Python语言基础与应用-02

陈斌 gischen@pku.edu.cn

目录

- 控制流和程序结构
- 基本模块介绍

  （时间、算术、持久化、文件、数据库、GUI、海龟）

运算语句：表达式、函数调用和赋值

- 各种类型的数据对象，可以通过各种运算组织成复杂的表达式
- 调用函数或者对象，也可以返回数据，所有可调用的事物称为callable

  调用函数或者对象，需要在其名称后加圆括号，如果有参数，写在圆括号里

  不加圆括号的函数或者对象名称仅是表示自己，不是调用
- 将表达式或者调用返回值传递给变量进行引用，称为赋值

```
>>> 12 * 34.5 + 23.4
437.4
>>> ('abc' + '123') * 3
'abc123abc123abc123'
>>>
>>> import math
>>> math.sqrt(12)
3.4641016151377544
>>> math.sqrt
<built-in function sqrt>
>>>
>>> n = 12 * 34
>>> n
408
>>> p2 = math.sqrt(2)
>>> p2
1.4142135623730951
>>> pfg = math.sqrt
>>> pfg
<built-in function sqrt>
>>> pfg(2)
1.4142135623730951
```

赋值语句的小技巧

- 级联赋值语句

```
x = y = z = 1
```

- 多个变量分解赋值

```
a, b = ['hello', 'world']
```

- 变量交换

```
a, b = b, a
```

- 自操作

```
i += 1
n *= 45
```

```
>>> x = y = z = 1
>>> x,y,z
(1, 1, 1)
>>> a, b= ['hello', 'world']
>>> a
'hello'
>>> b
'world'
>>>
>>> a, b = b, a
>>> a
'world'
>>> b
'hello'
>>>
>>> a += 'cup'
>>> a
'worldcup'
```

控制流语句：条件if

- 条件语句

```
if <逻辑条件>:
    <语句块>
elif <逻辑条件>: #可以多个elif
    <语句块>
else: #仅1个
    <语句块>
```

- 各种类型中某些值会自动被转换为False，其它值则是True：

```
None, 0, 0.0, '',
[], (), {}, set()
```

```
>>> a = 12
>>> if a > 10:
        print ("Great!")
elif a > 6:
        print ("Middle!")
else:
        print ("Low!")


Great!
```

控制流语句：while循环

- 条件循环while

```
while <逻辑条件>:
    <语句块>
    break #跳出循环
    continue #略过余下循环语句
    <语句块>
else: #条件不满足退出循环，则执行
    <语句块>
```

- else中可以判断循环是否遭遇了break

```
>>> n = 5
>>> while n > 0:
        n = n - 1
        if n < 2:
            break
        print (n)

4
3
2
```

```
>>> n = 5
>>> while n > 0:
        n = n - 1
        if n < 2:
            continue
        print (n)
else:
        print ('END!')

4
3
2
END!
```

控制流语句：for循环

- **迭代循环for：**

```
for <变量> in <可迭代对象>:
    <语句块>
    break #跳出循环
    continue #略过余下循环语句
else: #迭代完毕，则执行
    <语句块>
```

- **可迭代对象有很多类型**

象字符串、列表、元组、字典、集合等

也可以有后面提到的生成器、迭代器等

```
>>> for n in range(5):
        print (n)


0
1
2
3
4
```

```
>>> alist = ['a', 123, True]
>>> for v in alist:
        print (v)


a
123
True
```

```
>>> adic = {'name':'Tom', 'age':18, 'gender':'Male'}
>>> for k in adic:
        print (k, adic[k])


name Tom
age 18
gender Male
```

```
>>> for k, v in adic.items():
        print (k, v)


name Tom
age 18
gender Male
```

迭代的小技巧：zip()函数

- 由于for循环的迭代是对容器中的数据项进行枚举，但不带序号或下标
- 有时候我们需要数据项的序号
- 另外，有时我们需要并行迭代两组一一对应的数据项
- 使用zip()打包函数可以使代码更加简洁
- zip(list1，list2，list3...)返回由每个列表对应位置的数据项构成元组的列表。

```
weekdays = ['Sun', 'Mon', 'Tue', 'Wed', \
            'Thu', 'Fri', 'Sat']

for n, s in zip(range(7), weekdays):
    print (n, s)

for n in range(7):
    print (n, weekdays[n])

n = 0
while n < 7:
    print (n, weekdays[n])
    n = n + 1

drinks = ['coffee', 'tea', 'beer', \
          'water', 'milk', 'coca-cola', 'none']

for w, d in zip(weekdays, drinks):
    print ('drink', d, 'in', w)
```

```
0 Sun
1 Mon
2 Tue
3 Wed
4 Thu
5 Fri
6 Sat
```

```
drink coffee in Sun
drink tea in Mon
drink beer in Tue
drink water in Wed
drink milk in Thu
drink coca-cola in Fri
drink none in Sat
```

上机练习

① 给定n，计算1+2!+3!+...+n!的值

② 给定y和m，计算y年m月有几天？

注意闰年定义

③ 给定字符串s和数字n，打印把字符串s向右移动n位的新字符串

例如abcd和1，返回dabc

例如mnbol和2，返回olmnb

④ 给定一个英文数字字符串，打印相应阿拉伯数字字符串

例如：one-four-five-nine

返回：1459

函数function

- 函数用来对具有明确功能的代码段命名，以便复用（reuse）
- 定义函数：**def语句**；

```
def <函数名> (<参数表>):
    <缩进的代码段>
    return <函数返回值>
```

- 调用函数：<函数名>（<参数>）

注意括号！

无返回值：<函数名>(<参数表>)

返回值赋值：v = <函数名>(<参数表>)

```
1   def sum_list(alist):   # 定义一个带参数的函数
2       sum_temp = 0
3       for i in alist:
4           sum_temp += i
5       return sum_temp   # 函数返回值
6
7
8   print(sum_list)   # 查看函数对象sum_list
9
10  my_list = [23, 45, 67, 89, 100]
11  # 调用函数，将返回值赋值给my_sum
12  my_sum = sum_list(my_list)
13  print("sum of my list:%d" % (my_sum,))
```

```
<function sum_list at 0x10067a620>
sum of my list:324
```

定义函数的参数：固定参数 / 可变参数

- 定义函数时，参数可以有两种；
- 一种是在参数表中写明参数名key的参数，固定了顺序和数量

```
def func(key1, key2, key3…):
def func(key1, key2=value2…):
```

- 一种是定义时还不知道会有多少参数传入的可变参数

```
def func(*args): #不带key的多个参数
def func(**kwargs): #key=val形式的多个参数
```

```
16  def func_test(key1, key2, key3=23):
17      print("k1=%s,k2=%s,k3=%s" % (key1, key2, key3))
18
19
20  print("====func_test")
21  # 没有传入key3，用了缺省值
22  func_test('v1', 'v2')
23  # 传入了key3
24  func_test('ab', 'cd', 768)
25  # 使用参数名称就可以不管顺序
26  func_test(key2='KK', key1='K')
```

```
====func_test
k1=v1,k2=v2,k3=23
k1=ab,k2=cd,k3=768
k1=K,k2=KK,k3=23
```

定义函数的参数：固定参数 / 可变参数


```
29  # 可以随意传入0个或多个无名参数
30  def func_test2(*args):
31      for arg, i in zip(args, range(len(args))):
32          print("arg%d=%s" % (i, arg))
33
34
35  print("====func_test2")
36  func_test2(12, 34, 'abcd', True)
```

```
====func_test2
arg0=12
arg1=34
arg2=abcd
arg3=True
```

```
39  # 可以随意传入0个或多个带名参数
40  def func_test3(**kwargs):
41      for key, val in kwargs.items():
42          print("%s=%s" % (key, val))
43
44
45  print("====func_test3")
46  func_test3(myname="Tom", sep="comma", age=23)
```

```
====func_test3
sep=comma
age=23
myname=Tom
```

调用函数的参数：位置参数 / 关键字参数

- 调用函数的时候，可以传进两种参数；
- 一种是没有名字的位置参数

  func(arg1, arg2, arg3…)

  会按照前后顺序对应到函数参数传入
- 一种是带key的关键字参数

  func(key1=arg1, key2=arg2…)

  由于指定了key，可以不按照顺序对应
- 如果混用，所有位置参数必须在前，关键字参数必须在后

```
16  def func_test(key1, key2, key3=23):
17      print("k1=%s,k2=%s,k3=%s" % (key1, key2, key3))
18
19
20  print("====func_test")
21  # 没有传入key3，用了缺省值
22  func_test('v1', 'v2')
23  # 传入了key3
24  func_test('ab', 'cd', 768)
25  # 使用参数名称就可以不管顺序
26  func_test(key2='KK', key1='K')
```

```
====func_test
k1=v1,k2=v2,k3=23
k1=ab,k2=cd,k3=768
k1=K,k2=KK,k3=23
```

函数小技巧：map()函数

- 有时候，需要对列表中每个元素做一个相同的处理，得到新列表
- 例如所有数据乘以3
- 例如所有字符串转换为整数
- 例如两个列表对应值相加
- map(func, list1, list2....)
- 函数func有几个参数，后面跟几个列表

```
num = [10, 20, 40, 80, 160]
lst = [2, 4, 6, 8, 10]
def mul3(a):
    return a * 3

print (list( map(mul3, num) ))

def atob(a, b):
    return a + 1.0/b

print (list( map(atob, num, lst) ))
```

```
[30, 60, 120, 240, 480]
[10.5, 20.25, 40.166666666666664, 80.125, 160.1]
```

函数小技巧：匿名函数lambda

- 有时候，函数只用一次，其名称也就不重要，可以无需费神去def一个
- Lambda表达式可以返回一个匿名函数
- lambda <参数表>:<表达式>

```
num = [10, 20, 40, 80, 160]
lst = [2, 4, 6, 8, 10]
def mul3(a):
    return a * 3

print (list( map(mul3, num) ))

def atob(a, b):
    return a + 1.0/b

print (list( map(atob, num, lst) ))

print (list( map(lambda a:a * 3, num)))
print (list( map(lambda a,b:a+1.0/b, num, lst)))
```

上机练习

- 水仙花数判定：创建一个函数，接受一个参数n(n>=100)，判断这个数是否为水仙花数

  即满足如果这个数为m位数，则每个位上的数字的m次幂之和等于它本身,例如1^3 + 5^3 + 3^3 = 153, 1^4+6^4+3^4+4^4=1634），返回True或者False。

- 创建一个函数，接受一个参数max(max>=1000)，调用上题编写的判断函数，求100到max之间的水仙花数。

- 创建一个函数，接受两个字符串作为参数，返回两个字符串字符集合的并集。

  如接受的两个字符串为"abc"和"bcd"，返回set(['a', 'b', 'c' , 'd'])。

Python引用扩展模块：import

- **import <模块> [as <别名>]**
  将模块中的函数等名称导入当前程序
  “命名空间”namespace
  引用方法：<模块>.<名称>

- **dir(<名称>)函数**
  列出名称的属性

- **help(<名称>)函数**
  显示参考手册

- **from <模块> import <名称>**
  导入模块的部分名称

```
>>> import time
>>> dir(time)
['_STRUCT_TM_ITEMS', '__doc__', '__loader__', '__name__
', '__package__', '__spec__', 'altzone', 'asctime', 'cl
ock', 'ctime', 'daylight', 'get_clock_info', 'gmtime',
'localtime', 'mktime', 'monotonic', 'perf_counter', 'pr
ocess_time', 'sleep', 'strftime', 'strptime', 'struct_t
ime', 'time', 'timezone', 'tzname', 'tzset']
>>> time.tzname
('CST', 'CST')
>>> help(time.time)
Help on built-in function time in module time:

time(...)
    time() -> floating point number

    Return the current time in seconds since the Epoch.
    Fractions of a second may be present if the system
clock provides them.

>>> print(time.time())
1490280256.450634
```

时间相关：calendar模块

- 跟日历相关的若干函数和类，可以生成文本形式的日历
- calendar.calendar(<年>)
- calendar.month(<年>,<月>)
  返回多行字符串
- calendar.isleap(<年>)
  判别闰年
- calendar.prmonth(<年>,<月>)
- calendar.prcal(<年>)

```
>>> import calendar
>>> calendar.month(2017,3)
'     March 2017\nMo Tu We Th Fr Sa Su\n       1  2  3  4  5\n 6  7  8  9 10 11 12\n1
3 14 15 16 17 18 19\n20 21 22 23 24 25 26\n2
7 28 29 30 31\n'
>>> print (calendar.month(2017,3))
     March 2017
Mo Tu We Th Fr Sa Su
        1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

>>> calendar.isleap(2017)
False
>>>
```

时间相关：datetime模块

- 有4个主要的类
  - date处理年月日
  - time处理时分秒、毫秒
  - datetime处理日期加时间
  - timedelta处理时段（时间间隔）
- 常用函数 / 方法
  - datetime.date.today()
  - datetime.datetime.now()
  - datetime.datetime.isoformat()
- 两个时间相减就是timedelta

2天前

```
>>> import datetime
>>> datetime.date.today()
datetime.date(2017, 3, 23)
>>> d=datetime.datetime.now()
>>> d
datetime.datetime(2017, 3, 23, 23, 38, 35, 209753)
>>> d.isoformat()
'2017-03-23T23:38:35.209753'
>>> d-datetime.timedelta(days=2)
datetime.datetime(2017, 3, 21, 23, 38, 35, 209753)
```

时间相关：time模块

- time.time()浮点数表示的现在时间
  从1970-1-1 0:0:0开始的秒数
- time.struct_time结构化时间类
  time.localtime(<纪元时间>)->结构
  time.gmtime(<纪元时间>)->结构
  time.mktime(<结构化时间>)->纪元时间
- time.strftime(<格式>)表示格式化输出（结构化）时间
- time.strptime(<字串>,<格式>)按照格式识别字串，返回时间

```
>>> import time
>>> n = time.time()
>>> n
1490285666.071055
>>> time.localtime(n)
time.struct_time(tm_year=2017, tm_mon=3, tm
_mday=24, tm_hour=0, tm_min=14, tm_sec=26,
tm_wday=4, tm_yday=83, tm_isdst=0)
>>> time.gmtime(n)
time.struct_time(tm_year=2017, tm_mon=3, tm
_mday=23, tm_hour=16, tm_min=14, tm_sec=26,
tm_wday=3, tm_yday=82, tm_isdst=0)
>>> p=time.localtime(n)
>>> time.mktime(p)
1490285666.0
>>> p=time.gmtime(n)
>>> time.mktime(p)
1490256866.0
>>> time.strftime("%Y%m%d %H%M%S", p)
'20170323 161426'
>>> time.strptime("20170324", "%Y%m%d")
time.struct_time(tm_year=2017, tm_mon=3, tm
_mday=24, tm_hour=0, tm_min=0, tm_sec=0, tm
_wday=4, tm_yday=83, tm_isdst=-1)
```

基本模块简介：算术

- math：常用的算术函数、三角函数、幂指数等等
- cmath：支持复数的math函数
- decimal：十进制定点数
- fractions：有理数，比例
- random：随机数

  `random.randint(a,b)`

  `random.randrange(start,stop,step)`

- statistics：一些统计函数

```
>>> import random
>>> random.choice(["apple", "pie", "tea"])
'tea'
>>> random.randrange(10)
4
>>> random.randint(1,10)
5
```

```
>>> import statistics
>>> statistics.mean([12,34,56])
34.0
>>> statistics.median([12,34,56,100])
45.0
>>> statistics.median([12,34,56,100,101])
56
```

持久化：shelve

- 将任何数据对象，保存到文件中去
- 类似字典形式访问，可读可写
- shelve.open(<文件名>)
- f[key]=value
- value=f[key]
- del f[key]
- f.close()

无作用

```
import shelve

d = shelve.open(filename)  # open -- fi
                           # library

d[key] = data              # store data
                           # using an e
data = d[key]              # retrieve a
                           # if no such
del d[key]                 # delete dat
                           # if no such

flag = key in d            # true if th
klist = list(d.keys())     # a list of

# as d was opened WITHOUT writeback=Tru
d['xx'] = [0, 1, 2]        # this works
d['xx'].append(3)          # *this does

# having opened d without writeback=Tru
temp = d['xx']             # extracts t
temp.append(5)             # mutates th
d['xx'] = temp             # stores the

# or, d=shelve.open(filename,writeback=
# d['xx'].append(5) and have it work as
# consume more memory and make the d.cl

d.close()                  # close it
```

文本文件读写：内置文件对象

- open(<文件名>, <模式>)
- f.close()
- f.readline()：返回一行
- f.readlines()：返回所有行，列表
- f.writelines(<字符串列表>)：写入文本行

```
>>> f=open("my.txt", "w")
>>> f.writelines(["apple\n", "pie\n"])
>>> f.close()
>>> f=open("my.txt", "r")
>>> f.readlines()
['apple\n', 'pie\n']
>>> f.close()
```

上机练习

- 给算法计时，看看阶乘累加（n=1~100）各需要多长时间？
- 将一篇文章写入一个文本文件。
- 读出文本文件，统计单词数输出。
- 读出文本文件，随机输出其中的10个单词。

图形用户界面：easygui

- 可以显示各种对话框、文本框、选择框与用户交互

  easygui.egdemo()演示

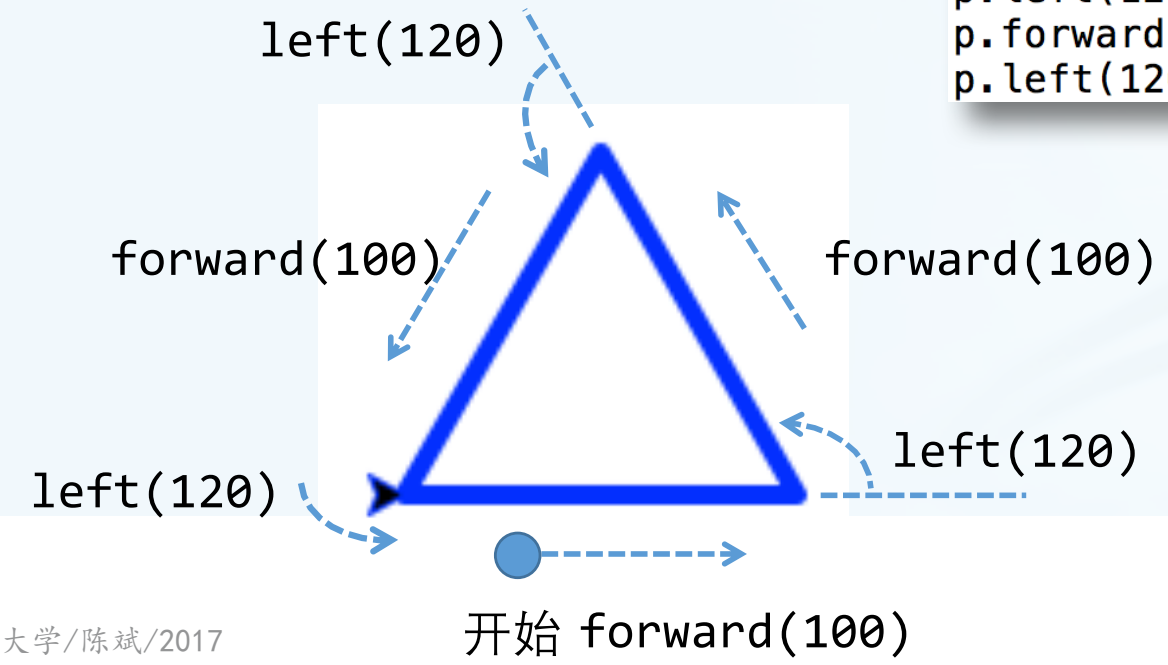
- easygui.msgbox
- easygui.fileopenbox
- easygui.choicebox
- easygui.textbox
- easygui.passwordbox
- 可以做出简单的图形界面程序

```
import easygui as g
import sys
while 1:
    g.msgbox('嗨，欢迎进入第一个GUI制作的小游戏~')
    msg = '你希望学习到什么知识呢？'
    title = '互动小游戏'
    choices = ['琴棋书画', '四书五经', '程序编写', '逆向分析']
    choice = g.choicebox(msg, title, choices)
    # note that we convert the choice to string, in case the user
    # cancelled the choice, and we got None.
    g.msgbox('你的选择是：' + str(choice), '结果')
    msg = '你希望重新开始小游戏么？'
    title = '请选择'
    if g.ccbox(msg, title):     # Show a Continue/Cancel dialog
        pass                    # user choose Continue
    else:
        sys.exit(0)             # user choose Cancel
```

海龟做图：turtle

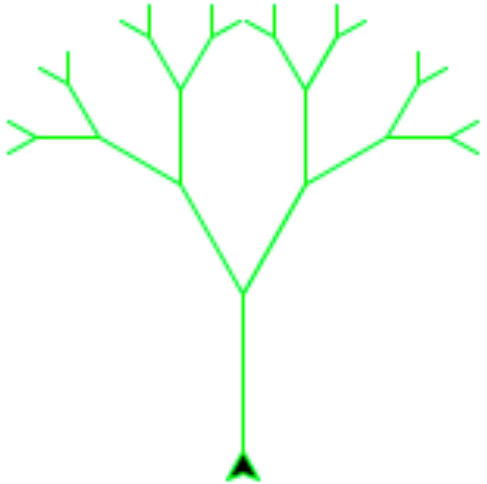
- 模拟海龟在沙滩上爬行所描绘的轨迹，从LOGO语言借鉴而来
- 海龟：显隐、外形、是否动画
- 画笔：抬起落下、颜色、粗细

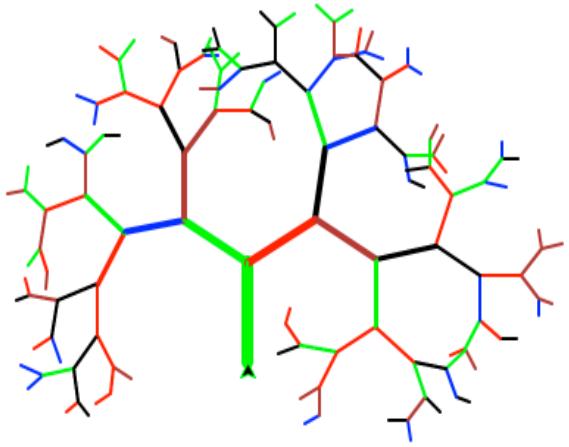
```
import turtle  # 导入turtle模块
p = turtle.Pen()  # 创建一支画笔（海龟）
p.pencolor('blue')  # 设置画笔颜色为黑色
p.pensize(5)  # 设置画笔的粗细为5
p.forward(100)  # 最初画笔（海龟）朝向正右方，向前画长度为100的直线
p.left(120)  # 画笔（海龟）向左转120度
p.forward(100)  # 向前画长度为100的直线
p.left(120)  # 画笔（海龟）向左转120度
p.forward(100)  # 向前画长度为100的直线
p.left(120)  # 画笔（海龟）向左转120度
```



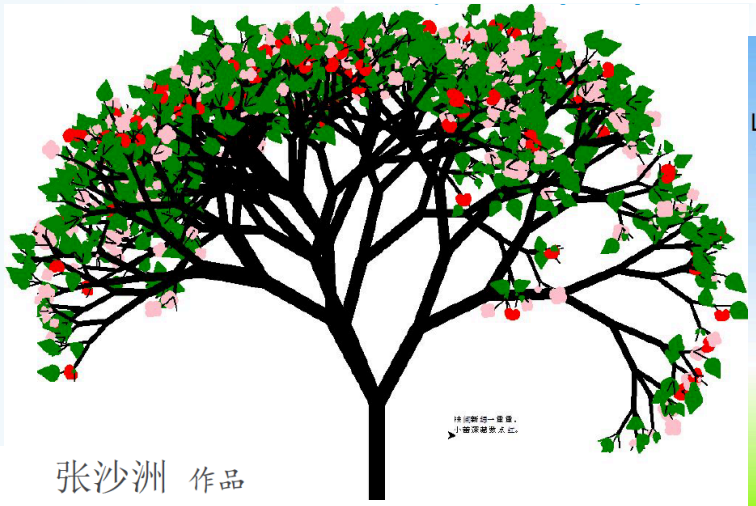
海龟做图：turtle

- 无需任何繁琐代码，立即开始绘图
- 海龟的外形可以设定，多个海龟，能够实现动画制作
- 极大激发学生的兴趣

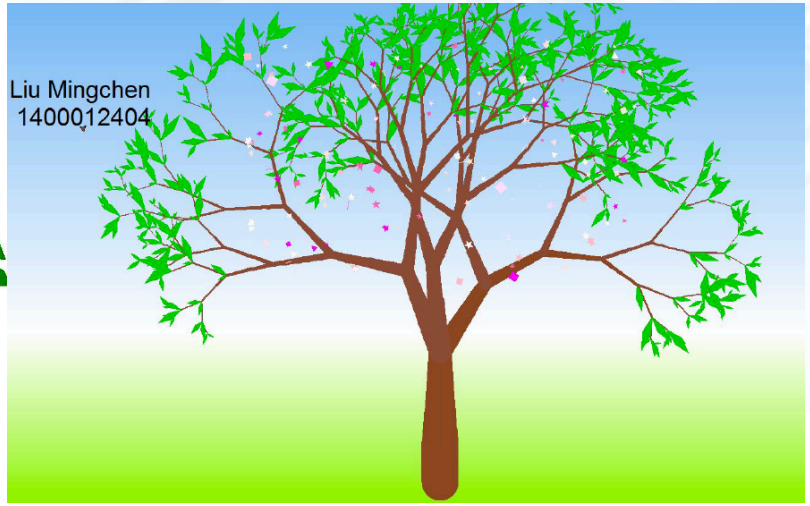




李然 作品



张沙洲 作品



上机练习

- 利用turtle模块，结合easygui，让用户选择一个图形进行绘制
- 选择“正方形”，绘制一个边长100的绿色正方形
- 选择“五角星”，绘制一个边长100的红色五角星。