CS 61A 控制和环境讨论1：2020年6月25日

夏季 2020

1控制

控制结构使用逻辑语句来指导程序的低端。为了
例如，条件句（如果-elif-else）允许程序跳过代码的某些部分，以及
迭代（虽然），允许程序重复一个部分。

如果语句

条件语句允许程序根据特定的条件执行不同的代码行。让我们回顾一下if-elif-else语法。

回忆以下几点：

- else和elif子句是可选的，您可以有任意数量的elif子句。
- **条件表达式是一个计算结果为真值（真值、非零整数等）的表达式。或一个错误的值(False、0、None、“”、[]等。).**
- **只执行条件表达式计算为真值的/elif下缩进的套件。**
- **如果没有一个条件表达式求值为真值，则将执行else套件。**条件语句中只能有一个子句！

如果<条件表达式>：
    <语句套装>elif<条件表达式>
:
    <语句套件>else：
    <语句套件>

布尔算子

Python还包括布尔运算符和，或，和不是。这些运算符用于组合和操作布尔值。

- 不返回以下表达式的相反真值（因此不会总是返回True或False)。
- 并按顺序计算表达式，一旦达到第一个假值就停止计算（短路），然后返回它。如果所有值都计算为真值，则返回最后一个值。
- 或在第一个真值处短路并返回它。如果所有值都计算为假值，则返回最后一个值。

>>>不是没有
真
>>>不是真
假的
>>> -1、0和1
0
>>> False或9999或1/0
9999

2控制和环境

问题

1.1阿方索只有在外面温度低于60度或下雨时才会穿一件夹克。

编写一个包含当前温度和布尔值指示的函数
如果下雨，如果阿方索穿夹克，否则穿假。

首先，尝试使用if语句来解决这个问题。

穿穿克（临时，下雨）：

```
"""
>>>穿一件夹克衫（90件，假）
假的
>>>穿一件夹克衫（40件，假）
真
>>>穿上了if（100，真）
真
"""
```

注意，我们将基于一个条件返回True或False
真实值也将是真的或假的。知道了这一点，试着写下这个
函数使用一行。

穿外套（温度，下雨）：

而循环

为了在一个程序中多次重复相同的语句，我们可以使用迭代。而<条件子句>：
在Python中，我们可以这样做的一种方法是使用一个while循环。<语句体>

只要<条件子句>的计算值为真值，<语句体就会变为>
将继续被执行。条件子句每次都会在
身体不执行。

问题

1.2评估以下代码的结果是什么？

```
def平方(x)：
    print("here!")
    返回x * x


def so_slow（num）：
    x = num
    而x > 0：
        x  =  x  +  1
    返回x / 0

正方形（so_slow (5)）
```

1.3写一个函数，如果一个正整数n是一个质数，则写一个函数返回True
否则为假。

素数n是一个除1以外的任何数字都不可整除的数字
和n本身。例如，13是素数，因为它只能被1和13整除，但是14
而不是，因为它可以被1、2、7和14整除。

提示：使用%运算符： x % y在除以y时返回x的剩余部分。

```
def is_prime (n)：
    """
    >>>是_prime（10）
    假的
    >>>是_prime(7)
    真
    """
```

4控制和环境

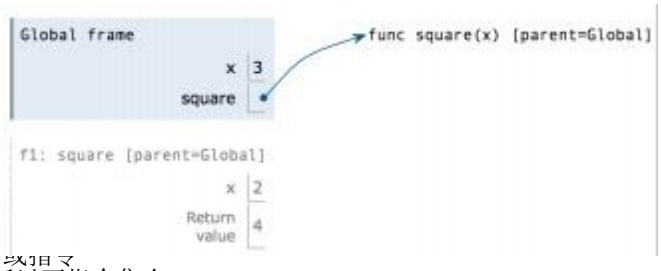
2环境图

环境图是我们用来跟踪所有变量的一个模型
那些已经被挖掘出来的价值和它们所被束缚住的价值。我们将使用这个工具
在整个课程中，了解涉及几个不同的复杂程序
分配和函数调用。

```
x  =  3

def平方(x)：
    返回x ** 2

正方形(2)
```



请记住，程序只是一组语句或指令
表示这些程序的ing图还包括以下指令集合
tions!让我们进去吧。

分配语句

赋值语句，如x = 3，程序中的定因变量。执行
一个在环境图中，记录变量名和值：

1. 评估=符号右侧的表达式
2. 在当前框架中写入变量名和表达式的值。

2.1使用这些规则为下面的分配语句绘制一个简单的图表。

```
x  =  10 %  4
y  =  x
x  **=  2
```

定义语句

def语句创建函数对象并将它们绑定到一个名称中。到图def
语句，记录函数名称，并将函数对象绑定到该名称。
写函数的父帧也很重要，这是
函数被设计。非常重要的注意：def语句的分配使用
指向函数的指针，它们的行为与原始分配是不同的。

1. 将函数对象绘制到帧的右侧，表示函数的内在名称、其参数和父帧(e。g.func广场(x)[父方= Global]。[1]
2. 在当前帧中写入函数名称，并从名称到函数对象。

2.2使用这些规则和赋值语句的规则来绘制下面的代码示图。

```
def双(x)：
    返回x * 2

def三重(x)：
    返回x * 3

帽子=双
双=三重
```

[1]导入函数时，我们仍然在环境图中创建一个函数对象
到已导入的函数的名称。但是，一个导入的函数的父级和参数
为未知，因此只包含函数的名称。例如，如果我们导入了函数添加，
函数对象将只是添加(。..)

调用表达式

调用表达式，如squre(2)，将函数应用于参数。当exe-
切割调用表达式，我们在图中创建一个新的框架来跟踪本地
可变因素

1. 计算运算符，它应该求算到一个函数。
2. 从左到右计算操作数。
3. 绘制一个新的框架，并贴上以下标签：[2]

   唯一索引（f1、f2、f3、...）

   函数的固有名称，即函数对象本身的名称。例如，如果函数对象为func正方形(x)[父=Global]，则内在名称为正方形。

   父框架（[父=全局]）
4. 将形式化参数绑定到在步骤2中获得的参数值(e。g.将x绑定到3)。
5. 在这个新帧中计算函数的主体，直到获得一个返回值。在帧中写下返回值。

如果一个函数没有返回值，则它将隐式地返回None。在这种情况下，
“返回值”框中应该包含无值。

2.3让我们把它们放在一起吧！绘制以下代码绘制环境图。

```
def双(x)：
    返回x * 2

嗯，=双
哇，=双(3)
hmmm (wow)
```

2.因为我们不知道如何内置函数像min（。..)或导入的函数，如添加(。..)
为了我们自己，我们不会画一个新的框架，当我们叫他们。

2.4绘制执行以下代码后得到的环境图。

```
def f (x):
        返回x

def g (x, y):
        如果为x (y)：
                返回而不是y
        返回y

x  =  3
x  =  g(f,  x)
f  =  g(f,  0)
```

1. **继续调用**

对于下表中的每个表达式，在计算表达式时编写交互式Python解释器显示的输出。输出可能有多行数据线。如果发生错误，请写“错误”，但包括在错误之前显示的所有输出。如果显示了一个函数值，请写下“函数”。第一行已作为一个示例提供。召回：交互式解释器显示已成功计算的表达式的值，除非为无。假设您已经启动了python3并执行了以下语句。对值的更改会在各个子问题中持续存在。

```
x  =  3

def p (rint):
        打印（rint）

def g (x, y):
        如果为x：
                print("one")
        elif x：
                打印（True，x)#x为truth-y是否会影响打印值？如果y：
                打印（真值，y）#y为真值是否会影响打印价值？其他的
                打印（False，y）#y为false-y是否会影响打印值？返回打印（p(y)）+ x
```

表达	**交互式输出**
打印（4、5) + 1	4 5 错误
2 * 2 * 1 + x * x	
打印（3 * 3 * 1）	
打印（x + 1 * x + 1）	
打印（打印（x+1*x+1））	
打印（打印（x+1*x+1）+1）	
打印（p（“rint”））	
x, y = 2, x g(y, x)	
g (y, p("rint"))	