Python 程序设计教案

<table>
<tr><td>本次授课内容</td><td colspan="2">1.1 Python 常用内置对象
1.2 2Python 运算符与表达式</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">理解变量类型的动态性
掌握运算符的用法</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">变量类型的动态性
运算符的多态性</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">粗略介绍 Python 内置对象
重点介绍变量类型的动态性
简单介绍数字、字符串、列表、元组、字典、集合的概念
重点介绍 Python 运算符的用法</td></tr>
<tr><td>实　验</td><td colspan="2">Python 运算符</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">课后习题 1-5 题，8-12 题。</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">第　一　节　课</td></tr>
</table>

课堂重点内容详解

对象类型	类型名称	示例	简要说明
数字	int float complex	1234 3.14, 1.3e5 3+4j	数字大小没有限制，内置支持复数及其运算
字符串	str	'swfu', "I'm student", '"Python "', r'abc', R'bcd'	使用单引号、双引号、三引号作为定界符，以字母 r 或 R 引导的表示原始字符串
字节串	bytes	b'hello world'	以字母 b 引导，可以使用单引号、双引号、三引号作为定界符
列表	list	[1, 2, 3] ['a', 'b', ['c', 2]]	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
字典	dict	{1:'food' ,2:'taste', 3:'import'}	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”
元组	tuple	(2, -5, 6) (3,)	不可变，所有元素放在一对圆括号中，元素之间使用逗号分隔，如果元组中只有一个元素的话，后面的逗号不能省略
集合	set frozenset	{'a', 'b', 'c'}	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;另外，set 是可变的，而 frozenset 是不可变的

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。

例如语句

```
>>> x = 3
```

创建了整型变量 x，并赋值为 3，再例如语句

```
>>> 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。

```
>>> x = 3
```

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

```
>>> 9999 ** 99                          #这里**是幂乘运算符，等价于内置函数 pow()
99014835352672348760226312475328262557055952889579105732432652912179483789405351346442217682691643393258692438667776624403200162375682140043297505120882020498009873555270384136230466997051069124380021820284037432937880069492030979195418511779843432959121215910629869993866990806757337472433120894242554489391091007320504903165678922088956073296292622630586570659359491789627675639684851490098999 9
```

```
>>> 0.3 + 0.2                           #实数相加
0.5
>>> 0.4 - 0.1                           #实数相减，结果稍微有点偏差
0.30000000000000004
>>> 0.4 - 0.1 == 0.3                    #应尽量避免直接比较两个实数是否相等
False
>>> abs(0.4-0.1 - 0.3) < 1e-6           #这里 1e-6 表示 10 的-6 次方
True
>>> x = 'Hello world.'                          #使用单引号作为定界符
```

```
>>> x = "Python is a great language."    #使用双引号作为定界符
>>> x = '''Tom said, "Let's go."'''      #不同定界符之间可以互相嵌套
>>> print(x)
Tom said, "Let's go."
>>> x = 'good ' + 'morning'              #连接字符串
>>> x
'good morning'
```

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

第 二 节 课

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，但如果操作数中有实数的话，结果为实数形式的整数
%	求余数，字符串格式化
**	幂运算
<、<=、>、>=、==、!=	（值）大小比较，集合的包含关系比较
or	逻辑或
and	逻辑与
not	逻辑非
in	成员测试
is	对象同一性测试，即测试是否为同一个对象或内存地址是否相同
|、^、&、<<、>>、~	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
@	矩阵相乘运算符

（1）+运算符

```
>>> [1, 2, 3] + [4, 5, 6]          #连接两个列表
[1, 2, 3, 4, 5, 6]
>>> (1, 2, 3) + (4,)               #连接两个元组
(1, 2, 3, 4)
>>> 'abcd' + '1234'                #连接两个字符串
'abcd1234'
>>> 'A' + 1                        #不支持字符与数字相加，抛出异常
TypeError: Can't convert 'int' object to str implicitly
>>> True + 3                       #Python 内部把 True 当作 1 处理
4
>>> False + 3                      #把 False 当作 0 处理
3
```

（2）*运算符

```
>>> True * 3
3
>>> False * 3
0
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> (1, 2, 3) * 3
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3)
>>> 'abc' * 3
'abcabcabc'
（3）运算符/和//
>>> 3 / 2                    #数学意义上的除法
1.5
>>> 15 // 4                  #如果两个操作数都是整数，结果为整数
3
>>> 15.0 // 4                #如果操作数中有实数，结果为实数形式的整数值
3.0
>>> -15//4                   #向下取整
-4
（4）%运算符
>>> 789 % 23                     #余数
7
>>> 123.45 % 3.2                 #可以对实数进行余数运算，注意精度问题
1.849999999999996
>>> '%c, %d' % (65, 65)       #把65分别格式化为字符和整数
'A, 65'
>>> '%f,%s' % (65, 65)         #把65分别格式化为实数和字符串
'65.000000,65'
（5）**运算符
>>> 3 ** 2                   #3的2次方，等价于pow(3, 2)
9
>>> pow(3, 2, 8)             #等价于(3**2) % 8
1
>>> 9 ** 0.5                 #9的0.5次方，平方根
3.0
>>> (-9) ** 0.5              #可以计算负数的平方根
(1.8369701987210297e-16+3j)
（6）Python关系运算符
>>> 1 < 3 < 5                    #等价于1 < 3 and 3 < 5
True
>>> 3 < 5 > 2
True
>>> 1 > 6 < 8
False
>>> 1 > 6 < math.sqrt(9)         #具有惰性求值或者逻辑短路的特点
False
>>> 1 < 6 < math.sqrt(9)         #还没有导入math模块，抛出异常
NameError: name 'math' is not defined
>>> import math
>>> 1 < 6 < math.sqrt(9)
False
>>> 'Hello' > 'world'              #比较字符串大小
False
>>> [1, 2, 3] < [1, 2, 4]          #比较列表大小
True
>>> 'Hello' > 3                    #字符串和数字不能比较
TypeError: unorderable types: str() > int()
```

```
>>> {1, 2, 3} < {1, 2, 3, 4}          #测试是否子集
True
>>> {1, 2, 3} == {3, 2, 1}            #测试两个集合是否相等
True
>>> {1, 2, 4} > {1, 2, 3}             #集合之间的包含测试
False
>>> {1, 2, 4} < {1, 2, 3}
False
>>> {1, 2, 4} == {1, 2, 3}
False
```

（7）成员测试运算符 in

```
>>> 3 in [1, 2, 3]                    #测试 3 是否存在于列表[1, 2, 3]中
True
>>> 5 in range(1, 10, 1)              #range()是用来生成指定范围数字的内置函数
True
>>> 'abc' in 'abcdefg'                #子字符串测试
True
>>> for i in (3, 5, 7):               #循环，成员遍历
    print(i, end='\t')
3   5   7
```

（8）集合的交集、并集、对称差集运算

```
>>> {1, 2, 3} | {3, 4, 5}             #并集，自动去除重复元素
{1, 2, 3, 4, 5}
>>> {1, 2, 3} & {3, 4, 5}             #交集
{3}
>>> {1, 2, 3} ^ {3, 4, 5}             #对称差集
{1, 2, 4, 5}
>>> {1, 2, 3} - {3, 4, 5}             #差集
{1, 2}
```

（9）逻辑运算符 and、or、not

```
>>> 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                           #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
```

<table>
<tr><td>本次授课内容</td><td colspan="2">2.3 Python 常用内置函数用法</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">掌握内置函数的用法
理解函数式编程模式</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">函数式编程模式</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">大致介绍常用的 Python 内置函数，重点介绍和演示 map、reduce、filter 这三个函数，让学生体会和理解函数式编程的模式。</td></tr>
<tr><td>实　验</td><td colspan="2">常用内置函数</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">课后习题 6、7、13</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
</table>

第　一　节　课

函数	功能简要说明
abs(x)	返回数字 x 的绝对值或复数 x 的模
all(iterable)	如果对于可迭代对象 iterable 中所有元素都等价于

课堂重点内容详解			
			True，则返回 True。对于空的可迭代对象也返回 True
		any(iterable)	只要可迭代对象 iterable 中存在等价于 True 的元素，则返回 True。对于空的可迭代对象，返回 False
		bin(x)	把整数 x 转换为二进制串表示形式
		complex(real, [imag])	返回复数
		chr(x)	返回 Unicode 编码为 x 的字符
		dir(obj)	返回指定对象或模块 obj 的成员列表，如果不带参数则返回当前作用域内所有标识符
		divmod(x, y)	返回包含整商和余数的元组((x-x%y)/y, x%y)
		enumerate(iterable[, start])	返回包含元素形式为(0, iterable[0]), (1, iterable[1]), (2, iterable[2]), ...的迭代器对象，start 表示索引的起始值
		eval(s[, globals[, locals]])	计算并返回字符串 s 中表达式的值
		filter(func, seq)	返回 filter 对象，其中包含序列 seq 中使得单参数函数 func 返回值为 True 的那些元素，如果函数 func 为 None 则返回包含 seq 中等价于 True 的元素的 filter 对象
		float(x)	把整数或字符串 x 转换为浮点数并返回
		globals()	返回包含当前作用域内全局变量及其值的字典
		help(obj)	返回对象 obj 的帮助信息
		hex(x)	把整数 x 转换为十六进制串
		id(obj)	返回对象 obj 的标识（内存地址）
		input([提示])	显示提示，接收键盘输入的内容，返回字符串
		int(x[, d])	返回实数（float）、分数（Fraction）或高精度实数（Decimal）x 的整数部分，或把 d 进制的字符串 x 转换为十进制并返回，d 默认为十进制
		isinstance(obj, class-or-type-or-tuple)	测试对象 obj 是否属于指定类型（如果有多个类型的话需要放到元组中）的实例
		len(obj)	返回对象 obj 包含的元素个数，适用于列表、元组、集合、字典、字符串以及 range 对象，不适用于具有惰性求值特点的生成器对象和 map、zip 等迭代对象
		list([x])、set([x])、tuple([x])、dict([x])	把对象 x 转换为列表、集合、元组或字典并返回，或生成空列表、空集合、空元组、空字典
		locals()	返回包含当前作用域内局部变量及其值的字典
		map(func, *iterables)	返回包含若干函数值的 map 对象，函数 func 的参数分别来自于 iterables 指定的一个或多个迭代对象，
		max(...)、min(...)	返回多个值中或者包含有限个元素的可迭代对象中所有元素的最大值、最小值，要求所有元素之间可比较大小，允许指定排序规则，参数为可迭代对象时还允许指定对象为空时返回的默认值
		next(iterator[, default])	返回可迭代对象 x 中的下一个元素，允许指定迭代结束之后继续迭代时返回的默认值
		oct(x)	把整数 x 转换为八进制串
		open(name[, mode])	以指定模式 mode 打开文件 name 并返回文件对象
		ord(x)	返回 1 个字符 x 的 Unicode 编码
		print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)	基本输出函数
		range([start,] end [, step])	返回 range 对象，其中包含左闭右开区间[start,end)内

	以 step 为步长的整数
reduce(func, sequence[, initial])	将双参数的函数 func 以迭代的方式从左到右依次应用至序列 seq 中每个元素，并把中间计算结果作为下一次计算的操作数之一，最终返回单个值作为结果。在 Python 3.x 中 reduce() 不是内置函数，需要从 functools 中导入再使用
reversed(seq)	返回 seq（可以是列表、元组、字符串、range 以及其他可迭代对象）中所有元素逆序后的迭代器对象，但不适用于具有惰性求值特点的生成器对象和 map、zip 等可迭代对象
round(x [, 小数位数])	对 x 进行四舍五入，若不指定小数位数，则返回整数
sorted(iterable, key=None, reverse=False)	返回排序后的列表，其中 iterable 表示要排序的序列或迭代对象，key 用来指定排序规则或依据，reverse 用来指定升序或降序
str(obj)	把对象 obj 直接转换为字符串
sum(x, start=0)	返回序列 x 中所有元素之和，要求序列 x 中所有元素支持加法运算
type(obj)	返回对象 obj 的类型
zip(seq1 [, seq2 [...]])	返回 zip 对象，其中元素为(seq1[i], seq2[i], ...)形式的元组，最终结果中包含的元素个数取决于所有参数序列或可迭代对象中最短的那个

第 二 节 课

```
>>> bin(555)                          #把数字转换为二进制串
'0b1000101011'
>>> oct(555)                          #转换为八进制串
'0o1053'
>>> hex(555)                          #转换为十六进制串
'0x22b'
>>> float(3)                          #把整数转换为实数
3.0
>>> float('3.5')                      #把数字字符串转换为实数
3.5
>>> float('inf')                      #无穷大，其中 inf 不区分大小写
inf
>>> complex(3)                        #指定实部
(3+0j)
>>> complex(3, 5)                     #指定实部和虚部
(3+5j)
>>> complex('inf')                    #无穷大
(inf+0j)
>>> ord('a')                        #查看指定字符的 Unicode 编码
97
>>> chr(65)                         #返回数字 65 对应的字符
'A'
>>> chr(ord('A')+1)                 #Python 不允许字符串和数字之间的加法操作
'B'
>>> chr(ord('国')+1)                #支持中文
'图'
>>> str(1234)                       #直接变成字符串
'1234'
>>> str([1,2,3])
'[1, 2, 3]'
>>> str((1,2,3))
'(1, 2, 3)'
>>> str({1,2,3})
'{1, 2, 3}'
>>> list(range(5))                    #把 range 对象转换为列表
[0, 1, 2, 3, 4]
>>> tuple(_)                           #一个下划线表示上一次正确的输出结果
(0, 1, 2, 3, 4)
>>> dict(zip('1234', 'abcde'))  #创建字典
{'4': 'd', '2': 'b', '3': 'c', '1': 'a'}
>>> set('1112234')                    #创建可变集合，自动去除重复
{'4', '2', '3', '1'}
>>> eval('3+5')
8
>>> eval(b'3+5')                    #引号前面加字母 b 表示字节串
8
>>> eval('9')                       #把数字字符串转换为数字
9
>>> eval('09')                      #抛出异常，不允许以 0 开头的数字
SyntaxError: invalid token
>>> int('09')                        #这样转换是可以的
9
>>> max(['2', '111'])                  #不指定排序规则
```

```
'2'
>>> max(['2', '111'], key=len)      #返回最长的字符串
'111'
>>> from random import randint
>>> lst = [[randint(1, 50) for i in range(5)] for j in range(30)]
                              #列表推导式，生成包含 30 个子列表的列表
                              #每个子列表中包含 5 个介于[1,50]区间的整数
>>> max(lst, key=sum)              #返回元素之和最大的子列表
>>> max(lst, key=lambda x: x[1]) #所有子列表中第 2 个元素最大的子列表
>>> max(lst, key=lambda x: (x[1], x[3]))
>>> x = input('Please input: ') #input()函数参数表示提示信息
Please input: 345
>>> x
'345'
>>> type(x)                      #把用户的输入作为字符串对待
<class 'str'>
>>> print(1, 3, 5, 7, sep='\t') #修改默认分隔符
1   3   5   7
>>> for i in range(10):          #修改 end 参数，每个输出之后
换行
    print(i, end=' ')

0 1 2 3 4 5 6 7 8 9
>>> x = list(range(11))
>>> import random
>>> random.shuffle(x)                  #shuffle()用来随机打乱顺序
>>> x
[2, 4, 0, 6, 10, 7, 8, 3, 9, 1, 5]
>>> sorted(x, key=lambda item:len(str(item)), reverse=True)
                                      #以指定规则降序排序
[10, 2, 4, 0, 6, 7, 8, 3, 9, 1, 5]
>>> list(enumerate('abcd'))                    #枚举字符串中的元素
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
>>> list(enumerate(['Python', 'Greate']))    #枚举列表中的元素
[(0, 'Python'), (1, 'Greate')]
>>> for index, value in enumerate(range(10, 15)):
    print((index, value), end=' ')

(0, 10) (1, 11) (2, 12) (3, 13) (4, 14)
>>> list(map(str, range(5)))      #把列表中的元素转换为字符串
['0', '1', '2', '3', '4']
>>> def add5(v):                   #单参数函数
    return v+5

>>> list(map(add5, range(10)))    #把单参数函数映射到一个序列的所有元素
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

>>> def add(x, y):                 #可以接收 2 个参数的函数
    return x+y
>>> list(map(add, range(5), range(5,10)))
                                  #把双参数函数映射到两个序列上
[5, 7, 9, 11, 13]
>>> from functools import reduce
```

```
>>> reduce(lambda x, y: x+y, range(1, 10)) #lambda 表达式相当于函数
45
>>> seq = ['foo', 'x41', '?!', '***']
>>> def func(x):
    return x.isalnum()                      #isalnum()是字符串的方法
                                            #用于测试 x 是否为字母或数字

>>> filter(func, seq)                       #返回 filter 对象
<filter object at 0x000000000305D898>
>>> list(filter(func, seq))                 #把 filter 对象转换为列表
['foo', 'x41']
>>> seq                                     #不对原列表做任何修改
['foo', 'x41', '?!', '***']
>>> range(5)                    #start 默认为 0，step 默认为 1
range(0, 5)
>>> list(_)
[0, 1, 2, 3, 4]
>>> list(range(1, 10, 2))      #指定起始值和步长
[1, 3, 5, 7, 9]
>>> list(range(9, 0, -2))      #步长为负数时，start 应比 end 大
[9, 7, 5, 3, 1]
>>> for i in range(4):         #循环 4 次
    print(3, end=' ')
3 3 3 3
>>> list(zip('abcd', [1, 2, 3]))          #压缩字符串和列表
[('a', 1), ('b', 2), ('c', 3)]
```