Python数据类型

对象(object)

- 万物皆对象

`isinstance(object, classinfo)`

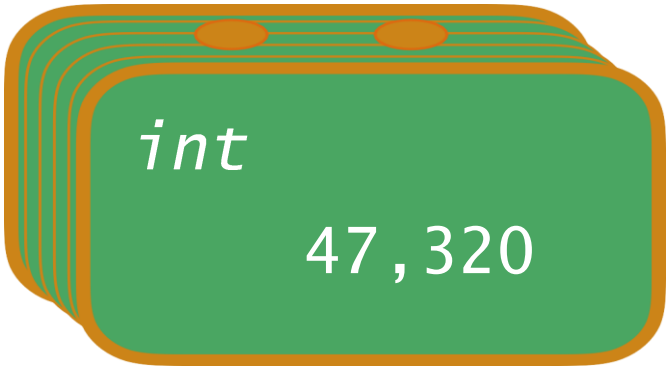
```
isinstance(4, object)                # => True

isinstance("Hello", object)          # => True

isinstance(None, object)             # => True

isinstance([1,2,3], object)          # => True
```

对象拥有identity

- 函数id()返回对象的”identity”
- identity在对象生命周期是唯一且固定的
- 对象在运行时绑定其类型(type)
- 对象包含指向其数据块的指针

```
>>> (4).__sizeof__()
14
>>> ("hello").__sizeof__()
30
>>>
```

An analogy



将Python对象看成是手提箱，箱子有不同的大小，用来装不同的值。每个手提箱有给定的类型，并且存储了一些值。

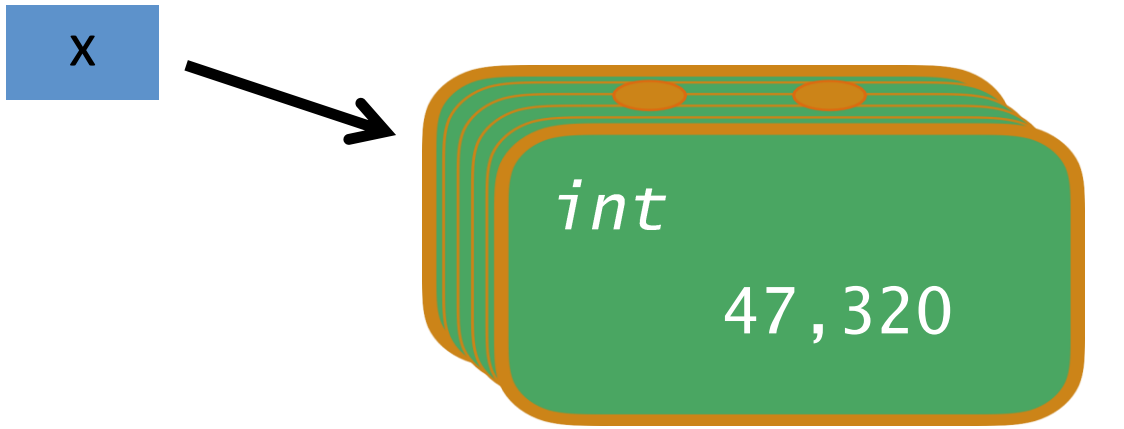
Python可以为你管理所有行李（一堆手提箱）。

变量

变量是对象的引用

用前面的类比，变量是手提箱的标签(label)

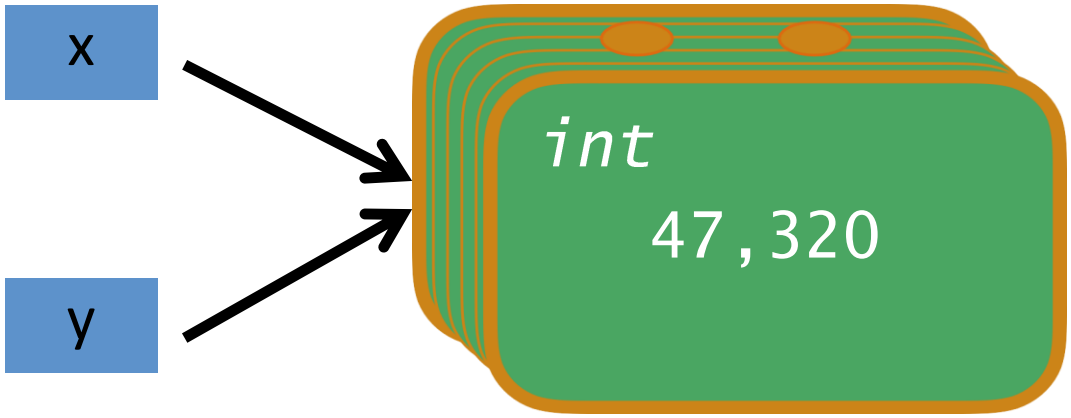
```
x = 47320
```



变量赋值

- 从一个变量进行赋值不拷贝一个对象，而是给同一个对象添加新的引用(reference)

```
x = 47320

y = x
```



is v.s. ==

我们已知等号测试：

1 == 1.0 True

但这两个对象本质上是有差异的：

type(1) != type(1.0)

int != float

is操作符检测identity，而不是equality

a is not b 是 not (a is b) 的语法糖

is v.s. ==

- is检测是否同一个手提箱，而==检测手提箱里边的物品是否相同

- 使用==比较值(values)（常用），使用is比较一致性(identities)

数字类型

数字类型

- 程序元素：010/10，存在多种可能
  - 表示十进制整数值10
  - 类似人名一样的字符串
- 数字类型对Python语言中数字的表示和使用进行了定义和规范

数字类型

Python语言包括三种数字类型

- 整数类型
- 浮点数类型
- 复数类型

整数类型

- 与数学中的整数概念一致，没有取值范围限制
- pow(x, y)函数：计算x^y
- 打开IDLE
  - 程序1：pow(2,10) , pow(2,15)
  - 程序2：pow(2, 1000)
  - 程序3：pow(2, pow(2,15))

整数类型

- 示例
  - 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, -0O456　(0o, 0O开头表示8进制数)

浮点数类型

- 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

浮点数类型

```
>>> 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)
>>>
```

浮点数类型

- 示例
  - 0.0, -77., -2.17
  - 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：

$$<a>e<b> = a * 10^{b}$$

复数类型

- 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

- 示例：

12.3+4j, -5.6+7j

复数类型

- 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

数字类型的操作

内置的数值运算操作符

操作符	描述
x + y	x与y之和
x - y	x与y之差
x * y	x与y之积
x / y	x与y之商
x // y	x与y之整数商，即：不大于x与y之商的最大整数
x % y	x与y之商的余数，也称为模运算
-x	x的负值，即：x*(-1)
+x	x本身
x**y	x的y次幂，即：x^y

内置的数值运算操作符

数字类型之间相互运算所生成的结果是“更宽”的类型，基本规则是：

- 整数之间运算，如果数学意义上的结果是小数，结果是浮点数；
- 整数之间运算，如果数学意义上的结果是整数，结果是整数；
- 整数和浮点数混合运算，输出结果是浮点数；
- 整数或浮点数与复数运算，输出结果是复数。

内置的数值运算函数

Python解释器提供了一些内置函数，在这些内置函数之中，有6个函数与数值运算相关

函数	描述
abs(x)	x的绝对值
divmod(x, y)	(x//y, x%y)，输出为二元组形式（也称为元组类型）
pow(x, y[, z])	(x**y)%z，[..]表示该参数可以省略，即：pow(x,y)，它与x**y相同
round(x[, ndigits])	对x四舍五入，保留ndigits位小数。round(x)返回四舍五入的整数值
max(x_1, x_2, …, x_n)	x_1, x_2, …, x_n的最大值，n没有限定
min(x_1, x_2, …, x_n)	x_1, x_2, …, x_n的最小值，n没有限定

数字类型的转换

- 数值运算操作符可以隐式地转换输出结果的数字类型
- 例如，两个整数采用运算符“/”的除法将可能输出浮点数结果。
- 此外，通过内置的数字类型转换函数可以显式地在数字类型之间进行转换

函数	描述
int(x)	将x转换为整数，x可以是浮点数或字符串
float(x)	将x转换为浮点数，x可以是整数或字符串
complex(re[, im])	生成一个复数，实部为re，虚部为im，re可以是整数、浮点数或字符串，im可以是整数或浮点数但不能为字符串

数字类型的转换

- 示例：
  - int(4.5) = 4 （直接去掉小数部分）
  - float(4) = 4.0 （增加小数部分）
  - complex(4) = 4 + 0J

数字类型的转换

- 示例：complex(4.5) = 4.5 + 0J

```
>>> float(4.5+0j)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    float(4.5+0j)
TypeError: can't convert complex to float
>>>
```

数字类型的判断

- 函数：type(x)，返回x的类型，适用于所有类型的判断
- 示例：

```
>>> type(4.5)
<class 'float'>
>>> type(z)
<class 'complex'>
>>>
```

math库的使用

math库概述

- math库是Python提供的内置数学类函数库
- math库不支持复数类型
- math库一共提供了4个数学常数和44个函数。
  - 44个函数共分为4类，包括：16个数值表示函数、8个幂对数函数、16个三角对数函数和4个高等特殊函数

math库概述

首先使用保留字import引用该库

- 第一种：import math

  对math库中函数采用math.<b>()形式使用

```
>>>import math
>>>math.ceil(10.2)
11
```

- 第二种，from math import <函数名>

  对math库中函数可以直接采用<函数名>()形式使用

```
>>>from math import floor
>>>floor(10.2)
10
```

math库解析

- math库包括4个数学常数

常数	数学表示	描述
math.pi	π	圆周率，值为3.141592653589793
math.e	e	自然对数，值为2.718281828459045
math.inf	∞	正无穷大，负无穷大为-math.inf
math.nan		非浮点数标记，NaN（Not a Number）

- math库包括16个数值表示函数

函数	描述
math.fabs(x)	返回x的绝对值
math.fmod(x, y)	返回x与y的模
math.fsum([x,y,…])	浮点数精确求和
math.ceil(x)	向上取整，返回不小于x的最小整数
math.floor(x)	向下取整，返回不大于x的最大整数
math.factorial(x)	返回x的阶乘，如果x是小数或负数，返回ValueError
math.gcd(a, b)	返回a与b的最大公约数
math.frexp(x)	返回(m, e)，当x=0，返回(0.0, 0)
math.ldexp(x, i)	返回x * 2^i运算值，math.frexp(x)函数的反运算
math.modf(x)	返回x的小数和整数部分
math.trunc(x)	返回x的整数部分
math.copysign(x, y)	用数值y的正负号替换数值x的正负号
math.isclose(a,b)	比较a和b的相似性，返回True或False
math.isfinite(x)	当x为有限大，返回True；否则，返回False
math.isinf(x)	当x为正数或负数无穷大，返回True；否则，返回False
math.isnan(x)	当x是NaN，返回True；否则，返回False

math库解析

```
math.inf == math.inf          # => True
math.nan == math.nan          # => False
math.isfinite(math.inf)       # => False
math.isfinite(math.nan)       # => False
math.isinf(math.nan)          # => False
math.isnan(math.inf)          # => False
```

```
>>> sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
0.9999999999999999
>>>
math.fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
1.0
```

math库解析

- math库中包括8个幂对数函数

函数	数学表示	描述
math.pow(x,y)	x^y	返回x的y次幂
math.exp(x)	e^x	返回e的x次幂，e是自然对数
math.expml(x)	e^x-1	返回e的x次幂减1
math.sqrt(x)	$\sqrt{x}$	返回x的平方根
math.log(x[,base])	$\log_{base} x$	返回x的对数值，只输入x时，返回自然对数
math.log1p(x)	$\ln(1+x)$	返回1+x的自然对数值
math.log2(x)	$\log x$	返回x的2对数值
math.log10(x)	$\log_{10} x$	返回x的10对数值

- math库包括16个三角运算函数

函数	数学表示	描述
math.degree(x)		角度x的弧度值转角度值
math.radians(x)		角度x的角度值转弧度值
math.hypot(x,y)		返回(x,y)坐标到原点(0,0)的距离
math.sin(x)	sin x	返回x的正弦函数值，x是弧度值
math.cos(x)	cos x	返回x的余弦函数值，x是弧度值
math.tan(x)	tan x	返回x的正切函数值，x是弧度值
math.asin(x)	arcsin x	返回x的反正弦函数值，x是弧度值
math.acos(x)	arccos x	返回x的反余弦函数值，x是弧度值
math.atan(x)	arctan x	返回x的反正切函数值，x是弧度值
math.atan2(y,x)	arctan y/x	返回y/x的反正切函数值，x是弧度值
math.sinh(x)	sinh x	返回x的双曲正弦函数值
math.cosh(x)	cosh x	返回x的双曲余弦函数值
math.tanh(x)	tanh x	返回x的双曲正切函数值
math.asinh(x)	arcsinh x	返回x的反双曲正弦函数值
math.acosh(x)	arccosh x	返回x的反双曲余弦函数值
math.atanh(x)	arctanh x	返回x的反双曲正切函数值

math库解析

- math库包括4个高等特殊函数

函数	数学表示	描述
math.erf(x)	$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$	高斯误差函数，应用于概率论、统计学等领域
math.erfc(x)	$\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt$	余补高斯误差函数，math.erfc(x)=1 - math.erf(x)
math.gamma(x)	$\int_0^\infty x^{t-1}e^{-x}dx$	伽玛（Gamma）函数，也叫欧拉第二积分函数
math.lgamma(x)	ln(gamma(x))	伽玛函数的自然对数

字符串类型及其操作

字符串

- Python语言中，字符串是用两个双引号“ ”或者单引号‘ ’括起来的一个或多个字符。
- 选用最方便的方式表述特殊字符

```
print('doesn\'t')      # => doesn't
print("doesn't")       # => doesn't

print('"Yes," he said.')          # => "Yes," he said.
print("\"Yes,\" he said.")        # => "Yes," he said.

print('"Isn\'t," she said.')      # => "Isn't," she said.
```

常用的字符串函数

```
greeting = "Hello world! "

greeting[4]                    # => 'o'
'world' in greeting            # => True
len(greeting)                  # => 13


greeting.find('lo')                  # => 3 (-1 if not found)
greeting.replace('llo', 'y')         # => “hey world!”
greeting.startswith(‘Hell')          # => True
```

常用的字符串函数

```
greeting = "Hello world! "


greeting.lower()                # => “hello world! "
greeting.title()                # => "Hello World! "


greeting.strip()                # => "Hello world!"
greeting.strip('! ')            # => "Hello world" (no '!')
```

字符串

- Python字符串中每个字符都是用Unicode编码表示
- chr(x)和ord(x)用于单字符和Unicode编码之间进行转换
- hex(x)和oct(x)函数分别返回整数x对应的十六进制和八进制的字符串形式

字符串

微实例：恺撒密码

凯撒密码是古罗马凯撒大帝用来对军事情报进行加密的算法，它采用了替换方法对信息中的每一个英文字符循环替换为字母表序列该字符后面第三个字符，对应关系如下：

原文：A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

密文：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

原文字符P，其密文字符C满足如下条件：

C = (P + 3) mod 26

解密方法反之，满足：

P = (C - 3) mod 26

字符串

```
plaincode = input("请输入明文: ")
for p in plaincode:
    if ord("a") <= ord(p) <= ord("z"):
        print(chr(ord("a") + (ord(p) - ord("a") + 3) % 26), end='')
    else:
        print(p, end='')
```

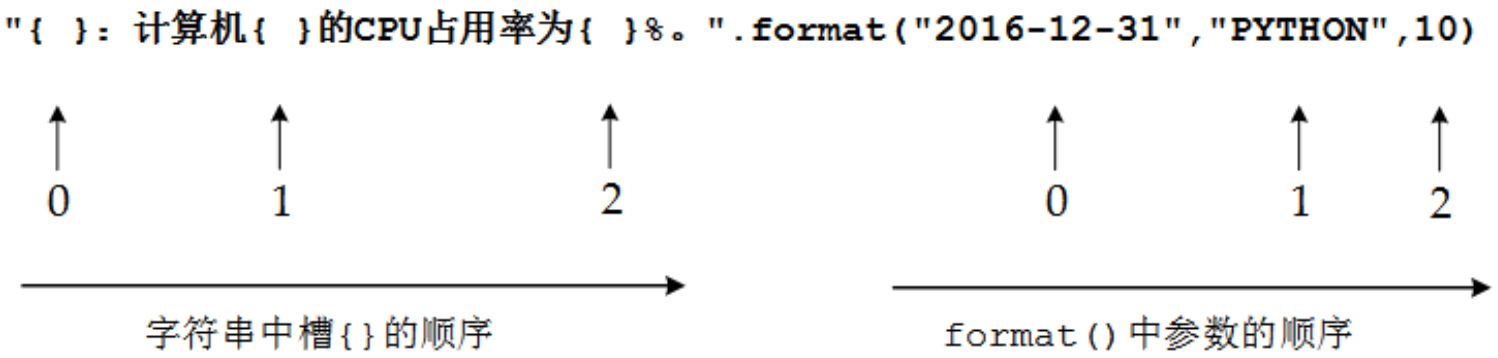
微实例运行结果如下：

```
>>>
请输入明文: python is an excellent language.
sbwkrq lv dq hafhoohqw odqjxdjh.
```

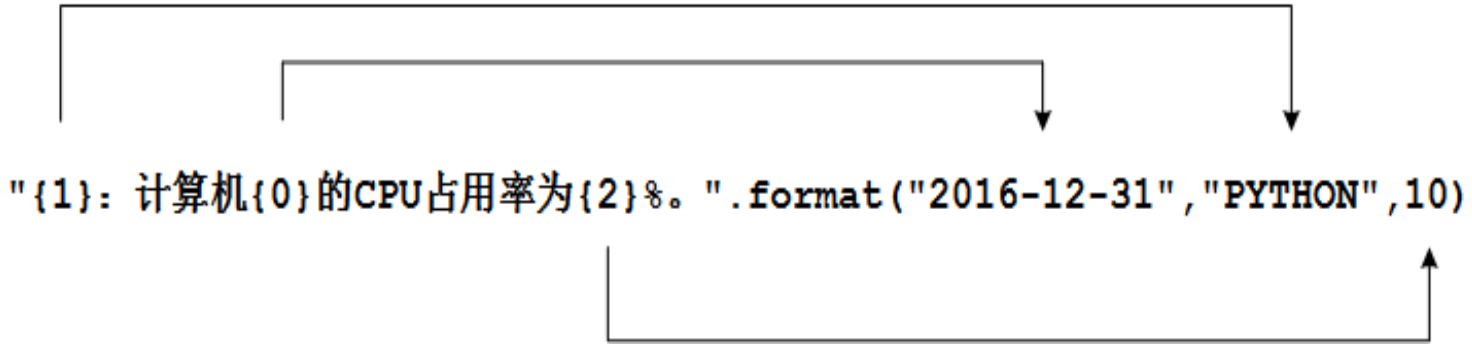
字符串格式化

字符串format()方法的基本使用格式是：

<模板字符串>.format(<逗号分隔的参数>)



字符串格式化



字符串格式化

- format()方法中模板字符串的槽除了包括参数序号，还可以包括格式控制信息。此时，槽的内部样式如下：{<参数序号>: <格式控制标记>}

- 其中，格式控制标记用来控制参数显示时的格式 。格式控制标记包括：<填充><对齐><宽度><,><.精度><类型>6个字段，这些字段都是可选的，可以组合使用，这里按照使用方式逐一介绍。

字符串格式化

:	<填充>	<对齐>	<宽度>	,	<.精度>	<类型>
引导符号	用于填充的单个字符	< 左对齐 > 右对齐 ^ 居中对齐	槽的设定输出宽度	数字的千位分隔符 适用于整数和浮点数	浮点数小数部分的精度 或 字符串的最大输出长度	整数类型 b, c, d, o, x, X 浮点数类型 e, E, f, %

字符串格式化

```
>>> s = "PYTHON"
>>> "{0:30}".format(s)
'PYTHON                        '
>>> "{0:>30}".format(s)
'                        PYTHON'
>>> "{0:*^30}".format(s)
'************PYTHON************'
>>> "{0:-^20,}".format(1234567890)
'---1,234,567,890----'
>>> "{0:.2f}".format(12345.67890)
'12345.68'
>>> "{0:H^20.3f}".format(12345.67890)
'HHHHH12345.679HHHHHH'
```

文本进度条

简单的开始

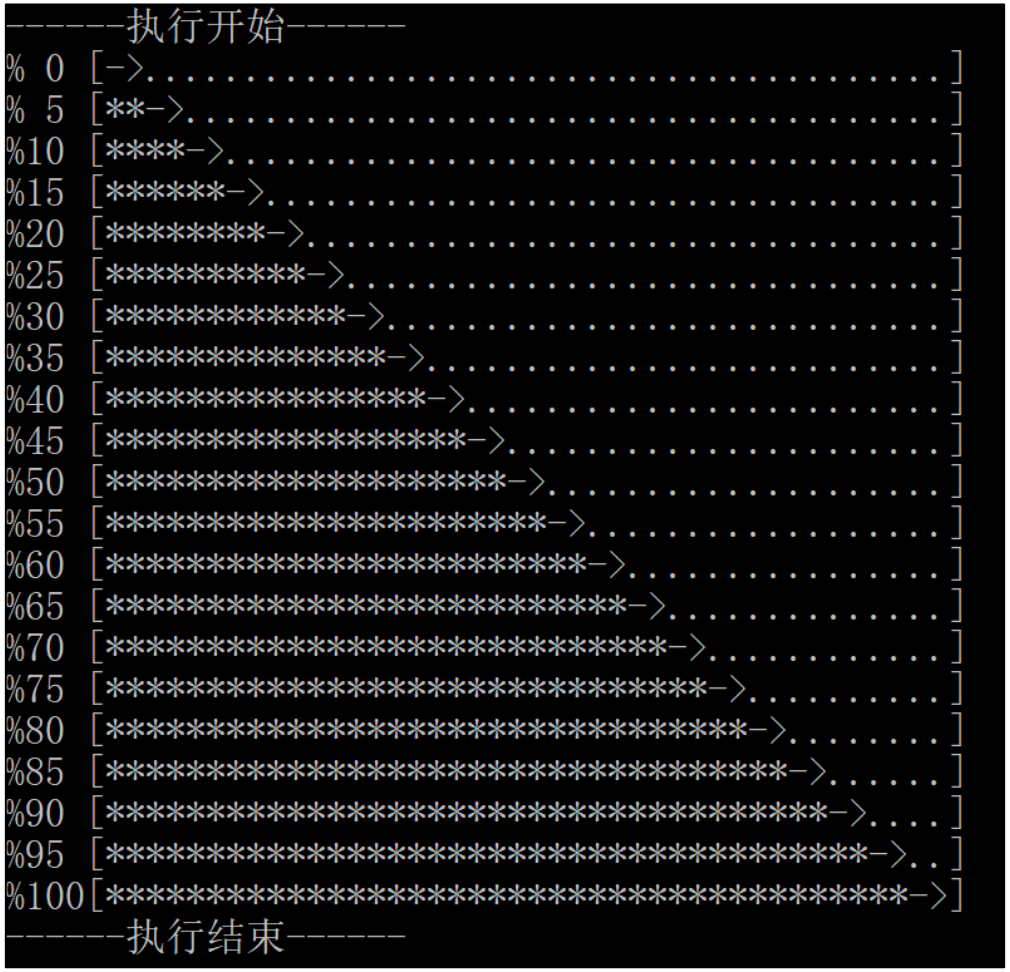
- 利用print()函数实现简单的非刷新文本进度条
- 基本思想是按照任务执行百分比将整个任务划分为100个单位，每执行N%输出一次进度条。每一行输出包含进度百分比，代表已完成的部分(**)和未完成的部分(..)的两种字符，以及一个跟随完成度前进的小箭头，风格如下：

```
%10 [*****->.....................................]
```

简单的开始

```
#TextProgress Bar.py
import time
scale = 20
print("------执行开始------")
for i in range(scale+1):
    a, b = '**' * i,'..' * (scale - i)
    c = (i/scale)*100
    print("%{:^3.0f}[{}->{}]" .format (c, a, b))
    time.sleep(0.1)
print("------执行结束------")
```

简单的开始



单行动态刷新

```
#TextProgressBar.py
import time
for i in range(101):
    print("\r{:2}%".format(i), end="")
    time.sleep(0.05)
```

单行动态刷新

- 上述程序在IDLE中的执行效果如下。

```
============
 0% 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%32%33%34%35%36%37%38%39%40
%41%42%43%44%45%46%47%48%49%50%51%52%53%54%55%56%57%58%59%6
0%61%62%63%64%65%66%67%68%69%70%71%72%73%74%75%76%77%78%79%
80%81%82%83%84%85%86%87%88%89%90%91%92%93%94%95%96%97%98%99
%100%
>>>
```

- 这是因为IDLE本身屏蔽了单行刷新功能，如果希望获得刷新效果，请使用控制台命令行执行TextProgressBar.py程序。

单行动态刷新

```
#TextProgress.py
import time
scale = 50
print("执行开始".center(scale//2,'-'))
t = time.clock()
for i in range(scale+1):
    a = '*' * i
    b = '.' * (scale - i)
    c = (i/scale)*100
    t -= time.clock()
    print("\r{:^3.0f}%[{}->{}]{:.2f}s".format(c,a,b,-t),end='')
    time.sleep(0.05)
print("\n"+"执行结束".center(scale//2,'-'))
```

单行动态刷新

- 采用命令行执行上述文件，效果如下：

```
----------执行开始----------
100%[**************************************************->]64.41s
----------执行结束----------
```

组合数据类型

组合数据类型概述

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；

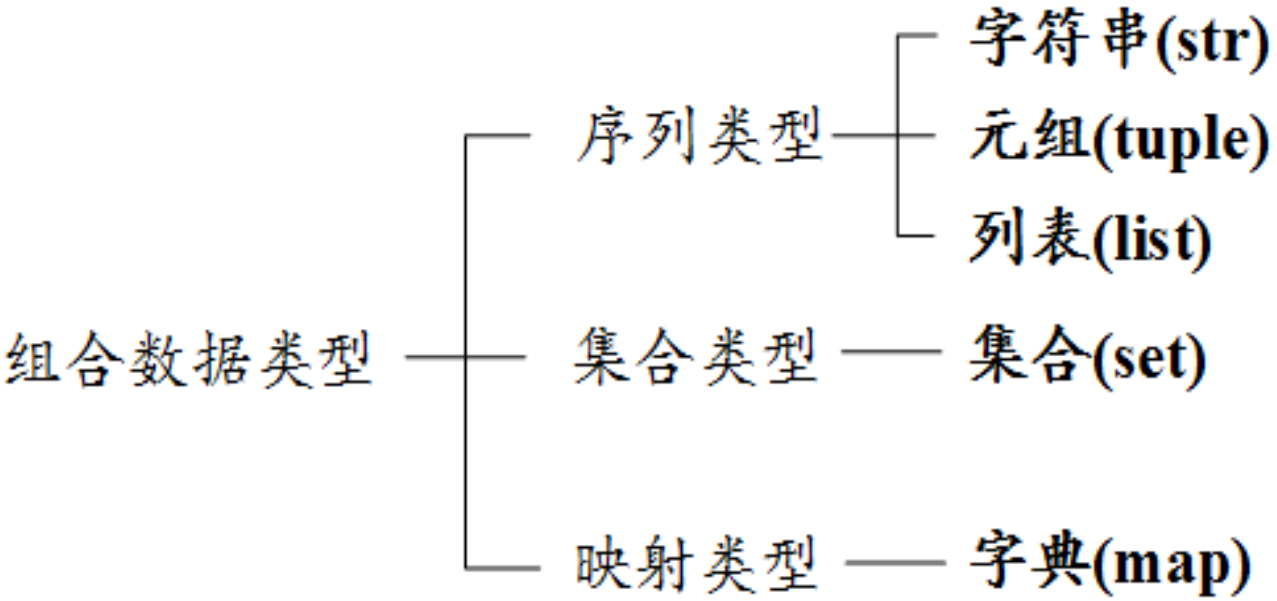
组合数据类型概述

- 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。

组合数据类型概述

- 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)。

组合数据类型概述

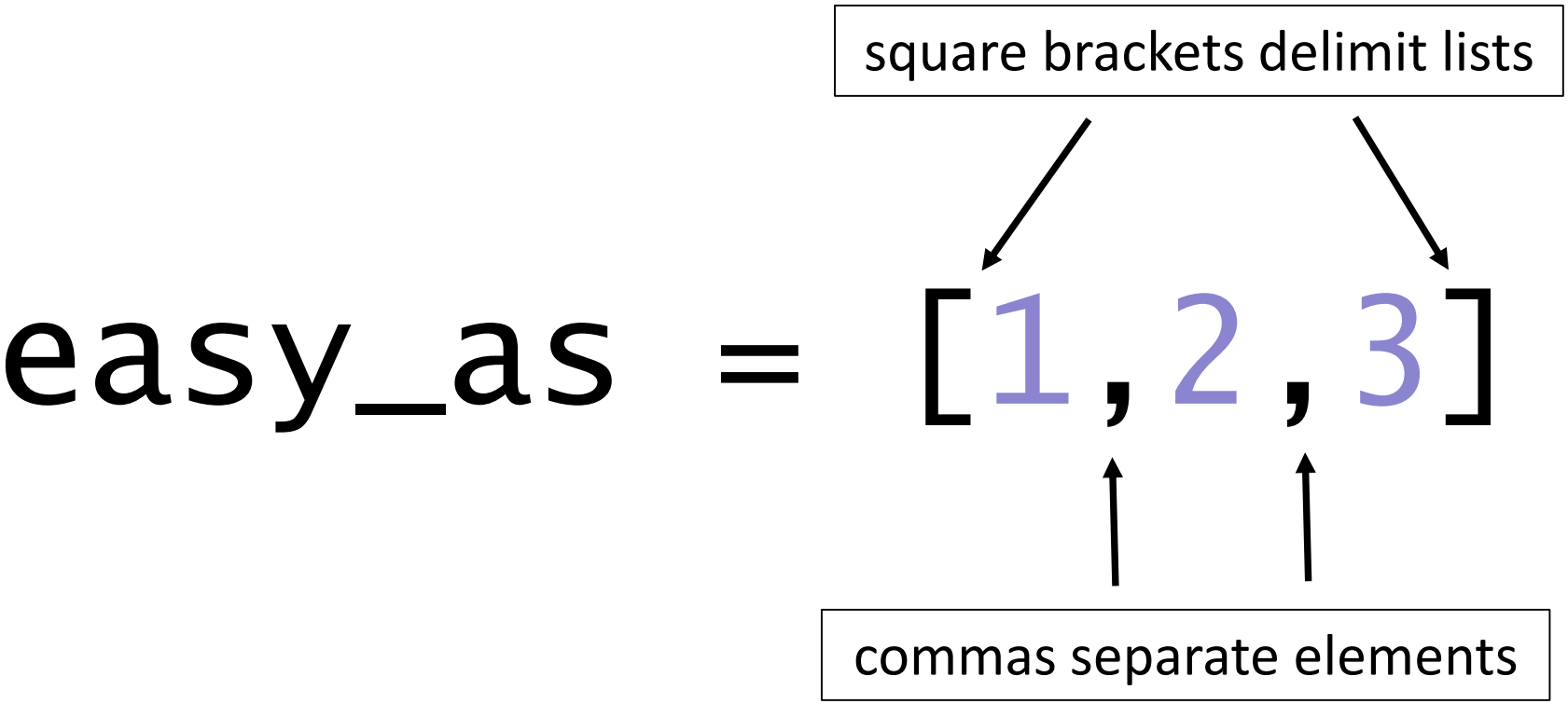


列表(lists)

列表

- 列表（list）是包含0个或多个对象引用的有序序列，属于序列类型。列表的长度和内容都是可变的，可自由对列表中数据项进行增加、删除或替换。列表没有长度限制，元素类型可以不同，使用非常灵活。

列表


square brackets delimit lists
easy_as = [1,2,3]
commas separate elements

列表

```
# Create a new list
empty = []
letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5]

# Lists can contain elements of different types
mixed = [4, 5, "seconds"]

# Append elements to the end of a list
numbers.append(7) # numbers == [2, 3, 5, 7]
numbers.append(11) # numbers == [2, 3, 5, 7, 11]
```

列表元素索引

```
# Access elements at a particular index
numbers[0]   # => 2
numbers[-1]  # => 11

# You can also slice lists - the usual rules
apply
letters[:3]    # => ['a', 'b', 'c']
numbers[1:-1]  # => [3, 5, 7]
```

嵌套列表

```
# Lists really can contain anything - even
other lists!
x = [letters, numbers]
x # => [['a', 'b', 'c', 'd'], [2, 3, 5, 7, 11]]
x[0] # => ['a', 'b', 'c', 'd']

x[0][1]  # => 'b'

x[1][2:] # => [5, 7, 11]
```

列表方法

```
# Extend list by appending elements from the iterable
my_list.extend(iterable)

# Insert object before index
my_list.insert(index, object)

# Remove first occurrence of value, or raise ValueError
my_list.remove(value)

# Remove all items
my_list.clear()
```

列表方法

```
# Return number of occurrences of value
my_list.count(value)

# Return first index of value, or raise ValueError
my_list.index(value, [start, [stop]])

# Remove, return item at index (default last) or
IndexError
my_list.pop([index])

# Stable sort *in place*
my_list.sort(key=None, reverse=False)

# Reverse *in place*.
my_list.reverse()
```

可迭代对象的通用方法

```
# Length (len)
len([])   # => 0
len("python")  # => 6
len([4,5,“seconds”])# => 3

# Membership (in)
0 in []   # => False
'y' in 'python'      # => True
'minutes' in [4, 5, 'seconds'] # => False
```

list.sort方法和内置函数sorted

- list.sort方法就地排序列表，返回值为None
- 内置函数sorted会新建一个列表进行排序，并将新建列表作为返回值

列表复制

- 切片

```
L = [1, 2, 3, 4, 5]
C = L[1:3]
C[0] = 8
print(C)
print(L)
```

```
>>>
[8, 3]
[1, 2, 3, 4, 5]
```

```
L = [1, 2, 3, 4, 5]
C = L[:]
C[0] = 8
print(C)
print(L)
```

```
>>>
[8, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

列表复制

- 内置list()函数

```
L = [1,2,3,4,5]
C = list(L)
C[0] = 8
print(C)
print(L)
```

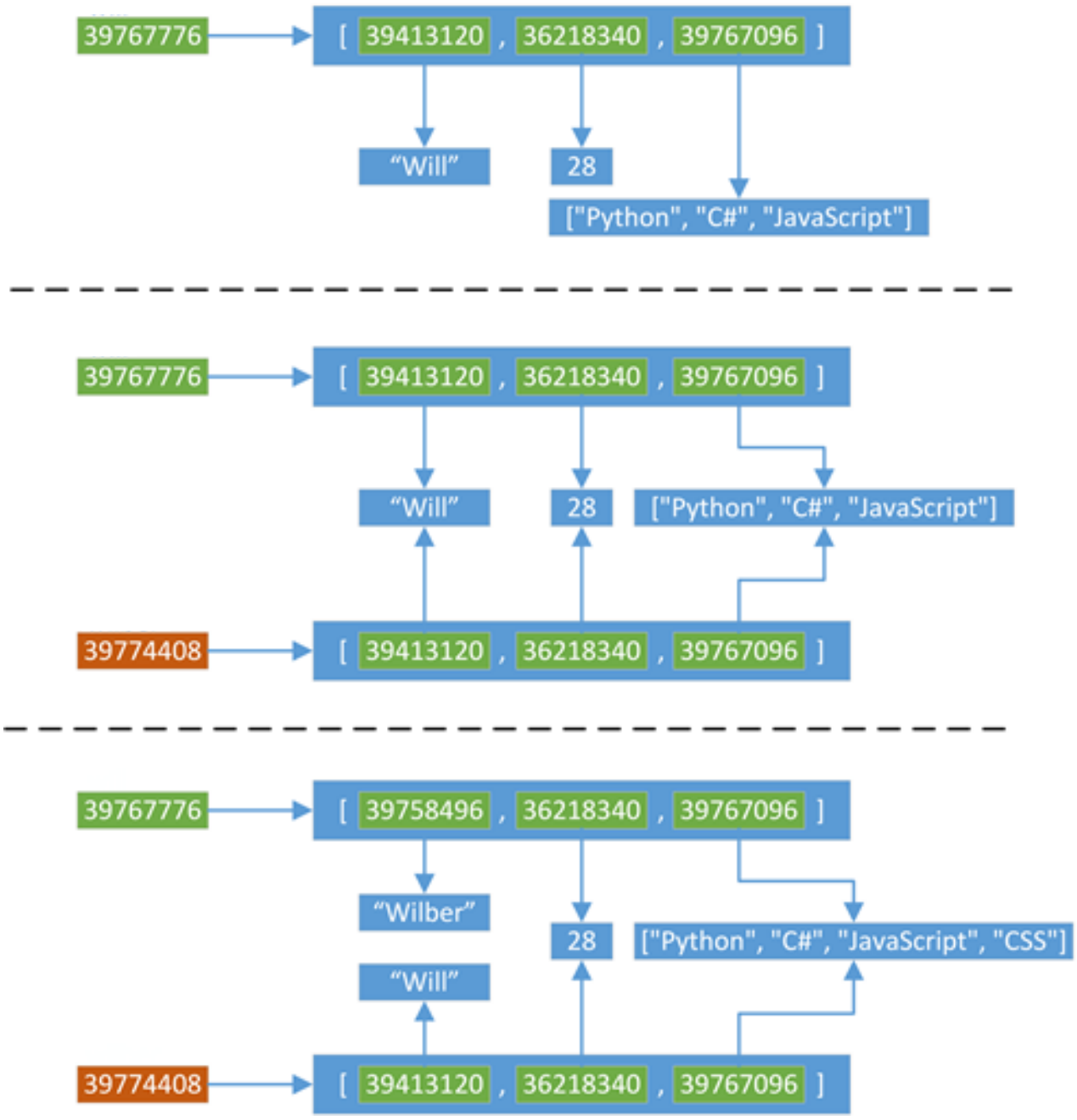
```
>>>
[8, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

列表复制

- copy()函数

```
import copy
student1 = ["Will", 28, ["Python", "C#", "JavaScript"]]
student2 = copy.copy(student1)
print(id(student1))
print(student1)
print([id(ele) for ele in student1])
print(id(student2))
print(student2)
print([id(ele) for ele in student2])

student1[0] = "Wilber"
student1[2].append("CSS")
print(id(student1))
print(student1)
print([id(ele) for ele in student1])
print(id(student2))
print(student2)
print([id(ele) for ele in student2])
```

```
39767776
['Will', 28, ['Python', 'C#', 'JavaScript']]
[39413120, 36218340, 39767096]
39774408
['Will', 28, ['Python', 'C#', 'JavaScript']]
[39413120, 36218340, 39767096]
39767776
['Wilber', 28, ['Python', 'C#', 'JavaScript', 'CSS']]
[39758496, 36218340, 39767096]
39774408
['Will', 28, ['Python', 'C#', 'JavaScript', 'CSS']]
[39413120, 36218340, 39767096]
```

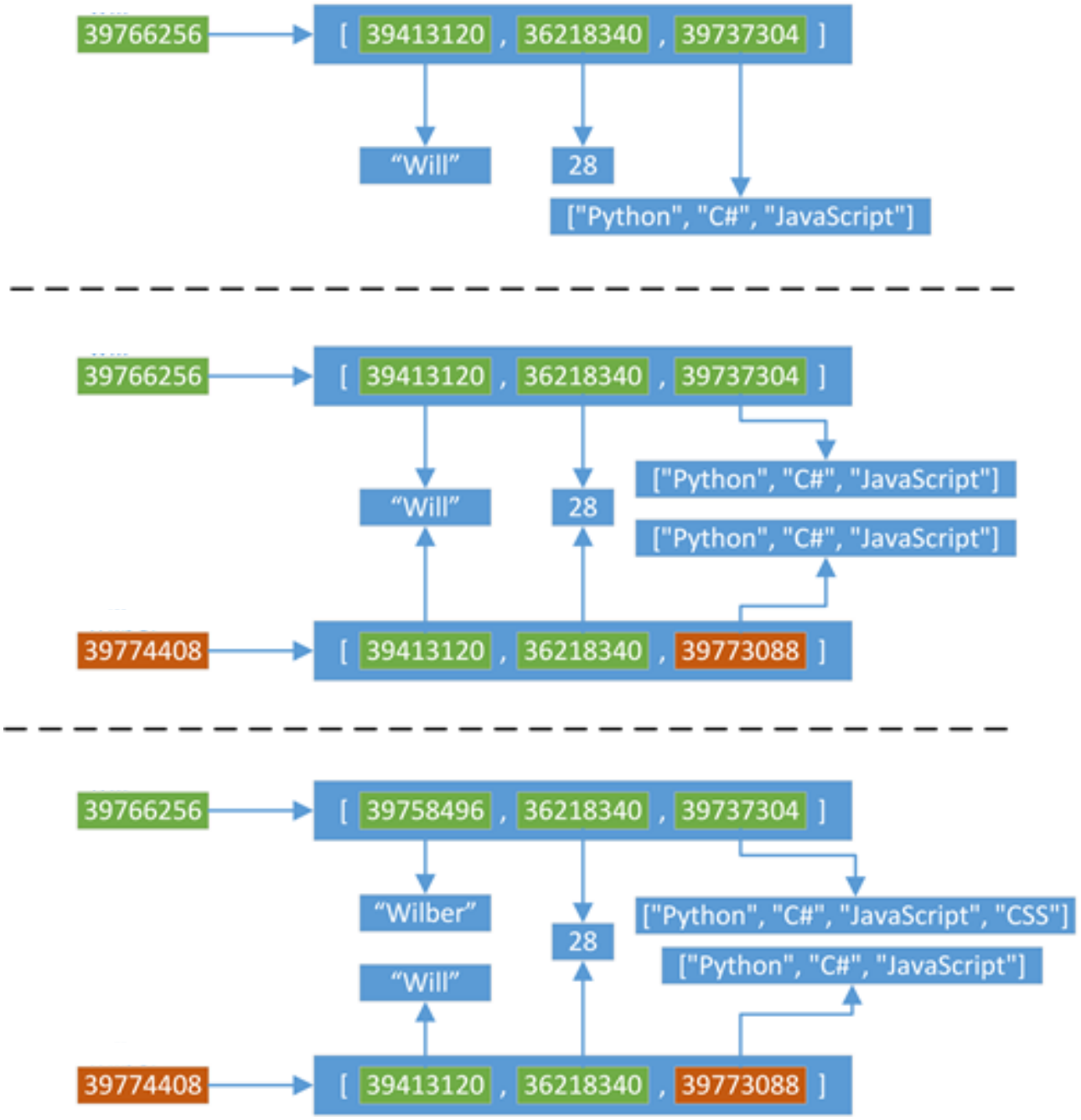

39767776
[39413120 , 36218340 , 39767096]
"Will"
28
["Python", "C#", "JavaScript"]
39767776
[39413120 , 36218340 , 39767096]
"Will"
28
["Python", "C#", "JavaScript"]
39774408
[39413120 , 36218340 , 39767096]
39767776
[39758496 , 36218340 , 39767096]
"Wilber"
28
["Python", "C#", "JavaScript", "CSS"]
"Will"
39774408
[39413120 , 36218340 , 39767096]

列表复制

- 深复制deepcopy()

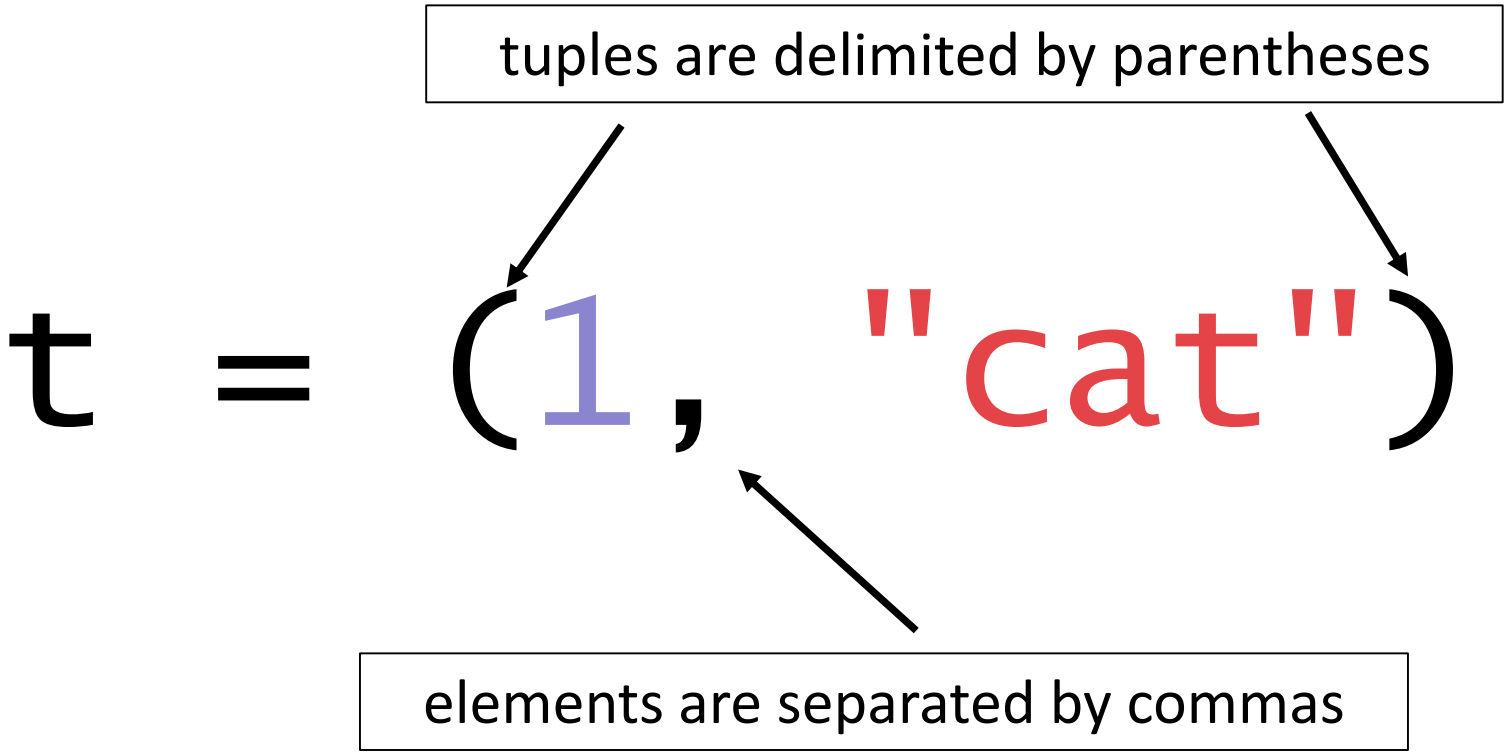
```
import copy
student1 = ["Will", 28, ["Python", "C#", "JavaScript"]]
student2 = copy.deepcopy(student1)
print(id(student1))
print(student1)
print([id(ele) for ele in student1])
print(id(student2))
print(student2)
print([id(ele) for ele in student2])

student1[0] = "Wilber"
student1[2].append("CSS")
print(id(student1))
print(student1)
print([id(ele) for ele in student1])
print(id(student2))
print(student2)
print([id(ele) for ele in student2])
```

```
39766256
['Will', 28, ['Python', 'C#', 'JavaScript']]
[39413120, 36218340, 39737304]
39767776
['Will', 28, ['Python', 'C#', 'JavaScript']]
[39413120, 36218340, 39773088]
39766256
['Wilber', 28, ['Python', 'C#', 'JavaScript', 'CSS']]
[39758496, 36218340, 39737304]
39767776
['Will', 28, ['Python', 'C#', 'JavaScript']]
[39413120, 36218340, 39773088]
```


39766256
[39413120 , 36218340 , 39737304]
"Will"
28
["Python", "C#", "JavaScript"]
39766256
[39413120 , 36218340 , 39737304]
"Will"
28
["Python", "C#", "JavaScript"]
["Python", "C#", "JavaScript"]
39774408
[39413120 , 36218340 , 39773088]
39766256
[39758496 , 36218340 , 39737304]
"Wilber"
"Will"
28
["Python", "C#", "JavaScript", "CSS"]
["Python", "C#", "JavaScript"]
39774408
[39413120 , 36218340 , 39773088]

元组(tuples)

元组


tuples are delimited by parentheses
t = (1, "cat")
elements are separated by commas

元组

```
fish = (1, 2, "red", "blue")
fish[0] # => 1
fish[0] = 7  # Raises a TypeError
```

元组中的元素不能变更

```
len(fish) # => 4
fish[:2] # => (1, 2)
"red" in fish    # => True
```

常用的序列方法仍然可用

元组解包

```
t = 12345, 54321, 'hello!'
print(t) # (12345, 54321, 'hello!')
type(t)   # => tuple

x, y, z = t
x # => 12345
y # => 54321
z # => ‘hello!’
```

逗号分隔的值合并成一个元组

交换赋值

Have

```
x = 5
y = 6
```

Want

```
x = 6
y = 5
```

First, y, x are packed into the tuple (6, 5)

```
x, y = y, x
```

Then, (6, 5) is unpacked into the variables x and y, respectively

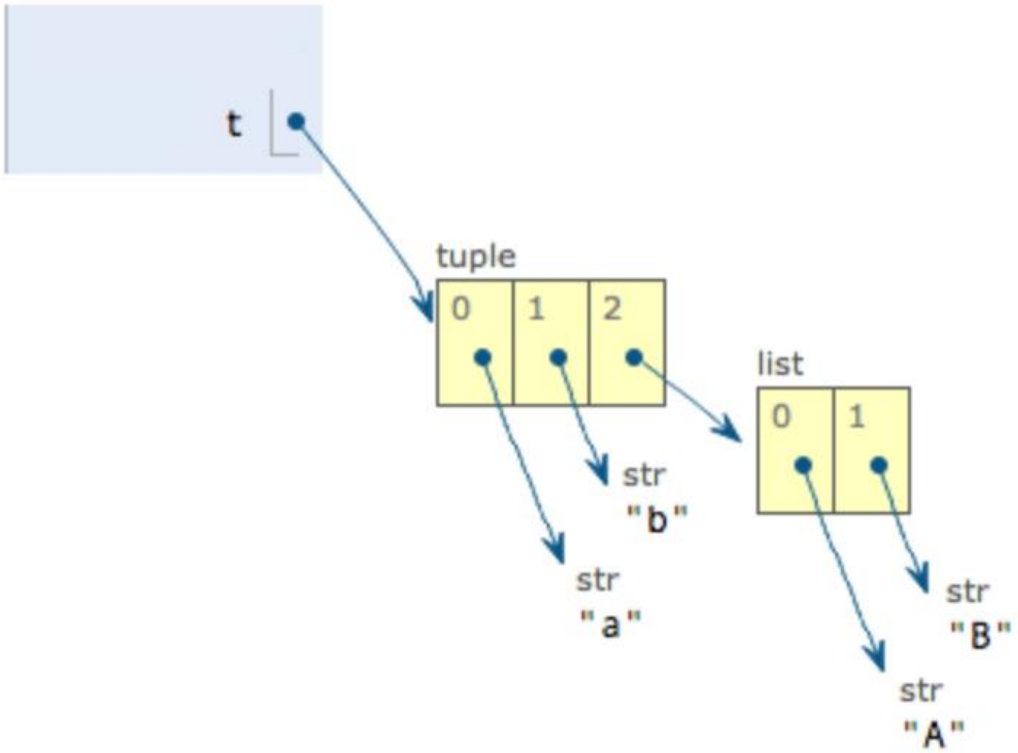
元组

```
empty = ()
number = (1)
t = (1,)          # Note: number != t
len(empty)        # => 0
len(t)            # => 1


v = ([1, 2, 3], ['a', 'b', 'c'])
v[0].append(4)
v     # => ([1, 2, 3, 4], ['a', 'b', 'c'])
```

tuples contain (immutable) references to underlying objects!

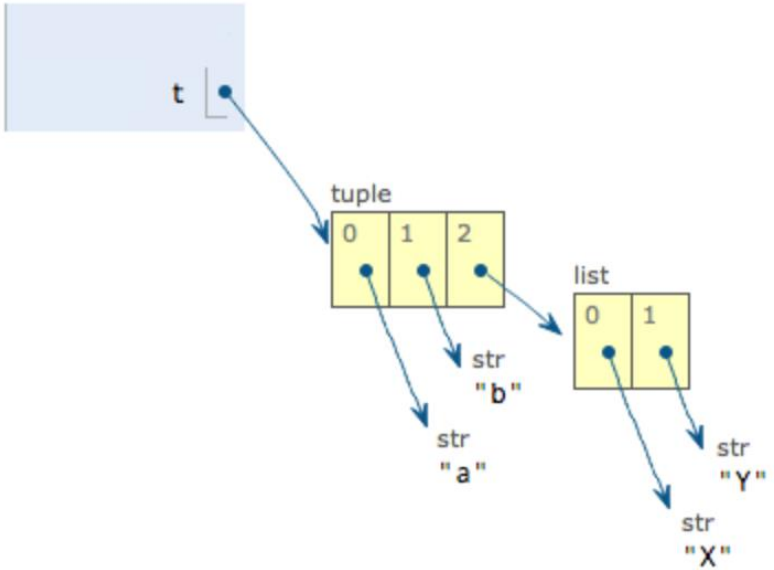
元组

```
t = ('a', 'b', ['A', 'B'])
```



元组

```
t = ('a', 'b', ['A', 'B'])
t[2][0] = 'X'
t[2].remove('B')
t[2].append('Y')
t # => ('a', 'b', ['X', 'Y'])
```



字典(dictionaries)

字典

- 通过任意键信息查找一组数据中值信息的过程叫映射，Python语言中通过字典实现映射。Python语言中的字典可以通过大括号({})建立，建立模式如下：

  {<键1>:<值1>, <键2>:<值2>, … , <键n>:<值n>}

- 其中，键和值通过冒号连接，不同键值对通过逗号隔开。

字典

```
empty = {}
type(empty)      # => <class ‘dict’>
empty == dict() # => True


a = dict(one=1, two=2, three=3)
b = {"one": 1, "two": 2, "three": 3}
a == b     # => True
```

字典索引

```
d = {"one": 1, "two": 2, "three": 3}

# Get
d['one']  # => 1
d['five'] # raises KeyError

# Set
d['two'] = 22 # Modify an existing key
d['four'] = 4 # Add a new key
```

字典

```
d = {"CS":[106, 107, 110], "MATH": [51, 113]}
d["COMPSCI"]   # raises KeyError
```

use get() method to avoid the KeyError

```
d.get("CS")    # => [106, 107, 110]
d.get("PHIL") # => None (not a KeyError!)
```

```
english_classes = d.get("ENGLISH", [])
num_english = len(english_classes)
```

works even if there were no English classes in our dictionary!

字典删除

```
d = {"one": 1, "two": 2, "three": 3}
```

```
del d["one"]
```

raises KeyError if invalid key

```
d.pop("three", default) # => 3
```

remove and return d['three'] or default value if not in the map

```
d.popitem() # => ("two", 2)
```

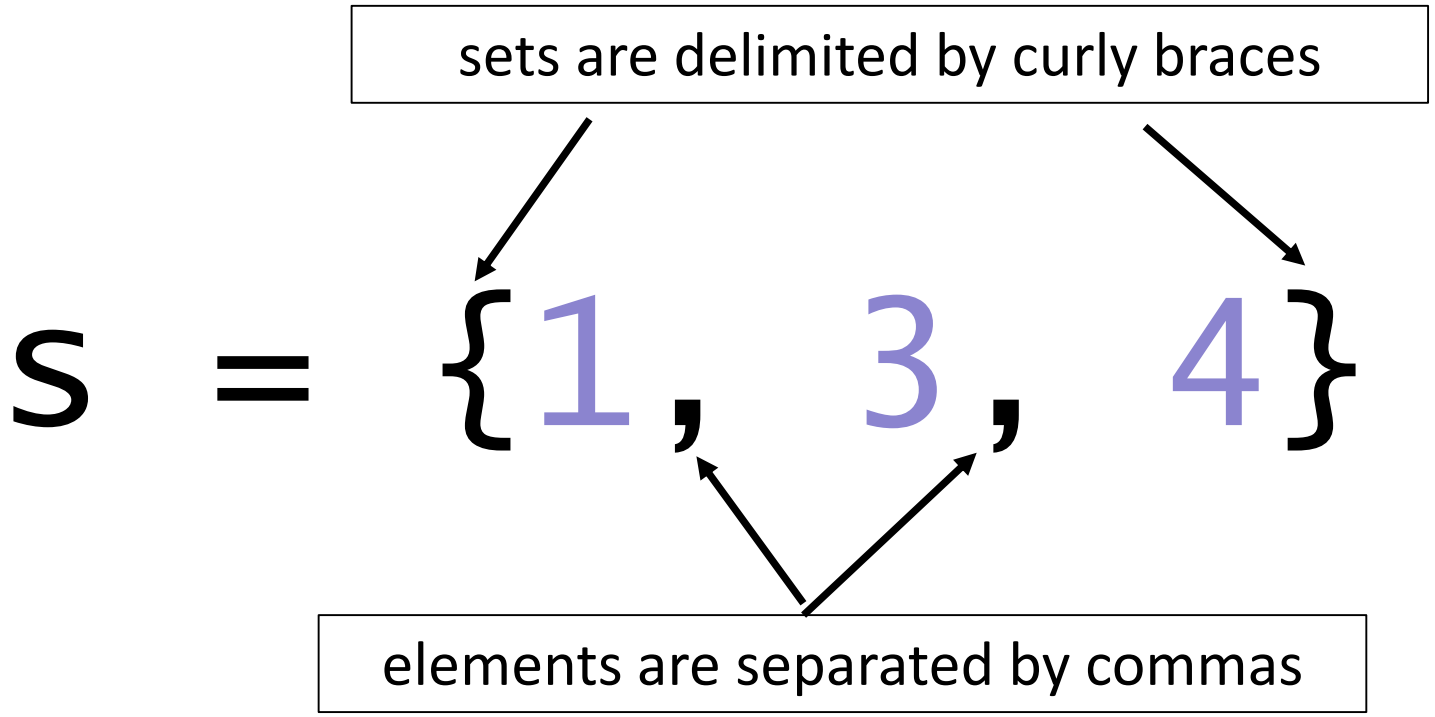
remove and return arbitrary (key, value) pair. Useful for destructive iteration

字典操作

```
len(d)

key in d  # equiv. to `key in d.keys()`

value in d.values

d.copy()

d.clear()

# equiv. to `for key in d.keys():`
for key in d:
    print(key)
```

集合(sets)

集合


sets are delimited by curly braces
s = {1, 3, 4}
elements are separated by commas

集合操作

```
empty_set = set()
set_from_list = set([1, 2, 1, 4, 3])    # => {1, 3, 4, 2}

basket = {"apple", "orange", "apple", "pear", "banana"}
len(basket)                  # => 4
"orange" in basket           # => True
"crabgrass" in basket        # => False

for fruit in basket:
    print(fruit, end='/')
# => pear/banana/apple/orange/
```

why not {}?

O(1) membership testing

集合操作

```
a = set("mississippi")  # {'i', 'm', 'p', 's'}

a.add('r')
a.remove('m') # raises KeyError if 'm' is not present
a.discard('x') # same as remove, except no error

a.pop()      # => 's' (or 'i' or 'p')
a.clear()
len(a)       # => 0
```

集合操作

```
a = set("abracadabra") # {'a', 'r', 'b', 'c', 'd'}
b = set("alacazam") # {'a', 'm', 'c', 'l', 'z'}

# Set difference
a - b  # => {‘r', 'd', 'b'}

# Union
a | b  # {'a', 'c', 'r‘, 'd', 'b', 'm', 'l', 'z'}

# Intersection
a & b # => {'a', 'c'}

# Symmetric difference
a ^ b # => {'r', 'd', 'b', 'm', 'z', 'l'}
```

循环

字典元素遍历

```
knights = {'gallahad': 'the pure', 'robin':
'the brave'}
for k, v in knights.items()
    print(k, v)

# =>
# gallahad the pure
# robin the brave
```

zip

```
questions = ['name', 'quest', 'favorite color']
answers = ['Lancelot', 'To seek the holy grail', 'Blue']
for q, a in zip(questions, answers):
    print('What is your {0}? {1}.'.format(q, a))
```

The zip() function generates pairs of entries from its arguments

```
# =>
# What is your name? Lancelot.
# What is your quest? To seek the holy grail.
# What is your favorite color? Blue.
```

反向遍历

```
for i in reversed(range(1, 10, 2)):
    print(i, end=', ')

# =>
# 9, 7, 5, 3, 1,
```

To loop over a sequence in reverse, first specify the sequence in a forward direction, and then call the reversed() function

排序遍历

```
basket = ['pear', 'banana', 'orange', 'pear',
'apple']
for fruit in sorted(basket):
    print(fruit)

# =>
# apple
# banana
# orange
# pear
# pear
```

sorted() function returns a new sorted list while leaving the source unaltered

推导式(comprehensions)

例子

```
squares = []
for x in range(100):
    squares.append(x**2)

print(squares[:5] + squares[-5:])
# [0, 1, 4, 9, 16, 9025, 9216, 9409, 9604,
9801]
```

推导式

```
[x ** 2
for x in
range(100)]
```

推导式

square brackets indicate that we're building a list

loop over the specified iterable

```
[f(xs) for xs in iter]
```

apply some operations to the loop variable(s) to generate new list elements

推导式

```
[f(xs) for xs in iter if pred(xs)]
```

only keep elements that satisfy a predicate condition

列表推导式例子

```
[word.lower() for word in sentence]
[word for word in sentence if len(word) > 8]


[(x, x ** 2, x ** 3) for x in range(10)]
[(i,j) for i in range(5) for j in range(i)]
```

练习

```
[0, 1, 2, 3] -> [1, 3, 5, 7]
[3, 5, 9, 8] -> [True, False, True, False]
range(10) -> [0, 1, 4, 9, …, 81]


["apple", "orange", "pear"] -> ["A", "O", "P"]
["apple", "orange", "pear"] -> ["apple",
"pear"]
[“apple”, “orange”, “pear”] ->
[(“apple”, 5), (“orange”, 6),
("pear"  4)]
```

其他推导式

```
# Dictionary Comprehensions
 key_func(vars):val_func(vars) for vars in
iterable}
{v:k for k, v in d.items()}

# Set Comprehensions
{func(vars) for vars in iterable}
{word for word in hamlet if
is_palindrome(word.lower())}
```

jieba库的使用

jieba库的概述

jieba是Python中一个重要的第三方中文分词函数库

```
>>>import jieba
>>>jieba.lcut("中国是一个伟大的国家")
['中国', '是', '一个', '伟大', '的', '国家']
```

jieba库是第三方库，不是安装包自带，需要通过pip指令安装

```
:\>pip install jieba      # 或者 pip3 install jieba
```

jieba库的解析

函数	描述
jieba.cut(s)	**精确模式**，返回一个可迭代的数据类型
jieba.cut(s, cut_all=True)	**全模式**，输出文本s中所有可能单词
jieba.cut_for_search(s)	**搜索引擎模式**，适合搜索引擎建立索引的分词结果
jieba.lcut(s)	精确模式，返回一个列表类型，建议使用
jieba.lcut(s, cut_all=True)	全模式，返回一个列表类型，建议使用
jieba.lcut_for_search(s)	搜索引擎模式，返回一个列表类型，建议使用
jieba.add_word(w)	向分词词典中增加新词w

jieba库的解析

```
>>>import jieba
>>>jieba.lcut("中华人民共和国是一个伟大的国家")
['中华人民共和国', '是', '一个', '伟大', '的', '国家']
>>>jieba.lcut("中华人民共和国是一个伟大的国家", cut_all=True)
['中华', '中华人民', '中华人民共和国', '华人', '人民', '人民共和国', '共和', '共和国', '国是', '一个', '伟大', '的', '国家']
>>>jieba.lcut_for_search("中华人民共和国是一个伟大的国家")
['中华', '华人', '人民', '共和', '共和国', '中华人民共和国', '是', '一个', '伟大', '的', '国家']
```

文本词频统计

《Hamlet》英文词频统计

```
#e10.1CalHamlet.py
def getText():
    txt = open("hamlet.txt", "r").read()
    txt = txt.lower()
    for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':
        txt = txt.replace(ch, " ")    #将文本中特殊字符替换为空格
    return txt
hamletTxt = getText()
words  = hamletTxt.split()
counts = {}
for word in words:
    counts[word] = counts.get(word,0) + 1
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

《Hamlet》英文词频统计

```
>>>
the          1138
and           965
to            754
of            669
you           550
a             542
i             542
my            514
hamlet       462
in            436
```

观察输出结果可以看到，高频单词大多数是冠词、代词、连接词等语法型词汇，并不能代表文章的含义。进一步，可以采用集合类型构建一个排除词汇库excludes，在输出结果中排除这个词汇库中内容。

```
#e10.2CalHamlet.py
excludes = {"the","and","of","you","a","i","my","in"}
def getText():
    txt = open("hamlet.txt", "r").read()
    txt = txt.lower()
    for ch in '!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~':
        txt = txt.replace(ch, " ")    #将文本中特殊字符替换为空格
    return txt
hamletTxt = getText()
words  = hamletTxt.split()
counts = {}
for word in words:
    counts[word] = counts.get(word,0) + 1
for word in excludes:
    del(counts[word])
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(10):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

《Hamlet》英文词频统计

运行程序后，输出结果如下

```
>>>
to              754
hamlet          462
it              416
that            391
is              340
not             314
lord            309
his             296
this            295
but             269
```

《三国演义》人物出场统计

```
#e10.3CalThreeKingdoms.py
import jieba
excludes = {}
txt = open("三国演义.txt", "r", encoding='utf-8').read()
words  = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:  #排除单个字符的分词结果
        continue
    else:
        counts[word] = counts.get(word,0) + 1
for word in excludes:
    del(counts[word])
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(15):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

《三国演义》人物出场统计

```
>>>
曹操          953
孔明          836
将军          772
却说          656
玄德          585
关公          510
丞相          491
二人          469
不可          440
荆州          425
玄德曰         390
孔明曰         390
不能          384
如此          378
张飞          358
```

《三国演义》人物出场统计

观察输出结果，同一个人物会有不同的名字，这种情况需要整合处理。同时，与英文词频统计类似，需要排除一些人名无关词汇，如“却说”、“将军”等。

```python
#e10.4CalThreeKingdoms.py
import jieba
excludes = {"将军","却说","荆州","二人","不可","不能","如此"}
txt = open("三国演义.txt", "r", encoding='utf-8').read()
words  = jieba.lcut(txt)
counts = {}
for word in words:
    if len(word) == 1:
        continue
    elif word == "诸葛亮" or word == "孔明曰":
        rword = "孔明"
    elif word == "关公" or word == "云长":
        rword = "关羽"
    elif word == "玄德" or word == "玄德曰":
        rword = "刘备"
    elif word == "孟德" or word == "丞相":
        rword = "曹操"
    else:
        rword = word
    counts[rword] = counts.get(rword,0) + 1
for word in excludes:
    del(counts[word])
items = list(counts.items())
items.sort(key=lambda x:x[1], reverse=True)
for i in range(5):
    word, count = items[i]
    print ("{0:<10}{1:>5}".format(word, count))
```

《三国演义》人物出场统计

输出排序前5的单词，运行程序后，输出结果如下：

```
>>>
曹操          1451
孔明          1383
刘备          1252
关羽           784
张飞           358
```

请继续完善程序，排除更多无关词汇干扰，总结出场最多的20个人物都有哪些。这里给一个参考：

曹操（1451）、孔明（1383）、刘备（1252）、关羽（784）、张飞（358）、吕布（300）、赵云（278）、孙权（264）、司马懿（221）、周瑜（217）、袁绍（191）、马超（185）、魏延（180）、黄忠（168）、姜维（151）、马岱（127）、庞德（122）、孟获（122）、刘表（120）、夏侯惇（116）