合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

Python
数据分析及可视化-I

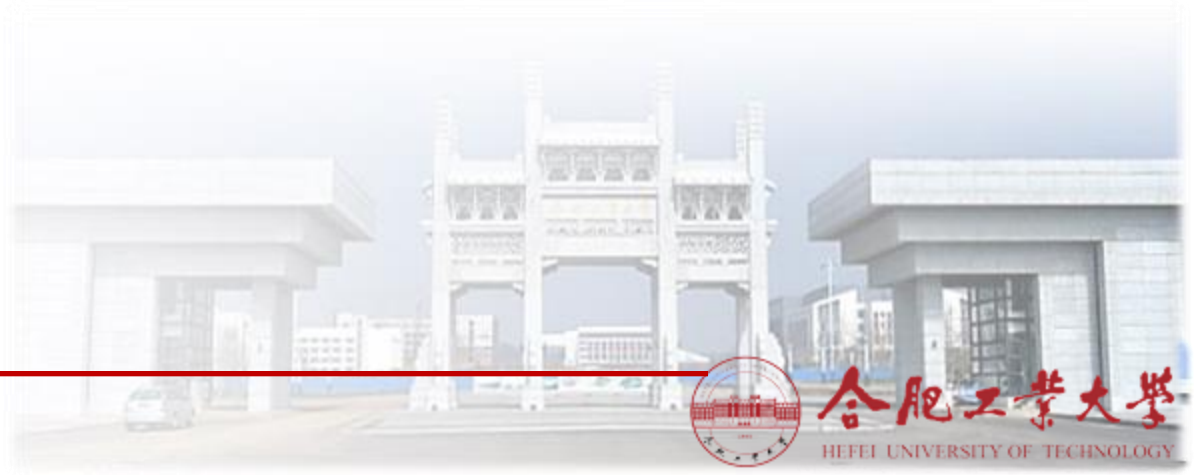
认识Python

HFUT计算机基础

2021.3

本章学习目标

- 了解Python数据对象
- 了解列表、元组、字典、集合、字符串的基本用法
- 理解加法运算符+对列表、元组、字符的连接作用
- 理解列表、元组、字符串的大小比较的原理
- 理解集合运算的原理
- 了解Python交互式编程的特点



数据对象

- 对象是python语言中最基本的概念，在python中处理的一切都是对象，对象具有属性、方法，对象的集合构成类，类又具有继承的特点。
- python中有许多内置对象可供编程者使用，内置对象可直接使用，如数字、字符串、列表、字典等。
- 非内置对象需要导入模块才能使用，如正弦函数sin(x)，随机数产生函数random( )等。

数据对象

对象类型	类型名称	示例	简要说明
数字	int、float、complex	88888888888888888888 9.8，3.14，6.626e-34，5+6j，5j	数字大小没有限制，且内置支持复数及其运算
字符串	str	'Readability counts.' "I'm a Python teacher." '''Tom sai, "let's go."''' r'C:\Windows\notepad.exe'	使用单引号、双引号、三引号作为定界符，不同定界符之间可以互相嵌套；前面加字母r或R表示原始字符串，任何字符都不进行转义
字节串	bytes	b'hello world'	以字母b引导
列表	list	[79, 89, 99] ['a',{3},(1,2),['c',2],{65:'A'}]	所有元素放在一对方括号中，元素之间使用逗号分隔，其中的元素可以是任意类型
元组	tuple	(1, 0, 0)、(0,)	所有元素放在一对圆括号中，元素之间使用逗号分隔，元组中只有一个元素时后面的逗号不能省略

数据对象

对象类型	类型名称	示例	简要说明
字典	dict	{'red': (1,0,0), 'green':(0,1,0), 'blue':(0,0,1)}	所有元素放在一对大括号中，元素之间使用逗号分隔，元素形式为“键:值”，其中“键”不允许重复并且必须为不可变类型，“值”可以是任意类型的数据
集合	set	{'bread', 'beer', 'orange'}	所有元素放在一对大括号中，元素之间使用逗号分隔，元素不允许重复且必须为不可变类型
布尔型	bool	True, False	逻辑值，首字母必须大写
空类型	NoneType	None	空值，首字母必须大写

数据对象

对象类型	类型名称	示例	简要说明
异常	NameError、ValueError、TypeError、KeyError.....		Python内置异常类
文件		f = open('test.txt', 'w', encoding='utf8')	Python内置函数open()使用指定的模式打开文件，返回文件对象
其他可迭代对象		生成器对象、range对象、zip对象、enumerate对象、map对象、filter对象等等	具有惰性求值的特点，空间占用小，适合大数据处理
编程单元		函数（使用def定义） 类（使用class定义） 模块（类型为module）	类和函数都属于可调用对象，模块用来集中存放函数、类、常量或其他对象

数据对象

数字类型

Python语言包括三种数字类型

- 整数类型
  - 1010，99，-217
  - 0x9a，-0X89　（0x，0X开头表示16进制数）
  - 0b010，-0B101　（0b，0B开头表示2进制数）
  - 0o123，-00456　（0o，00开头表示8进制数）

Python中的整数与数学中的整数概念一致，没有取值范围限制。
例如：pow(x，y)函数：计算x^y

```
>>> print(pow(2,1000))
>>> print(pow(2, pow(2,15)))
```

数字类型

浮点数类型

- 0.0, -77., -2.17
- 带有小数点及小数的数字，python语言中浮点数的数值范围存在限制，小数精度也存在限制。这种限制与在不同计算机系统有关96e4, 4.3e-3, 9.6E5 （科学计数法）
- 科学计数法使用字母“e”或者“E”作为幂的符号，以10为基数。科学计数法含义如下：
  <a>e<b> = a * 10b

```
>>> import sys
>>> sys.float_info
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308,
min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53,
epsilon=2.220446049250313e-16, radix=2, rounds=1)
```

数字类型

复数类型

- 12.3+4j,　　-5.6+7j
- 与数学中的复数概念一致，z = a + bj，　a是实数部分，b是虚数部分，a和b都是浮点类型，虚数部分用j或者J标识
- z = 1.23e-4+5.6e+89j（实部和虚部是什么？）
- 对于复数z，可以用z.real获得实数部分，z.imag获得虚数部分
- z.real = 0.000123　　z.imag = 5.6e+89

数字类型扩展

三种类型存在一种逐渐“扩展”的关系：　整数 -> 浮点数 -> 复数

（整数是浮点数特例，浮点数是复数特例）

不同数字类型之间可以进行混合运算，运算后生成结果为最宽类型。（整数 + 浮点数 = 浮点数）

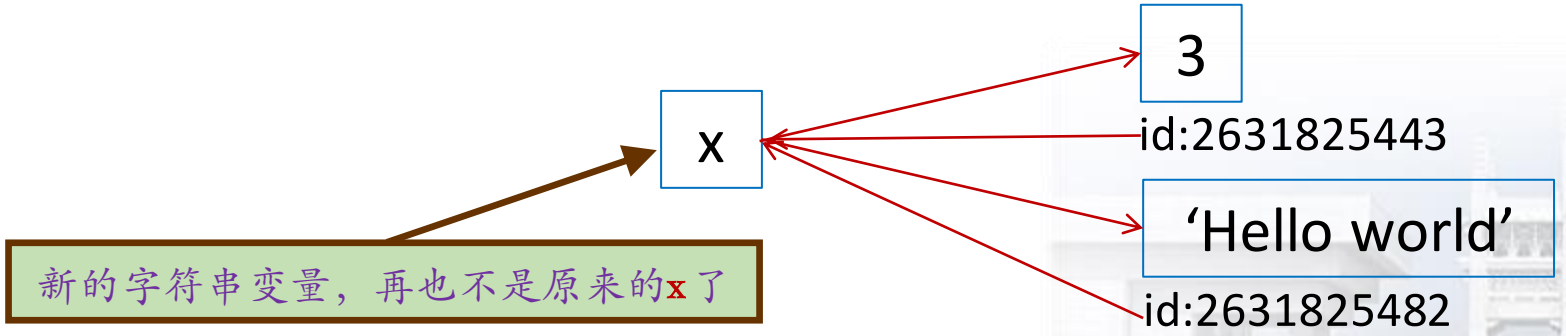
123 + 4.0 = 127.0

常量与变量

◆在Python中，不需要事先声明变量名及其类型，直接赋值即可创建各种类型的对象变量。这一点适用于Python任意类型的对象。

```
>>> x = 3                          #创建了整型变量x，并赋值为3
>>> x = 'Hello world.'             #创建了字符串变量x,并赋值为'Hello world.'
```

◆赋值语句的执行过程是：首先把等号右侧表达式的值计算出来，然后在内存中寻找一个位置把值存放进去，最后创建变量并指向这个内存地址。



◆Python中的变量并不直接存储值，而是引用了值的内存地址，这也是变量类型随时可以改变的原因。

数据对象

常量与变量

- Python属于强类型编程语言，解释器会根据赋值或运算来自动推断变量类型。
- Python还是一种动态类型语言，变量的类型也是可以随时变化的。

```
>>> x = 3
>>> print(type(x))
<class 'int'>
>>> x = 'Hello world.'
>>> print(type(x))                    #查看变量类型
<class 'str'>
>>> x = [1,2,3]
>>> print(type(x))
<class 'list'>
>>> isinstance(3, int)                #测试对象是否是某个类型的实例
True
>>> isinstance('Hello world', int)
False
```

组合数据

计算机不仅对单个变量表示的数据进行处理，更多情况，计算机需要对一组数据进行批量处理。一些例子包括：

◆给定一组单词{python, data, function, list, loop}，计算并输出每个单词的长度；

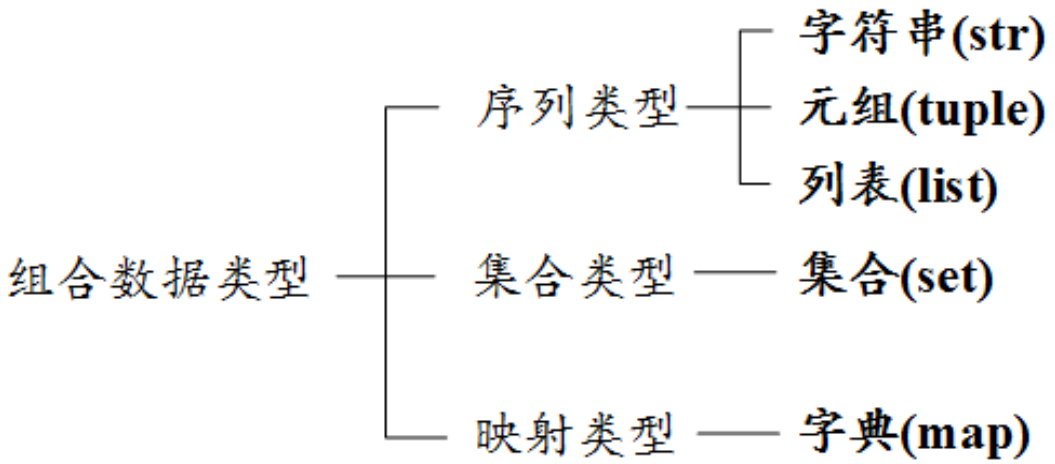
◆给定一个学院学生信息，统计一下男女生比例；

◆一次实验产生了很多组数据，对这些大量数据进行分析；

组合数据

组合数据类型能够将多个同类型或不同类型的数据组织起来，通过单一的表示使数据操作更有序更容易。根据数据之间的关系，组合数据类型可以分为三类：

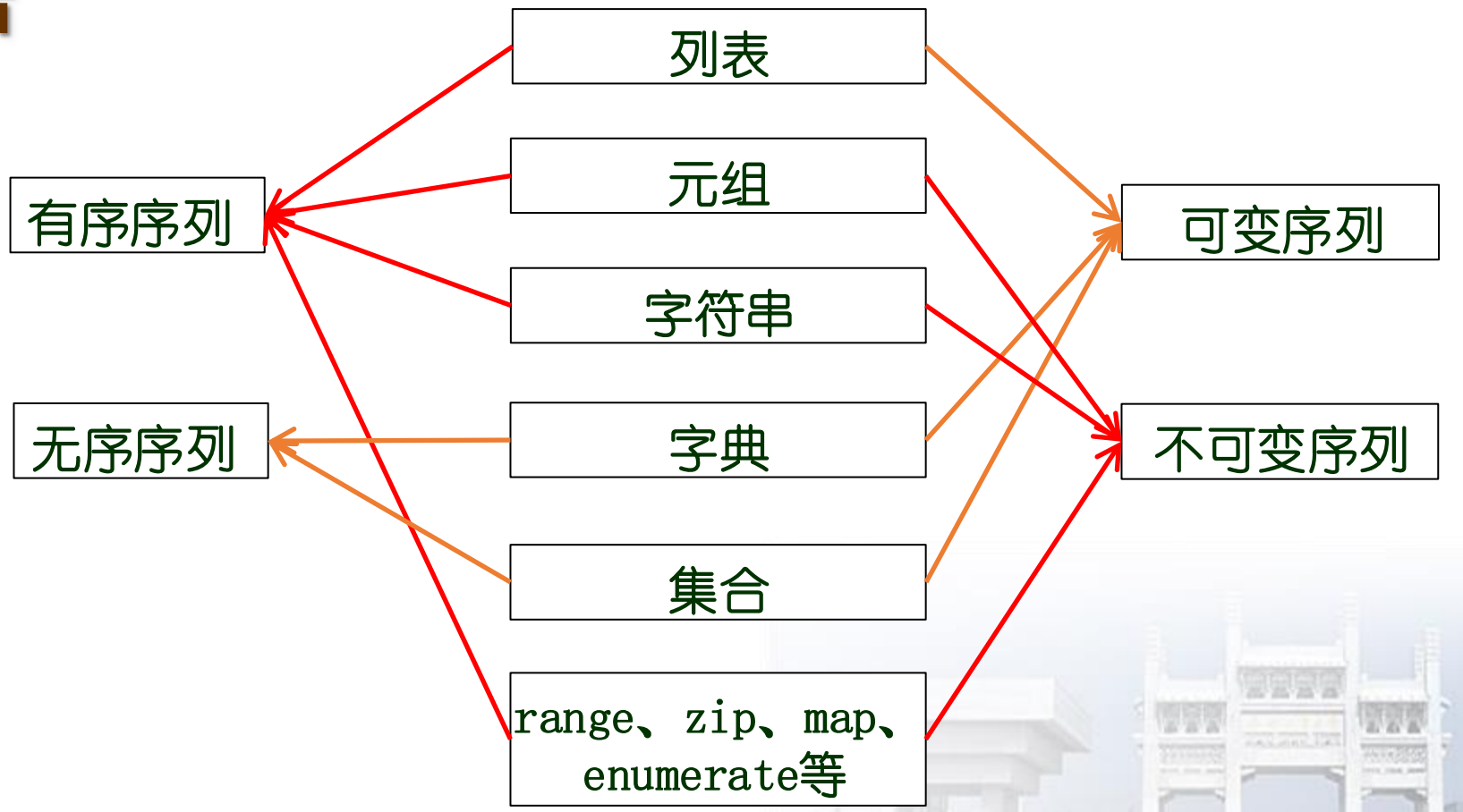
序列类型、集合类型和映射类型。



- 序列类型是一个元素向量，元素之间存在先后关系，通过序号访问，元素之间不排他。
- 集合类型是一个元素集合，元素之间无序，相同元素在集合中唯一存在。
- 映射类型是“键-值”数据项的组合，每个元素是一个键值对，表示为(key, value)。

列表、元组、字典、集合

组合数据



列表、元组、字典、集合

组合数据特点

	字符串	列表	元组	集合	字典
类型名称	Str	list	tuple	set	dict
定界符	'',"","''' '''	方括号[]	圆括号()	大括号{}	大括号{}
是否可变	否	是	否	是	是
是否有序	是	是	是	否	否
是否支持下标	是(使用序号作为下标)	是(使用序号作为下标)	是(使用序号作为下标)	否	是(使用“键”作为下标)
元素分隔符	无	逗号	逗号	逗号	逗号
元素形式要求	无	无	无	必须可哈希	键:值
元素值要求	无	无	无	必须可哈希	“键”必须可哈希
元素是否可重复	是	是	是	否	“键”不允许重复，“值”可以重复
元素查找速度	一般	非常慢	很慢	非常快	非常快
新增和删除元素速度	不允许	尾部操作快 其他位置慢	不允许	快	快

组合数据

```
>>> x_str = "ABCDEFG"                          #创建字符串对象
>>> x_list = [1, 2, 3]                         #创建列表对象
>>> x_tuple = (1, 2, 3)                        #创建元组对象
>>> x_dict = {'a':97, 'b':98, 'c':99}          #创建字典对象
>>> x_set = {1, 2, 3}                          #创建集合对象
>>> print(x_str[2])                            #使用下标访问指定位置的元素   B
>>> print(x_dict['a'])                         #字典对象的下标是“键”      97
>>> 3 in x_set                                 #成员测试      True
>>> print(len(x_list))                         # 查看列表长度，也就是其中元素的个数 3
>>> print(x_tuple.index(2))                    # 查看元素2在元组中首次出现的位置     1
>>> for key, value in x_dict.items():
        if value == 98:
            print(key)                         # 查看字典中哪些“键”对应的“值”为98   b
>>> print(max(x_set))                          # 查看集合中元素的最大值       3
```

组合数据

```
>>> text = '''Beautiful is better than ugly.Explicit is better than implicit.Simple
        is better than complex.Complex is better than complicated.Flat is better
        than nested.Sparse is better than dense.Readability counts.'''

>>> print(len(text))                    # 字符串长度，即所有字符的数量
>>> print(text.count('is'))             # 字符串中单词is出现的次数
>>> print('beautiful' in text)          # 测试字符串中是否包含单词beautiful
>>> print('='*20)                       # 字符串重复
>>> print('Good '+'Morning')            # 字符串连接
```

运算符与表达式

运算符	功能说明
+	算术加法，列表、元组、字符串合并与连接，正号
-	算术减法，集合差集，相反数
*	算术乘法，序列重复
/	真除法
//	求整商
%	求余数
**	幂运算
<、<=、>、>=、==、!=	（值）大小比较，集合的包含关系比较
and、or、not	逻辑与、逻辑或、逻辑非
in	成员测试
is	测试两个对象是否为同一个对象的引用
|、^、&、<<、>>、~	位或、位异或、位与、左移位、右移位、位求反
&、|、^	集合交集、并集、对称差集

运算优先级

```
1  lambda
2  逻辑运算：or
3  逻辑运算：and
4  逻辑运算：not
5  成员测试：in, not in
6  同一性测试：is, is not
7  比较：<, <=, >, >=, !=, ==
8  按位或：|
9  按位异或：^
10 按位与：&
11 移位：<<, >>
12 加法和减法：+, -
13 乘法、除法与取余：*, /, %
14 正负号：+x, -x
15 按位翻转：~x
16 指数：**
```

运算符与表达式

算术运算

```
>>> str1='abc'+'def'                    #字符相加
>>> lst1=[1,2]+[3,4]                    #列表相加
>>> tup1=(1,2)+(3,)                     #元组相加
>>> dic1={1,2,3}-{3,4,5}                #集合相减
>>> str2='重要的事情说三遍！' * 3        #字符串扩展为3倍
>>> lst2=[0] * 5                        #列表扩展为3倍
>>> tup2=(0,) * 3                       #元组扩展为3倍
>>> n1=365 % 7                          #整数求余
>>> print( '%c,%d,%f' %(65,65,65))      # 把65转换成不同的字符串格式输出
>>> print(2 ** 4)                       #输出整数平方
>>> print(9 ** 0.5)                     #输出整数平方根
```

关系运算

```
>>> print(3+2 < 7+8)                          # 关系运算符优先级低于算术运算符
>>> print(3 < 5 > 2)                          # 等价于3<5 and 5>2
>>> print(3 == 3 < 5)                         # 等价于3==3 and 3<5
>>> print('12345' > '23456')                  # 第一个字符'1'<'2'，直接得出结论
>>> print('abcd' > 'Abcd')                    # 第一个字符'a'>'A'，直接得出结论
>>> print([85, 92, 73, 84] < [91, 82, 73])    # 第一个数字85<91，直接得出结论
>>> print([180, 90, 101] > [180, 90, 99])     # 前两个数字相等，第三个数字101>99
>>> print({1, 2, 3, 4} > {3, 4, 5})           # 第一个集合不是第二个集合的超集
>>> print({1, 2, 3, 4} <= {3, 4, 5})          # 第一个集合不是第二个集合的子集
>>> print([1, 2, 3, 4] > [1, 2, 3])           # 前三个元素相等，并且第一个列表有多余的元素
```

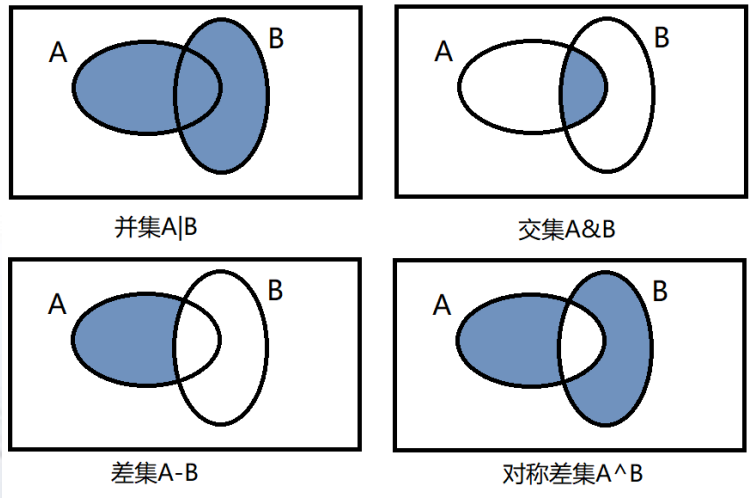
运算符与表达式

成员测试

```
>>> print(60 in [70, 60, 50, 80])          True
>>> print('abc' in 'a1b2c3dfg')            False
>>> print([3] in [[3], [4], [5]])          True
>>> print('3' in map(str, range(5)))        True
>>> print(5 in range(5))                    False
```

集合运算

```
>>> A = {35, 45, 55, 65, 75}
>>> B = {65, 75, 85, 95}
>>> print(A | B)      {35,45,55,65,75,85,98}
>>> print(A & B)      {65,75}
>>> print(A - B)      {35,45,55}
>>> print(B - A)      {85,95}
>>> print(A ^ B)      {35,45,55,85,95}
```



逻辑运算符

- 逻辑运算符and、or、not常用来连接条件表达式构成更加复杂的条件表达式，并且and和or具有惰性求值或逻辑短路的特点，当连接多个表达式时只计算必须要计算的值。
- 另外要注意的是，运算符and和or并不一定会返回True或False，而是得到最后一个被计算的表达式的值，但是运算符not一定会返回True或False。

```
>>> 3>5 and a>3                #注意，此时并没有定义变量a
False
>>> 3>5 or a>3                 #3>5的值为False，所以需要计算后面表达式
NameError: name 'a' is not defined
>>> 3<5 or a>3                 #3<5的值为True，不需要计算后面表达式
True
>>> 3 and 5 or 0                   #最后一个计算的表达式的值作为整个表达式的值
5
>>> 3 and 5>2
True
>>> 3 not in [1, 2, 3]         #逻辑非运算not
False
>>> 3 is not 5                 #not的计算结果只能是True或False之一
True
```

补充说明

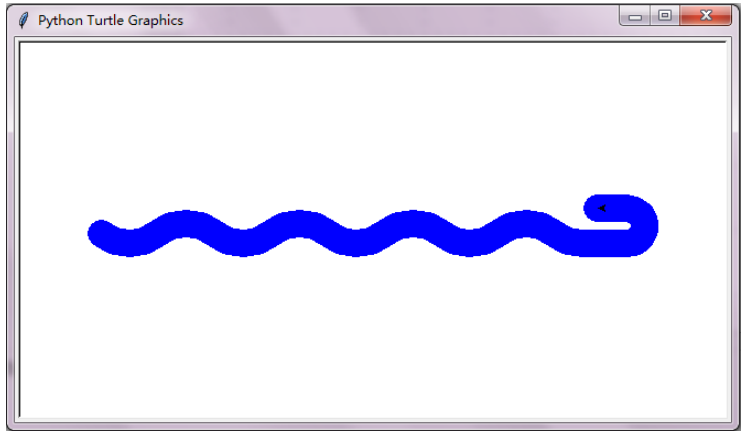
◆Python还有赋值运算符=和+=、-=、*=、/=、//=、**=、|=、^=等大量复合赋值运算符。

◆Python不支持++和--运算符，虽然在形式上有时候似乎可以这样用，但实际上是另外的含义，要注意和其他语言的区别。

```
>>> i = 3
>>> ++i                          #正正得正
3
>>> +(+3)                        #与++i等价
3
>>> i++                          #Python不支持++运算符，语法错误
SyntaxError: invalid syntax
>>> --i                          #负负得正，等价于-(-i)
3
>>> ---i                         #等价于-(-(-i))
-3
>>> i--                          #Python不支持--运算符，语法错误
SyntaxError: invalid syntax
```

认识Python程序

turtle库和蟒蛇绘制程序



通过下面的例子，来实践用Python语言输出图形效果。

*P046-e2.1DrawPython.py - F:\PPT\Python程序设计\Pyth...

File Edit Format Run Options Window Help

```
#e2.1DrawPython.py
import turtle
turtle.setup(650, 350, 200, 200)
turtle.pensize(25)
turtle.pencolor(0/255,0/255,255/255)
turtle.penup()
turtle.fd(-250)
turtle.pendown()
turtle.seth(-40)
for i in range(4):
    turtle.circle(40, 80)
    turtle.circle(-40, 80)
turtle.circle(40, 80/2)
turtle.fd(40)
turtle.circle(16, 180)
turtle.fd(40 * 2/3)
```

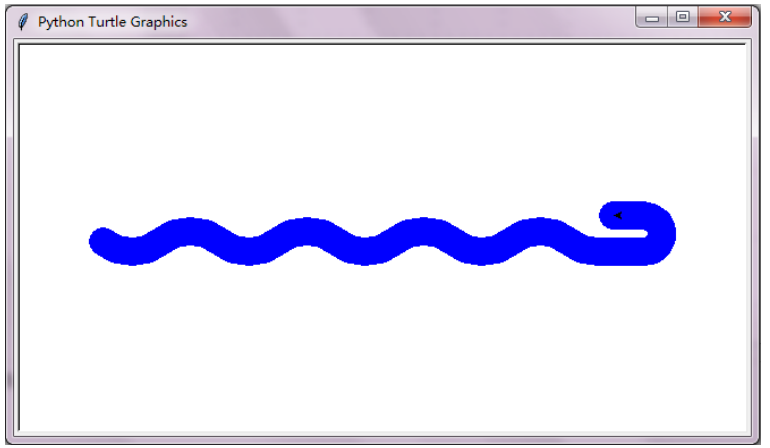
Ln: 17 Col: 0

认识Python程序

程序简析：

- import turtle

import是一个关键字，用来引入一些外部库，这里的含义是引入一个名字叫turtle的python标准函数库

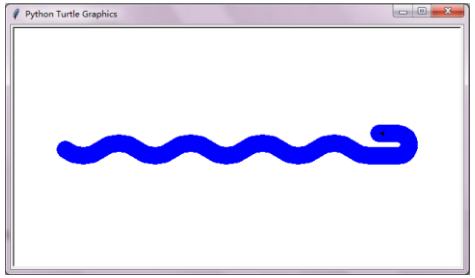


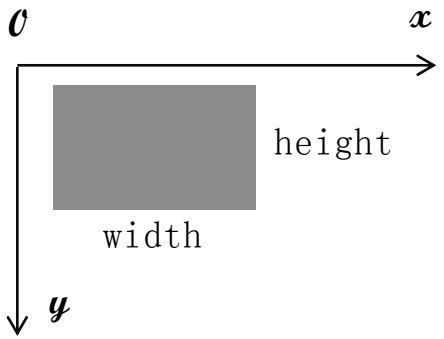
注：Turtle库是Python语言中一个很流行的绘制图像的函数库，使用turtle库，同学们头脑里需要有这样一个概念：

想象一个小乌龟，在一个横轴为x、纵轴为y的坐标系的窗口中从原点0(0,0)位置（窗口的中心）开始，它根据一组函数指令的控制，在这个平面坐标系中爬行，从而以它爬行的轨迹绘制了图形。

认识Python程序

程序简析：





我们所使用的显示屏幕也是一个坐标系，该坐标系以左上角为原点，向左和向下分别是x轴和y轴。

turtle.setup()函数

- Turtle中的turtle.setup()函数用于启动一个图形窗口，它有四个参数（turtle.setup(width, height, startx, starty)
- 分别是：启动窗口的宽度和高度及窗口左上角在屏幕中的坐标位置。
- 蟒蛇程序代码启动一个650像素宽、350像素高的窗口，该窗口的左上角距离屏幕左边框和上边框都是200。

认识Python程序

程序简析：

turtle.pensize()函数

- Turtle中的turtle.pensize()函数表示小乌龟运动轨迹的宽度。
- 它包含一个输入参数，这里我们把它设为25像素。

turtle.pencolor()函数

- Turtle中的turtle.pencolor()函数表示小乌龟运动轨迹的颜色。
- 它包含一个输入参数，这里我们把它设为蓝色，blue，其他颜色单词也可以使用。Turtle采用RGB方式来定义颜色，如果希望获得和图片中颜色一致的小蛇，请输入turtle.pencolor(“#0000FF”)

turtle.penup()函数

- turtle.penup()函数功能是将小乌龟抬起“离开”坐标系，以使得爬行时不留下轨迹。

turtle.pendown()函数

- turtle.pendown()函数功能是将抬起的小乌龟放下，以“回到”坐标系，以使得爬行时能留下轨迹。

认识Python程序

程序简析：

turtle.fd()函数

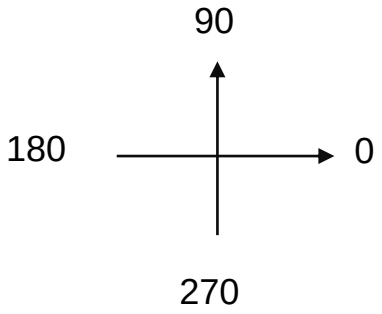
- turtle.fd()函数也可以用turtle.forward()表示乌龟直线爬行移动。
- 它有一个参数表示爬行的距离，正数表示小乌龟向前直线爬行，负数表示向后直线爬行

turtle.circle()函数

- turtle.circle(rad,angle)函数让小乌龟沿着一个圆形爬行。
- 参数rad描述圆形轨迹的半径，rad为正值，小乌龟逆时针爬行，如果rad为负值，小乌龟顺时针爬行。
- 参数angle表示小乌龟沿着圆形爬行的弧度值

认识Python程序

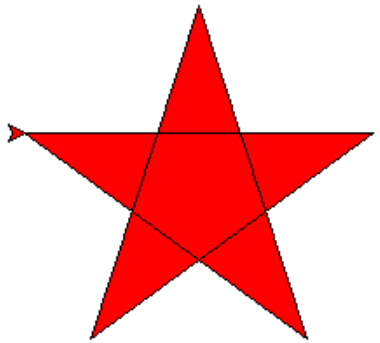
程序简析： **turtle.seth()函数**

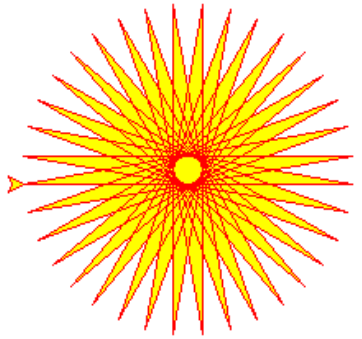


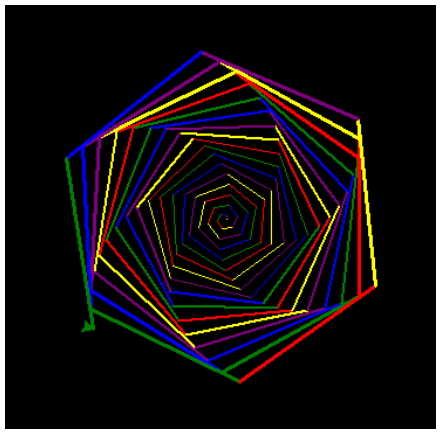
standard mode	logo mode
0 - east	0 - north
90 - north	90 - east
180 - west	180 - south
270 - south	270 - west

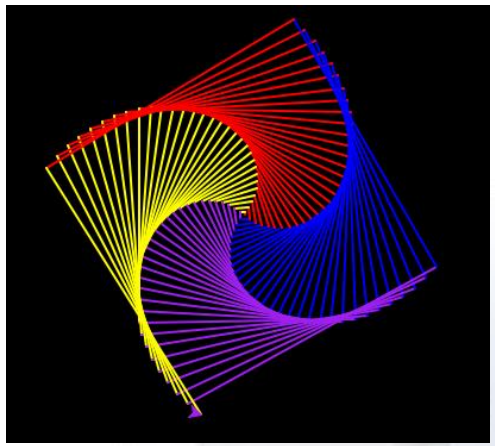
- Turtle中的turtle.seth(angle)函数表示小乌龟启动时运动的方向。它包含一个输入参数angle，是角度值。
- 其中，0表示向东，90度向北，180度向西，270度向南，逆时针方向；负值表示相反（顺时针）方向。
- 程序中，我们让小乌龟向-40度启动爬行，即：向东南方向40度。

Python实例展示









Python实例展示

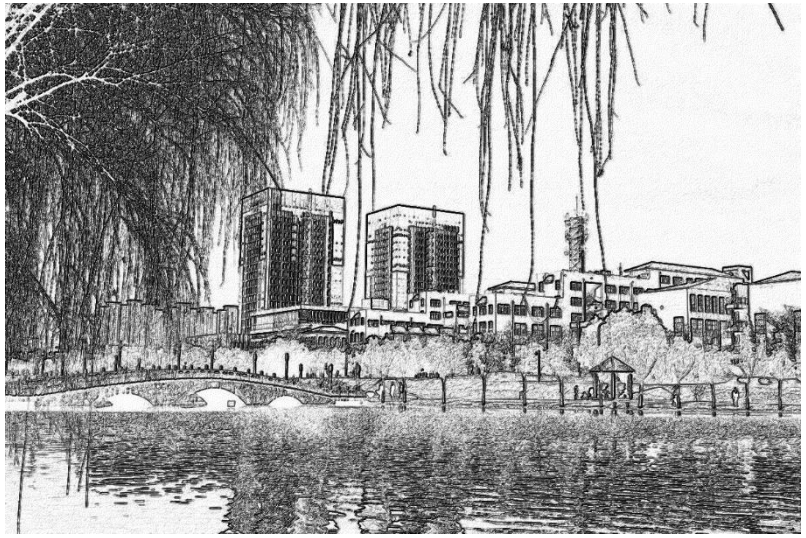
4行代码

图像的轮廓效果

```
from PIL import Image,ImageFilter
im = Image.open( 'Hfut.jpg')
om = im.filter(ImageFilter.CONTOUR)
om.save( 'HfutContour.jpg')
```

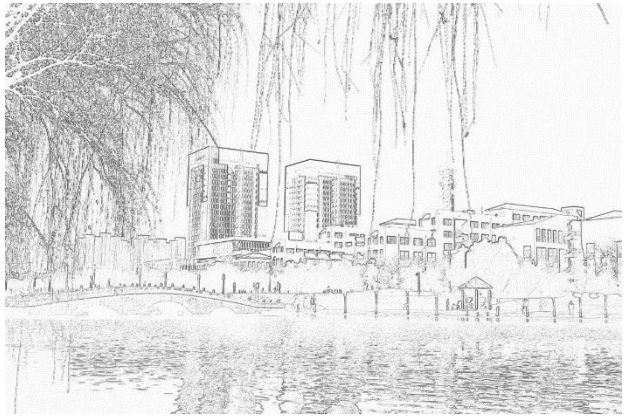
Python实例展示

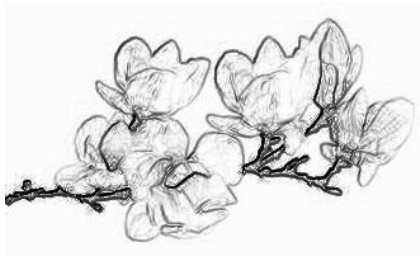
```
from PIL import Image
import numpy as np
a = np.array(Image.open( 'hfut.jpg').convert('L')).astype('float')
depth = 10
grad = np.gradient(a)
grad_x, grad_y = grad
grad_x = grad_x*depth/100
grad_y = grad_y*depth/100
A = np.sqrt(grad_x**2 + grad_y**2 + 1)
uni_x = grad_x/A
uni_y = grad_y/A
uni_z = 1/A
vec_el = np.pi/2.2
vec_az = np.pi/4
dx = np.cos(vec_el)*np.cos(vec_az)
dy = np.cos(vec_el)*np.sin(vec_az)
dz = np.sin(vec_el)
b = 255*(dx*uni_x + dy*uni_y + dz*uni_z)
b = b.clip(0,255)
im = Image.fromarray(b.astype('uint8')
im.save( 'hfutHD.jpg')
```

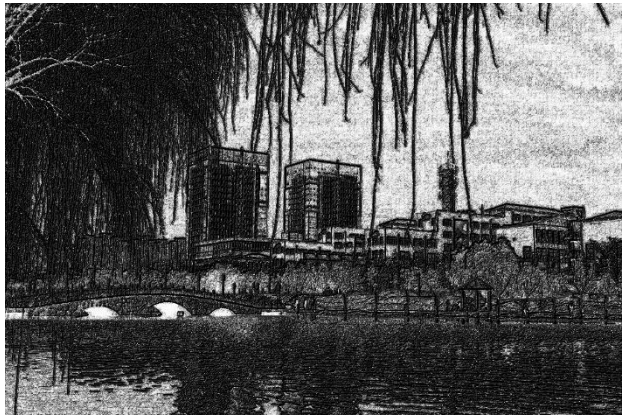


图像的手绘效果

Python实例展示







合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

祝学习进步！

计算机基础教育研究所

2021.3