法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

第三周 python使用基础

1) 基本概念
2) 运算符与表达式
3) 逻辑控制结构
4) 函数
5) 面向对象编程
6) 异常处理
7) 模块与包
8) 输入、输出、文件和目录操作

1）基本概念

1. 常量
2. 数
3. 字符串
4. 变量
5. 数据类型
6. 逻辑行与物理行
7. 缩进

1.1)常量

- 常量无名
- 不需要修饰

1.2)数

- 整数
- 长整数
- 浮点数
- 复数

互动作业：复数的使用和表示法

1.3)字符串

- 单引号　‘abc’
- 双引号 “abc”
- 三引号 ‘’’abc’’’ “””abc”””
- 转义符　\
- 原生字符串 r”hello world!\n”
- **常用的索引相关操作**
- **切割操作**
- **邪恶的eval()**

互动作业：讨论eval的各种用法

1.4)变量

- 首字符为字母（大小写均可）或为下划线（_）
- 其他部分字符为字母（大小写均可）、数字、下划线
- 区分大小写，myname与myName是两个不同的变量

1.5)数据类型

- 数值
- 字符串
- 线性容器
  - 字符串也是一种线性容器
  - List
  - tuple
- Hash容器
  - Dict
  - set
- None
- 逻辑类型（True, False）

互动作业：什么是hash

1.6)逻辑行与物理行

- 物理行是您在编辑器中所看见的。逻辑行是Python能够识别的。一般，Python认为一个物理行对应一个逻辑行。
- 如果您要在一个物理行使用多个逻辑行，那么就要用分号（;）分割开，分号表示一个逻辑行或语句的结束。我们建议不要这样写，可读性差。

例子：i = 5;print(i)

相当于

```
i = 5
print(i)
```

- 如果您要在多个物理行中写一个逻辑行，那么你用反斜杠（\）来连接。

例子：
```
s = "This is a\
      string"
```

相当于

```
s = "This is a string"
```

互动作业：什么叫逻辑概念和物理概念

1.7)缩进

- 缩进在Python中试重要的。行首的空白是重要的，例子：

```
print("Hello")
  print("world")
```

- 第二个print语句会报错，因为它前面有一个错误的空白。
- 同意层次的语句必须有相同的缩进。
- 不要混合使用空白和制表符，这样在不同平台中将无法工作。
- 强烈建议在缩进时，使用单个制表符或两个或四个空格
- 建议一如既往地使用四个空格。

2）运算符与表达式

1. 运算符
2. 运算符的优先级
3. 表达式
4. repr()

互动作业：repr的各种使用

2.1)运算符

运算符	名称	说明	举例
&	按位与	数的按位与	5 & 3得到1 。
|	按位或	数的按位或	5 | 3得到7
^	按位异或	数的按位异或	5 ^ 3得到6 。
~	按位翻转	x的按位翻转是-(x+1)	~5得到-6 。
<	小于	返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。	5 < 3返回0（即False）而3 < 5返回1（接True）。比较可以被任意连接：3 < 5 < 7返回True。
>	大于	返回x是否大于y	5 > 3返回True。如果两个操作数都是数字，它们首先被转换为一个共同的类型。否则，它总是返回False。
<=	小于等于	返回x是否小于等于y	x = 3; y = 6; x <= y返回True。
>=	大于等于	返回x是否大于等于y	x = 4; y = 3; x >= y返回True。
==	等于	比较对象是否相等	x = 2; y = 2; x == y返回True。x = "str"; y = "stR"; x == y返回False。x = "str"; y = "str"; x == y返回True。

2012.12.1

2.1)运算符

运算符	名称	说明	举例
!=	不等于	比较两个对象是否不相等	x = 2; y = 3; x != y返回True。
not	布尔”非“	如果x为True，返回False。如果x是False，它返回True。	x = True; not x返回False。
and	布尔”与“	如果x为False，x and y返回False，否则它返回y的计算值。	x = False; y = True; x and y，由于x是False，返回False。在这里Python不会计算y，因为它知道这个表达式的值肯定是False（因为x是False）。这个现象称为短路计算。
or	布尔”或“	如果x是True，x or y返回True，否则它返回y的计算值。	x = True; y = False; x or y返回True。短路计算在这里也适用。

2.2)运算符优先级

运算符	描述
lambda	Lambda表达式
or	布尔”或“
and	布尔”与“
not x	布尔”非“
in, not in	成员测试
is, is not	同一性测试
<, <=, >, >=, !=, ==	比较
|	按位或
^	按位异或
&	按位与
<<, >>	移位
+, -	加法与减法

*, /, %	乘法、除法与取余
+x, -x	正负号
~x	按位翻转
**	指数
x.attribute	属性参考
x[index]	下标
x[index:index]	寻址段
f(arguments...)	函数调用
(expression,...)	绑定或元组显示
[expression,...]	列表显示
{key:datum,...}	字典显示
'expression,...'	字符串转换

2.3)表达式

- （常量，变量，函数，对象） + 运算符 + 优先级
- 字符串表达式
- 数值表达式
- 逻辑表达式
- 函数式表达式
- eval()和repr()

互动作业：表达式在计算机中如何展开的

3）逻辑控制结构

- if
- while
- for
- break
- continue

互动作业：为什么python循环控制结构会有else

4）函数

1. 简单函数
2. 带形参函数
3. 变量作用域
4. 默认参数值
5. 关键参数
6. 文档字符串
7. **Lambda**
8. 闭包

互动作业：闭包的使用

5）面向对象编程

1. 面向对象编程
2. 类的定义
3. 类的实例化
4. 类的封装
5. 专用类方法
6. 类属性介绍
7. 私有函数

5.1)面向对象编程

- 封装（Encapsulation）
- 继承（Inheritance ）
- 多态（polymorphism）

5.2)类的定义

- Class a:

```
def __init__(self):
    self.m=1
```

- Class b(a):

```
def __inti__(self):
    self.n=1
```

5.3)类的实例化

- c=a()
- c.__class__
- c.__doc__
- type(c)
- str(c)

5.4)类的封装

```
class UserDict:                                          ❶
    def __init__(self, dict=None):                       ❷
        self.data = {}                                   ❸
        if dict is not None: self.update(dict)           ❹ ❺

    def clear(self): self.data.clear()                   ❶
    def copy(self):                                      ❷
        if self.__class__ is UserDict:                   ❸
            return UserDict(self.data)
        import copy                                      ❹
        return copy.copy(self)
    def keys(self): return self.data.keys()              ❺
    def items(self): return self.data.items()
    def values(self): return self.data.values()
```

5.5)类专有方法

```
def __getitem__(self, key): return self.data[key]

>>> f = fileinfo.FileInfo("/music/_singles/kairo.mp3")
>>> f
{'name':'/music/_singles/kairo.mp3'}
>>> f.__getitem__("name")  ❶
'/music/_singles/kairo.mp3'
>>> f["name"]              ❷
'/music/_singles/kairo.mp3'
```

```
def __repr__(self): return repr(self.data)      ❶
def __cmp__(self, dict):                        ❷
    if isinstance(dict, UserDict):
        return cmp(self.data, dict.data)
    else:
        return cmp(self.data, dict)
def __len__(self): return len(self.data)        ❸
def __delitem__(self, key): del self.data[key]  ❹
```

5.6)类属性介绍

```
class MP3FileInfo(FileInfo):
    "store ID3v1.0 MP3 tags"
    tagDataMap = {"title"   : (  3,  33, stripnulls),
                  "artist"  : ( 33,  63, stripnulls),
                  "album"   : ( 63,  93, stripnulls),
                  "year"    : ( 93,  97, stripnulls),
                  "comment" : ( 97, 126, stripnulls),
                  "genre"   : (127, 128, ord)}

>>> import fileinfo
```

```
>>> fileinfo.MP3FileInfo                    ❶
<class fileinfo.MP3FileInfo at 01257FDC>
>>> fileinfo.MP3FileInfo.tagDataMap         ❷
{'title': (3, 33, <function stripnulls at 0260C8D4>),
'genre': (127, 128, <built-in function ord>),
'artist': (33, 63, <function stripnulls at 0260C8D4>),
'year': (93, 97, <function stripnulls at 0260C8D4>),
'comment': (97, 126, <function stripnulls at 0260C8D4>),
'album': (63, 93, <function stripnulls at 0260C8D4>)}
>>> m = fileinfo.MP3FileInfo()              ❸
>>> m.tagDataMap
{'title': (3, 33, <function stripnulls at 0260C8D4>),
'genre': (127, 128, <built-in function ord>),
'artist': (33, 63, <function stripnulls at 0260C8D4>),
'year': (93, 97, <function stripnulls at 0260C8D4>),
'comment': (97, 126, <function stripnulls at 0260C8D4>),
'album': (63, 93, <function stripnulls at 0260C8D4>)}
```

5.7)私有函数

- 与大多数的语言不同，一个 Python 函数，方法，或属性是私有还是公有，完全取决于它的名字。
- 如果一个 Python 函数，类方法，或属性的名字以两个下划线开始（但不是结束），它是私有的；其它所有的都是公有的。

互动作业：为什么python私有函数会这样表示，及作用？

6）异常处理

```
>>> fsock = open("/notthere", "r")                    ❶
Traceback (innermost last):
  File "<interactive input>", line 1, in ?
IOError: [Errno 2] No such file or directory: '/notthere'
>>> try:
...     fsock = open("/notthere")                     ❷
... except IOError:                                   ❸
...     print "The file does not exist, exiting gracefully"
... print "This line will always print"              ❹
The file does not exist, exiting gracefully
This line will always print
```

互动作业：什么叫防御式编程

7）模块与包

- __init__.py
- 文件、模块与包
- 编码问题
- #!
- 导入
- import
- from import

互动作业：__init__.py的妙用

8）文件和目录操作

- open
- write
- read
- readlines
- Seek
- Os.listdir
- Os.walk

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn