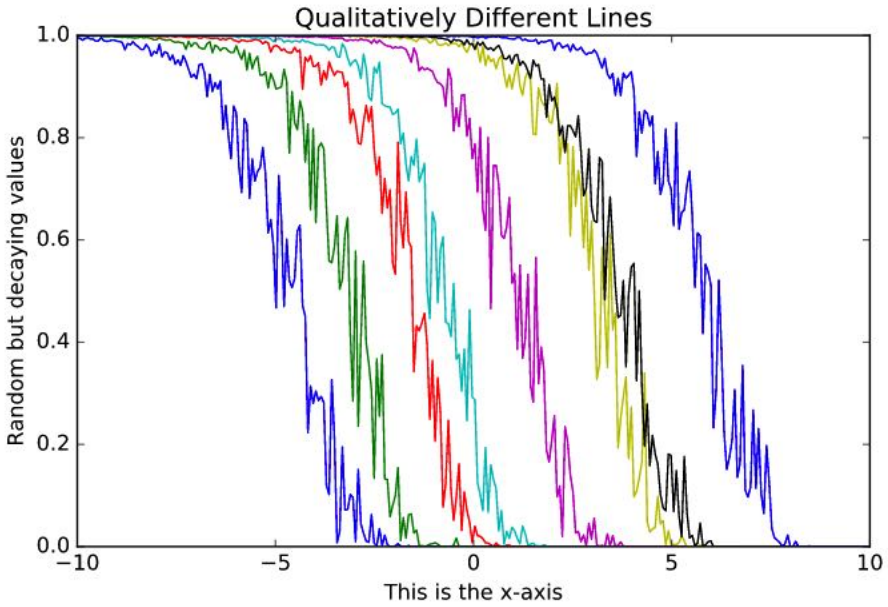
Python语言基础与应用07

北京大学 陈斌

2019.07.10

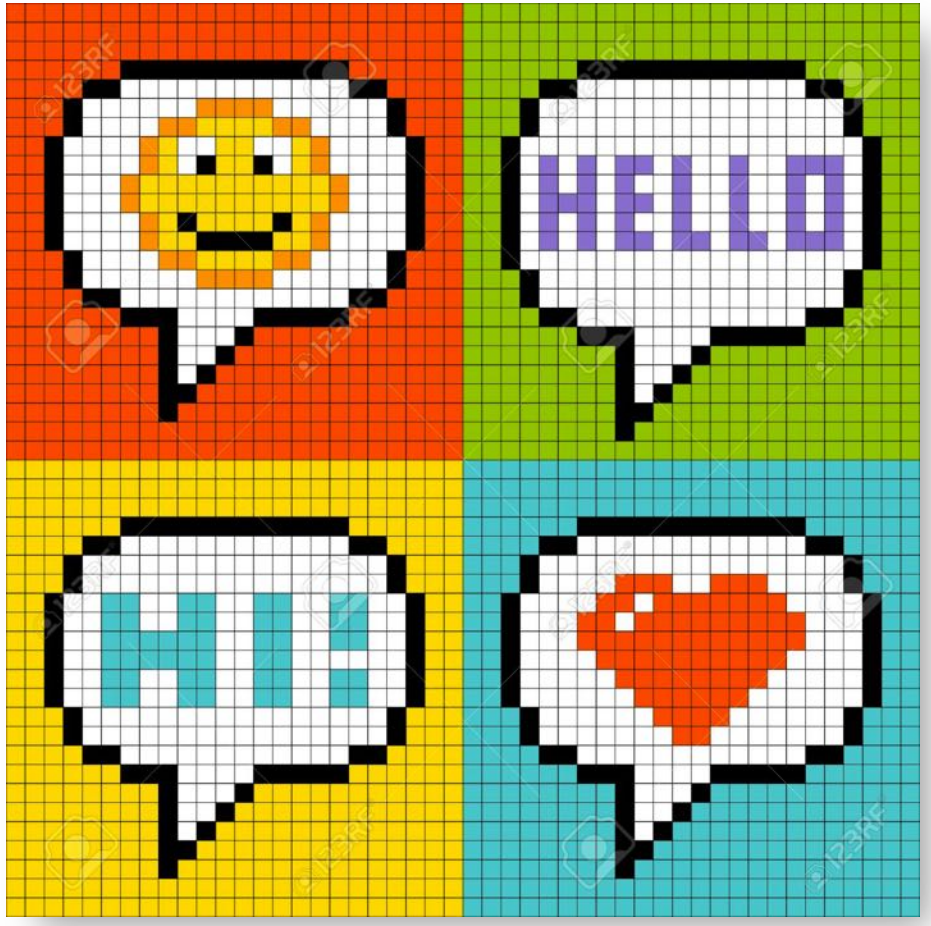
目录

- 图形图像基本知识
- 图像处理PIL
- 图表可视化matplotlib
- 多媒体应用pygame zero
- Web App和网络爬虫

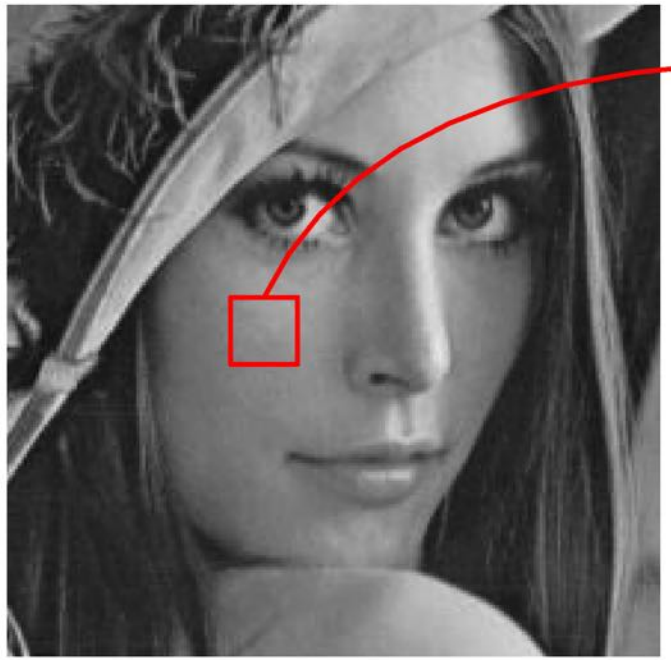


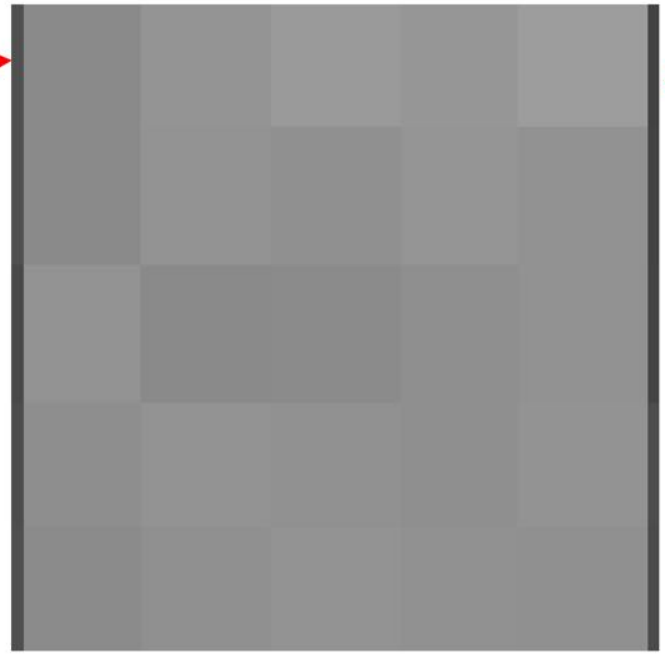
二维图形表示：像素和图像

- 像素Pixel
  - 组成图像的基本小方格，具有大小和位置，规则排列
- 像素的属性
  - 形状、大小、位置、颜色值
- 图像Image
  - 由规则排列的像素构成的矩形，可以描绘各种视觉形象
- 图像的属性
  - 分辨率、像素密度、颜色模型



二维图形表示：像素和图像



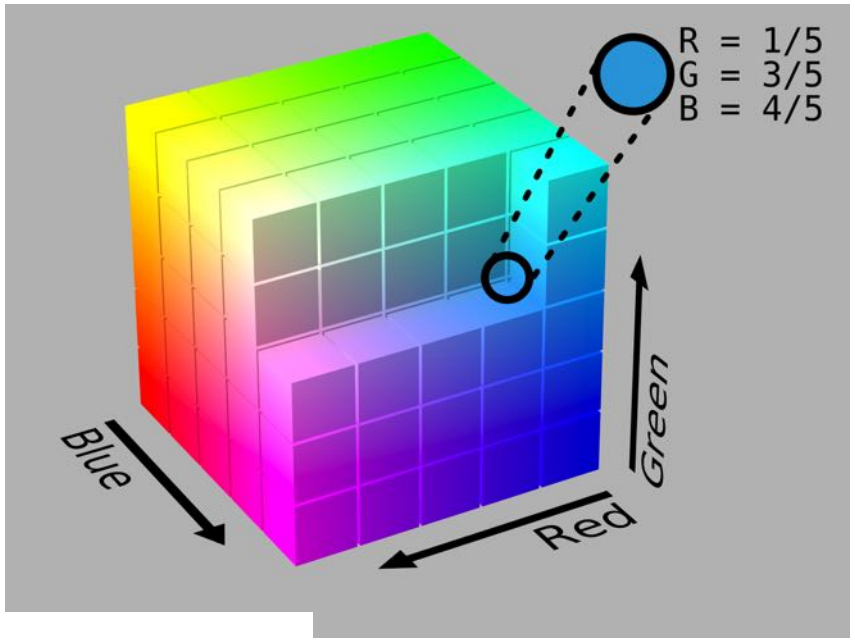


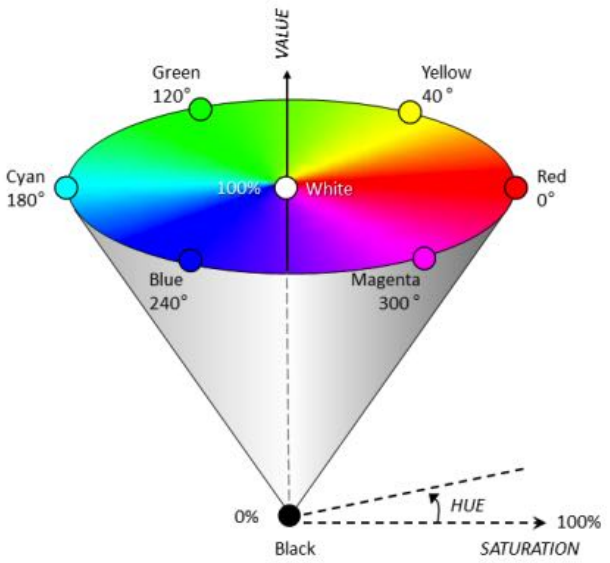
156	165	170	167	173
156	164	161	166	163
164	156	157	160	163
159	164	162	160	164
157	161	164	162	161

(a) original image (b) Region of red rectangle (c) Gray-scale value

图像颜色模型：RGB

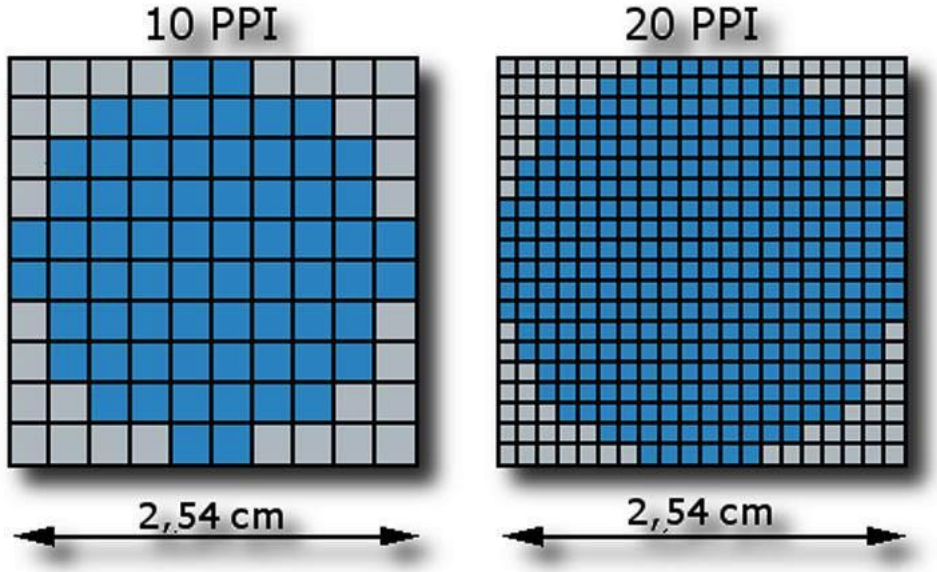
- 三原色模型RGB
- 用3个字节表示颜色
  - 分别表示红、绿、蓝颜色值
  - 0-255，一共有255*255*255种
- 引入第四个字节表示透明度的RGBA模型
- 另一种常用颜色模型HSV
  - 辉度、饱和度、亮度





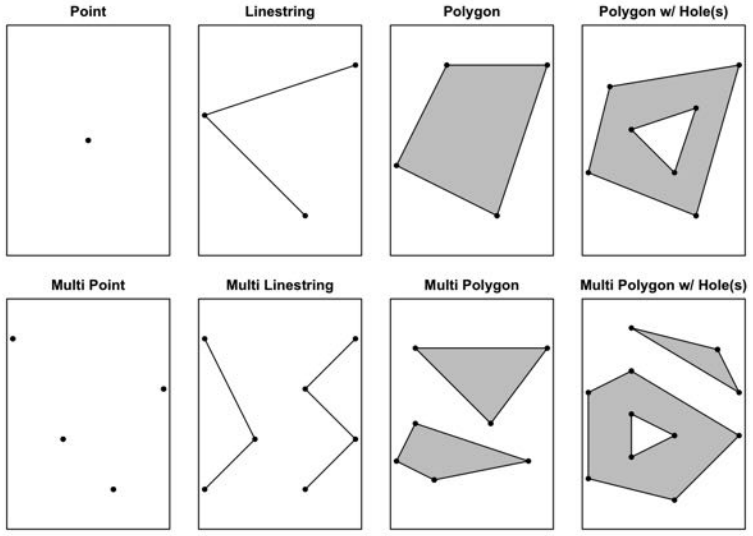
像素密度：PPI（Pixel Per Inch）

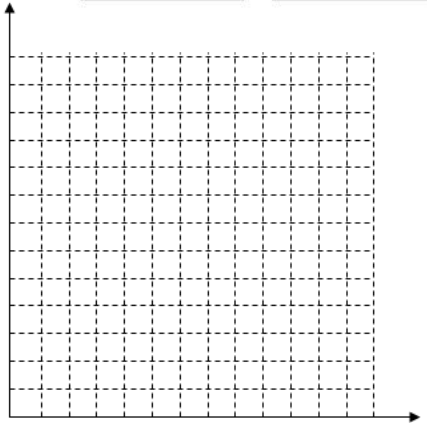
- 每英寸像素点数量
- 密度越高图像越精细
- 视网膜分辨率
  - 人眼在常规距离上无法分辨出视网膜屏幕的像素点
  - 标准视力5.0，看手机的距离，300ppi达到无法分辨像素点



二维坐标平面和矢量图形

- 在二维坐标平面，用顶点和顶点的连接来表示点、线、面
- 通过点符号、线型、填充模式来表达形状，并指定颜色
- 可以绘制高质量图形，并任意缩放





PIL：图像处理库

- Python 3安装Pillow
- Python上事实标准库
- 功能强大，可以对图像做各种处理
- 如：缩放、裁剪、旋转、滤镜、文字、调色板等等

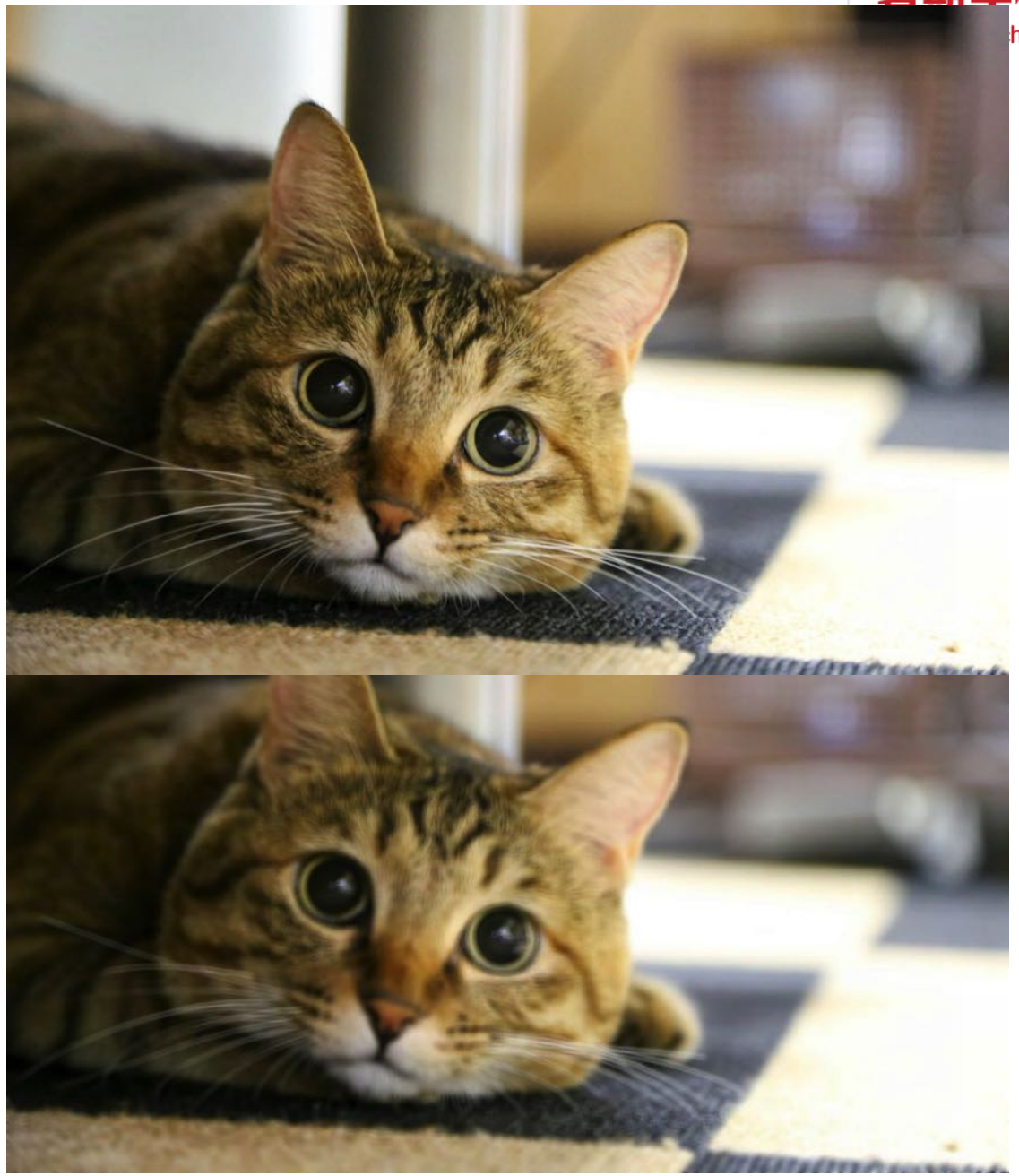
PIL缩放图像操作

```
from PIL import Image

# 打开一个jpg图像文件，注意是当前路径:
im = Image.open('test.jpg')
# 获得图像尺寸:
w, h = im.size
print('Original image size: %sx%s' % (w, h))
# 缩放到50%:
im.thumbnail((w//2, h//2))
print('Resize image to: %sx%s' % (w//2, h//2))
# 把缩放后的图像用jpeg格式保存:
im.save('thumbnail.jpg', 'jpeg')
```

```
============== RESTART: /Users
Original image size: 900x600
Resize image to: 450x300
```

PIL模糊效果

```
from PIL import Image, ImageFilter

# 打开一个jpg图像文件，注意是当前路径:
im = Image.open('test.jpg')
# 应用模糊滤镜:
im2 = im.filter(ImageFilter.BLUR)
im2.save('blur.jpg', 'jpeg')
```



PIL查看图像信息

```
1  from PIL import Image
2
3  im = Image.open("images/4.jpg")
4  print(im.format, im.mode, im.size)
5  px = list(im.getdata())
6  print(len(px))
7  print(px[:900])
8  im.show()
```

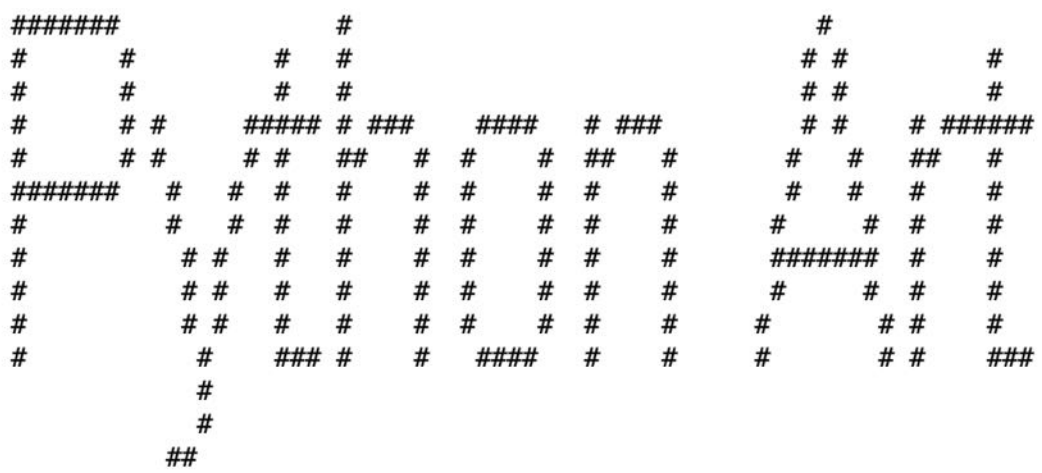
```
JPEG RGB (500, 585)
292500
[(255, 255, 255), (255, 255, 255), (255, 255, 255), (255,
```

PIL生成验证码


```python
from PIL import Image, ImageDraw, ImageFont, ImageFilter

import random

# 随机字母:
def rndChar():
    return chr(random.randint(65, 90))

# 随机颜色1:
def rndColor():
    return (random.randint(64, 255), \
            random.randint(64, 255), \
            random.randint(64, 255))

# 随机颜色2:
def rndColor2():
    return (random.randint(32, 127), \
            random.randint(32, 127), \
            random.randint(32, 127))

# 240 x 60:
width = 60 * 4
height = 60
image = Image.new('RGB', (width, height), (255, 255, 255))
# 创建Font对象:
font = ImageFont.truetype('Arial.ttf', 36)
# 创建Draw对象:
draw = ImageDraw.Draw(image)
# 填充每个像素:
for x in range(width):
    for y in range(height):
        draw.point((x, y), fill=rndColor())
# 输出文字:
for t in range(4):
    draw.text((60 * t + 10, 10), rndChar(), font=font, fill=rndColor2())
# 模糊:
image = image.filter(ImageFilter.BLUR)
image.save('code.jpg', 'jpeg')
```

ASCII字符图形艺术

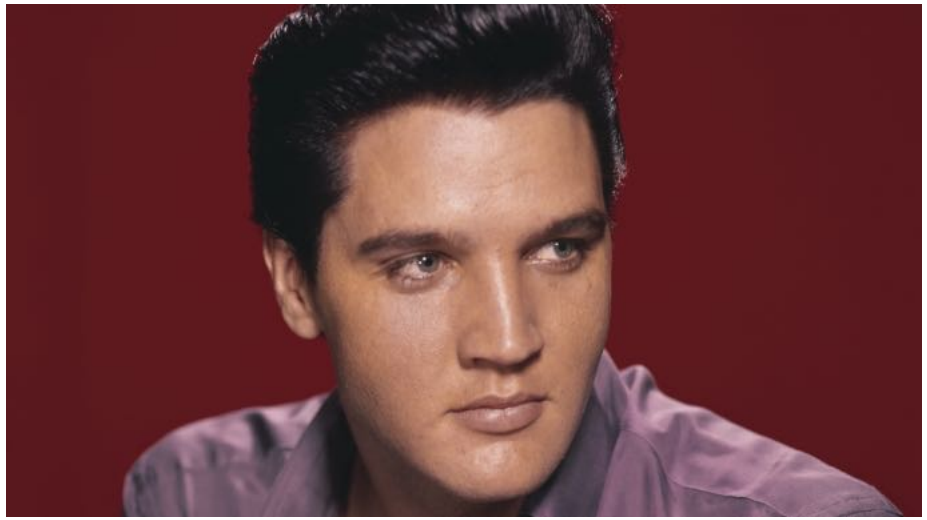
```
大小= (70, 17)
```



```
######           #                                  #
#     #        #  #                                 # #        #
#     #        #  #                                 # #        #
#     # #    ##### # ###    ####   # ###            # #    # ######
#     # #     # #  ##   #  #    #  ##   #          #  #   ##   #
######  #   # #  #    #  #    #  #    #          #  #   #    #
#       #   # #  #    #  #    #  #    #         #     #  #    #
#        # #  #  #    #  #    #  #    #         ###### #    #
#        # #  #  #    #  #    #  #    #         #     #  #    #
#        # #  #  #    #  #    #  #    #        #      # #    #
#         #   ### #    #   ####   #    #        #      # #    ###
#
#
##
```

```python
from PIL import Image, ImageFont, ImageDraw

# 绘制文字
words = "Python Art"
# 准备一个字体
font = ImageFont.truetype("Arial", 15)
# 看这段文字绘制后的大小
size = font.getsize(words)
print('大小=', size)
# 新建一个图像，黑白模式，底色白色
im = Image.new('1', size, 'white')
draw = ImageDraw.Draw(im)
# 在图像上绘制文字
draw.text((0, 0), words, font=font)
im.show()
# 循环每个像素，黑色就记为"#"，白色就记为" "
asc = []
for p in list(im.getdata()):
    if p == 0:  # 黑色
        asc.append("#")
    else:
        asc.append(" ")

# 逐行打印出来，size=(列数，行数)
for row in range(size[1]):
    for col in range(size[0]):
        print(asc[row*size[0]+ col], end='')
    print()
```

如何做成这样呢？



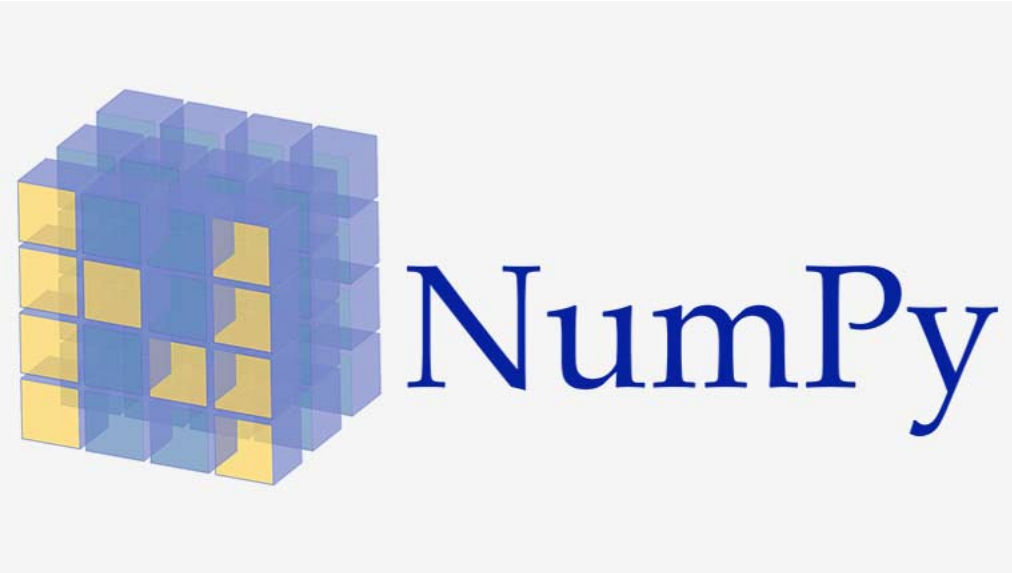
```
list(' .,:;irsXA253hMHGS#9B&@')
```

上机练习

- 利用PIL生成一批图片的统一大小(200*100)的缩略图
  - 0.jpg, 1.jpg, 2.jpg……
  - 生成s0.jpg, s1.jpg, s2.jpg……
  - 从课程网站下载images.zip
- 编写一个从彩色图形转化为ASCII字符图形艺术的程序
  - 从网上下载图片

numpy库

- numpy是Python用于处理大型矩阵的一个速度极快的数学库
  - 可以做向量和矩阵的运算，包括各种创建矩阵的方法，以及一般的矩阵运算、求逆、求转置
- 它的很多底层的函数都是用C写的，可以得到在普通Python中无法达到的运行速度



numpy库

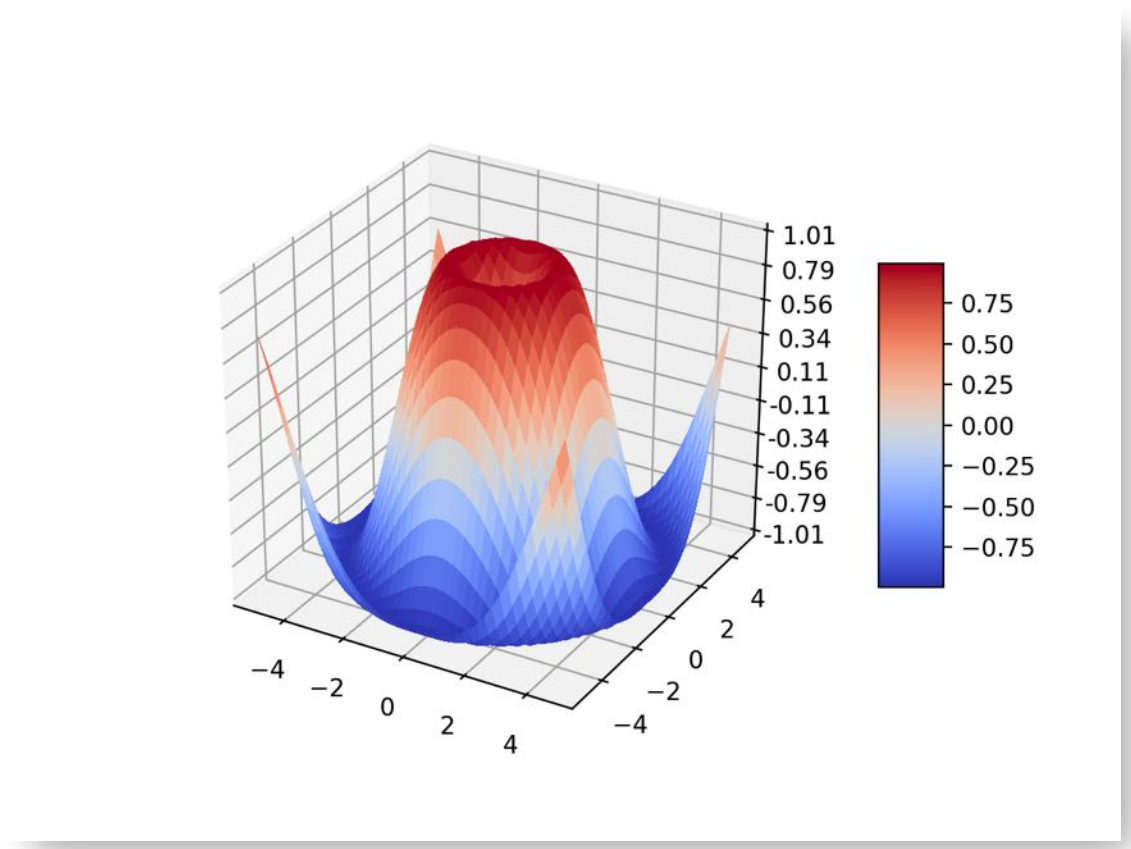
- 矩阵计算
  - 创建矩阵 a = np.matrix([])
  - 矩阵求逆 a.I
  - 矩阵转置 a.T
  - 矩阵乘法 a*b或np.dot(a,b)
- 对象属性
  - np.shape 数组形状，矩阵则为n行m列
  - np.size 对象元素的个数
  - np.dtype 指定当前numpy对象的整体数据

```
>>> import numpy as np
>>> a= np.matrix([[1,2],[3,4]])
>>> a.I
matrix([[-2. ,  1. ],
        [ 1.5, -0.5]])
>>> a.T
matrix([[1, 3],
        [2, 4]])
>>> a.I * a
matrix([[  1.00000000e+00,   0.00000000e+00],
        [  1.11022302e-16,   1.00000000e+00]])
>>> b= np.matrix([[7,6],[5,4]])
>>> a*b
matrix([[17, 14],
        [41, 34]])
```

```
>>> a.shape
(2, 2)
>>> a.size
4
>>> a.dtype
dtype('int64')
```

matplotlib

- matplotlib 是 Python 的一个绘图库。它包含了大量的工具，可以使用这些工具创建各种图形
  - 包括简单的散点图，折线图，甚至是三维图形、动画等，Python 科学计算社区经常使用它完成数据可视化的工作。
- 功能异常强大
  - http://matplotlib.org/gallery.html

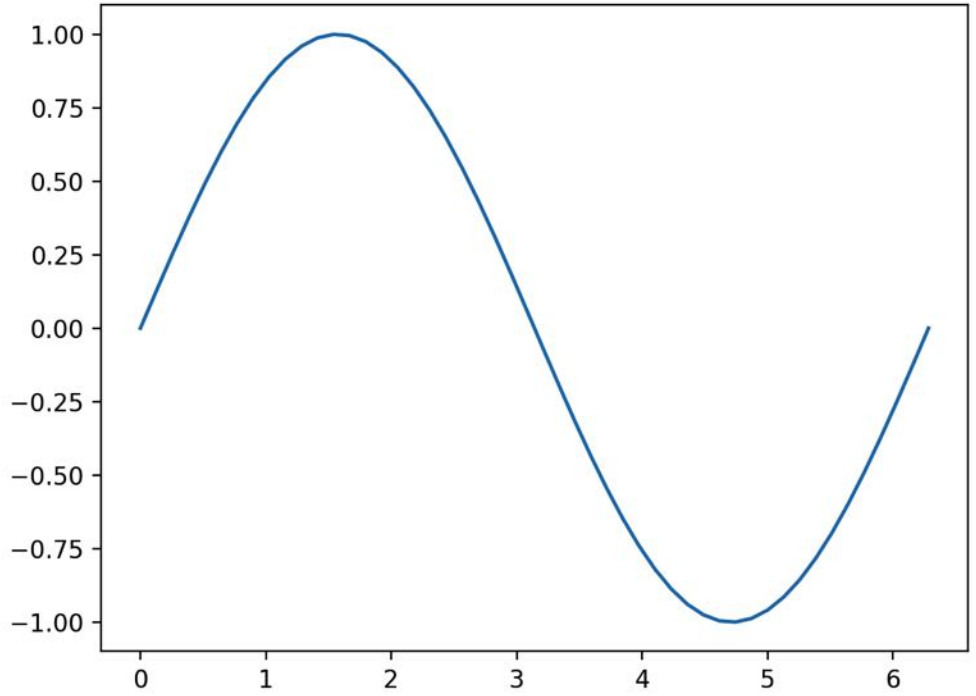


绘制函数图像基本思路

- 基本思路
  - 通过将图像上一些点的坐标连接起来，即可绘制函数的近似图像，当点越多时，所绘图像越接近函数图像
- numpy库的linspace( )函数生成数组
  - numpy.linspace(<start>,<stop>,<num>)
  - 生成一个存放等差数列的数组，数组元素为浮点型，包含三个参数，分别是：数列起始值、终止值（默认包含自身）、数列元素个数
- matplotlib库的plot( )函数用来画图
  - 可以设定图形颜色、线条线型、以及做标注等

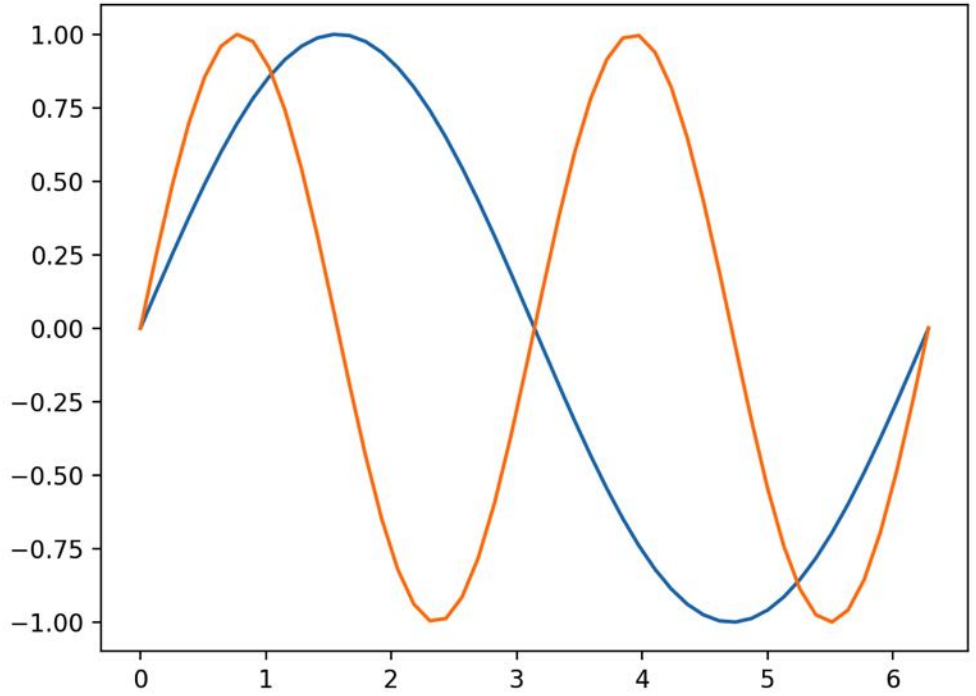
matplotlib：简单图形

```
import matplotlib.pyplot as plt
import numpy as np

# 简单的绘图
x = np.linspace(0, 2 * np.pi, 50)

# 如果没有第一个参数 x，图形的 x 坐标默认为数组的索引
plt.plot(x, np.sin(x))

plt.show() # 显示图形
```



matplotlib：多个简单图形

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 2 * np.pi, 50)
plt.plot(x, np.sin(x),
         x, np.sin(2 * x))
plt.show()
```



定制线型

- plot()函数的绘制样式参数表示
- 颜色

颜色	表示方法	颜色	表示方法
blue	‘b’	yellow	‘y’
cyan	‘c’	white	‘w’
green	‘g’	red	‘r’
black	‘k’	magenta	‘m’

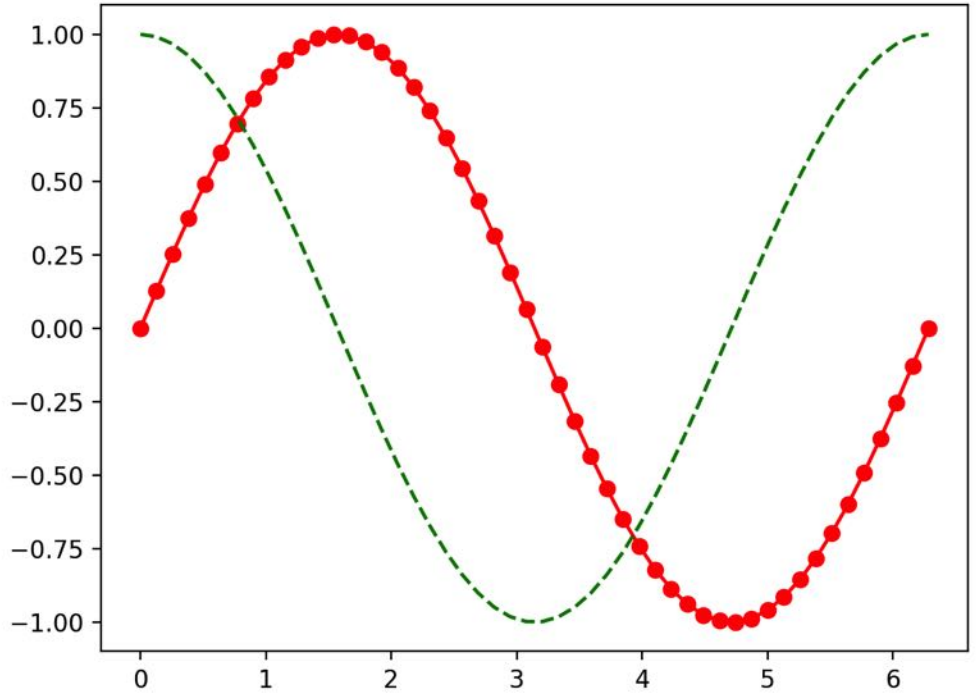
- 线型与点型

线型	表示方法
实线	-
短线	--
短点相间线	-.
虚点线	:

点型	表示方法
圆形	o
叉	x、+
三角形	^、v、<、>
五角星	*

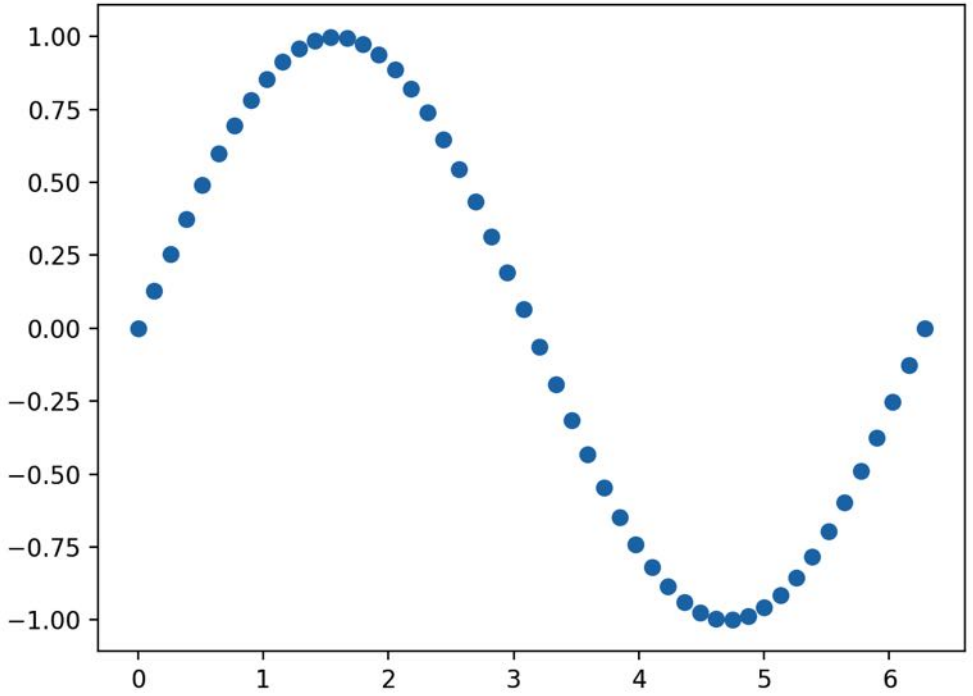
matplotlib：定制线型

```
import matplotlib.pyplot as plt
import numpy as np

# 自定义曲线的外观
x = np.linspace(0, 2 * np.pi, 50)
plt.plot(x, np.sin(x), 'r-o',
         x, np.cos(x), 'g--')
plt.show()
```



matplotlib：散点图

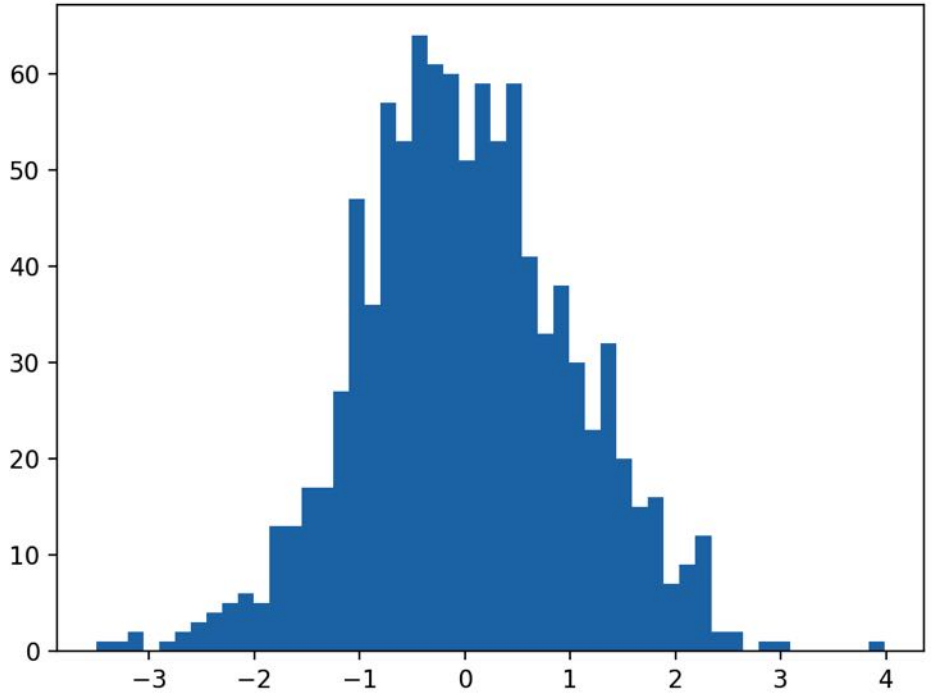
```
import matplotlib.pyplot as plt
import numpy as np

# 简单的散点图
x = np.linspace(0, 2 * np.pi, 50)
y = np.sin(x)
plt.scatter(x,y)
plt.show()
```



直方图

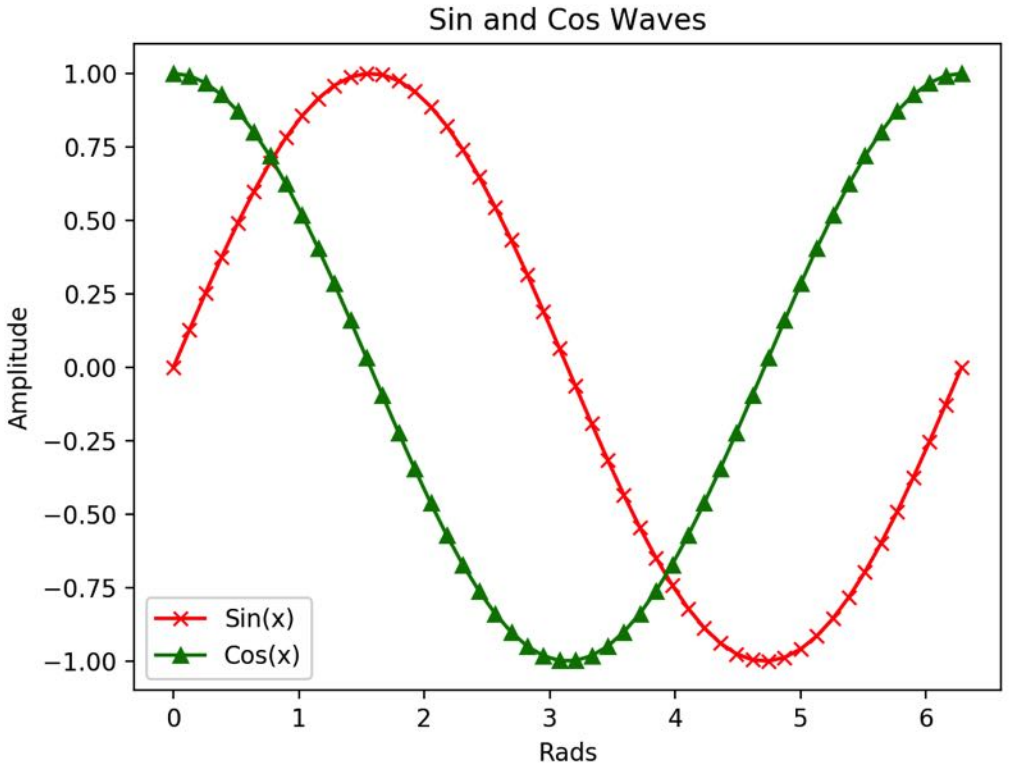
```
import matplotlib.pyplot as plt
import numpy as np

# 直方图
x = np.random.randn(1000)
plt.hist(x,50)
plt.show()
```

正态分布



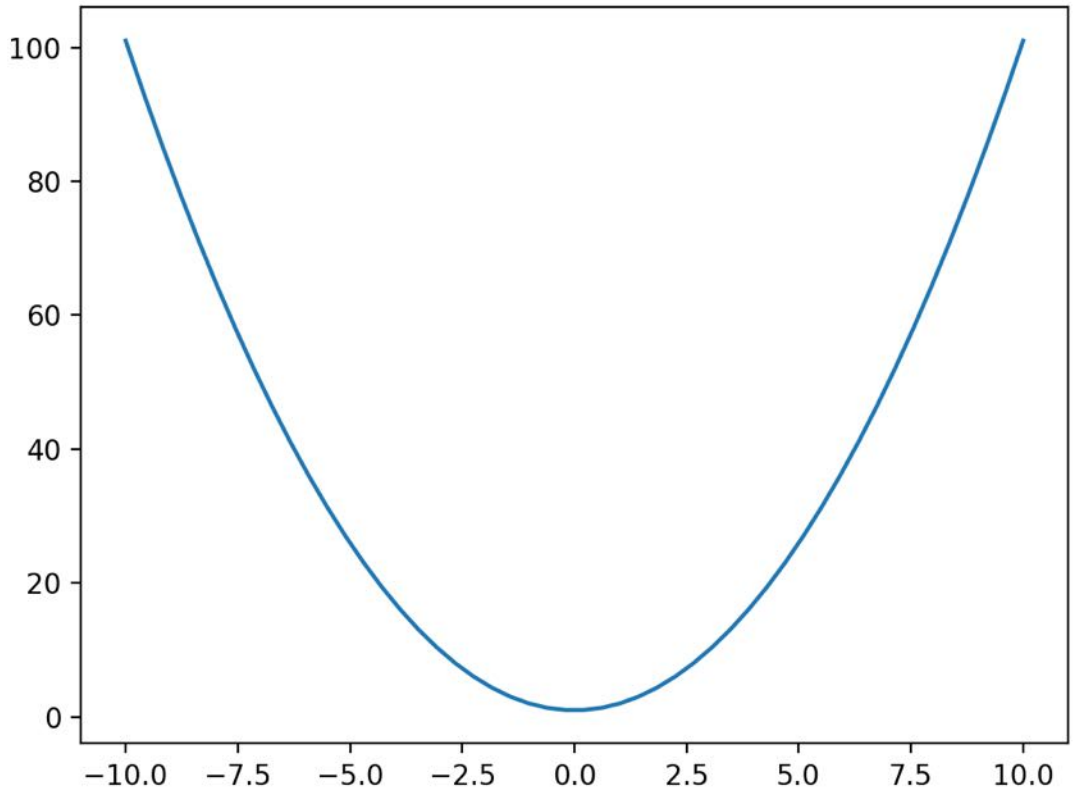
标题，标签和图例

```
import matplotlib.pyplot as plt
import numpy as np

# 添加标题，坐标轴标记和图例
x = np.linspace(0, 2 * np.pi, 50)
plt.plot(x, np.sin(x), 'r-x', label='Sin(x)')
plt.plot(x, np.cos(x), 'g-^', label='Cos(x)')
plt.legend() # 展示图例
plt.xlabel('Rads') # 给 x 轴添加标签
plt.ylabel('Amplitude') # 给 y 轴添加标签
plt.title('Sin and Cos Waves') # 添加图形标题
plt.show()
```



自定义函数

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x = np.linspace(-10, 10, 50)
5
6  def y(x):
7      return x**2+1
8
9  plt.plot(x, list(map(y, x)))
10
11 plt.show() # 显示图形
```



上机练习

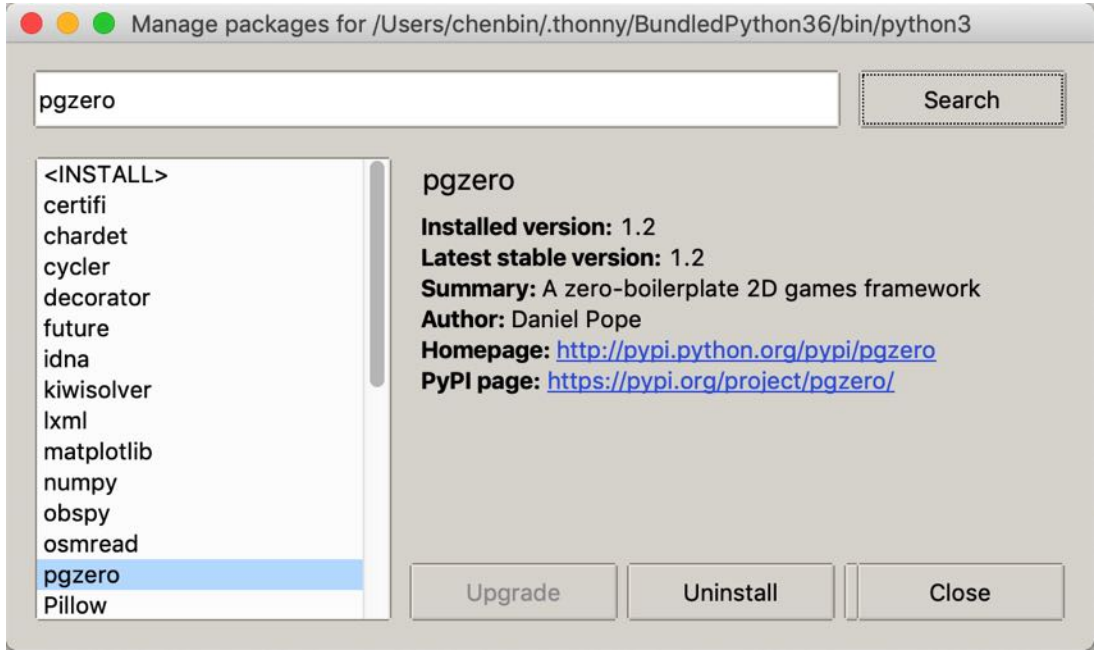
- 用matplotlib绘制一个某课程总评成绩的直方图
  - 在课程网站下载grades.txt
- 用matplotlib绘制一个带标题、标签和图例的组合函数图像(x=-10..10):
  - y=x**2-2*x
  - y=2*x+3
  - y=6*sin(x)

多媒体程序开发pygame zero

- Pygame Zero基础
- 事件驱动式应用
- 设计实现“接宝物”游戏

Pygame Zero安装

- 打开Thonny
- 菜单“工具”->“管理模块”
- 输入“pgzero”
- 点击搜索
- 点击“install”
- 安装成功！



创建第一个窗口！

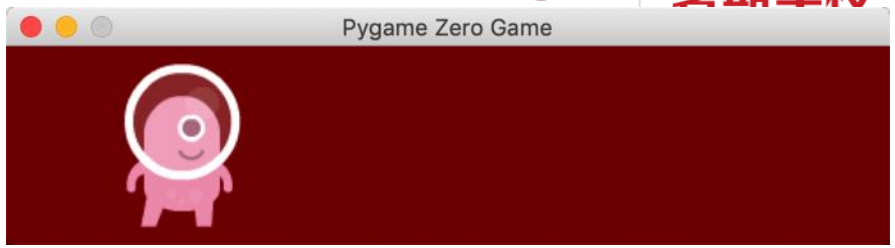
- 图形化的游戏都有一个窗口
- 我们来写程序创建一个300*300的正方形红色窗口
  - 注意line1和15是必须的
- 注意！打开的窗口要按组合键CTRL-Q才能关闭
  - L6/7指定窗口的大小
  - L11/12是在刷新窗口的时候设置红色填充
- 每秒要调用draw函数60次！


```
1   import pgzrun
2
3   # 绘制窗口背景为暗红色
4
5   # 窗口的宽和高设置
6   WIDTH = 300
7   HEIGHT = 300
8
9
10  # 每次需要刷新窗口的时候，会自动调用draw函数
11  def draw():
12      screen.fill((128, 0, 0))
13
14
15  pgzrun.go()
```

让游戏动起来

- 游戏中的图形分为：
  - 不动、或者虽然活动但没有影响的背景
  - 运动、或者虽然不动但会互相碰撞、互相作用的角色
  - 经常更新的各种信息显示
- 要让图形动起来，就要不停地做两件事
  - 更新：背景数据、角色位置
  - 重画：根据新的数据和位置重新绘制画面

创建游戏角色



- 在源代码文件目录下新建两个目录
  - images：存放图片
  - sounds：存放声音
- 用Actor类创建一个精灵对象
  - 指定图片、位置
- 在draw中画出精灵
  - 精灵图片将出现在指定的位置
  - alien.pos

images/alien.png

```
1   import pgzrun
2
3   # 绘制一个精灵
4
5   # 1，创建一个精灵
6   alien = Actor('alien')
7   alien.pos = 100, 56
8
9   # 2，设定窗口大小
10  WIDTH = 500
11  HEIGHT = alien.height + 20
12
13
14  # 3，每次需要刷新窗口的时候，会自动调用draw函数
15  def draw():
16      screen.clear()
17      screen.fill((128,0,0))
18      alien.draw()
19
20
21  pgzrun.go()
```

让精灵动起来

- 首先创建精灵alien
- 然后定义每帧的刷新需要做的两件事情
  - 记住每秒要刷新60次！
- 更新update
  - 改变精灵的位置，每秒会改变60次！
- 绘制draw
  - 填充窗口和画出精灵

```
1   import pgzrun
2
3   # 让精灵运动起来
4   # 1，创建一个精灵
5   alien = Actor('alien')
6   alien.topright = 0, 10
7
8   # 2，设定窗口大小
9   WIDTH = 500
10  HEIGHT = alien.height + 20
11
12  # 3，每次需要刷新窗口的时候，会自动调用draw函数
13  def draw():
14      screen.clear()
15      screen.fill((128, 0, 0))
16      alien.draw()
17
18  # 4，每一帧都会自动调用update函数
19  def update():
20      alien.left += 2
21      if alien.left > WIDTH:
22          alien.left = 0
23
24
25  pgzrun.go()
```

与精灵交互

用鼠标点击

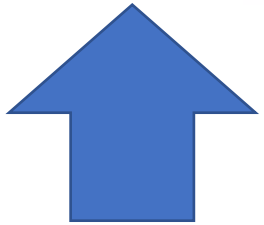
pos就是鼠标的位置

```
24  # 5, 当鼠标点击当时候调用这个函数
25  def on_mouse_down(pos):
26      if alien.collidepoint(pos):
27          print("Ekk")
28      else:
29          print("You missed me!")
```

还是用图形和声音来响应更好

- 当鼠标点击时
  - 测试鼠标位置是否与外星人角色重叠
  - 播放音效
  - 外星人形象切换为hurt
- 有个小毛病
  - 切换是一次性的
  - 没有恢复成正常的alien
- 修改方案让外星人1秒后恢复?
  - sleep?

```
27  # 5，当鼠标点击当时候调用这个函数
28  # 切换外星人形象，播放音效
29  def on_mouse_down(pos):
30      if alien.collidepoint(pos):
31          sounds.eep.play()
32          alien.image = 'alien_hurt'
```



images/alien.png
images/alien_hurt.png
sounds/eep.wav

定时器clock

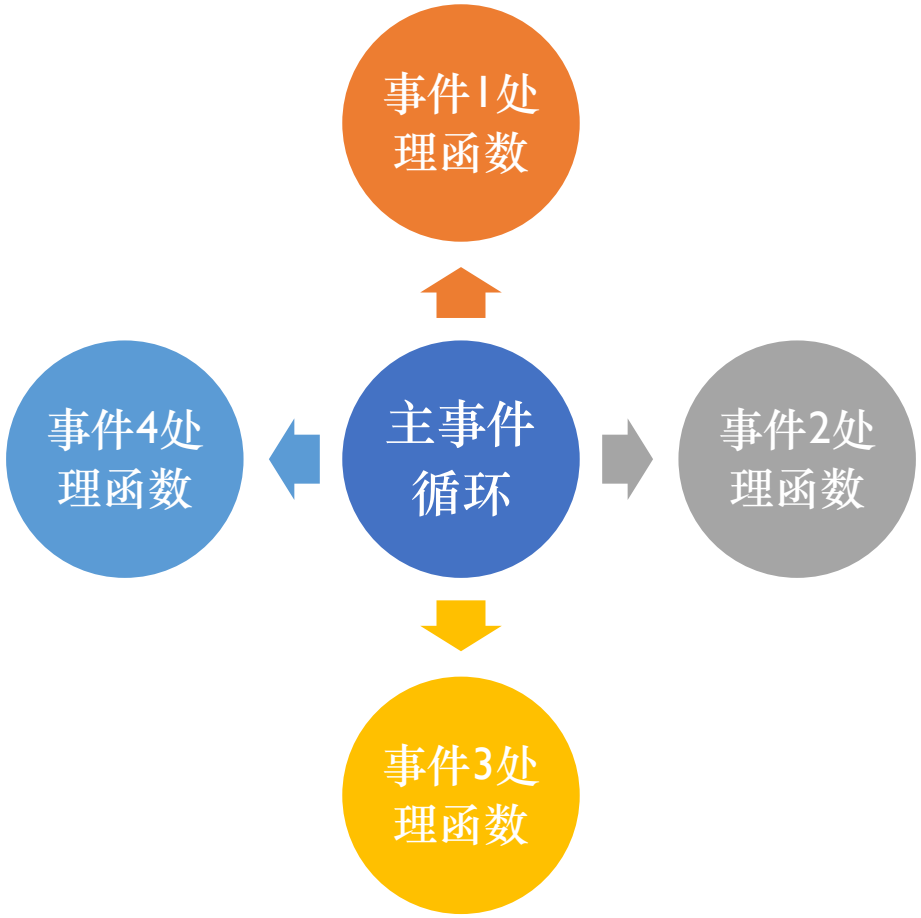
- 定时器clock可以安排在一段时间后自动调用某个函数
  - clock.schedule_unique()
- 将外星人受伤和恢复变为两个函数
  - set_alien_hurt
  - set_alien_normal
- 鼠标点击时调用受伤函数
- 在受伤函数里安排1s后恢复

```
26  # 5，当鼠标点击当时候调用这个函数
27  # 切换外星人形象，播放音效
28  def on_mouse_down(pos):
29      if alien.collidepoint(pos):
30          set_alien_hurt()
31
32  # 6，受伤外星人
33  def set_alien_hurt():
34      alien.image = 'alien_hurt'
35      sounds.eep.play()
36      # 设置定时器，1s后自动恢复
37      clock.schedule_unique(set_alien_normal, 1.0)
38
39  # 7，外星人恢复
40  def set_alien_normal():
41      alien.image = 'alien'
```

大功告成！

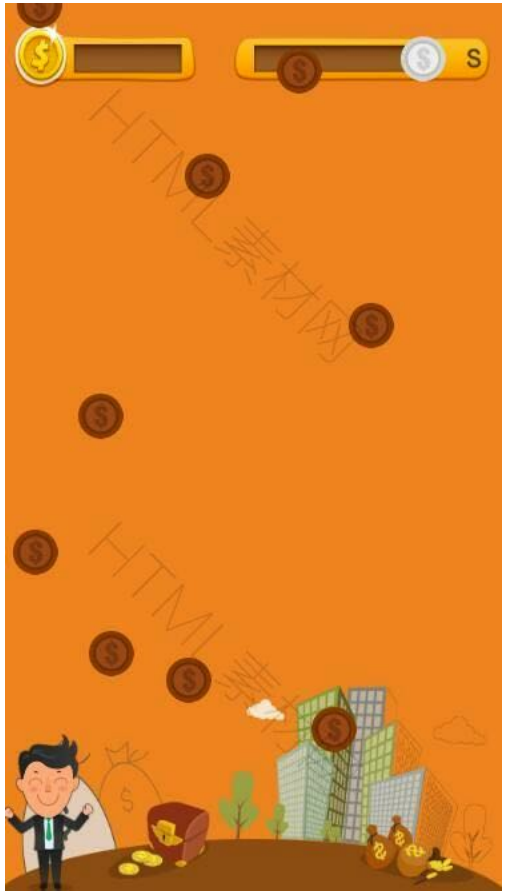
- 设定窗口大小WIDTH/HEIGHT
- 创建一批角色Actor
- 写好每次刷新都要做的事情
  - 更新：修改背景和角色的数据
  - 重画：画一遍窗口的内容
- 写好交互的动作
  - 鼠标：on_mouse_down(pos)
  - 键盘：on_key_down(key)

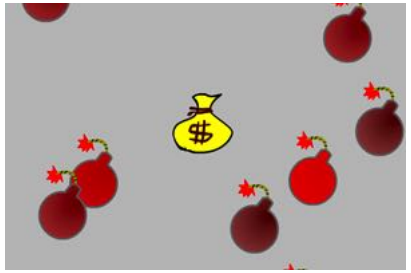
顺序过程程序和事件驱动程序



写一个接宝物的游戏

- 角色确定
  - 哪几种类型角色?
  - 角色的运动? 控制?
  - 角色之间的碰撞?
- 画面设计
  - 背景画面、显示信息
  - 角色的绘制位置
- 交互设计
  - 鼠标控制?



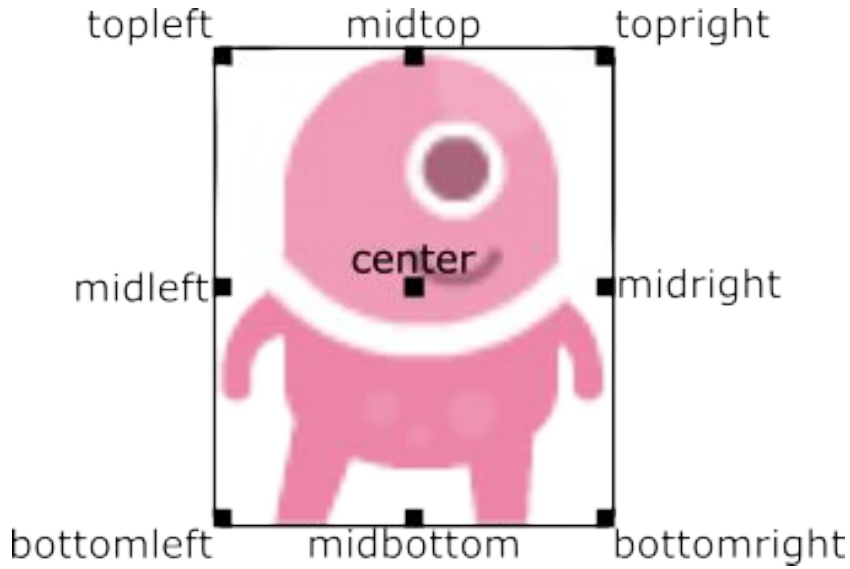


一些进一步的参考

- 颜色可以是(r,g,b)也可以是字符串名称
  - (128,0,0)或者'red'
- 窗口写字：颜色可以是(r,g,b)也可以是字符串名称
  - screen.draw.text(str, (x,y), color=颜色, background=颜色)
- 绘制背景图像
  - screen.clear()
  - screen.fill(颜色)
  - screen.blit(背景图名, (x,y))
- 音效sounds
  - sounds.<名称>.play(loops=<重复次数>)
  - sounds.<名称>.stop()

一些进一步的参考

- 定时器clock
  - 安排时间：clock.schedule(<函数>, <延迟时间>)
  - 会覆盖以前的安排：clock.schedule_unique(<函数>, <延迟时间>)
  - 定期反复调用：clock.schedule_interval(<函数>, <重复时间>)
  - 取消安排：clock.unschedule(<函数>)
- 角色Actor
  - 位置：center=(x,y)
  - 角度旋转：angle=角度值
  - 相对距离：distance_to(<另一个角色>/(x,y))
  - 相对角度：angle_to(<另一个角色>/(x,y))



一些进一步的参考

- 角色Actor
  - image=图片
- Actor碰撞检测
  - collidepoint(<另一个角色>/(x,y))
  - colliderect(<另一个角色>/<矩形Rect()>/((x1,y1),(x2,y2)))
- 动画效果
  - animate(<角色>, pos=(x,y))
- 合成的音符
  - tone.play('E4', 0.5)
  - 创建一个，后续调用play： beep = tone.create('A3', 0.5)
  - beep.play()

一些进一步的参考

- 鼠标事件
  - 鼠标左右键：mouse.LEFT, mouse.RIGHT, mouse.MIDDLE
  - on_mouse_move(pos, rel, buttons)
    - if mouse.LEFT in buttons:
  - on_mouse_down(pos, button)
- 键盘事件
  - on_key_down(key)
  - 键名称：key.A, key.B, key.ESC…

接宝物游戏程序分析

```
1   import pgzrun
2   import random
3
4   # 捡宝物的小游戏
5   # 宝物角色
6   # 玩家角色（5种表情）
7   # 背景：草原（1000*606像素）
8   # 1，创建一系列精灵
9   # 名称
10  names = ["peach", "cherryf", "apple", "orange", "lemon", "grape", "banana",
11           "egg", "donut", "taco", "pizza", "burger", "fries", "leg", "dango",
12           "cake", "cookie", "ice", "candy",
13           "cherry", "bouquet", "leaf4", "sunflower", "tulip", "rose",
14           "tiger", "cat", "bear"]
15
16  # 保存正在下落的宝物
17  things = []
18
19  # 玩家、位置和得分
20  player = Actor("face_sweat")
21  player.center = 500, 580
22  player.score = 0
23
24  # 2，设定窗口
25  WIDTH = 1000
26  HEIGHT = 606
```

接宝物游戏程序分析

```
29      # 3，每次需要刷新窗口的时候，会自动调用draw函数
30      def draw():
31          # 清除窗口，设置背景
32          screen.clear()
33          screen.blit("backimg", (0, 0))
34
35          # 画上宝物和玩家
36          for t in things:
37              t.draw()
38          player.draw()
39
40          # 画上分数
41          screen.draw.text("SCORE:%d" % player.score, (400, 10))
```

接宝物游戏程序分析

```
44  # 4，每一帧都会自动调用update函数
45  def update():
46      # 平均每秒随机生成一个宝物加入到things列表里
47      if random.randrange(60) == 0:
48          # 随机生成宝物，和随机的顶部位置
49          t = Actor(random.choice(names))
50          t.center = random.randrange(1000), 0
51          things.append(t)
52      # 每个宝物下降，判断是不是碰到玩家
53      for t in things:
54          # 下降4个像素，可以调节速度
55          t.y += 4
56          # 如果调出底线了，就删除，玩家扣分
57          if t.y >= 606:
58              things.remove(t)
59              player.score -= 4
60              # 玩家换一个表情
61              set_player_sad()
62          # 判断是否被玩家接到了
63          elif t.colliderect(player):
64              # 碰到玩家了，被吃掉，玩家加分
65              things.remove(t)
66              player.score += 1
67              # 玩家换一个表情
68              set_player_happy()
```

接宝物游戏程序分析

```
71  # 5，鼠标移动玩家
72  def on_mouse_move(pos):
73      player.x = pos[0]
74
75
76  # 换成欢乐的表情
77  def set_player_happy():
78      player.image = "face_cool"
79      sounds.exp.play()
80      # 设置定时器，1秒后恢复
81      clock.schedule_unique(set_player_normal, 1.0)
82
83  # 换成悲伤表情
84  def set_player_sad():
85      player.image="face_cry"
86      sounds.blip.play()
87      # 设置定时器，1秒后恢复
88      clock.schedule_unique(set_player_normal, 1.0)
89
90  # 换回正常表情
91  def set_player_normal():
92      player.image = "face_sweat"
93
94
95  pgzrun.go()
```

- 改进：随分数改变速度?
- 加退出按钮?
  - game.exit()
- 各种物体分数不一样?

上机练习

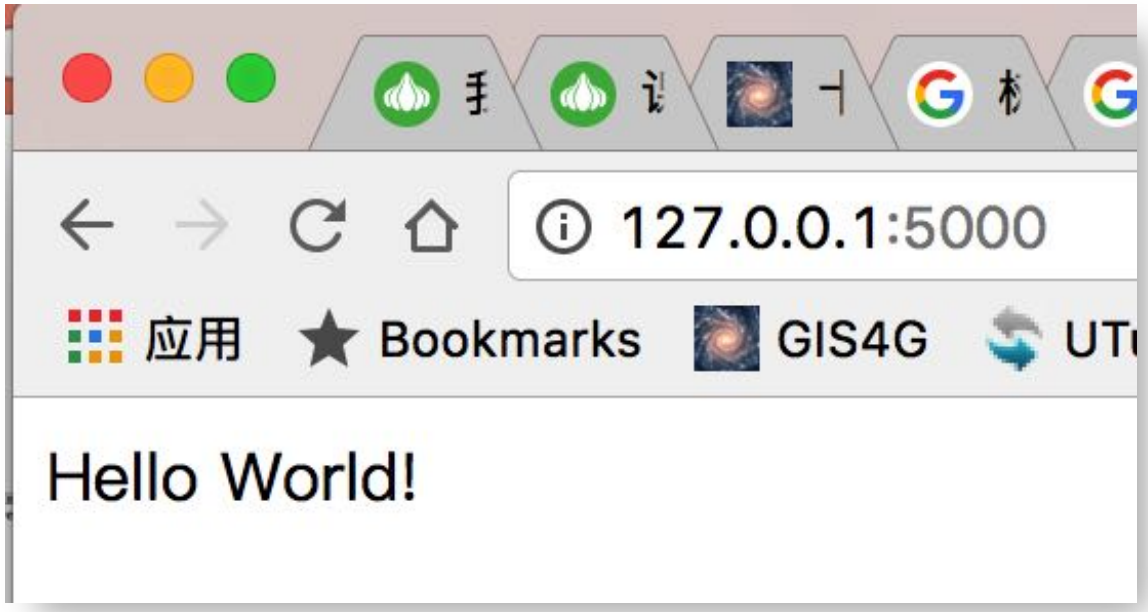
- 从接宝物游戏出发，修改为宝物射击游戏
  - 点击鼠标发射小球设计下落的宝物
  - 躲宝物，以免碰撞
  - 计分
  - 随分数升高，越来越快

Flask

- Web应用已经成为目前最热门的应用软件形式
- Web应用通过Web服务器提供服务，客户端采用浏览器或者遵循HTTP协议的客户端
- 由于需要处理HTTP传输协议，很多web开发框架涌现
- flask是一种非常容易上手的Python web开发框架，只需要具备基本的python开发技能，就可以开发出一个web应用来

Flask小例子

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

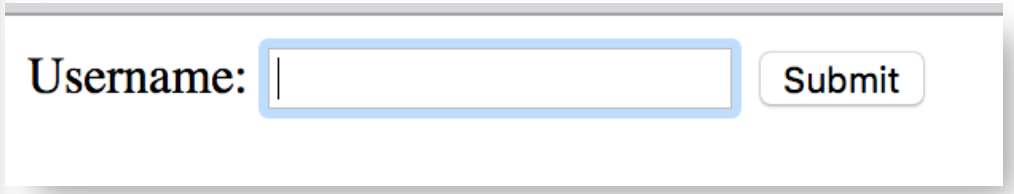



```
============= RESTART: /Users/chenbin/Documents/homework/flsk.py ==========
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [31/Mar/2017 02:47:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Mar/2017 02:48:00] "GET /favicon.ico HTTP/1.1" 404 -
```

更复杂一些的例子：表单插件Flask-WTF

```python
from flask_wtf import Form
from wtforms import StringField
from wtforms.validators import DataRequired

class MyForm(Form):
    user = StringField('Username', validators=[DataRequired()])

from flask import Flask, render_template

app = Flask(__name__)
app.secret_key = '1234567'

@app.route('/login', methods=('GET', 'POST'))
def login():
    form = MyForm()
    if form.validate_on_submit():
        # if form.user.data == 'admin':
        if form.data['user'] == 'admin':
            return 'Admin login successfully!'
        else:
            return 'Wrong user!'
    return render_template('login.html', form=form)

if __name__ == "__main__":
    app.run()
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>登录表单</title>
</head>
<body>
<form method="POST" action="{{ url_for('login') }}">
    {{ form.hidden_tag() }}
    {{ form.user.label }}: {{ form.user(size=20) }}
    <input type="submit" value="Submit">
</form>
</body>
</html>
```



网络爬虫

- 爬虫是按照一定规则，自动地提取并保存网页中信息的程序
  - 蜘蛛沿着网络抓取猎物
  - 通过一个节点之后，顺着该节点的连线继续爬行到下一个节点，最终爬完整个网络的全部节点
- 通过向网站发起请求获取资源，提取其中有用的信息

requests库

- Python实现的一个简单易用的HTTP库
  - 支持HTTP持久连接和连接池、SSL证书验证、cookies处理、流式上传等
- 向服务器发起请求并获取响应，完成访问网页的步骤

〉简洁、容易理解，是最友好的网络爬虫库

- http请求类型
  - requests.request()：构造一个请求
  - requests.get()：获取HTML网页
  - requests.head()：获取HTML网页头信息
  - requests.post()：提交POST请求
  - requests.put()：提交PUT请求
  - requests.patch()：提交局部修改请求
  - requests.delete()：提交删除请求
  - requests.options()：获取http请求

〉返回的是一个response对象

requests库

- response对象
  - 包含服务器返回的所有信息，例如状态码、编码形式、文本内容等；也包含请求的request信息
  - .status_code：HTTP请求的返回状态
  - .text：HTTP响应内容的字符串形式
  - .content：HTTP响应内容的二进制形式
  - .encoding：(从HTTP header中)分析响应内容的编码方式
  - .apparent_encoding：(从内容中)分析响应内容的编码方式

requests库

```
Python 3.6.6
>>> import requests as rq
>>> r=rq.get("http://www.pku.edu.cn")
>>> r.status_code
200

>>> r.text[:200]
'<!-- ONLINE: 2015-05-04 -->\n<!doctype html>\n<html>\n<head>\n<meta charset="utf-8">\n<me
evice-width, initial-scale=1.0,minimum-scale=1.0"/>\n<meta name="apple-mobile-web-'

>>> r.encoding
'UTF-8'

>>> r.apparent_encoding
'utf-8'

>>> r.content[:200]
b'<!-- ONLINE: 2015-05-04 -->\n<!doctype html>\n<html>\n<head>\n<meta charset="utf-8">\n<m
device-width, initial-scale=1.0,minimum-scale=1.0"/>\n<meta name="apple-mobile-web-'
```

requests库

- 定制请求头
  - requests的请求接口有一个名为headers的参数，向它传递一个字典来完成请求头定制
- 设置代理
  - 一些网站设置了同一IP访问次数的限制，可以在发送请求时指定proxies参数来替换代理，解决这一问题

```
proxies = {
            "http": "http://10.10.10.10:1010",
            "https": "http://10.11.10.14:1011",
          }
r = requests.get(url, proxies = proxies)
```

Beautiful Soup

- 页面解析器
  - 使用requests库下载了网页并转换成字符串后，需要一个解析器来处理HTML和XML，解析页面格式，提取有用的信息

〉解析器类型

解析器	使用方法	优势
python标准库	BeautifulSoup(markup,“html.parser”)	- Python的内置标准库 - 文档容错能力强
lxml HTML解析器	BeautifulSoup(markup,“lxml”)	- 速度快 - 文档容错能力强
lxml XML解析器	BeautifulSoup(markup, [“lxml-xml”]) BeautifulSoup(markup,“xml”)	- 速度快 - 唯一支持XML的解析器
Html5lib	BeautifulSoup(markup,“html5lib”)	- 最好的容错性 - 以浏览器的方式解析文档 - 生成HTML5格式的文档

Beautiful Soup

- 搜索方法
  - find_all(name, attrs, recursive, string, **kwargs)
  - 返回文档中符合条件的所有tag，是一个列表
  - find(name, attrs, recursive, string, **kwargs)
  - 相当于find_all()中limit = 1，返回一个结果
  - name：对标签名称的检索字符串
  - attrs: 对标签属性值的检索字符串
  - recursive: 是否对子节点全部检索，默认为True
  - string: <>...</> 中检索字符串
  - **kwargs：关键词参数列表

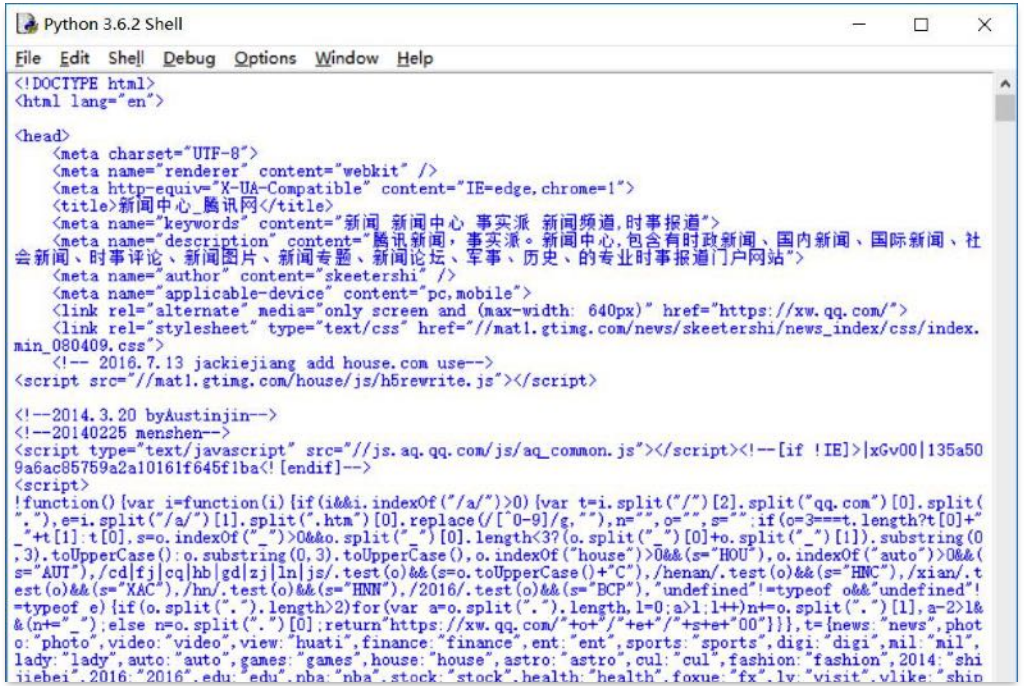
爬虫的基本流程

- 分析网页结构

爬虫的基本流程

- 爬取页面
  - 通过requests库向目标站点发送请求，若对方服务器正常响应，能够收到一个response对象，它包含了服务器返回的所有信息

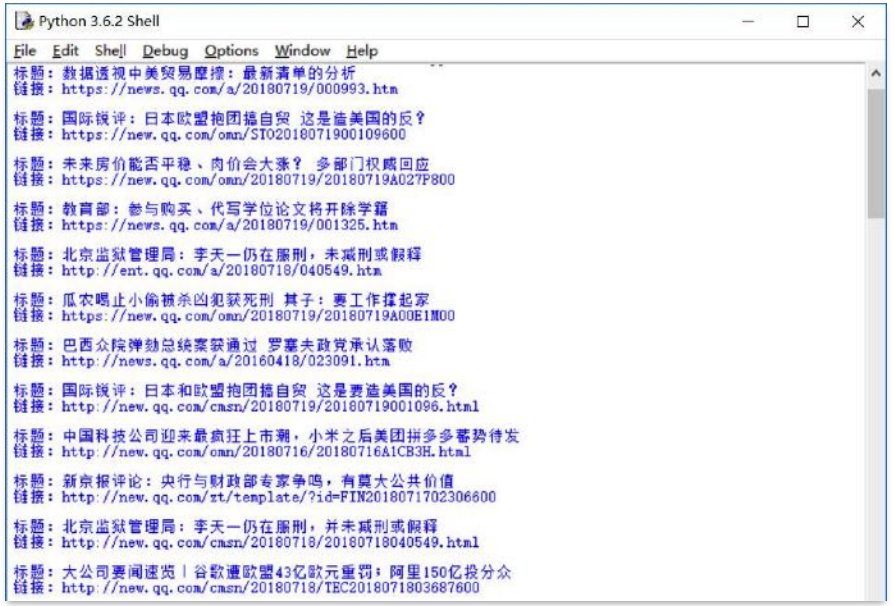
```
import requests
url = "http://news.qq.com/"
r = requests.get(url, timeout = 30)
print(r.text)
```



爬虫的基本流程

- 解析页面
  - HTML代码-网页解析器
  - Json数据-json模块，转换成Json对象
  - 二进制数据-以wb形式写入文件，再做进一步处理
  - 此处使用bs4进行解析

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(r.text, 'lxml')
for news in soup.find_all('div', class_ = 'text'):
    info = news.find('a')
    if len(info) > 0:
        title = info.get_text()
        link = str(info.get('href'))
        print('标题: ' + title)
        print('链接: ' + link + '\n')
```



上机练习

- 编写一个flask服务器，保存两个网页，URL分别是/, /city
  - 在／下面的网页带链接：
  - 城市
- 利用爬虫库和解析库，从新闻网站抓回所有新闻标题和URL链接
  - http://sports.qq.com/
  - https://news.163.com/