Python及其应用

主讲人：钱惠敏
E-mail: amandaqian@hhu.edu.cn

第2讲 数据类型与基本运算

2.1 引言

2.2 数字类型

2.3 常量、变量与关键字

2.4 语句与表达式

2.5 运算符和操作对象

2.6 字符串

2.7元组和列表

2.1 引言

- 类型：编程语言对数据的一种划分。
- Python3 中有六个标准的数据类型

Number（数字或数值）、String（字符串）、List（列表）、Tuple（元组）、Sets（集合）、Dictionary（字典）

- Python3支持三种不同的数字类型

整型（int）、浮点型（float）、复数（complex）

2.2 数字类型

- 三种数字类型

✓整型(int)

没有取值范围限制，注意区分除法的运算符 / //(地板除)

地板除就是取整

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)

2.2 数字类型(续1)

➢ 三种数字类型(2)

✓浮点型(float)

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。

示例

0.0, 33., -3.13

96e4, 4.3e-3, 9.6E5 （科学计数法）

2.2 数字类型（续2）

- 三种数字类型(3)

✓复数(complex)

Z=a+bj 或 Z=a+bJ

示例

2+3j, 5-3.13j

z=6e4+4.3e-3j

2.2 数字类型（续3）

- 三种数字类型(4)

✓三种类型的扩展关系：整数→浮点数→复数

✓数据类型的转换

int(x)，float(x), complex(x),
complex(x, y)—实部为x，虚部为y

示例

int(3.13)=3
float(3)=3.0
complex(3)=3+0j

✓数据类型的判断 type(x)

2.2 数字类型（续4）

- 数字类型的运算

运算符和运算函数	操作含义
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	不大于x与y之商的最大整数
x%y	x与y之商的余数
+x	x
-x	x的负值
x**y	x的y次幂
abs(x)	x的绝对值
divmod(x,y)	(x//y,x%y)
pow(x,y)	x的y次幂

2.3 常量、变量与关键字

- 常量，通常用大写表示常量。

  圆周率π：PI

  自然常数e：E

- 变量
  - ✓ 变量名可以是由数字、字符、下划线组成的任意长度的字符串，但必须以字母开头，区分大小写。
  - ✓ 变量的使用环境宽松，采用“=”，可把任意数据类型赋值给变量。

2.3 常量、变量与关键字（续1）

➢关键字（33个）

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

2.4 语句与表达式

- 语句：Python解释器可以运行的一个代码单元，也可以理解为可以执行的命令。

目前已经使用过两种语句：print打印语句和赋值语句。

- 表达式：值、变量和操作符的组合。单独一个值也被看作一个表达式。单独的变量也可以看作一个表达式。

- 表达式是某事，而语句是做某事，说的通俗点就是告诉计算机做什么。

2.5 运算符和操作对象

- 运算符：

算术运算符、比较（关系）运算符、赋值运算符、逻辑运算符、位运算符、成员运算符、身份运算符。

- 运算符优先级

- 操作对象：由运算符连接起来的对象

2.5.1 算术运算符

a=10; b=5

运算符	描述	实例
+	加：两个对象相加	a + b =15
-	减：得到负数或是一个数减去另一个数	a - b =5
*	乘：两个数相乘或是返回一个被重复若干次的字符串	a * b =50
/	除：x除以y	a / b =2
%	取模：返回除法的余数	b % a =0
**	幂：返回x的y次幂	a**b =100000
//	取整除（地板除）：返回商的整数部分	9//2 = 4 , 9.0//2.0 =4.0

2.5.2 比较运算符

a=10; b=20

运算符	描述	实例
==	等于：比较对象是否相等	(a == b) 返回 False
!=	不等于：比较两个对象是否不相等	(a != b) 返回 True
>	大于：返回x是否大于y	(a > b) 返回 False
<	小于：返回x是否小于y	(a < b) 返回 True
>=	大于等于：返回x是否大于等于y	(a >= b) 返回 False
<=	小于等于：返回x是否小于等于y	(a <= b) 返回 True

2.5.3 赋值运算符

运算符	描述	实例
-=	减法赋值	c -= a 等效于 c = c – a
*=	乘法赋值	c *= a 等效于 c = c * a
/=	除法赋值	c /= a 等效于 c = c / a
%=	取模赋值	c %= a 等效于 c = c % a
**=	幂赋值	c **= a 等效于 c = c ** a
//=	取整（地板）除赋值	c //= a 等效于 c = c // a

2.5.4 位运算符

a=60; b=13

运算符	描述(二进制)	实例
&	按位与	(a & b) , 12
|	按位或	(a | b) , 61
^	按位异或	(a ^ b) , 49
~	按位取反	(~a) , -61
<<	左移动	(a << 2), 240
>>	右移动	(a >> 2), 15

2.5.5 逻辑运算符

a=10; b=20

运算符	逻辑表达式	描述	实例
And	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
Or	x or y	布尔“或” - 如果 x 是非 0，它返回 x 的值，否则它返回 y 的计算值。	(a or b) 返回 10。
Not	not x	布尔“非” - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b)返回 False

2.5.6 成员运算符

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x在y序列中,如果x在y序列中返回True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x不在y序列中,如果x不在y序列中返回 True。

2.5.7 身份运算符

运算符	描述	实例
is	is判断两个标识符是不是引用自一个对象	x is y,如果id(x)等于id(y) , is 返回结果1
is not	**is not用于判断两个标识符是不是引用自不同对象**	**x is not y,如果id(x)不等于id(y).is not 返回结果1**

2.5.8 运算符优先级（从高至低）

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not or and	逻辑运算符

2.6 字符串

- 字符串：用引号('或")来创建字符串。
- 字符串索引：

从左至右，从0开始；从右至左，从-1开始

h	e	l	l	o		w	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

访问方式 <string>[<索引>] #访问字符

<string>[<start>:<end>] #访问子串

2.6 字符串(续1)

➢字符串连接 + *

加法操作 连接两个字符串

乘法操作 复制字符串

➢字符串操作

len(string) --返回字符串的长度

str() – 将其它数据类型转换为字符串

for <var> in <string>:

操作

--循环遍历字符串中的每个字符

转义字符表

转义字符	描述	转义字符	描述
\(在行尾时)	续行符	\n	换行
\\	反斜杠符号	\v	纵向制表符
\'	单引号	\t	横向制表符
\"	双引号	\r	回车
\a	响铃	\f	换页
\b	退格(Backspace)	\oyy	八进制数，yy代表的字符，例如：\o12代表换行
\e	转义	\xyy	十六进制数，yy代表的字符，例如：\x0a代表换行
\000	空	\other	其它的字符以普通格式输出

2.6.1 字符串格式化

➢ 字符串格式化操作符：百分号%

```
>>> print ('hello,%s' % 'world')
hello,world
>>> print ('小智今年%s岁了' % 10)
小智今年10岁了
```

待格式化的字符串

格式化的值

注：“%s” 称为转换说明符或占位符，它标记了需要放置转换值的位置。

格式化符号

符号	描述	符号	描述
%c	格式化字符及其ASCII码	%f	格式化浮点数字，可指定小数点后的精度
%s	格式化字符串	%e	用科学计数法格式化浮点数
%d	格式化整数	%E	作用同%e，用科学计数法格式化浮点数
%u	格式化无符号整型	%g	%f和%e的简写
%o	格式化无符号八进制数	%G	%f 和 %E 的简写
%x	格式化无符号十六进制数	%p	用十六进制数格式化变量的地址
%X	格式化无符号十六进制数（大写）		

2.6.1 字符串格式化（续）

- 字符串格式化的值是任意的

```
>>> print ('今年是%s年，中国女排夺得本届奥运会%s，中国共获%d得枚金牌'%('2016','冠军',26))
```

格式化的值是元组

今年是2016年，中国女排夺得本届奥运会冠军，中国共获26得枚金牌

注：“%s” 称为转换说明符或占位符，它标记了需要放置转换值的位置。

2.6.2 字符串方法

操作	含义
+	连接
*	重复
<string>[ ]	索引
<string>[:]	剪切
len(<string>)	长度
<string>.upper()	字符串中字母大写
<string>.lower()	字符串中字母小写
<string>.strip()	去两边空格及去指定字符
<string>.split()	按指定字符分割字符串为数组
<string>.join()	连接两个字符串序列
<string>.find()	搜索指定字符串
<string>.replace()	字符串替换
for <var> in <string>	字符串迭代

2.6.2 字符串方法（续1）

➢find()方法检测字符串中是否包含指定子字符串

string.find(str, beg=0, end=len(string))

str，指定检索的子字符串，

beg，开始索引，默认为0，

end，结束索引，默认为字符串的长度。

如果包含子字符串，返回子串所在位置的最左端索引，如果没有找到则返回-1。

```
>>> field='do it now'
>>> field.find('now')
6
>>> field.find('python')
-1
```

2.6.2 字符串方法（续2）

➢len()方法返回字符串的长度

string.find(str, beg=0, end=len(string))

str，指定检索的子字符串，

beg，开始索引，默认为0，

end，结束索引，默认为字符串的长度。

如果包含子字符串，返回子串所在位置的最左端索引，如果没有找到则返回-1。

```
>>> field='do it now'
>>> field.find('now')
6
>>> field.find('python')
-1
```

2.6.2 字符串方法（续3）

- join()方法将序列中的元素以指定的字符连接生成一个新的字符串。

str.join(sequence)

```
>>> dirs='','home','data','hdfs'
>>> print('路径：','/'.join(dirs))
路径： /home/data/hdfs
>>> field=['1','2','3','4','5']
>>> print('连接字符串列表：', '+'.join(field))
连接字符串列表： 1+2+3+4+5
```

2.6.2 字符串方法（续4）

➢lower()方法转换字符串中所有大写字符为小写。

str.lower()

```
>>> field='DO IT NOW'
>>> print('调用lower得到字符串：',field.lower())
调用lower得到字符串： do it now
>>> greeting='Hello,World'
>>> print('调用lower得到字符串：',greeting.lower())
调用lower得到字符串： hello,world
```

2.6.2 字符串方法（续5）

➢upper()方法将字符串中的小写字母转为大写字母。

str.upper()

```
>>> field='do it now'
>>> print('调用upper得到字符串：',field.upper())
调用upper得到字符串： DO IT NOW
>>> greeting='Hello,World'
>>> print('调用upper得到字符串：',greeting.upper())
调用upper得到字符串： HELLO,WORLD
```

2.6.2 字符串方法（续6）

➢swapcase() 方法用于对字符串的大小写字母进行转换，将字符串中大写转换为小写，小写转换为大写。

str.swapcase()

```
>>> field='Just do  it, NOW'
>>> print('原字符串：', field)
原字符串： Just do  it, NOW
>>> print('调用swapcase方法后得到字符串：', field.swapcase())
调用swapcase方法后得到字符串： jUST DO  IT,now
```

2.6.2 字符串方法（续7）

➢replace()方法把字符串中的old（旧字符串）替换成new(新字符串)，如果指定第三个参数max，则替换不超过max次。

str.replace(old, new [, max])

```
>>> field='do it now, do right now'
>>> print('原字符串：',field)
原字符串： do it now, do right now
>>> print('新字符串：', field.replace('do','Just do'))
新字符串： Just do it now, Just do right now
```

2.6.2 字符串方法（续8）

- split()方法通过指定分隔符st(默认为空格)，对字符串进行切片，如果参数num 有指定值，则仅分隔 num 个子字符串。它是join的逆方法。

str.split(st="", num=string.count(str))

```
>>> field='do it now'
>>> print('不提供任何分割符分隔后的字符串：', field.split())
不提供任何分割符分隔后的字符串： ['do', 'it', 'now']
>>> print('根据i分隔后的字符串：', field.split('i'))
根据i分隔后的字符串： ['do ', 't now']
```

2.6.2 字符串方法（续9）

➢strip()方法用于移除字符串头尾指定的字符chars（默认为空格）。

str.strip([chars])

```
>>> field='----do it now----'
>>> print('原字符串： ', field)
原字符串：  ----do it now----
>>> print('新字符串： ', field.strip('-'))
新字符串：  do it now
```

2.6.2 字符串方法（续10）

- translate()方法根据table给出的翻译表(包含256个字符)转换字符串的字符, 要过滤掉的字符放到deletechars参数中。

str.translate(table [, deletechars])

maketrans() 方法用于创建intab的字符至outtab的字符的映射关系转换表

```
>>> intab='adfas'
>>> outtab='12345'
>>> trantab=str.maketrans(intab,outtab)
>>> st='just do it'
>>> print('st调用translate方法后：', st.translate(trantab))
st调用translate方法后： ju5t 2o it
```

2.7 元组和列表

- 通用序列操作

索引（indexing）、分片（slicing）、序列相加（adding）、乘法（multiplying）、成员资格、长度、最小值和最大值。

2.7.1 索引

- 序列是Python中最基本的数据结构。序列中的每个元素都分配一个数字，代表它在序列中的位置，或索引，第一个索引是0，第二个索引是1，依此类推。

- 序列中所有的元素都是有编号的，可以通过编号分别对序列的元素进行访问。

- 从左向右索引称为正数索引，从右向左称为负数索引

2.7.2 分片

➢分片：对一定范围内的元素进行访问，分片通过冒号相隔的两个索引来实现。

sequence[indexStart : indexEnd : stride]

注：第一个索引（indexStart）的元素是包含在分片内的，第二个（ indexEnd ）则不包含在分片内，stride是步长

2.7.3 序列相加

- 用加号进行序列的连接

✓数字序列可以和数字序列通过加号连接，连接后的结果还是数字序列，字符串序列也可以通过加号连接，连接后的结果还是字符串序列。

✓只有类型相同的序列才能通过加号进行序列连接操作，不同类型的序列不能通过加号进行序列连接操作。

2.7.4 乘法

- 序列乘法：用一个数字x乘以一个序列会生成新的序列，在新的序列中，原来的序列将被重复x次。

✓序列的乘法便于实现重复操作、空列表和None初始化操作

2.7.5 成员资格

- in运算符用于检验某个条件是否为真，检查一个值是否在序列中，并返回检验结果，检验结果为真返回True，结果为假则返回False。

2.7.6 长度、最小值和最大值

- Python提供了长度、最大值和最小值的内建函数，对应的内建函数分别为len、max和min。

2.7.7 元　组

元组（tuple）的概念

- 元组是包含多个元素的数据类型，元组的元素不能修改。
- 创建元组的方法很简单：如果你使用逗号分隔了一些值，那么你就自动创建了元组。
- 元组可以是空的　　t2=()
- 元组可以只包含一个元素　　t3=123,
- 元组外侧可以使用括号，也可以不使用

2.7.7 元　组（续1）

元组的特点

- 元组中元素可以是不同类型，一个元组也可以作为另一个元组的元素，此时，作为元素的元组需要增加括号。
- 元组定义后不能更改，也不能删除。
- tuple函数：参数为序列，把参数转换为元组；参数是元组，原样返回参数。

2.7.7 元　组（续2）

➢元组的基本操作：

✓访问元组：使用下标索引来访问元组中的值

✓修改元组：元组中的元素值是不允许修改的，但可以对元组进行连接组合

✓删除元组：元组中的元素值是不允许删除的，但可以使用del语句来删除整个元组

✓元组索引、截取：可以访问元组中的指定位置的元素，也可以截取索引中的一段元素

2.7.7 元　组（续3）

- 元组内置函数：
  - ✓len(tuple)：计算元组元素个数
  - ✓max(tuple)：返回元组中元素最大值
  - ✓min(tuple)：返回元组中元素最小值
  - ✓tuple(seq)：将列表转换为元组

2.7.8 列表

列表（list）的概念

- 列表是有序的元素集合；列表中每个元素类型可以不一样

  列表用[]或list()函数创建

- 列表与元组和字符串的不同：列表是可变的（mutable），即列表的内容是可改变的。

2.7.8 列表 (续1)

- 元素赋值：通过的元素，并对该位置元素重新编号来标记某个特定位置赋值

  注：可以对一个列表中的元素赋不同类型的值

- 增加元素：通过append() 在列表末尾添加新的对象

  list.append(obj)

  list --- 列表，obj --- 需要添加到list列表末尾的对象

- 删除元素：使用del删除列表中的元素

- 分片赋值：通过分片赋值直接对列表做变更

  注：可以对一个列表中的元素赋不同类型的值

2.7.8 列表（续2）

列表的方法

- 方法是一个与某些对象有紧密联系的函数

- 方法的调用语法如下：

  对象.方法(参数)

2.7.8 列表（续3）

- list.append(obj) — 在列表末尾添加新对象
- list.count(obj) — 统计某个元素在列表中出现的次数

```
field=list('hello,world’)
field.count('o’)
2
```

- list.extend(seq) —用新列表扩展原来的列表

```
a=['hello','world']
b=['python','is','funny']
a.extend(b)
a
['hello', 'world', 'python', 'is', 'funny']
```

2.7.8 列表（续4）

➢list.index(obj) —找出列表中某个值的第一个匹配项索引位置

```
field=['hello', 'world', 'python', 'is', 'funny']
print('hello的索引位置为：',field.index('hello'))
hello的索引位置为： 0
```

➢list.insert(index,obj) —将指定对象插入到列表中的指定位置

```
num=[1,2,3]
print('插入之前的num：',num)
num.insert(2,'插入位置在2之后，3之前')
print('插入之后的num：',num)
插入之前的num： [1, 2, 3]
插入之后的num： [1, 2, '插入位置在2之后，3之前', 3]
```

2.7. 列表（续5）

➢list.pop(obj)—移除列表中的一个元素，并且返回该元素的值

```
field=['hello', 'world', 'python', 'is', 'funny']
field.pop() #不传参数，默认移除最后一个元素 'funny'
print('移除元素后的field： ',field)
移除元素后的field： ['hello', 'world', 'python', 'is']
field.pop(3) #移除编号为3的元素'is'
```

➢list.remove(obj) —移除列表中某个值的第一个匹配项

```
field=['女排','精神','中国','精神','学习','精神']
print('移除前列表field： ',field)
移除前列表field： ['女排', '精神', '中国', '精神', '学习', '精神']
field.remove('精神')
print('移除后列表field： ',field)
移除后列表field： ['女排', '中国', '精神', '学习', '精神']
```

2.7.8 列表（续6）

➢list.reverse() —反向列表中元素，该方法不需要传入参数

```
num=[1,2,3]
print('列表反转前num：',num)
列表反转前num： [1, 2, 3]
num.reverse()
print('列表反转后：',num)
列表反转后： [3, 2, 1]
```

➢list.clear() —清空列表，类似于 del field[:]

```
field=['study','python','is','happy']
field.clear()
print('field调用clear方法后的结果:',field)
```

field调用clear方法后的结果: []

2.7.8 列表（续7）

➢list.copy() —复制列表

```
field=['study','python','is','happy']
copyfield=field.copy()
print('复制操作结果:',copyfield)
```

复制操作结果: ['study', 'python', 'is', 'happy']

➢ list.sort(func) —对原列表进行排序，如果指定参数，则使用参数指定的比较方法进行排序（可选参数：key和reverse ）

```
field=['study','python','is','happy']
field.sort(key=len) #按字符串由短到长排序
field
['is', 'study', 'happy', 'python']
field.sort(key=len, reverse=True) #按字符串由长到短排序，传递两个参数
field
['python', 'study', 'happy', 'is']
```