nsd1906_py02_day04

re模块

```
>>> import re
# 在food的开头匹配f..，匹配到返回匹配对象，否则返回None
>>> re.match('f..', 'food')
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>> print(re.match('f..', 'seafood'))
None
# 在字符串中匹配f..
>>> re.search('f..', 'seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> m = re.search('f..', 'seafood')
# 匹配到之后，用group方法返回匹配结果
>>> m.group()
'foo'

>>> re.findall('f..', 'seafood is food')
['foo', 'foo']
# finditer返回的是由匹配对象构成的生成器
>>> list(re.finditer('f..', 'seafood is food'))
[<_sre.SRE_Match object; span=(3, 6), match='foo'>, <_sre.SRE_Match object; span=(11, 14),
match='foo'>]
>>> for m in re.finditer('f..', 'seafood is food'):
...   m.group()
...
'foo'
'foo'

# 以.和-作为分隔符切割字符串
>>> re.split('-|\.', 'hello-world.tar.gz')
['hello', 'world', 'tar', 'gz']
# 把X替换成tedu
>>> re.sub('X', 'tedu', 'Hello X. Hi X')
'Hello tedu. Hi tedu'

# 为了提升效率，建议将正则表达式的模式先编译
>>> patt = re.compile('f..')
>>> m = patt.search('seafood')
>>> m.group()
'foo'
>>> patt.findall('seafood is food')
['foo', 'foo']
```

字典排序

- 字典本身没有顺序，不能排序

- 需要将其转换成其他序列类型

```
>>> result = {'172.40.58.150': 10, '172.40.58.124': 6, '172.40.58.101': 10, '127.0.0.1':
121, '192.168.4.254': 103, '192.168.2.254': 110, '201.1.1.254': 173, '201.1.2.254': 119,
'172.40.0.54': 391, '172.40.50.116': 244}
>>> alist = list(result.items())
>>> alist
[('172.40.58.150', 10), ('172.40.58.124', 6), ('172.40.58.101', 10), ('127.0.0.1', 121),
('192.168.4.254', 103), ('192.168.2.254', 110), ('201.1.1.254', 173), ('201.1.2.254', 119),
('172.40.0.54', 391), ('172.40.50.116', 244)]
```

复杂列表排序：

- 列表的sort方法，支持一个名为key的参数
- key应该是一个函数
- 该函数处理列表的每一项，处理结果作为排序依据

```
>>> def get_second(seq):
...   return seq[-1]
...
>>> alist.sort(key=get_second)
>>> alist
[('172.40.58.124', 6), ('172.40.58.150', 10), ('172.40.58.101', 10), ('192.168.4.254',
103), ('192.168.2.254', 110), ('201.1.2.254', 119), ('127.0.0.1', 121), ('201.1.1.254',
173), ('172.40.50.116', 244), ('172.40.0.54', 391)]
# 上面写法，可以用以下一行解决，并实现降序排列
>>> alist.sort(key=lambda seq: seq[-1], reverse=True)
>>> alist
[('172.40.0.54', 391), ('172.40.50.116', 244), ('201.1.1.254', 173), ('127.0.0.1', 121),
('201.1.2.254', 119), ('192.168.2.254', 110), ('192.168.4.254', 103), ('172.40.58.150',
10), ('172.40.58.101', 10), ('172.40.58.124', 6)]
```

数据库

安装py软件包

```
# 1. 在线直接安装
(nsd1906) [root@room8pc16 day04]# pip install pymysql
# 直接安装，将访问python官方站点，速度慢，可以使用国内镜像站点，方法如下：
(nsd1906) [root@room8pc16 day04]# mkdir ~/.pip/
(nsd1906) [root@room8pc16 day04]# vim ~/.pip/pip.conf
[global]
index-url = http://pypi.douban.com/simple/
[install]
trusted-host=pypi.douban.com


# 2. 本地安装
# ls /linux-soft/05/zzg_pypkgs.tar.gz
```

```
# 将该文件解压：
[root@room8pc16 zzg_pypkgs]# ls
ansible-cmdb_pkgs  matplotlib_pkgs  requests_pkgs
ansible_pkg        paramiko_pkgs    sqlalchemy_pkgs
dj_pkgs            pymysql_pkgs     wordcloud_pkgs
jenkins            python3_pkg
(nsd1906) [root@room8pc16 day04]# pip install pymysql_pkgs/*
```

pymysql模块

- 操作mysql数据库
- 主要是通过python执行sql语句

创建小型数据库

为一个小公司创建数据库，用于记录员工情况以及发工资的信息

字段：员工ID、姓名、性别、部门、职位、出生日期、联系方式、工资日、基本工资、奖金、扣税、实发工资

数据库范式

- 第一范式、第二范式、第三范式、巴斯-科德范式、第四范式、第五范式六种。
- 所谓第一范式（1NF）是指在关系模型中，对域添加的一个规范要求，所有的域都应该是原子性的。如联系方式，可以分为家庭住址、电话号码、email等
- 第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或记录必须可以被唯一地区分。简单来说，就是需要有一个主键
- 第三范式（3NF）是第二范式（2NF）的一个子集，即满足第三范式（3NF）必须满足第二范式（2NF）。3NF要求：非主属性不能依赖于其他非主属性。如实发工资，它是通过基本工资和奖金算出来的。

根据数据库范式，需要把字段拆分到不同的表中：

员工表：员工ID、姓名、生日、email、部门ID

部门表：部门ID、部门名

工资表：id、工资日、员工ID、基本工资、奖金

```
[root@room8pc16 zzg_pypkgs]# mysql -uroot -ptedu.cn
MariaDB [(none)]> CREATE DATABASE nsd1906 DEFAULT CHARSET utf8;
```

SQLALCHEMY

- 不限数据库
- 不用编写SQL语句
- 采用ORM（对象关系映射）
    - Object：对象
    - Relationship：关系
    - Mapper：映射
    - 数据库的表与class映射

- 表中的字段与类变量映射
- 数据库的数据类型与sqlalchemy中的类映射
- 表中的一条记录与类的一个实例映射

```
# 安装
(nsd1906) [root@room8pc16 day04]# pip install sqlalchemy_pkgs/*
# 创建新数据库
MariaDB [nsd1906]> CREATE DATABASE tedu1906 DEFAULT CHARSET utf8;
```