



School of Economics and Management, Beihang University

现代程序设计技术

赵吉昌

- Python基础
 - 关键字
 - 代码格式
 - 标识符与变量
 - 数据类型

- 关键词

- `import keyword`
- `print(keyword.kwlist)`
- `with open("test.txt", "w") as f:`
 - **上下文管理器**，常在资源管理中用到，能够处理异常
- `None`和任何其他数据类型比较永远返回`False`
- `nonlocal`在函数或其他作用域中使用外层（非全局）变量
- `yield`在生成器和协程部分会详细讨论

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

- 缩进
 - 缩进空格数可变，但是同一个代码块的语句必须包含相同的缩进空格数
 - 建议使用4个空格
- 多语句
 - 可以用；在同一行显示多条语句
 - 语句很长时可使用\来实现多行语句
 - 在 [], {}, 或 () 中的多行语句不需要使用\
- 注释
 - 单行注释用#
 - 多行注释用'''或"""

- 标识符

- 标识符由字母、数字、下划线(_)组成
- 所有标识符可以包括英文、数字以及下划线，但不能以数字开头

- 非ASCII码亦可

类：骆驼命名

属性和方法：Unix式命名

- 标识符区分大小写
- 以下划线开头的标识符有特殊意义

- 以单下划线开头代表不能直接访问的类属性
 - 以双下划线开头代表类的私有成员
 - 以双下划线开头和结尾代表 Python 里特殊方法专用的标识（比较关键）
 - 在类编程部分会详细介绍和使用

- 变量

- 变量不需要声明

- 每个变量在使用前都必须赋值

- 赋值以后该变量才会被创建

- 变量没有类型

- Python是动态类型语言

- `a, b, c = 1, 2, "string"`

- del 语句删除对象引用

- `del var_a, var_b`

- 删除后不能再引用，除非再次赋值

- 变量
 - 获取变量所指对象的内存地址
 - `id(var)`
 - `print(hex(id(var)))`
 - `a=2`
 - `del a`
 - `print(id(2))`
 - `c=2`
 - `print(id(c))`

- 标准数据类型
 - 变量所指的内存中对象的类型
 - Number(int, bool, float, complex) : 数字
 - String(str) : 字符串
 - List(list) : 列表
 - Tuple(tuple) : 元组
 - Set(set) : 集合
 - Dictionary(dict) : 字典

- 类型的划分

- 不可变数据 : Number、String、Tuple
- 可变数据 : List、Dictionary、Set

- 类型的查询

- `type()` 函数可以用来查询变量所指的对象类型
- `type()` 不会认为子类是一种父类类型

- 类型的判断

- `isinstance(a, int)` 可以用来判断是否是某类型
- `isinstance()` 会认为子类是一种父类类型

- 数字(Number)类型
 - 整数、布尔型、浮点数和复数。
 - `int` (整数), 长整型
 - `bool` (布尔), 如 `True`和`False`.
 - `type(True)`和`isinstance(True, int)`的返回结果?
 - `float` (浮点数), 如3.14
 - `complex` (复数), 如`1+3j`
 - 数据类型不允许改变,这就意味着如果改变数字数据类型的值, 将重新分配内存空间
 - `a=20`
 - `b=20`
 - `id(a)==id(b)` 是否成立? 如果让`b=30`, 其地址会变化吗?
 - 可变类型`list`呢?

- 内存分配

- 为提高内存利用效率,对于简单对象如int对象或字符串对象等,会采取重用对象内存的办法

- 解释器依赖,不同的解释器可能有不同的实现

- x=2

- y=2

- print(id(x))

- print(id(y))

- x=3

- print(id(3))

- print(id(x))

- 数字类型运算

- 常规运算比较简单

- 需要注意的一些地方

- True 和 False 关键字的值是 1 和 0，它们可以和数字相加

- if -1 if 0 if 2 如何判断?

- /：除法，得到浮点数

- //：除法，得到整数 (不一定，分子分母为浮点时得到浮点)

- $2 // 3 = 0.0$ 或 $4.1 // 3 = 1.0$

- **：幂

- complex：a + bj,或者complex(a,b)表示，复数的实部a和虚部b都是浮点型

- 比较操作可以传递 (a<b==c)

- a,b=2,3

- print(a<b==3)

- 字符串(String)
 - 单引号和双引号使用完全相同
 - 使用三引号(''或''')可以指定一个多行字符串。
 - 转义符 '\'
 - 反斜杠可以用来转义，使用r可以让反斜杠不发生转义
 - `print(r'\n')`
 - 按字面意义级联字符串，如"this " "is " "string"会被自动转换为this is string
 - 字符串可以用 + 运算符连接在一起，用 * 运算符重复。

- 字符串(String)

- 字符串有两种索引方式，从左往右以 0 开始，从右往左以 -1 开始。

- 字符串不能改变

- `s = 'abc'`

- `s[0] = 'a'` # 可否修改？

- `s += 'b'`

- 这时还是不是同样的内存地址？

- `s2 = 'abc'` # s2的地址为何？

- 没有单独的字符类型，一个字符就是长度为 1 的字符串

- 字符串(String)
 - 字符串常用的函数
 - `find()`
 - `strip()`
 - `split()`
 - `zfill(width)`
 - `a = ''`
 - `b = a.zfill(10)`
 - `id(a) == id(b) ??`

- 字符串模板

- 格式化输出（后续内容中还会提及）

- `values = {'price': '5.99', 'dis': '50'}`
 - `t1 = string.Template("""PRICE:
$price$$\nDISCOUNT: $dis%""")`
 - `print(t1.substitute(values))`

- 字符串比较

- `import difflib`
 - `d = difflib.Differ()`
 - `diff = d.compare(s1, s2)`

- 字符串对齐

- `import textwrap`
- `textwrap.dedent(sample_text)`
- `textwrap.indent(wrap, '>')`
- `textwrap.fill(wrap, width=50)`
- `textwrap.fill(wrap, initial_indent=' '*4, subsequent_indent=' '*4, width=50)`
- `textwrap.shorten(sample_text, 30, placeholder='...')`

- 列表(List)

- 使用最频繁的数据类型

- `list1=['a','b','c']`
 - `list2=['d','e','f']`
 - `print(list1*2)`
 - `print(list1+list2)`

- 有步长的元素截取

- `list1[1:4:2]`
 - `nl=[0,1,2,3,4,5,6,7,8,9,10]`
 - `print(nl[0::2])` 将输出？
 - `print(nl[::2])` 将输出？
 - `print(nl[1::2])` 将输出？

- 列表(List)

- 逆序

- `l1=['A','B','C','D','E','F','G']`
 - `print(l1[-1::-1])`
 - `print(l1.reverse())`

- 元素的删除

- `del l1[0]`
 - `l1.remove('A')`
 - `a.clear()` 相当于 `del a[:]`
 - `del a`

- 元素的增加

- `a.append(x)` 相当于 `a[len(a):]=[x]`
 - `a.extend(1)` 相当于 `a[len(a):]=1`

- 列表(List)

- 排序

- `a.sort()`

- 复制

- `a.copy()`

- **slice copy** , 在可能对原列表有修改的操作中较常见

- `b=a[:]` ? 深浅拷贝在后面会继续讨论

- 推导式

- `a=[x**2 for x in range(6)]`
 - `a=list(map(lambda x:x**2,range(6)))`
 - `pis=[str(round(math.pi, i)) for i in range(1,6)]`
 - `a=[(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]` (**跟zip的区别**)

- 内存分配

- `L = [1, 2, 3]`

- `L2 = [1, 2, 3]`

- `id(L) == id(L2) ??`

- `L1 = L[:]`

- `id(L1) == id(L) ??`

- `L.append(4)`

- `id(L) == ?`

- 列表(List)

- 一些复杂但可能有用的操作

- `l1=[[1,3,4],[5,6,7,8],[9,10,11]]`
 - `l1_f=[e for ele in l1 for e in ele]` #flatten list

- 矩阵交换行列

- `matrix=[[1,2,3],[4,5,6],[7,8,9]]`
 - `matrix2=[[row[i] for row in matrix] for i in range(3)]`

- 按列遍历矩阵

- 比较大小

- 序列对象可以与其它序列对象比较
 - 注意运算规则，不建议直接使用

• 列表排序

- `d1=[random.randint(0,1000) for i in range(100)]`
- `d1.sort()`
- `d1s=sorted(d1, reverse=True)`
- `d1s=[]`
- `import heapq`
- `for n in d1:`
 - `heapq.heappush(d1s,n)`
- `heapq.heapify(d1)`

• 高效合并多个有序列表

- `d1=[random.randint(0,1000) for i in range(100)]`
- `d2=[random.randint(0,1000) for i in range(100)]`
- `d1.sort()`
- `d2.sort()`
- `d=list(heapq.merge(d1,d2))`

- 列表排序

- 如何在插入新元素时自动更新顺序

- `import bisect`

- `nums = [random.random() for i in range(10)]`

- `l=[]`

- `for n in nums:`

- `position = bisect.bisect(l, n)`

- `print(position)`

- `bisect.insort(l,n)`

- `#bisect.insort_left(l,n)`

- `print(l)`

- 元组 (tuple)

- 与列表类似，不同之处在于元素不可修改

- 可以把字符串看作一种特殊的元组

- 虽然tuple的元素不可改变，但它可以包含可变的对象

- `tup1 = ([1, 2, 3], 0)`

- `tup1[0] = [1, 2, 3, 4] ??`

- `tup1[0].append(4) ??` (联想到深浅拷贝的区别)

- 构造包含 0 个或 1 个元素的元组

- `tup1 = ()` # 空元组

- `tup2 = (20,)` # 一个元素，需要在元素后添加逗号，否则含义不明确

- 元组中的元素值不允许删除，但`del`能删除整个元组

- 命名元组

- namedtuple

- 元组中的元素可通过名称进行访问

- Student =

- `collections.namedtuple('Student', 'name credit hometown')`

- `s1=Student(name='zjc', credit='65', hometown='gs')` # 元素列表要完整

- `print(s1)`

- `print(s1.credit)`

- `s1.hometown='bj' #????`

- 集合(Set)

- 一般用于进行成员关系测试和删除重复元素
- 可以使用大括号 { } 或者 set() 函数创建集合
 - `s1={e1,e2,e3}`
 - `s2=set(value)`
- 创建空集合必须用 set() 而非 { }
 - { } 用来创建一个空字典
- 集合的运算
 - - | &
 - update() 可实现批量添加元素
- 集合的实现
 - 散列实现, 不能包含可变类型
 - 包含list的元组可否加入set? ?

- 字典(Dictionary)

- 一种映射类型，其元素是键值对

- 无序的键(key): 值(value) 集合

- 键(key)必须使用不可变类型

- 同一个字典中键(key)必须是唯一的

- 函数 `dict()` 可以直接从键值对序列中构建字典

- 构造函数

- `dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])`

- `dict(sape=4139, guido=4127, jack=4098)`

- `dict(list(enumerate(['one', 'two', 'three'], start=1))))`

- 散列实现

- ChainMap

- 组合多个字典使其在逻辑上表现为一个整体
- 创建了一个单一的可更新视图
- 键查找时返回第一次出现的目标键
- 注意对其修改会影响原始的字典
- 常用于上下文参数或配置的管理
- `d1={'path': '/c/d/e', 'cmd': 'clear', 'pwd': '$'}`
- `d2={'root': '/', 'cmd': 'cls', 'pwd': '#', 'prompt': 'True'}`
- `d=collections.ChainMap(d1,d2)`
- `print(d['path'])`
- `print(d['cmd'])`
- `d['cmd']='update'`
- `print(d1['cmd'])`
- `print(d2['cmd'])`

- Counter

- 常用于计数（如文本处理中）
- 可自动进行计数的更新
- `freq=collections.Counter(['a', 'b', 'c', 'a', 'a', 'b'])`
- `freq.update('abca')`
- `for l in 'abc':`
 - `print("{}:{}".format(l, freq[l]))`

- defaultdict

- 可以设定键的默认值

- 避免无键时出现错误

- `def default_value():` #该可调用对象用来设定默认值

- `return -1`

- `dc =`

- `collections.defaultdict(default_value,`
`zjc='38')` #default_value可以是None

- `print(dc['zjc'])`

- `print(dc['lsy'])`

- `dc['lsy']+=1`

- `print(dc['lsy'])`

- 字典(Dictionary)

- 如何有序地输出？

- `for key in sorted(dic.keys()):`

- `print(key, dic[key])`

- 如何构建有序字典

- 即对字典排序

- `import collections`

- `dic = collections.OrderedDict()`

- `dic=collections.OrderedDict(sorted(un
sorted_d.items(), key=lambda dc:dc[1],
reverse = True))`

- 判断相等

- is 运算符

- `a is b` 相当于 `id(a) == id(b)`

- is 用于判断两个变量引用对象是否为同一个

- == 用于判断引用变量的值是否相等

• 强制类型转换

- `int(x [,base])` 将字符串`x`转换为一个整数
- `float(x)` 将`x`转换到一个浮点数
- `complex(real [,imag])` 创建一个复数
- `str(x)` 将对象 `x` 转换为字符串
- `repr(x)` 返回对象 `x` 的字符串表达
- `eval(str)` 用来计算在字符串中的有效Python表达式,并返回一个对象
 - `eval('print(2+3)')`
 - `eval('exit()')`
- `tuple(s)` 将序列 `s` 转换为一个元组
- `list(s)` 将序列 `s` 转换为一个列表

- 强制类型转换

- `set(s)` 转换为可变集合
- `dict(d)` 创建一个字典, `d` 必须是一个 `(key, value)` 元组序列。
- `frozenset(s)` 转换为不可变集合
- `chr(x)` 将一个整数转换为一个字符
- `ord(x)` 将一个字符转换为它的整数值
- `hex(x)` 将一个整数转换为一个十六进制字符串

- 随机函数

- `choice(seq)` 从序列中随机挑选一个元素
- `randrange([start,] stop [,step])`
从指定范围内，按指定基数递增的集合中获取一个随机数，基数默认值为 1
- `random()` 随机生成下一个实数，在 $[0, 1)$ 范围内
- `seed()` 改变随机数生成器的种子 `seed`
 - 或固定
- `shuffle(list)` 将序列的所有元素随机排序
- `uniform(x, y)` 随机生成下一个实数，在 $[x, y]$ 范围内

- 随机函数

- `randint(a, b)` : 随机生成 $[a, b]$ 之间的随机整数
- `choices(seq, k)` : 随机抽取长度为 k 的子序列, k 可以大于序列长度
- `sample(seq, k)` : 随机抽取长度为 k 的不包含重复元素的子序列, k 不能大于序列长度

- 正式表达

- `import re`

- <https://learnku.com/docs/pymotw/re-regular-expression/3362>

本周作业



- 利用python数据结构 (`list`, `dict`, `set`等) 完成简单的文本分析任务
 - B站弹幕数据