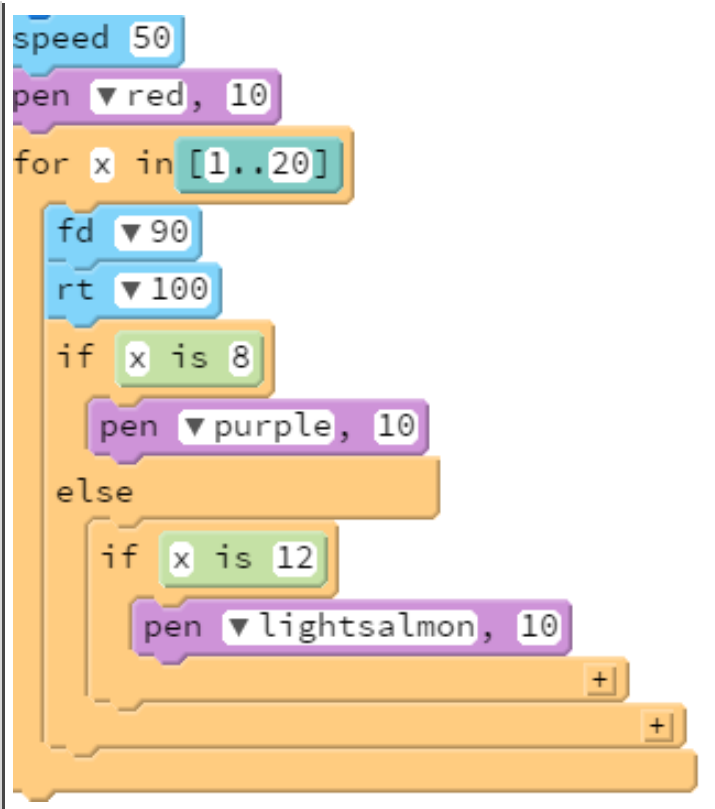
6.1.2 标准

CSTA标准	CSTA链	CSTA学习目标覆盖
Level 3 A (Grades 9 - 12)	计算思维(CT)	解释如何通过排序, 选择, 迭代和递归构成算法块
Level 3 A (Grades 9 - 12)	CT	解释程序执行进程
Level 3 A (Grades 9 - 12)	CT	描述计算中如何使用数学与统计功能, 集合和逻辑

6.1.3 课程计划一

这节课聚焦于控制块, If语句和If Else语句。这节课应当花费一个课时的30到40分钟, 剩余的时间让学生仿照这个程序编写一个相似的程序。

详细内容	教学建议	时间
<div>Code</div> <div><pre>Day = 'Monday' play 'c G/G/ AG z' say 'Good Morning! It is time to wake up' if Day is 'Monday' say 'You have to go to school today' else say 'Good bye!'</pre></div> <div></div>	<p>解释布尔表达式计算的关键概念。</p> <p>展示控制块和‘IF’语句</p> <p>输入左一系列的代码 (注意: 对于这个程序, 你可能需要一个扬声器。你可以使用write块取代say块)</p> <p>教学提示: 你可以通过在整段代码外添加一个循环使其以一个固定周期多次 (如: 5次) 播放这个曲调来扩展这个课程。解释这个程序给出一个基于变量“Day”被设定的值的输出。</p>	示范: 20分钟
<div>Code</div> <div><pre>speed 50 pen red, 10 for x in [1..20] fd 90 rt 100 if x is 8 pen purple, 10 else if x is 12 pen lightsalmon, 10</pre></div> <div></div>	<p>使用绘图给出另一个例子。示范条件语句如何影响绘图案的颜色。</p> <p>教学提示: 改变条件里的值来观察图案是如何变化的。</p> <p>增加另一个if语句来显示另一种颜色。(指出一个嵌套的if语句的使用。)</p> <p>添加一个‘if’语句来改变速度。这里有这个程序的拷贝。</p> <div></div> <p>现在学生能够开始写它们自己的程序了。学要求生写出示范的两个程序。学生应该在下课前 (课程时间中的15分钟) 完成这两个程序。</p> <div></div>	示范: 20分钟 学生练习: 15分钟



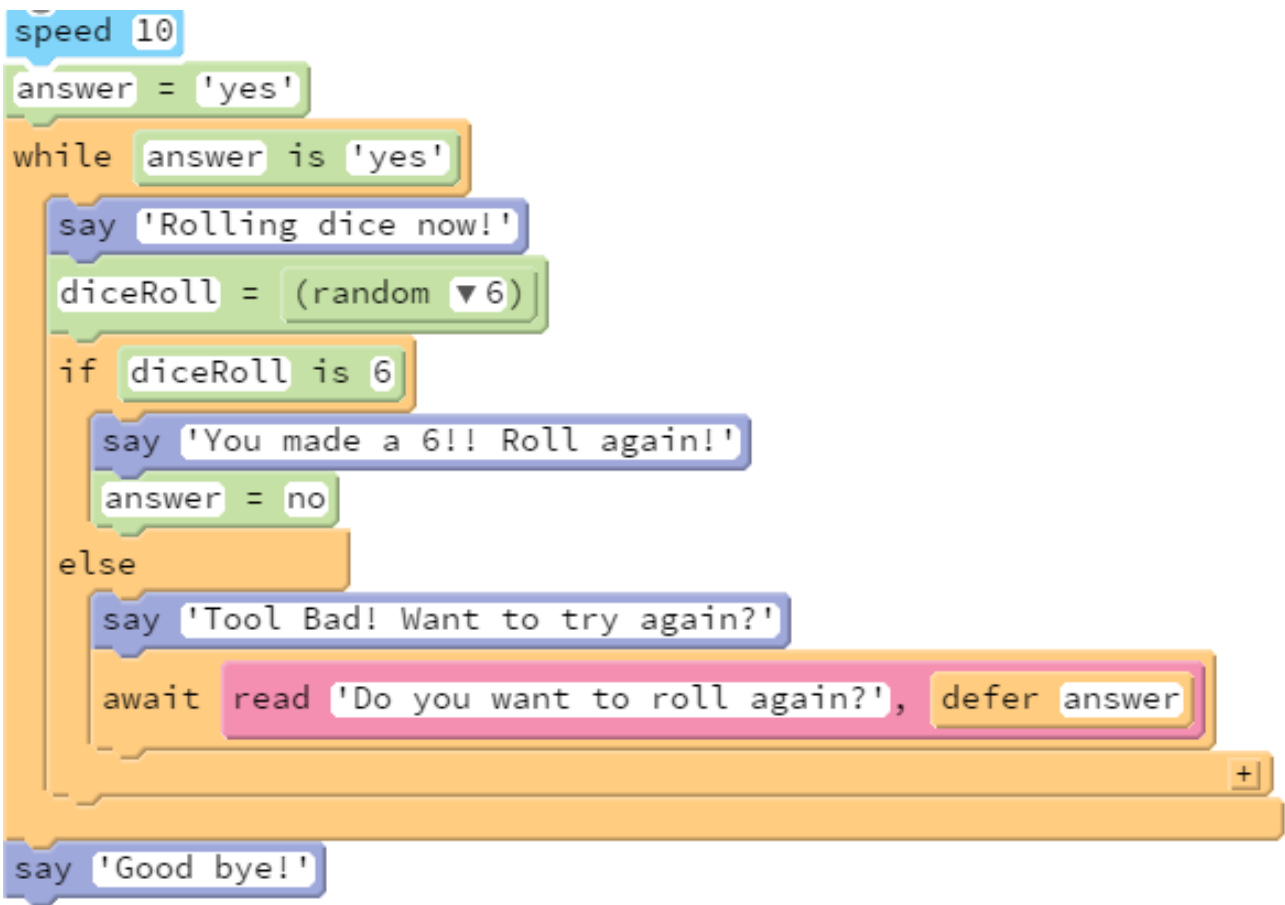
Code

```
speed 10
answer = 'yes'
hide()
while answer is 'yes'
  diceRoll = (random 6)
  label
String.fromCharCode(9856 +
diceRoll), 100
  say 'Rolling dice now!'
  if diceRoll is 6
    say 'You made a 6!! Roll
again!'
    answer = no
  else
    write diceRoll
    say 'Tool Bad! Want to
try again?'
    await read 'Roll
again?', defer answer
    say 'Good bye!'
```

添加一些有趣的元素到程序里, 以展示如何使用一个循环语句和一个条件语句增加程序的能力。

模拟掷骰子的[代码](#)。(这是一个快艇骰子游戏的初级程序。附加练习部分提供了快艇骰子游戏的规范。)

演示并遍历学生的代码。指出随机数的使用, 如何设计一个退出条件使得在最后退出循环和 If...Else块。



示范:55分钟

6.1.4 课程计划二

这节课修改在第3,4, 5章节介绍的问答机器人的循环赋值。学生将会用他们之前学到的所有概念加上条件语句来扩展这个程序。

详细内容	教学建议	时间
<div>Code</div> <pre># chatbot # CS1004 example chatbot using loop and variable prompts write 'Bob: Hi! My name is Bob.' await read 'Bob: What\'s your name?', defer name write 'Bob: Hello ' + name done = false while not done prompt = name + ' can you guess who I am?' + ':' await read prompt, defer q if (q.match /quit give up/) write 'Bob: OK, nice talking to you!' done = true else if (q.match /bot/) write 'Bob: Close... But I am a human, of course.' else write 'Bob: Good guesswork!'</pre>	<p>改进问答机器人(循环版本)程序。询问学生如何扩展它的功能。</p> <p>尽管这个程序问了四个人的名字, 它没有“记住”他们。</p> <p>通过问学生一个好的问题来使课堂更加有趣。遍历代码并强调条件语句的能力。</p> <p>这个例子同时介绍了操作x.match()功能。在block-模式, 它看起来是这样的:</p>	示范:20分钟

	<pre># chatbot # CS1004 example chatbot using loop and variable prompts write 'Bob: Hi! My name is Bob.' await read 'Bob: What\'s your name?', defer name write 'Bob: Hello ' + name done = false while not done prompt = (name + ' can you guess who I am?') + ':' await read prompt, defer q if (q.match /quit exit give up/) write 'Bob: OK, nice talking to you!' done = true else if (q.match /robot human chatbot/) write 'Bob: Close... But I am a human, of course.' else (write 'Bob: Good guesswork!')</pre>	
--	--	--

6.1.5 课程计划三

这节课计划示范复杂if语句的使用, 特别是, 组合多个布尔表达式。

详细内容	教学建议	时间
<div>Code</div> <pre>x = random [1..3] write x if x is 1 or x is 2 write 'Today is your lucky day!' else write 'I cannot see your future.' # Demonstrate complex IF Statements</pre> <div><pre>x = random [1..3] write x if x is 1 or x is 2 write 'Today is your lucky day!' else write 'I cannot see your future.'</pre><pre># Demonstrate complex IF Statements</pre></div>	<p>解释评测多布尔表达式的关键概念。解释AND/OR操作符组合两个表达式并评测它们。</p> <p>这里提供的例子一个简单程序程序, 扮演一个“精灵”并预测基于产生的随机数的一日气运。</p> <p>教学提示: 解释or操作符如何起作用。用这个只要表达式中的一个为真‘if’语句就会被执行的例子来解释。</p> <p>让学生拷贝提供的代码并测试这个程序。</p>	示范: 20分钟
<div>Code</div> <pre>x = random [1..3] write x if x is 1 or x is 2 write 'Today is your lucky day!' else if `` is `` write 'Stay low. Let everything happen tomorrow' else write 'I cannot see your future.' # Demonstrate complex IF Statements</pre>	<p>现在用左侧展示的已修改的程序来深入解释。</p> <p>教学提示: 解释如何AND操作符如何起作用。用这个*当且仅当*两个表达式都为真时‘if’语句才会执行的程序来解释。</p> <p>让学生拷贝提供的代码并测试这个程序。</p> <p>给学生10分钟的时间来试验if语句并修改代码。让他们测试其他程序员的代码并看看精灵对他们这天有什么预测!</p>	示范: 20分钟 学生练习: 10分钟

6.1.6 课程计划四

现在学生将开始与一个合作者一起设计一个猜大小游戏。附加练习部分给出了问题描述。开始练习前, 学生将会看到一个结对编程方面的短视频。下面的课程计划聚焦于如何与合作者一起设计一个程序。学生将会花费半节课的时间来设计这个程序。剩余的半节课以及另外的一节课或者作业来完成这个安排。(注意: 作业可能导致不是真的合作工作, 因为这增加了在网络上或从队伍以外的人处寻找答案的诱惑。)

详细内容	教学建议	时间
https://www.youtube.com/watch?v=vgkahOzFH2Q	让学生观看这个关于结对编程的视频。	学生练习: 5分钟
把学生分组。参考部分提供了建立合作小组的资源。		
使用“Rally Robin”合作的学习®结构, 让学生写出猜大小游戏的伪代码。他们的伪代码应当指明: <div><div>i.</div><div>使用的变量</div><div>ii.</div><div>需要的控制结构</div><div>iii.</div><div>输入/输出语句</div></div>		
学生练习: 20分钟		
Rally Robin 说明: 在一张纸上, 轮流写下如何设计这个猜大小游戏的设计说明。 每一个说明应当有一个数字标记(如: 1, 2,...) 学生1写下第一个说明。 学生2写下第二个说明。 重复这个过程直到所有的说明都被记录下来。 下一步, 退回去轮流修正这些说明, 直到连个合作者都满意。 将你的工作提交给你的老师以得到分数。 这里是一个 猜大小 程序的样例。 学生练习: 20分钟		
在你的电脑上, 用结对编程视频上的结对编程的方法与你的合作者一起输入程序。 学生练习: 40分钟		

6.1.7 课程计划五

这节互动式的课程包括学生设计一个赛车游戏。我们建议学生用CoffeeScript输入这些文本。鼓励学生在text-模式和block-模式之间切换以使他们走出舒适区并增加他们的自信。按照这个活动表里提供的乌龟赛跑轨道活动来做。

详细内容	教学建议	时间
http://activity.pencilcode.net/home/worksheet/race.html	教学建议: 让班里所有学生同时开始这个工作表。打印出这个工作表并把它给学生。这将消除复制-粘贴代码的诱惑。按照这个链接完成步骤1和2。 指导学生输入文本并保持使用text-模式。	学生练习: 10分钟
http://activity.pencilcode.net/home/worksheet/race.html	让学生在教室里口头回答问题1和2。	学生练习: 15分钟
http://activity.pencilcode.net/home/worksheet/race.html	现在让学生走出他们自己的步子作为他们回应挑战1至4。学生可以保持使用text-模式或者切换回block-模式。如果他们切换回block-模式, 鼓励他们切换到文本查看他们的代码是什么样子的。 解决代码用CoffeeScript给出。 教学提示: 如果学生输入文本, 鼓励他们缩进他们的代码来表达内容。学生应该遵循所有好的编程习惯(在Appendix A中提到)。	学生练习: 20分钟
赛车游戏活动能被学生完成。 1. http://activity.pencilcode.net/home/worksheet/race.html - 基础的游戏 2. http://activity.pencilcode.net/home/worksheet/race-car.html - 使小车看起来像一个真正的车, 这里强调了对象”的思想, 这个对象的行为受你控制 3. http://activity.pencilcode.net/home/worksheet/race-two.html - 这里介绍了“第二个对象”——两个实例——现在你可以分开控制它们 4. http://activity.pencilcode.net/home/worksheet/race-track.html - 这是一个绘图的回顾, 但是用于一种非常不同的目的——创建一个轨道形状 5. http://activity.pencilcode.net/home/worksheet/race-speed.html - 这是一个变量的说明, 用来记录每一辆车跑得多快 6. http://activity.pencilcode.net/home/worksheet/race-time.html - 这里用了另一个变量, 这个变量用来记录过去了多少时间 7. http://activity.pencilcode.net/home/worksheet/race-menu.html - 这是一个用函数来分离你打程序为子程序的例子		

6.2 资源

附加练习

快艇骰子游戏

设计一个被满意的修改版本的快艇骰子游戏, 这个游戏中你掷三个骰子并基于下面的类别来计算分数。这个游戏在每一个用户三次机会之后结束。在三次机会结束以后得到最高的分数的用户是获胜者。三个类别如下：

- i. 3个相同 —— 5分
- ii. 两个相同 —— 10分
- iii. 快艇 —— 你赢了!游戏结束!

猜大小游戏

设计一个简单的游戏, 你的电脑产生一个随机数, 用户必须猜测这个数字。电脑通过给出像“大了或者小了”的回应来给用户帮助。一旦正确的数字被给出, 游戏显示电脑产生的数字并说“游戏结束”。如果你能添加一个新特性到这个游戏, 你可以额外得到五分——创造力分。