第2章　Python的基本语法概述

2.1 Python语言的编程规范

2.2 常量、变量与对象

2.3 数据类型

2.4 Python的语句概述

2.5 输入/输出函数

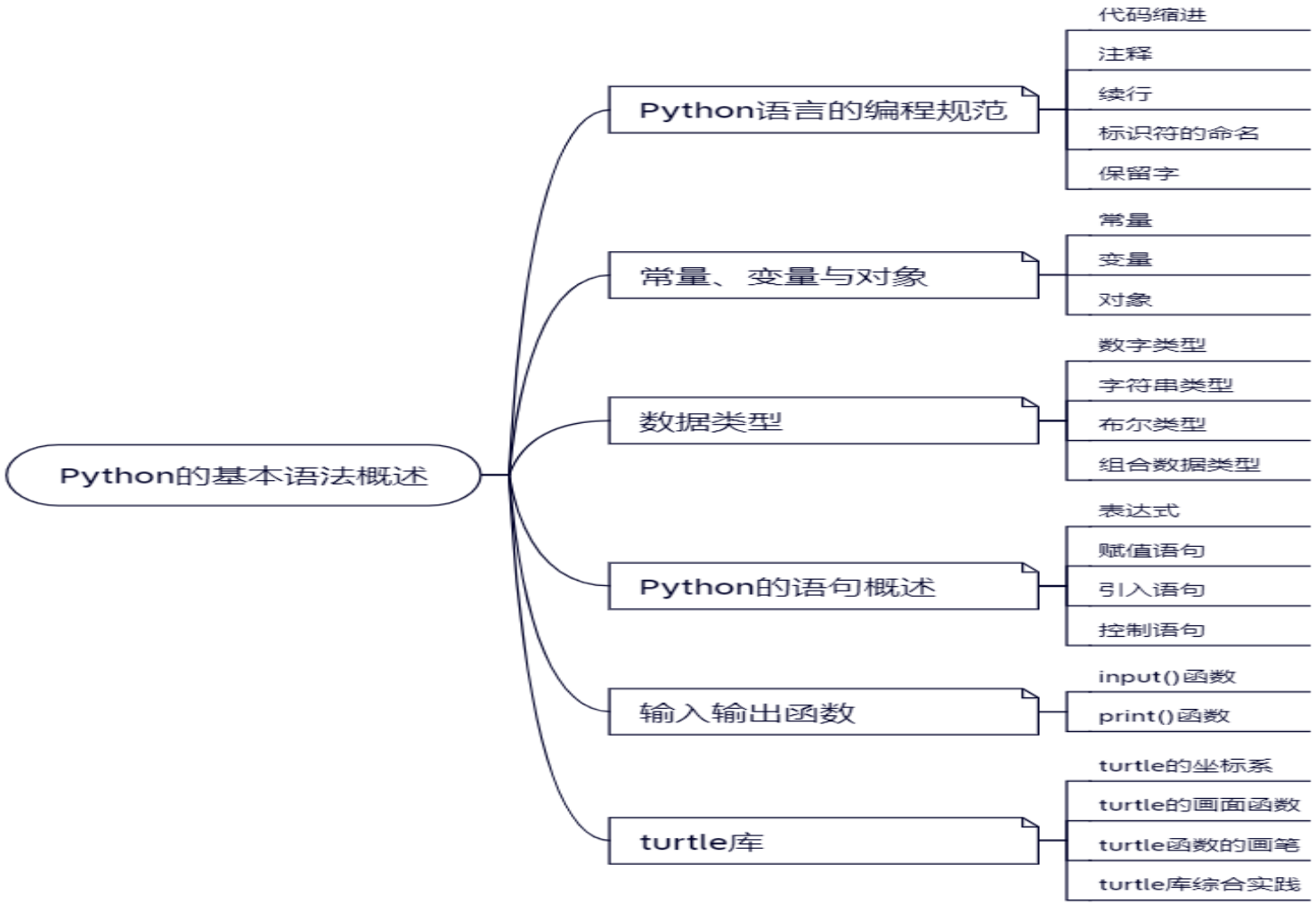
2.6 turtle库

考纲考点

- 程序的基本语法元素：程序的格式框架、缩进、注释、变量、命名、保留字、数据类型、赋值语句、引用语句、控制语句

- 基本输入输出函数：input()、eval()、print()
- Turtle库

知识导图


Python的基本语法概述
Python语言的编程规范
代码缩进
注释
续行
标识符的命名
保留字
常量、变量与对象
常量
变量
对象
数据类型
数字类型
字符串类型
布尔类型
组合数据类型
Python的语句概述
表达式
赋值语句
引入语句
控制语句
输入输出函数
input()函数
print()函数
turtle库
turtle的坐标系
turtle的画面函数
turtle函数的画笔
turtle库综合实践

2.1 Python语言的编程规范

2.1.1代码缩进

- Python语言采用严格的“缩进”来表明程序的格式框架。缩进指每一行代码开始前的空白区域，用来表示代码之间的包含和层次关系。
- 1个缩进 = 4个空格
- 缩进是Python语言中表明程序框架的唯一手段

缩进

- 当表达分支、循环、函数、类等程序含义时，在if、while、for、def、class等保留字所在完整语句后通过英文冒号（:）结尾，表示缩进的开始。
- python程序是依靠代码块的缩进来体现代码之间的逻辑关系的，缩进结束就表示一个代码块结束了。
- 同一个级别的代码块的缩进量必须相同

例2-1 编程，根据输入的年份和月份计算出该月的天数。

代码如下：

```
year=int(input("请输入年份："))
month=int(input("请输入月份："))
if month==2:
        if year % 4 == 0 and year % 100!=0 or
year % 400 ==0:
                days=29
        else:
                days=28
elif month in [4,6,9,11]:
        days=30
else:
        days=31
print("本月天数是：",days)
```

2.1.2 注释

- 注释是代码中的辅助性文字，会被编译或解释器略去，不被计算机执行，一般用于程序员对代码的说明。Python语言采用#表示一行注释的开始，

```
#这是一个单行注释
print("你好，中国！") #这是一个单行注释
```

- 注释可以在一行中任意位置通过“#”开始，其后的本行内容被当作注释，而之前的内容仍然是python执行程序的一部分。

注释

- 多行注释可以用三个单引号（’’’）括起来。

```
"""
这是一个多行注释。
你可以在这里写入多行文本，Python 解释器会忽视它们。
"""
print("你好，中国！")
```

2.1.3 续行

- 续行符可以提高代码的可读性和可维护性。当一行代码过长时，不仅会导致代码难以阅读和修改，还会影响代码的性能。
- 可以使用反斜杠（\）来实现代码的续行

1.交互方式下的续行符“\”，示例代码如下：

```
>>> a=1+2+3\
  +4+5+6+\
  7+8+9
>>> a
45
```

2.文件方式下的续行符“\”，示例代码如下：

```
a=1+2+3+\
  4+5+6+\
  7+8+9
```

- 还可以使用括号续行。Python会自动将这几行代码当作一行代码来处理，文件方式和交互方式下均可使用。

```
a=(1+2+3+
4+5+6+
7+8+9)
```

2.1.4 标识符的命名

- 标识符是用户编程时使用的名字。对标识符命名是编程语言规则的一部分，它规定了如何为变量、函数、类、模块等编程元素取名字。

- 一般采用字母、数字、下划线，甚至汉字等字符及其组合进行命名，但要注意的是，首字符不能是数字，不允许使用空格和特殊字符（如感叹号、问号、冒号等）

  例如：变量名my_variable，函数名_my_function，类名MyClass
  等等

- 标识符对大小写敏感，即china和China是两个不同的名字；对标识符命名还需要注意的是不能与python的保留字相同。

- 续行符可以提高代码的可读性和可维护性。当一行代码过长时，不仅会导致代码难以阅读和修改，还会影响代码的性能。
- 可以使用反斜杠（\）来实现代码的续行

1.交互方式下的续行符“\”，示例代码如下：

```
>>> a=1+2+3\
  +4+5+6+\
  7+8+9
>>> a
45
```

2.文件方式下的续行符“\”，示例代码如下：

```
a=1+2+3+\
  4+5+6+\
  7+8+9
```

2.1.5 保留字

- 保留字，也称为关键字，指被编程语言内部定义并保留使用的标识符。
- 程序员编写程序不能定义与保留字相同的标识符。
- 每种程序设计语言都有一套保留字，保留字一般用来构成程序整体框架、表达关键值和具有结构性的复杂语义等。
- 掌握一门编程语言首先要熟记其所对应的保留字。

- Python 3.x保留字列表（35个）

and	as	assert	async	await
break	class	continue	def	del
elif	else	except	False	finally
for	from	global	if	import
in	is	lambda	nonlocal	not
or	pass	raise	return	try
True	while	with	yield	None

2.2 常量、变量与对象

2.2.1 常量

- 所谓常量，一般是指不需要改变也不能改变的字面值

  例如一个数字5，一个字符串“hello!”，一个列表[1,2,3], 等等。

- 在python中，还会出现一类符号常量，一般用大写字母来表示

  例如：MAX_VALUE、PI

- Python中并没有真正的符号常量，由于 Python中没有强制符号常量类型的定义方式，这些所谓的符号常量类型只是程序员约定俗成的表示方式而已，在实际操作中，其本质还是变量。

```
>>> MAX_VALUE=100
>>> print(MAX_VALUE)
100
>>> MAX_VALUE=200
>>> print(MAX_VALUE)
200
```

2.2.2 变量

- 变量是保存和表示数据值的一种语法元素，在程序中十分常见。顾名思义，变量的值是可以改变的，能够通过赋值（使用等号=表达）方式被修改，例如（注意空格）：

```
>>> x=5
>>> x
5
>>> x="中国"
>>> x
'中国'
```

2.2.3 对象

- 在Python中一切皆对象。
- 每个对象由：标识（identity），类型(type)，值(value)组成

- 标识用于唯一标识对象，通常对应于对象在计算机内存中的地址。使用内置函数id(obj)可返回对象的标识

- 类型用于表示对象存储的“数据”类型。可以限制对象的取值范围以及可执行的操作。使用内置函数type(obj)可返回对象的所属类型

- 值表示对象所存储的数据的信息。使用print(obj)可以直接打印出值

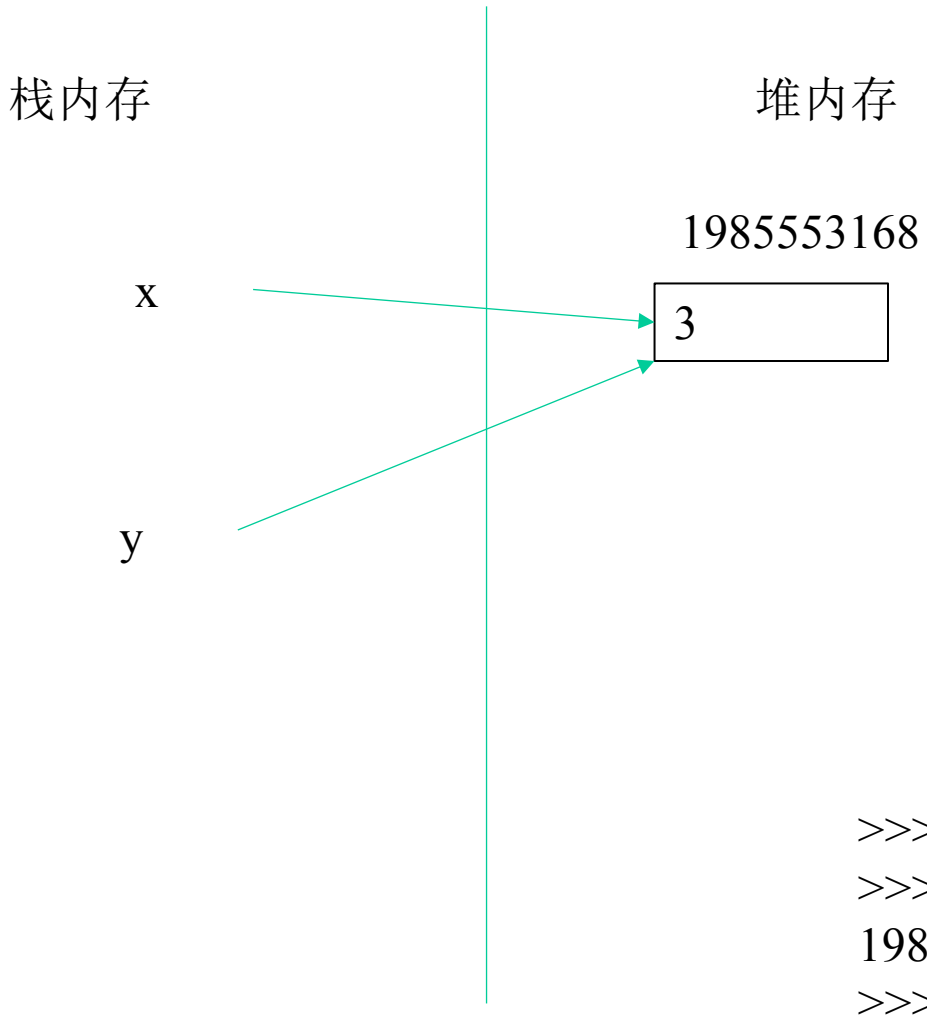
- 其本质：一个内存块，拥有特定的值，支持特定类型的相关操作
- 特别说明：

在python中，变量其实是对象的引用，因为变量存储的是对象的地址，变量通过地址引用了“对象”。

变量位于：栈内存

对象位于：堆内存

对象没有被引用的时候，会被回收到垃圾堆

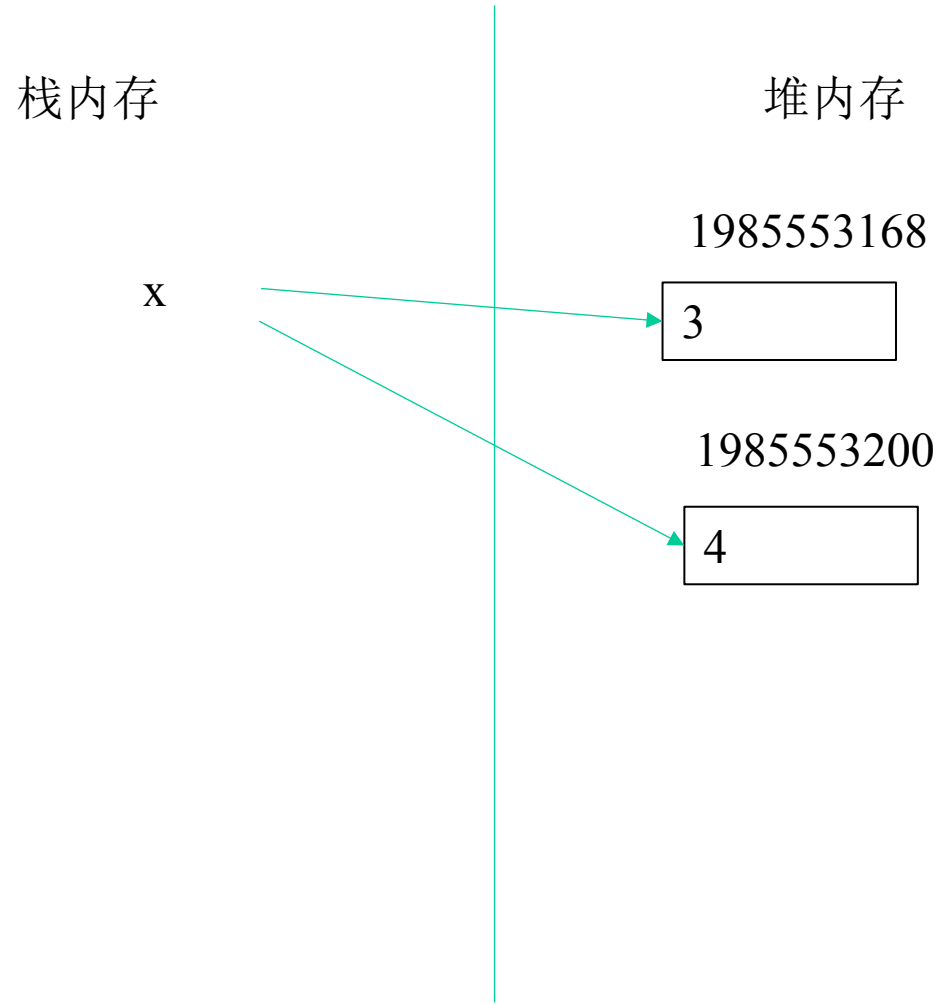



```
>>> x=3
>>> id(x)                    #查看x的引用地址
1985553168
>>> id(3)                    #查看对象3的地址
1985553168
```

不可变对象

- 在Python中，对象可以分为可变对象和不可变对象。

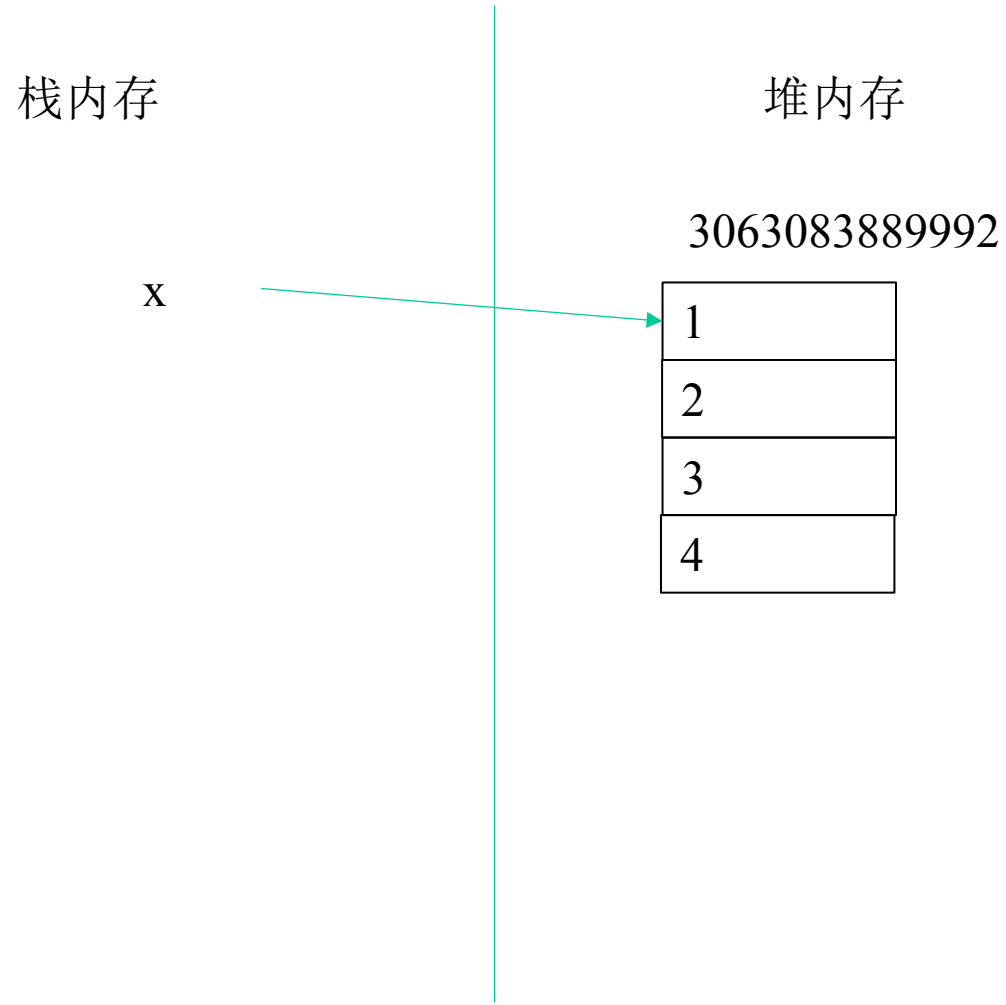
- 不可变对象是指该对象所指向的内存中的值不能被改变。当改变某个变量时，由于其所指的值不能被改变，因此会开辟一个新的地址，变量再指向这个新的地址。在Python中，数值类型（int和float）、字符串（str）和元组（tuple）都是不可变对象。




```
>>> x=3
>>> id(x)
1985553168
>>> id(3)
1985553168
>>> x=x+1
>>> id(x)
1985553200
>>> id(3)
1985553168
>>> id(4)
1985553200
```

可变对象

- 可变对象是指该对象所指向的内存中的值可以被改变。变量（准确的说是引用）改变后，实际上是其所指的值直接发生改变，并没有发生复制行为，也没有开辟出新地址，通俗点说就是原地改变。在Python中，列表（list）、字典（dict）和集合（set）是可变对象。




```
>>> x=[1,2,3]                #x为列表
>>> id(x)                    #查看x的引用地址
3063083889992
>>> x.append(4)              #给x增加元素4
>>> x
[1, 2, 3, 4]                 #此时x值发生变化
>>> id(x)                    #x的引用地址没变
3063083889992
```

- Python中，可变对象和不可变对象其值都能够被引用，就是都可以做“读”操作。但是“写”操作只能对可变对象进行。

2.3 数据类型

- Python语言支持多种数据类型，最简单的包括数字类型、字符串类型，略微复杂的包括元组类型、集合类型、列表类型、字典类型等。

2.3.1 数字类型

- 表示数字或数值的数据类型称为数字类型，Python语言提供3种数字类型：整数、浮点数和复数，分别对应数学中的整数、实数和复数。

整数

- 一个整数值可以表示为十进制、十六进制、八进制和二进制等不同进制形式。

在Python3中，整数没有大小限制，可以是任意大小的整数。

十进制：　1010

十六进制：0x3F2

八进制：　0o1762

二进制：　0b001111110010

浮点数

- 一个浮点数可以表示为带有小数点的一般形式，也

可以采用科学计数法表示，只有十进制形式。

例如：浮点数123.456，两种表示方式如下：

一般形式：　　123.456

科学计数法：　　1.23456e2

整数与浮点数混合运算时，表达式自动转成浮点数

复数

- Python中的复数由实部和虚部组成，可以使用a+bj或者complex(a,b)表示，其中a和b都是实数，且a称为复数的实部，b称为复数的虚部。可以使用j或J表示复数的虚部

```
>>> x=5+6j
>>> y=7+8j
>>> x+y
(12+14j)
>>> x*y
(-13+82j)
```

2.3.2 字符串类型

- Python语言中，字符串类型数据主要用于处理一些文本类信息，有字符串类型的常量和变量，单个字符也是字符串。
- Python使用单引号、双引号、三个单引号，三个双引号作为定界符表示字符串，且不同的定界符之间可以相互嵌套使用

```
>>> x="456"
>>> type(x)
<class 'str'>
>>> x='happy'
>>> type(x)
<class 'str'>
>>> x='"He said,"Happy Birthday"""      #为了输出双引号，引号嵌套使用
>>> x
'He said,"Happy Birthday"'
```

2.3.3 布尔类型

- 布尔类型数据只有两个值：True和False。布尔类型的特点是占用内存空间小，使用灵活，常用于if语句、循环语句、条件表达式中，进行逻辑判断和条件判断

- 布尔类型可以与其他数据类型进行转换，如与整数类型（int）进行转换时，True对应1，False对应0。

```
>>> True and True                    #与运算
True
>>> True and False
False
>>> False and True
False
>>> True or True                     #或运算
True
>>> True or False
True
>>> False or  True
True
>>> not True                         #非运算
False
>>> not False
True
>>> 3*True                           #布尔值可以直接参
与整数运算
3
>>> 3*False
0
```

- and是逻辑与运算，or是逻辑或运算，not是逻辑非运算

2.3.4 组合数据类型

- 在Python中，组合数据类型是指可以包含不同类型的元素的数据结构。Python中的组合数据类型主要包括列表（list）、元组（tuple）、字典（dictionary）和集合（set）。

列表

- 列表是一种有序的元素集合，可以随时添加或删除其中的元素。列表中的元素可以是任何类型，例如字符串、整数、浮点数、其他列表等。列表使用“[]”作为定界符。

```
>>> x=[36,"happy",3.14]
>>> type(x)
<class 'list'>
```

元组

- 元组与列表类似，是一个有序的元素集合，但元组是不可变对象，一旦创建就不能更改。元组中的元素也可以是任何类型。元组使用“（）”作为定界符。

```
>>> x=(56,"happy",[3,5,7])
>>> x
(56, 'happy', [3, 5, 7])
>>> type(x)
<class 'tuple'>
```

字典

- 字典是无序的键值对集合。字典中的元素以键值对的形式存在，你可以通过键来访问其对应的值。字典中的键必须是唯一的。值可以是任何类型，包括列表、元组、字典等。字典使用“{}”作为定界符。

```
>>> x={'name':'John','age':18,'hobbies':['fishing','reading']}
>>> x
{'name': 'John', 'hobbies': ['fishing', 'reading'], 'age': 18}
>>> type(x)
<class 'dict'>
```

集合

- 集合是一个无序的唯一元素集合。集合中的元素都是唯一的，不会重复。集合支持一些数学集合的操作，如并集、交集、差集等。集合中的元素只能是不可变对象。集合也使用“{}”作为定界符。

```
>>> x={3.14,4,'abc',(4,6)}
>>> x
{3.14, 4, (4, 6), 'abc'}
>>> type(x)
<class 'set'>
```

2.4 Python的语句概述

2.4.1 表达式

- 产生或计算新数据值的代码片段称为表达式。 表达式类似数学中的计算公式，以表达单一功能为目的，运算后产生运算结果，运算结果的类型由操作符或运算符决定。

- 表达式一般由数据和操作符等构成，这是构成Python语句的重要部分，单个常量或变量可以看作最简单的表达式。