人工智能程序设计

计算机学院

李晶晶

華南師範大學
South China Normal University

问题

- 猜数游戏设计
  - 编写一个程序模拟猜数游戏，系统在指定范围内生成一个随机数，然后提示用户进行猜测，并根据用户输入进行必要提示（猜对了，太大了，太小了），如果猜对结束程序，如果次数用完没有猜对，提示游戏结束并给出正确答案。

问题分析

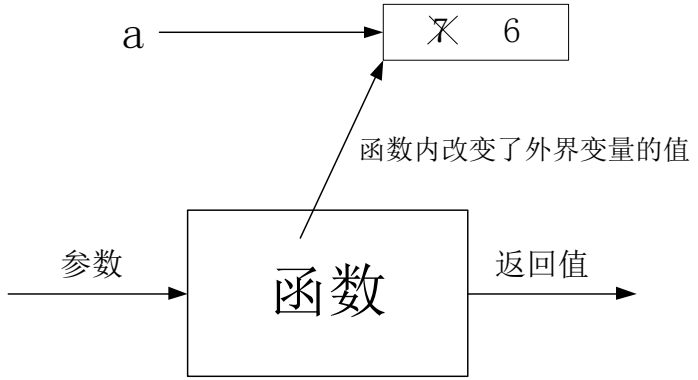
- 输入：猜测的数字（指定为1到某个最大值的整数）
- 处理：用一个函数随机生成一个给定范围内的整数数字，判断输入的数字
- 输出：提示（猜对了，太大了，太小了）或者猜测次数用完的提示
- 设计算法
  - 设计函数实现猜数字游戏

目录

- 函数概述
- 自定义函数
- 参数和返回值
- 变量作用域
- 匿名函数和高阶函数

函数概述

- 为什么需要函数？
- 函数的思想是什么？
- 什么是完美函数？



Python函数分类

- 内置函数
- 标准库函数
- 第三方库函数
- 用户自定义函数

自定义函数

def 函数名([形参列表]):
函数体

```
def my_function(parameter):
    print(parameter)
    return 'parameter is '+ parameter
```

- 不需要说明形参类型
- 不需要指定函数返回值类型
- 如果函数没有明确的返回值，Python认为返回空值None
- 即使该函数不需要接收任何参数，也必须保留一对空的英文半角圆括号
- 函数头部括号后面的冒号必不可少
- 函数体相对于def关键字必须保持一定的空格缩进

函数参数

- 位置参数
- 关键字参数
- 默认参数
- 可变参数

默认参数

- def 函数名(… , 参数名=默认值):
- 注意：在定义带有默认参数的函数时，默认参数只能出现在所有参数的最右端，并且任何一个默认参数的右侧都不能再定义非默认参数

默认参数

```
# <例4.1: 默认参数示例>
def interest(money, day = 1, interest_rate  = 0.05):
    income = 0
    income = money*interest_rate*day/365
    print(income)

interest(5000)
interest(10000)
```

可变参数

- 在定义函数时无法得知参数个数的情况，在Python中使用*args和**kwargs可以定义可变参数
- 可变参数之前可以定义0到任意多个参数
- 可变参数永远放在参数的最后面

任意数量的位置可变参数

- 在传递参数时，可以在原有的参数后面添加0个或多个参数，这些参数将会被放在元组内并传入函数。

```
# <例4.2: 任意数量的位置可变参数>
def exp(x,y,*args):
    print('x:', x)
    print('y:', y)
    print('args:', args)

exp(1, 5, 66, 55, 'abc')
```

任意数量的位置可变参数

```
# <例4.3: 任意位置参数>
def fun(*p):
    num = p.count(0)
#count(a)函数为元组内置函数，统计元组中a的个数
    if num>0:
        print("The number of 0 in the parameter is:%d" %num)
    else:
        print("There is no number 0 in the parameter.")

fun('a', 'bcd', 0, 'n') #输出结果为The number of 0 in the
parameter is:1
```

任意数量的关键字可变参数

- 参数传递时，这些参数会被放到字典内并传入函数中。
- 带两个星号前缀的参数必须在所有带默认值的参数之后

```
# <例4.4: 任意数量的关键字可变参数>
def exp(x, y, *args, **kwargs):
    print('x:', x)
    print('y:', y)
    print('args:', args)
    print('kwargs:', kwargs)

exp(1, 2, 2, 4, 6, a = 'c', b = 1)
```

函数参数

- 参数可以理解为“随机应变的容器”，不需要声明参数类型，所以编程时最好主动检查所传参数类型的正确性

函数返回值

- return函数返回的运行结果可以保存为一个对象供其他函数调用
- 返回值的数据类型没有限制，个数可以是一个或者一组

```
# <例4.5: 返回函数值>
def interest(money, day = 1, interest_rate  = 0.05):
    income = 0
    income = money*interest_rate*day/365
    return income

x = interest(1000)
print(x)
y = interest(5000)
print(y)
```

函数返回值

```
#<例4.6：参数与返回值举例>
#检验一个给定的序列中指定位置的字符是否是要查找的字符
def find(str,pos,key):
    if(len(str)<pos):          #指定的位置大于字符串的长度
        return False,-99
    else:
        if(str[pos]==key):
            return True,True
        else:
            return True,False
mystr = "abcdefgh"
correct,res = find(mystr, 2, 'c')
if(not correct): print("Error:position exceeds length of string")
elif res: print("Find it!")
else: print("Not in it!")
```

调用函数

- 位置参数调用
  - 严格函数定义时的位置传入，顺序不可以调换，否则会影响输出结果或者直接报错
- 关键字参数调用
  - 可以不严格按照位置
  - 关键字参数可以和位置参数混用，但必须在位置参数后面
- 可变参数调用
  - *arg可变参数列表直接将元组或者列表转换为参数
  - **kwargs关键字参数列表直接将字典转换为关键字参数

局部变量和全局变量

- 局部变量是只能在特定的函数中可以访问的变量
- 全局变量是定义在所有函数最外面的变量

```
def sum(*arg):
    sum1 = 0
    for i in range(len(arg)):
        sum1 += arg[i]
    return sum1
```

```
sum0 = 10
def fun():
    sum_global = sum0 + 100
    return sum_global
```

局部变量和全局变量

- 分辨局部与全局变量的规则
  - 假设有一个变量为a，它出现在函数f()里面，应怎样判断它在函数内是什么变量。可以使用如下规则来判定：
    - ✓如果有global关键字修饰变量a，则a为全局变量。
    - ✓否则，假如a是参数或者出现在等号左边，则a是局部变量。
    - ✓否则，a与函数f外层的变量a的属性相同。

局部变量和全局变量

```
#<例4.7：局部变量与全局变量举例1>
a = 1  #所有函数最外面的变量，全局变量
def fun(x,y):
    global a    #global关键字表明a是全局变量
    a = x+y
    return a
sum = fun(10, 100)
print(a)
```

局部变量和全局变量

```
#<例4.8：局部变量与全局变量举例2>
a = 1              #所有函数最外面的变量，全局变量
def fun(x,y):
    a = x+y
#a在函数内的等号左侧，局部变量，不改变全局变
#量的值。
    return a
sum = fun(10, 100)
print(a)        #打印的是函数外的变量a
```

局部变量和全局变量

```
#<例4.9：a, b, c, d是否为局部变量？>
b, c = 2, 4  #在所有函数最外层，即全局变量。
def g_func():
    a = b*c         #a是局部变量
    d = a           #d是局部变量，b和c都是全局变量。
    print(a, d,';', end="")

g_func()
print(b, c)
```

局部变量和全局变量

```
#例4.10：复杂运算例子>
def do_sub(a, b):
    c=a-b              #a, b, c都是do_sub()中的局部变量
    print(c)
    return c
def do_add(a, b):      #参数a和b是do_add()中的局部变量
    global c
    c=a+b              #全局变量c，修改了c的值
    c=do_sub(c, 1)     #再次修改了全局变量c的值
    print(c)
#所有函数外先执行:
a=3                    #全局变量a
b=2                    #全局变量b
c=1                    #全局变量c
do_add(a, b)           #全局变量a和b作为参数传递给do_add()
print(c)               #全局变量c
```

嵌套函数

- 嵌套函数是指在函数内定义的函数
- 定义在其他函数内部的函数称为内建函数，包含有内建函数的函数称为外部函数
- 嵌套函数如同局部变量般是个“局部”函数，它只能在外层定义它的函数中使用

嵌套函数

```
#<程序：嵌套函数举例>
def selection_sort(L):
    def find_min(L): #返回L中最小值所在的索引
        min =0       #这个min是find_min中的局部变量
        for i in range(len(L)):
            if L[i]<L[min]:  min = i
        return min
        for i in range(len(L)-1):
            min = find_min(L[i:])    #min是selection_sort中的局部变量
            L[min+i],L[i] = L[i],L[min+i]
        return L
```

嵌套函数下的局部和全局变量

- 假设有一个变量为a，它出现在函数f里面，应该怎样判断它在函数f内是什么变量。我们可以定义如下规则。
  - 如果有global关键字修饰变量a，那么不管函数f()是不是嵌套函数， a都为全局变量。
  - 否则，假如a是参数或者出现在“=”左边，则a是局部变量。
  - 否则，a应继承上层函数中a的属性。如果函数f不是嵌套函数，那么a为全局变量。如果f是嵌套函数，a就是上层的a。

嵌套函数下的局部和全局变量

```
#<例4.11：局部变量与全局变量举例3>
a = 1   #全局变量
def F3():
    def F():
        global a  #a是最外层的全局变量
        print("In F3's F, a =", a)
    a = 3
    F()
F3()
```

嵌套函数下的局部和全局变量

```
#<例4.12：局部变量与全局变量举例4>
a=1
def F4():
    global a
    def F():
        a=2     #a是F的局部变量
        print("In F4's F,a=", a)
    F()
    print("In F4, a =", a) #a是全局变量
F4()
```

嵌套函数下的局部和全局变量

```
#<例4.13：局部变量与全局变量举例5>
a=1
def F5():
    def F():  #a 不是F的局部变量，那就继承上层函数中a的属性
        print(“In F5's F, a =", a)
    a=3
    F()
F5()
```

上机实践

- 方差计算公式的一个推导公式为平方的均值减去均值的平方，如下所示：

$$S^2 = \frac{x_1^2+x_2^2+x_3^2+\cdots+x_n^2}{n} - M^2$$

请构建求方差函数var，请在求平方函数中内建求平方和函数sums及求均值函数mean，并在求均值函数中在内建一个求和函数sum。

匿名函数

- 使用lambda语句创建匿名函数，即函数没有具体的名称
- Lamda表达式的基本格式为

  lambda arg1,arg2,arg3...: <expression>

  冒号前是函数参数，冒号后是返回值

匿名函数

- lambda定义的是单行函数，如果需要复杂的函数，应使用def语句
- lambda语句可以包含多个参数
- lambda语句有且只有一个返回值
- lambda语句中的表达式不能含有命令，且仅限一条表达式

匿名函数

- lambda语句作为对象赋值给变量，然后使用变量名调用

```
#<例4.14: lambda语句创建对数函数>
from math import log   #引入python数学库的对数函数
#此函数用于返回一个以base为底的匿名对数函数
def make_log_function(base):
    return lambda x: log(x, base)
#创建一个以3为底的匿名对数函数，并赋值
my_log = make_log_function(3)
#调用匿名函数my_log， 底数已经设置为3， 只需设置真数
#如果用log函数，则需要同时设置真数和底数
print(my_log(9))
```

高阶函数

- map函数

  map(func, list) func是一个函数，list是一个序列对象

  - 执行时，序列对象中的每个元素，按照从左到右的顺序通过把函数func依次作用在list的每个元素上，得到一个新的list返回

高阶函数

- map函数

```
#<例4.15:map函数>
def add(x):
    x += 3
    return x

numbers = list(range(10))
num1 = list(map(add, numbers))
num2 = list(map(lambda x: x+3, numbers))
print(numbers)
print(num1)
print(num2)
```

高阶函数

- filter函数

  filter(func, list) func是一个函数，list是一个序列对象

  - func的作用是对每个元素进行判断，通过返回True或False来过滤掉不符合条件的元素，符合条件的元素组成的新list

程序实现

- 演示代码
  - 输入：猜测的数字（指定为1到某个最大值的整数）
  - 处理：
    - ✓用函数实现随机生成一个给定范围内的整数数字以及判断输入的数字
  - 输出：提示（猜对了，太大了，太小了）或者猜测次数用完的提示或者输入的数字错误


```
#程序：猜数字游戏>
from random import randint

def guessNumber(maxValue = 10, maxTime = 3):
    #随机生成一个整数
    value = randint(1, maxValue)
    for i in range(maxTime):
        if i == 0:
            print("开始猜数字")
        else:
            print("剩余次数为", maxTime-i, "请再猜一次")

        print("输入一个整数，范围在1到", maxValue)
        x = eval(input("请输入："))

        while (not isinstance(x, int)) or x < 1 or x > maxValue:
            print("必须输入一个整数，范围在1到", maxValue)
            x = eval(input("请输入："))
            if x == value:
                print("猜对了，恭喜！")
                break
            elif x > value:
                print("太大了")
            else:
                print("太小了")
    else:
        #次数用完了，还没有猜对，游戏结束，提示正确答案
        print('游戏结束，你输了！')
        print("正确的值为", value)

guessNumber()
```

小结

- 自定义函数
  - 参数与返回值
- 全局变量和局部变量
  - 嵌套函数
- 匿名函数
- 高阶函数