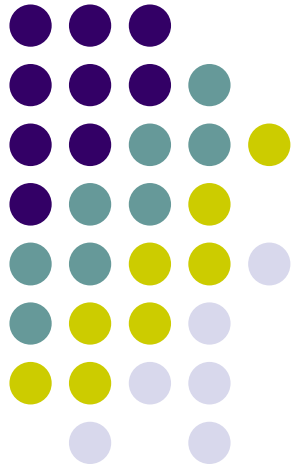
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

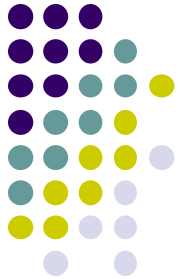
第5章图形编程
Graphic Programming



申丽萍

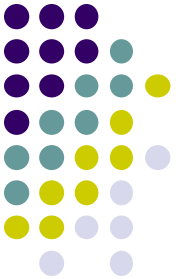
lpshen@sjtu.edu.cn

第5章 图形编程



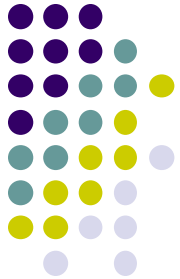
- 计算可视化
- 面向对象基本概念
  - 面向对象思想与编程在第7章介绍
- 图形编程库
- graphics图形编程
  - 本章介绍基本的绘图功能
  - 图形用户界面编程在第8章介绍
- 编程案例

计算可视化



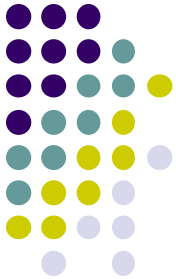
- 可视化:将抽象事物和过程转变成视觉可见的、形象直观的图形图像表示
  - 柱状图,直方图,散点图,网络图,流程图,树,地图,图像,动画,...
- 科学可视化:将科学与工程计算、实验中的数据用直观的计算机图形图像呈现出来,以便人们理解数据、增强对事物现象的认识和对内在规律的洞察.
- 数据可视化:将海量数据转化为数据图像,以帮助人们直观地观察数据. 可以进而发展到更高层次的信息可视化和知识可视化.
- 工程设计可视化
- ...

图形是复杂数据



- 包含的信息是复杂的
  - 圆形:圆心(元组)和半径(数值);内部及边界的颜色(RGB元组);...
- 图形操作是复杂的
  - 求面积,周长等传统计算;还有移动位置,改颜色等等
- 用对象表示复杂数据
  - 包含数据表示
  - 包含对数据的操作
- 图像编程提供友好的Graphical User Interface (GUI)
  - 第8章介绍

第5章 图形编程



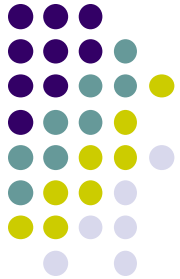
- 计算可视化
- 面向对象基本概念
  - 面向对象思想与编程在第7章介绍
- 图形编程库
- graphics图形编程
  - 本章介绍基本的绘图功能
  - 图形用户界面编程在第8章介绍
- 编程案例

数据与操作:传统观点

- 数据类型
  - 某种值的集合
  - 运算(操作)的集合
- 计算就是对数据进行操作
  - 数据与操作分离
  - 数据是被动的,操作是主动的
- 例如:string类型
  - 值是'abc'等
  - 对串的操作有+,*,len()等

```
>>> list1=[2,5,1,9,6,8]
>>> list2=sorted(list1)
>>> list2
[1, 2, 5, 6, 8, 9]
>>> list1
[2, 5, 1, 9, 6, 8]
```

数据与操作:面向对象观点



- 对象(*Object*):集数据与操作于一身.
  - 对象知道一些信息
  - 对象能对那些信息进行处理
- 计算:向对象发出请求操作的消息.
  - 消息即请求执行对象的操作
- 面向对象(*Object-Oriented*):软件系统由各种对象组成,对象之间通过消息进行交互.
- 现代软件系统几乎都是OO设计和实现.

```
>>> list1
[2, 5, 1, 9, 6, 8]
>>> list1.sort()
>>> list1
[1, 2, 5, 6, 8, 9]
```

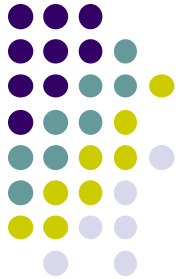
OO基本概念

- 类(*class*):描述同类对象的共性
  - 包含的数据
    - 任何类型的数据,甚至可以是对其他对象的引用.
  - 能执行的操作:方法(*method*)
- 对象(*object*):类的实例(*instance*)
  - 同类的不同对象可有不同的数据值(实例变量),但能执行的操作是一样的
- 创建对象:使用类的构造器(*constructor*).

  <类名>(<参量1>,<参量2>,…)
- 消息:请求对象执行它的方法.

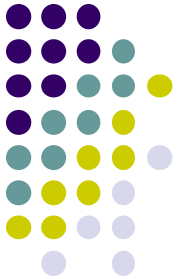
  <对象>.<方法名>(<参量1>,<参量2>,…)

对象的方法



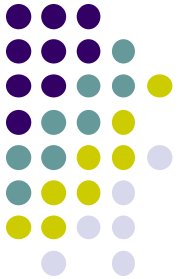
- 不同对象当然提供不同的操作
- 对象一般都提供读取它的实例变量值的方法,统称为*accessor*.
  - 例如Point对象的getX()和getY(), Line对象的getP1()和getP2().
- 对象一般也提供修改其实例变量的方法,统称为*mutator*.
  - 例如所有图形对象都有move(dx,dy)方法.

第5章 图形编程



- 计算可视化
- 面向对象基本概念
  - 面向对象思想与编程在第7章介绍
- 两个图形编程库
- graphics图形编程
  - 本章介绍基本的绘图功能
  - 图形用户界面编程在第8章介绍
- 编程案例

图形编程模块

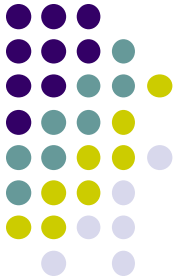


- 图形编程模块
  - Python的标准库:*Tkinter*
  - 面向对象简单易学的图形库:*graphics.py*
  - 提供窗口、菜单、按钮等图形构件
- 使用图形编程模块
  - 导入

```
import Tkinter                    import graphics
From Tkinter import *             from graphics import *
```

  - 调用其中的函数编写图形程序

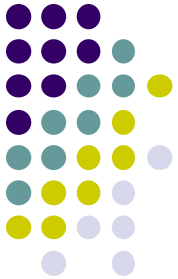
Tkinter图形编程



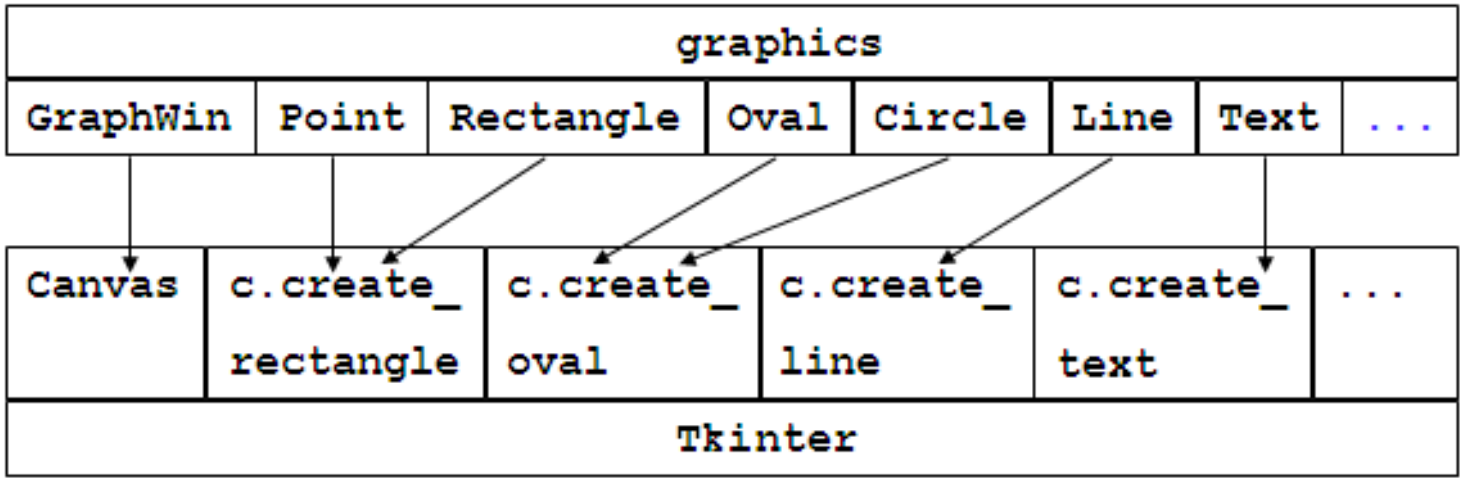
- 15种控件 （对象）

  Button 按钮
  Canvas 画布，可在上面绘制直线，矩形等图形
  也可含位图
  Checkbutton 核对按钮
  Radiobutton 无线按钮
  Entry 单行文本框，文字可输入和修改
  Text 多行文本框，文字可输入和修改
  Label 标签
  Listbox 列表框，用户可从中选择
  Messagebox 消息框，类似于Label,但可以多行
  Scale 进度条，可设置起始值和终了值
  Scrollbar 滚动条
  Menu 菜单，点下菜单后弹出选项列表，用户可从中选择
  Menubutton 菜单按钮，下拉和层叠菜单组件

面向对象graphics库

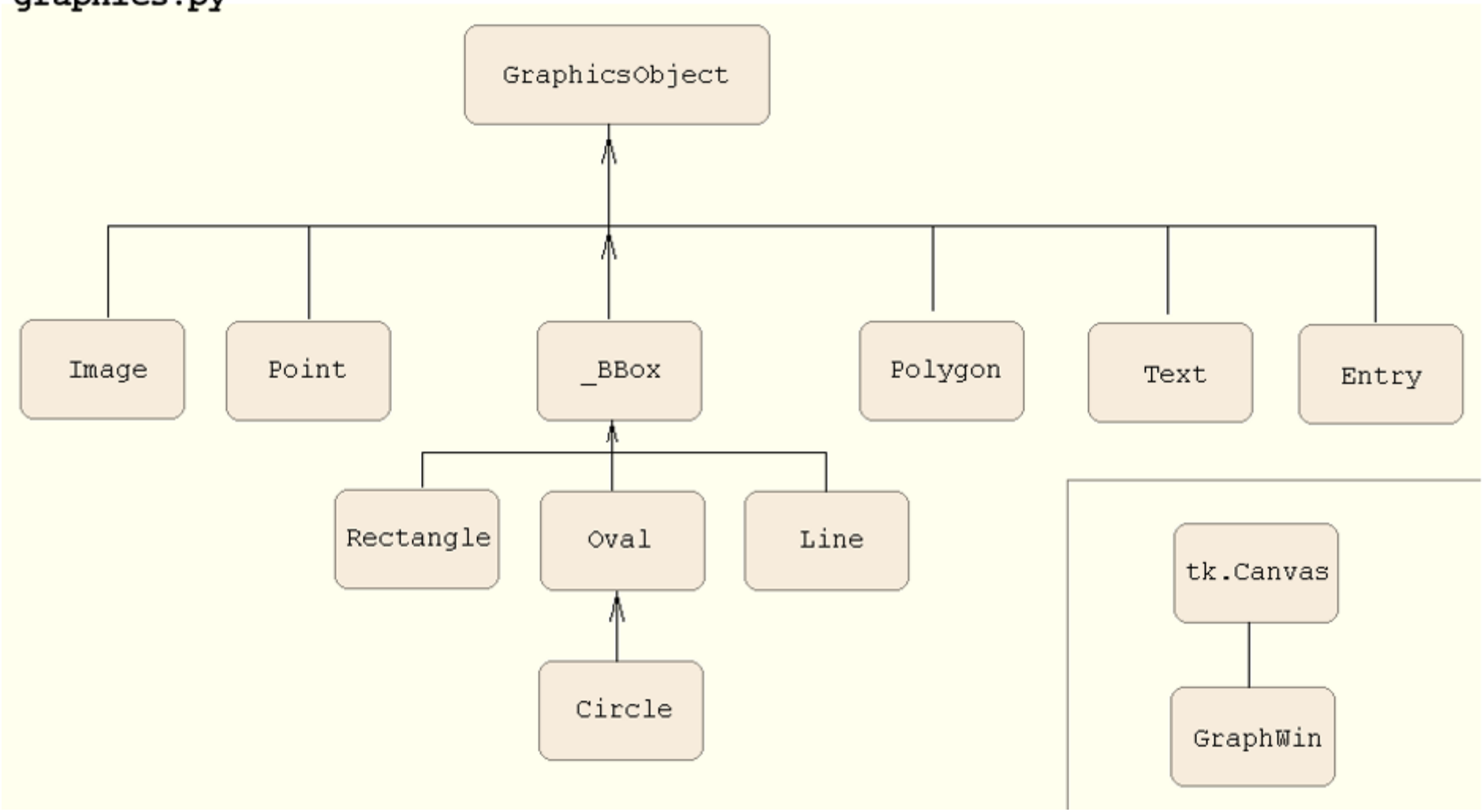


- 在Tkinter之上设计的另一个图形模块graphics.py
  - 因为他认为Tkinter较复杂,初学者学起来有点难
  - graphics将Tkinter的功能封装成了更易用的类
  - 这是非标准模块,需要自己下载.

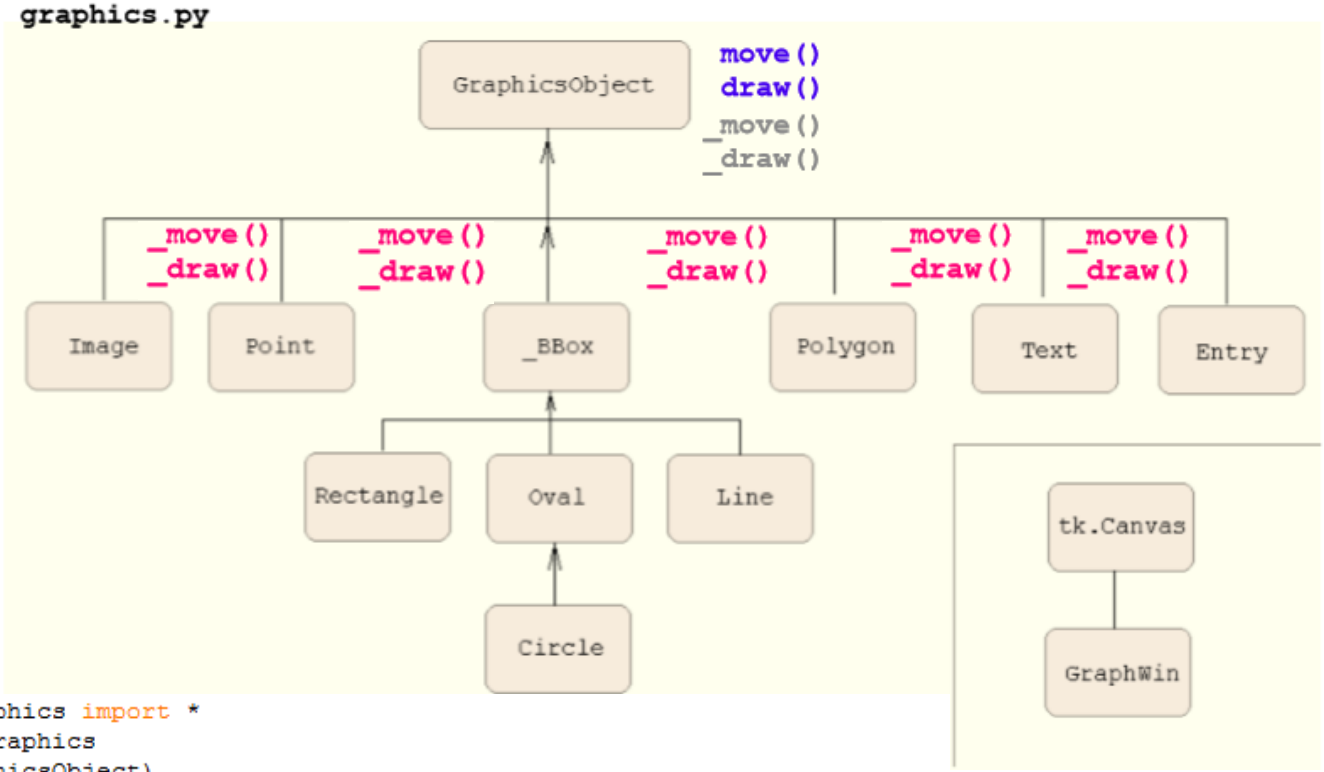


面向对象graphics库

graphics.py



Graphics库:继承

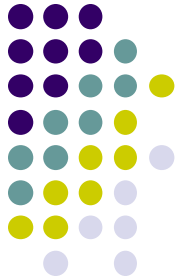



```
>>> from graphics import *
>>> import graphics
>>> dir(GraphicsObject)
['__doc__', '__init__', '__module__', '_draw', '_move', '_reconfig', 'draw'
, 'move', 'setFill', 'setOutline', 'setWidth', 'undraw']
>>> dir(graphics._BBox)
['__doc__', '__init__', '__module__', '_draw', '_move', '_reconfig', 'draw'
, 'getCenter', 'getP1', 'getP2', 'move', 'setFill', 'setOutline', 'setWidth
', 'undraw']
>>> dir(Rectangle)
['__doc__', '__init__', '__module__', '_draw', '_move', '_reconfig', 'clone
', 'draw', 'getCenter', 'getP1', 'getP2', 'move', 'setFill', 'setOutline',
'setWidth', 'undraw']
>>> dir(_BBox)

Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    dir(_BBox)
NameError: name ' BBox' is not defined
```

built-in attributes or methods

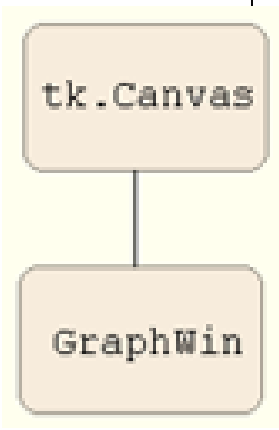
private attributes or methods

第5章 图形编程

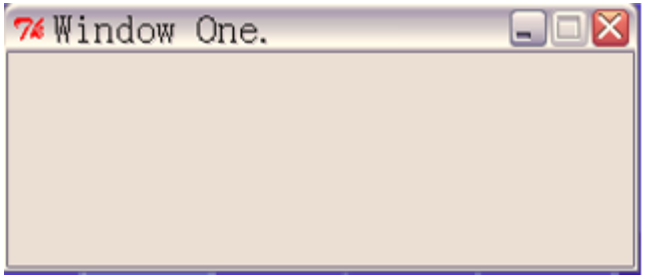


- 计算可视化
- 面向对象基本概念
  - 面向对象思想与编程在第7章介绍
- 图形编程库
- graphics图形编程
  - 本章介绍基本的绘图功能
  - 图形用户界面编程在第8章介绍
- 编程案例

图形窗口 GraphWin类




```
>>> import graphics
>>> win = graphics.GraphWin('window one', 300, 100)
>>> win.close()
```



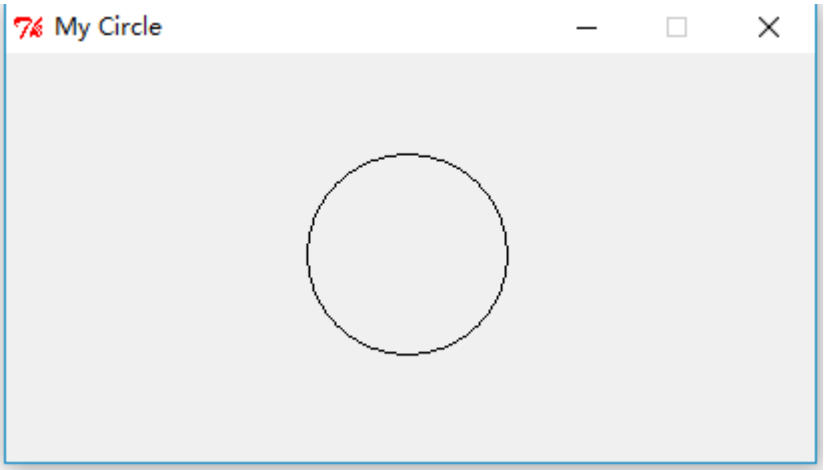

```
>>> from graphics import *
>>> win = GraphWin('window one', 300, 100)
>>> win.close()
```

```
__init__(self, title="Graphics Window",
         width=200, height=200, autoflush=True)
```

- 图形窗口:画图的地方
- 窗口是由像素组成的:默认大小200*200
- 画图:实际就是控制各像素的颜色.

图形窗口 GraphWin类

```
>>> from graphics import *
>>> def main():
        win = GraphWin("My Circle", 400, 200)
        c = Circle(Point(200,100), 50)
        c.draw(win)
        # pause for click in window win.close()
        win.getMouse()
        win.close()


>>> main()
```



python.py

2 GraphWin Objects

A `GraphWin` object represents a window on the screen where graphical image
A program may define any number of `GraphWins`. A `GraphWin` understan
methods:

`GraphWin(title, width, height)` Constructs a new graphics window for
screen. The parameters are optional, the default title is "Graphics Wi
default size is 200 x 200.

`plot(x, y, color)` Draws the pixel at (x, y) in the window. Color is optio
default.

`plotPixel(x, y, Color)` Draws the pixel at the "raw" position (x, y) ign
dinate transformations set up by `setCoords`.

`setBackground(color)` Sets the window background to the given color. T
ground is gray. See Section 5.8.5 for information on specifying colors.

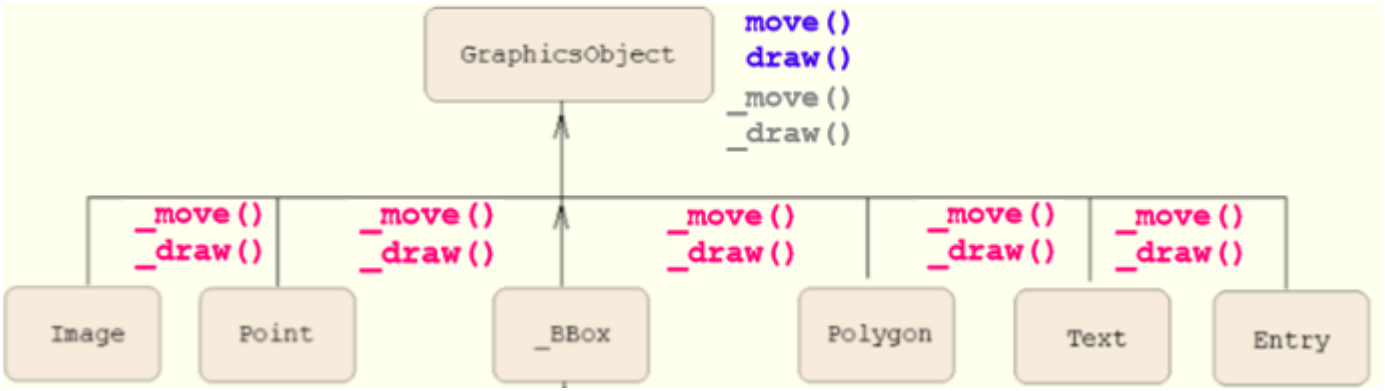
`close()` Closes the on-screen window.

`getMouse()` Pauses for the user to click a mouse in the window and returns
was clicked as a `Point` object.

`checkMouse()` Similar to `getMouse`, but does not pause for a user click. Re
point where the mouse was clicked or `None` if the window as not been
previous call to `checkMouse` or `getMouse`. This is particularly usefu
simple animation loops.

`setCoords(xll, yll, xur, yur)` Sets the coordinate system of the wind

图形对象 GraphicsObject基类




```
def _draw(self, canvas, options):
    """draws appropriate figure on canv
    Returns Tk id of item drawn"""
    pass # must override in subclass


def _move(self, dx, dy):
    """updates internal state of object
    pass # must override in subclass
```

3 Graphics Objects

The module provides the following classes of drawable objects: `Point`, `Line`, `Circle`, `C`
`Rectangle`, `Polygon`, and `Text`. All objects are initially created unfilled with a black out
All graphics objects support the following generic set of methods:

`setFill(color)` Sets the interior of the object to the given color.

`setOutline(color)` Sets the outline of the object to the given color.

`setWidth(pixels)` Sets the width of the outline of the object to this many pixels. (
not work for `Point`.)

`draw(aGraphWin)` Draws the object into the given `GraphWin`.

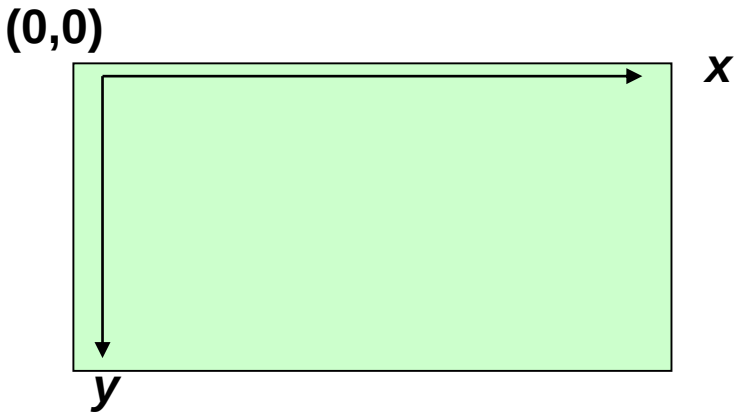
`undraw()` Undraws the object from a graphics window. This produces an error if the ob
is not currently drawn.

`move(dx,dy)` Moves the object `dx` units in the x direction and `dy` units in the y direc
If the object is currently drawn, the image is adjusted to the new position.

`clone()` Returns a duplicate of the object. Clones are always created in an undrawn s
Other than that, they are identical to the cloned object.

点 Point类

- 点:图形窗口的一个像素,位置用坐标(*x*,*y*)表示.
- 坐标系统
  - 原点(0,0):左上角
  - *x*轴:自左向右
  - *y*轴:自顶向下
- 小测试:默认大小的图形窗口,右下角坐标是?



点Point类

- 创建Point类的对象　Point(<*x*坐标>,<*y*坐标>)
- 对Point对象的操作

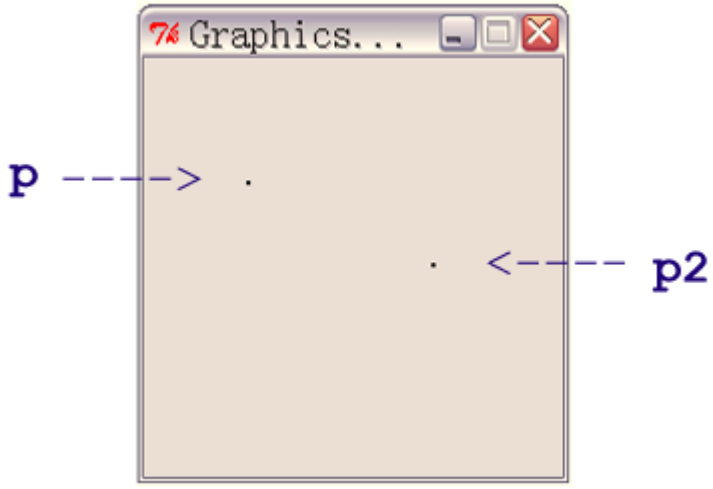
p.getX(), p.getY(), p.draw(win), p.undraw()

p.move(dx,dy), p.setFill(color), p.setOutline(color)

```
class Point(GraphicsObject):
    def __init__(self, x, y):
        GraphicsObject.__init__(self, ["outline", "fill"])
        self.setFill = self.setOutline
        self.x = x
        self.y = y

    def _draw(self, canvas, options):
        x,y = canvas.toScreen(self.x,self.y)
        return canvas.create_rectangle(x,y,x+1,y+1,options)

    def _move(self, dx, dy):
        self.x = self.x + dx
        self.y = self.y + dy

    def clone(self):
        other = Point(self.x,self.y)
        other.config = self.config.copy()
        return other

    def getX(self): return self.x
    def getY(self): return self.y
```

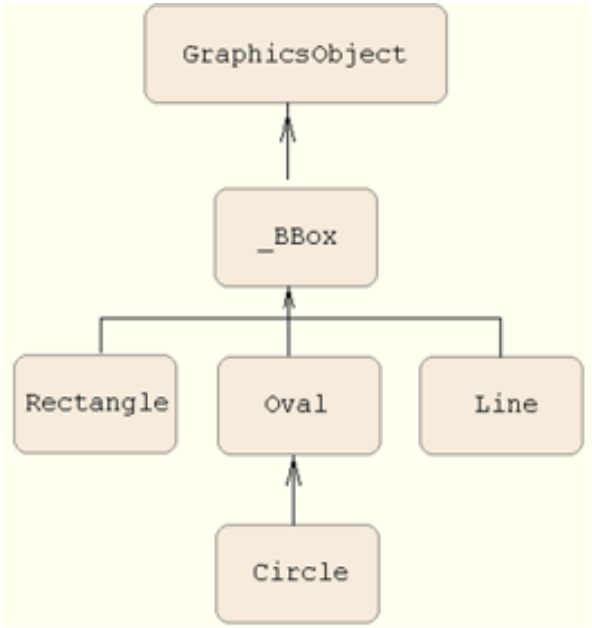
点Point类

```
>>> from graphics import *
>>> win = GraphWin()
>>> p1 = Point(50,60)
>>> p2 = Point (140,100)
>>> p1.draw(win)
>>> p2.draw(win)
>>> p1.undraw()
>>> win.close()
```

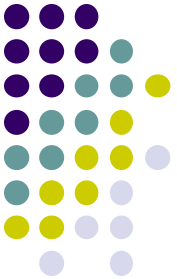


限定框_BBox基类

```
class _BBox(GraphicsObject):
    # Internal base class for objects represented by bounding box
    # (opposite corners) Line segment is a degenerate case.

    def __init__(self, p1, p2, options=["outline","width","fill"]):
        GraphicsObject.__init__(self, options)
        self.p1 = p1.clone()
        self.p2 = p2.clone()

    def _move(self, dx, dy):
        self.p1.x = self.p1.x + dx
        self.p1.y = self.p1.y + dy
        self.p2.x = self.p2.x + dx
        self.p2.y = self.p2.y  + dy

    def getP1(self): return self.p1.clone()

    def getP2(self): return self.p2.clone()

    def getCenter(self):
        p1 = self.p1
        p2 = self.p2
        return Point((p1.x+p2.x)/2.0, (p1.y+p2.y)/2.0)
```



线段 Line类



- 创建Line类的对象

  Line(<端点1>,<端点2>)

  - 例如

  line = Line(Point(0,0),Point(100,100))

- 对Line对象的操作

  line.draw(win),line.undraw()

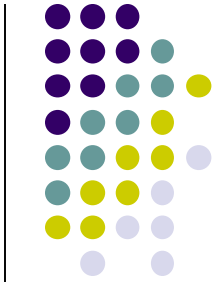
  line.move(dx,dy)

  line.setFill(color),line.setOutline(color)

  line.setArrow(arrow)

  line.clone()

圆形Circle类



- 创建Circle类的对象

  Circle(<圆心>,<半径>)

  - 例如

  c = Circle(Point(50,50),40)

- 对Circle对象的操作

  c.draw(win),c.undraw()

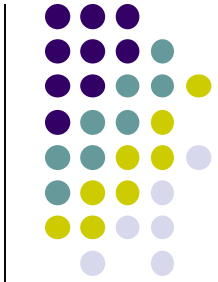
  c.setFill(color),c.setOutline(color)

  c.move(dx,dy)

  c.getRadius()

  c.clone()

椭圆Oval类



- 创建Oval类的对象

  Oval(<限定框左上角>,<限定框右下角>)

  - 例如

  o = Oval(Point(20,150),Point(180,199))

- 对Oval对象的操作

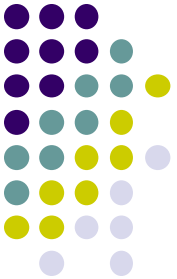
  o.draw(win),o.undraw()

  o.setFill(color),o.setOutline(color)

  o.move(dx,dy), o.getCenter()

  o.clone()

矩形Rectangle类



- 创建Rectangle类的对象

  Rectangle(<左上角>,<右下角>)

  - 例如

  r = Rectangle(Point(5,8),Point(30,40))

- 对Rectangle对象的操作

  r.draw(win),r.undraw()

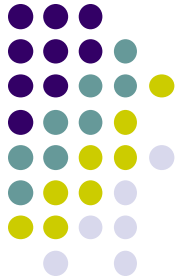
  r.setFill(color),r.setOutline(color)

  r.move(dx,dy)

  r.getP1(), r.getP2(), r.getCenter()

  r.clone()

多边形Polygon类



- 创建Polygon类的对象

  Polygon(<顶点1>,<顶点2>,<顶点3>,…)

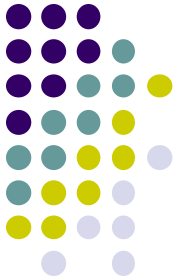
  - 例如

  ```
  p = Polygon(Point(10,10),Point(30,30),Point(10,30))
  ```

- 对Polygon对象的操作

  ```
  p.draw(win),p.undraw()
  p.setFill(color),p.setOutline(color)
  p.move(dx,dy)
  p.getPoints()
  p.clone()
  ```

文本Text类



- 创建Text类的对象

  `t = Text(<中心位置点>,<字符串>)`

  - 例如

  `t = Text(Point(99,99),”text here”)`

- 对Text对象的操作

  `t.draw(win),t.undraw()`

  `t.setText("txt"), t.getText()`

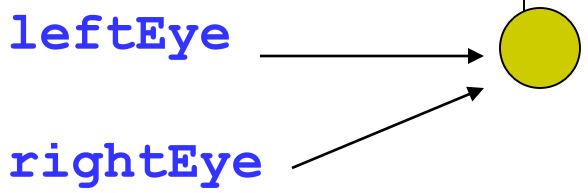
  `t.setTextColor(color)`

  `t.clone()`

如何复制对象:画两眼

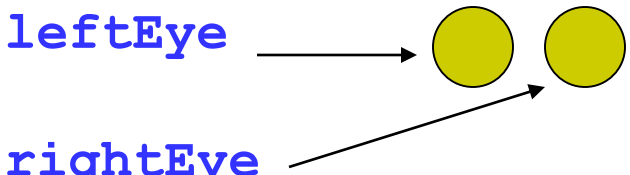
- 错误代码

```
leftEye = Circle(Point(80,50),5)
leftEye.setFill('yellow')
leftEye.setOutline('red')
rightEye = leftEye    #Python变量是引用,而不是值的复制!
rightEye.move(20,0)
```

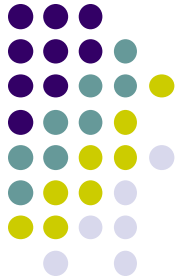


- 正确代码

```
leftEye = Circle(Point(80,50),5)
leftEye.setFill('yellow')
leftEye.setOutline('red')
rightEye = Circle(Point(100,50),5)
rightEye.setFill('yellow')
rightEye.setOutline('red')
```



如何复制对象:画两眼(续)

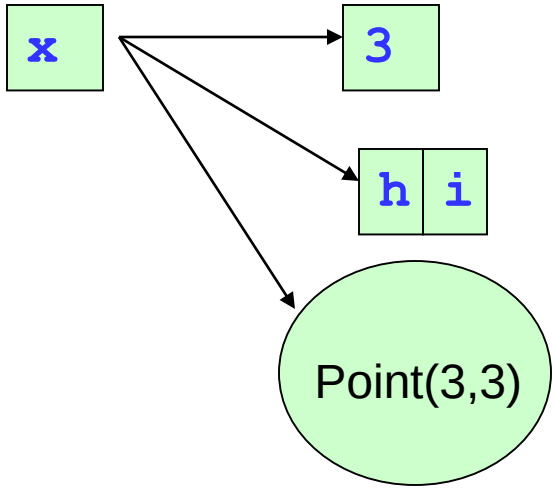


- 更美观的代码

```
leftEye = Circle(Point(80,50),5)
leftEye.setFill('yellow')
leftEye.setOutline('red')
rightEye = leftEye.clone()     #使用clone()复制对象
rightEye.move(20,0)
```

回顾与总结:Python的变量赋值

```
>>>x = 3
>>>type(x)
>>>x = 'hi'
>>>type(x)
>>>x = Point(3,3)
>>>type(x)
```

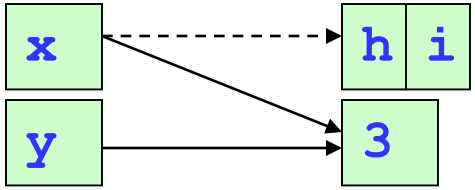


- 回顾:Python变量
  - 随时创建
  - 无需事先声明类型
  - 可随时"改变类型"
- 因为Python是动态类型化的.
  - 数据是在内存中创建的对象
  - 变量在赋值时创建
  - 变量是对数据对象的引用
  - 类型从属于数据对象,而非变量
  - 改变变量实际是改变变量所指的对象
  - 只有在执行时,才能确定变量所指的是哪个对象

回顾与总结:共享引用

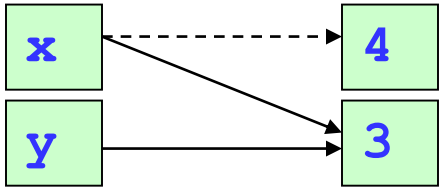
- 共享引用:多个变量引用同一个数据对象

```
>>>x = 3
>>>y = x
>>>x = 'hi'
```



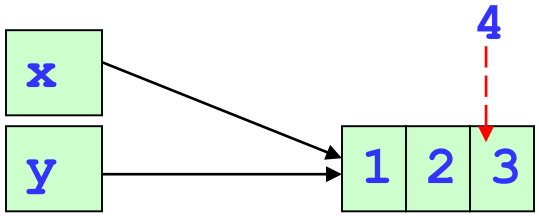
- 不可变对象的共享

```
>>>x = 3
>>>y = x
>>>x = x + 1
```

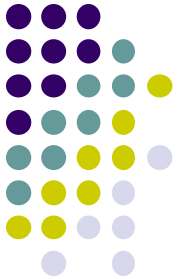


- 可变对象的共享

```
>>>x = [1,2,3]
>>>y = x
>>>x[2] = 4
```



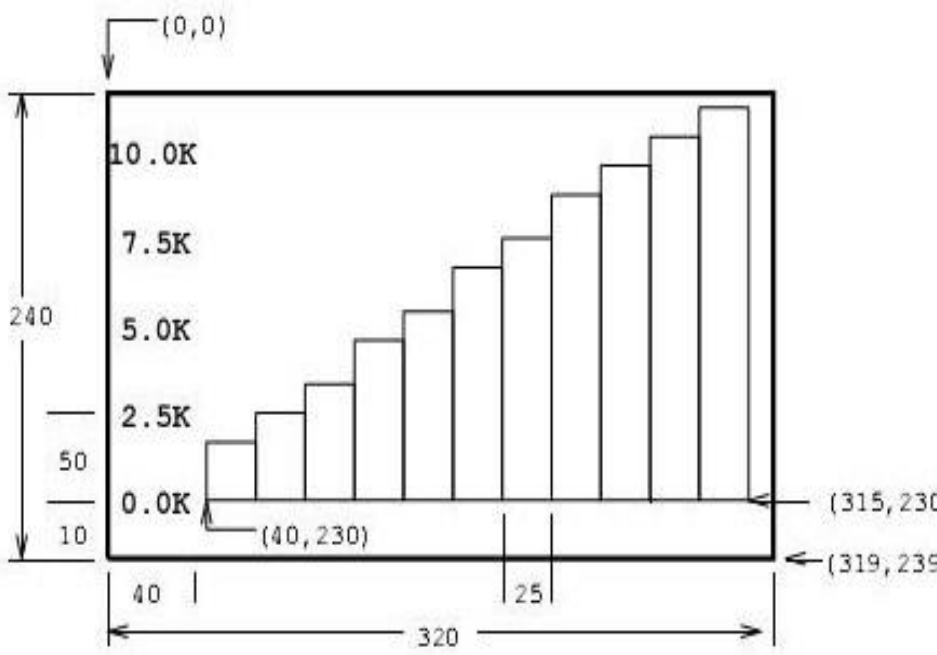
第5章 图形编程



- 计算可视化
- 面向对象基本概念
  - 面向对象思想与编程在第7章介绍
- 图形编程库
- graphics图形编程
  - 本章介绍基本的绘图功能
  - 图形用户界面编程在第8章介绍
- 编程案例

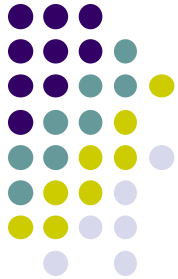
编程实例:投资收益图形版

- 用柱状图表示10年的投资收益
- 算法
  - 输入本金和年利率
  - 创建图形窗口
  - 窗口左边画上刻度
  - 0号位置画对应初始本金的柱子
  - 对接下来的1到10年:
  - principal=principal*(1 + rate)
  - 在相应位置画对应的柱子



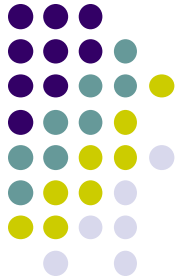

```
win = GraphWin("Investment Growth Chart", 320, 240)
```

编程实例:投资收益图形版(续)



- 确定刻度的内容和位置
  - 内容:0.0K,2.5K,5.0K,7.5K,10.0K
    - 右对齐,左边填充空格
  - 位置:确定水平方向40像素为中心;垂直方向 ($10,000)平分200像素.
- 算法精化
  - Draw label " 0.0K" at (40, 230)
  - Draw label " 2.5K" at (40, 180)
  - Draw label " 5.0K" at (40, 130)
  - Draw label " 7.5K" at (40, 80)
  - Draw label "10.0K" at (40, 30)

编程实例:投资收益图形版(续)



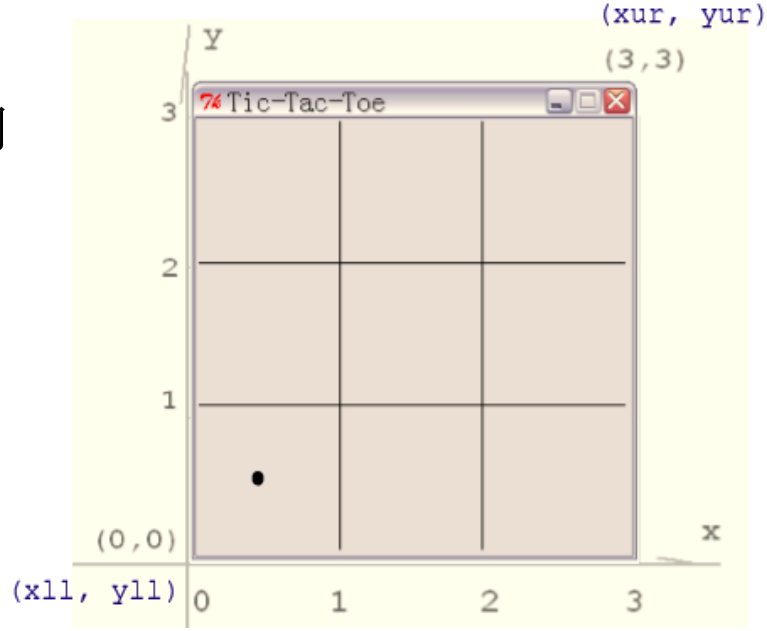
- 确定0号柱子的位置和高度
  - 位置: 左下角坐标为(60,230)，右上角坐标为（80，230–principal*0.02)
  - 高度: 230–principal*(200/10000).故右上角y坐标为(230–principal*0.02)
  - 宽度: (320–40)/11 ≈ 25, 故右上角x坐标为(80)
- 其他柱子的位置和高度
  - 位置: 左下角为(year*20+60, 230).
  - 高度: principal*0.02, 故右上角y坐标为(230–principal*0.02)
  - 宽度: 20,故右上角x坐标为(左下角x坐标+20)

自定义坐标系

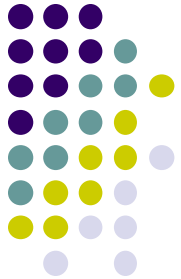
- 默认坐标系以像素为单位,编程很麻烦.
  - 坐标变换:实际数据需映射成窗口像素位置.
- Graphics模块提供自定义坐标系的功能,使坐标变换自动完成 setCoords(xll,yll,xur,yur)。好处：
  - 编程简单直观
  - 改变窗口像素尺寸对程序几乎没影响
- 例如

```
win = GraphWin(“Tic-Tac-Toe”,300,300)
win.setCoords(0.0, 0.0, 3.0, 3.0)
line=Line(Point(1,0), Point(1,3))
line.draw(win)
```

  - 新坐标系为:左下角(0,0),右上角(3,3)
  - 系统自动把新坐标变换成3*3像素坐标



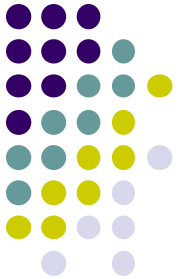
捕获鼠标点击



- GraphWin类有一个方法getMouse():调用时等待用户点击,并返回点击位置(是个Point对象).
- 例如:

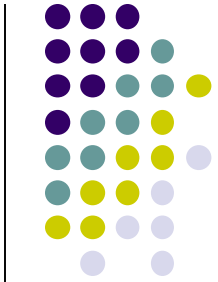
```
win = GraphWin("Click Me!")
for i in range(10):
    p = win.getMouse()
    print "You clicked (%d, %d)" % (p.getX(),
    p.getY())
```

编程实例:triangle.py



```
from graphics import *
def main():
    win = GraphWin("Draw a Triangle")
    win.setCoords(0.0, 0.0, 10.0, 10.0)
    message = Text(Point(5, 0.5), "Click on three points")
    message.draw(win)
    p1 = win.getMouse()
    p1.draw(win)
    p2 = win.getMouse()
    p2.draw(win)
    p3 = win.getMouse()
    p3.draw(win)
    triangle = Polygon(p1,p2,p3)
    triangle.draw(win)
    message.setText("Click anywhere to quit.")
    win.getMouse()

main()
```

图形窗口中的文本输入



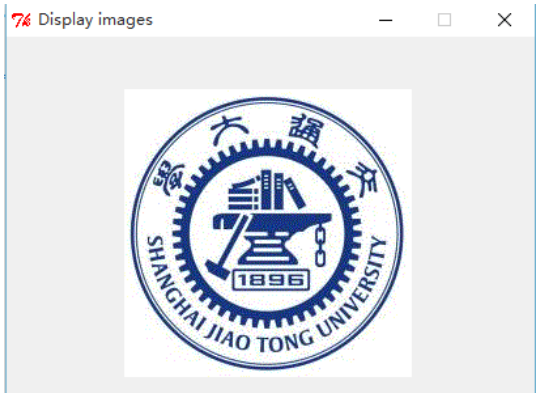
- Entry对象:提供一个可编辑文本的框
  - Entry(<中心点>,<宽度>)
  - 方法:setText(<字符串>)和getText()等
- 实例

```
input = Entry(Point(2,3), 5)
input.setText("0.0")
input.draw(win)
n = eval(input.getText())
```

显示图片

- Image对象:显示GIF/PPM图像
  - Image(<中心点>,<图像文件>)
  - 方法: getPixel(x, y), setPixel(x,y, color), save(filename)等
- 实例

```
from graphics import *

def main():
    win = GraphWin("Display images", 400, 300)
    win.setCoords(0.0, 0.0, 8.0, 6.0)
    # display image using Image class
    # which can only disply Gif or PPM files
    image = Image(Point(4,3), "sjtu.gif")
    image.draw(win)

    win.getMouse()
    win.close()
```



sjtu.gif 文件的位置：

python command line:

与 python.exe 同一子目录

display_image.pyc:

与 display_image.pyc 同一子目录

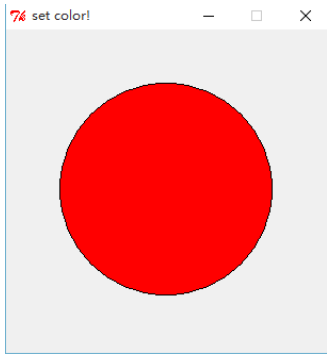
IDLE - Run Module:

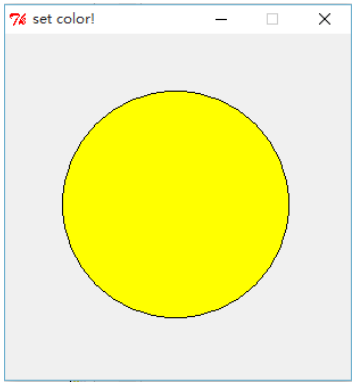
与 display_image.py 同一子目录

颜色设置

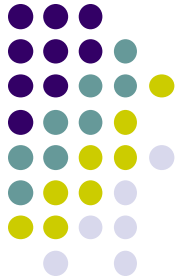
- `p.setFill(color),p.setOutline(color)`
- value of color
  - normal colors such as ’red’, ’purple’, ’green’, ’cyan’, etc
  - the function color rgb(red, green, blue)
- 实例

```
win=GraphWin('set color!')
win.setCoords(0,0,6,6)
center=Point(3,3)
circ=Circle(center,2)
circ.setFill('red')
circ.draw(win)

win.getMouse()
# set color to yellow=rgb(255,255,0)
myColor=color_rgb(255,255,0)
circ.setFill(myColor)
```

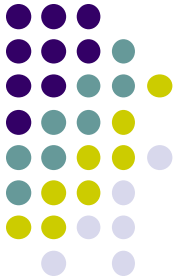




图形交互界面



- GUI
  - 图形元素用于输入输出
  - 事件驱动的编程
- 事件:移动鼠标,点击按钮,选菜单等.
  - 也是对象
  - 事件发生后,有相应事件处理程序来处理.
- 第8章学习



End