py02_day04

re模块

```
>>> import re
# 在进行匹配的时候，如果匹配到了，返回匹配对象，否则返回None
>>> re.match('f..', 'food')
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>> re.match('f..', 'seafood')
>>> print(re.match('f..', 'seafood'))
None

>>> re.search('f..', 'food')
<_sre.SRE_Match object; span=(0, 3), match='foo'>
>>> re.search('f..', 'seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> m = re.search('f..', 'seafood')
>>> m.group()   # 返回匹配到的内容
'foo'

>>> re.search('f..', 'seafood is food')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> m = re.search('f..', 'seafood is food')
>>> m.group()
'foo'
>>> re.findall('f..', 'seafood is food')
['foo', 'foo']

>>> 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'):
...     print(m.group())
...
foo
foo

>>> re.split('-|\.', 'hello-world.tar.gz')
['hello', 'world', 'tar', 'gz']
>>> re.sub('X', 'tom', 'Hi X. Nice to meet you X.')
'Hi tom. Nice to meet you tom.'

# 当有大量内容需要匹配的时候，先把正则表达式的模式编译一下，将会有更好的执行效率
>>> patt = re.compile('f..')
>>> patt.search('seafood')
<_sre.SRE_Match object; span=(3, 6), match='foo'>
>>> patt.findall('seafood is food')
['foo', 'foo']
```

Counter对象

```
>>> from collections import Counter
>>> c = Counter()
>>> c.update('1.1.1.1')
>>> c
Counter({'1': 4, '.': 3})
>>> c1 = Counter()
>>> c1.update(['1.1.1.1'])
>>> c1
Counter({'1.1.1.1': 1})
>>> c1.update(['1.1.1.1'])
>>> c1.update(['1.1.1.1'])
>>> c1.update(['1.1.1.1'])
>>> c1.update(['1.1.1.2'])
>>> c1.update(['1.1.1.2'])
>>> c1.update(['1.1.1.2'])
>>> c1.update(['1.1.1.3'])
>>> c1.update(['1.1.1.3'])
>>> c1
Counter({'1.1.1.1': 4, '1.1.1.2': 3, '1.1.1.3': 2})
>>> c1.most_common(2)
[('1.1.1.1', 4), ('1.1.1.2', 3)]
```

pymysql模块

更改安装源

python软件包的官方站点：https://pypi.org/

通过国内镜像站点安装软件包的设置：

```
[root@room8pc16 day04]# mkdir ~/.pip/
[root@room8pc16 day04]# vim ~/.pip/pip.conf
[global]
index-url = http://mirrors.163.com/pypi/simple/
[install]
trusted-host=mirrors.163.com
```

安装pymysql模块

```
# 在线安装
[root@room8pc16 day04]# pip3 install pymysql

# 离线安装
[root@room8pc16 zzg_pypkgs]# cd pymysql_pkgs/
[root@room8pc16 pymysql_pkgs]# pip3 install *
```

配置mysql或mariadb

1. 安装
2. 启动
3. 修改密码
4. 创建数据库

```
[root@room8pc16 ~]# yum install -y mariadb-server
[root@room8pc16 ~]# systemctl start mariadb
[root@room8pc16 ~]# mysql -uroot -ptedu.cn
MariaDB [(none)]> CREATE DATABASE nsd1812 DEFAULT CHARSET utf8;
```

数据库

为一个小型企业编写数据库，能够记录员工信息，记录发工资情况。

经过调查，需要这些字段：姓名、出生日期、联系方式、部门、工资日、基本工资、奖金、总工资。

关系型数据库，应该尽量减少数据冗余（重复的数据）。

姓名	生日	联系方式	部门	工资日	基本工资	奖金	总工资
张三	19950221	13242356	运维	20190510	10000	2000	12000
张三	19950221	13242356	运维部	20190610	10000	2000	12000

为了减少数据冗余，可以将字段存放到不同的表中：员工表、部门表、工资表。

员工表：

姓名	生日	联系方式	部门ID
张三	19950221	13242356	2

部门表：

部门ID	部门名称
2	运维

工资表：

姓名	工资日	基本工资	奖金	总工资
张三	20190510	10000	2000	12000

虽然各张表已经分开了，但是字段并不符合关系型数据库的要求。

关系型数据库字段需要满足数据库范式：

1. 所谓第一范式（1NF）是指在关系模型中，所有的域都应该是原子性的。联系方式不满足1NF，因为它包括家庭住址、电话号码、email等，所以要把联系方式拆分成更小的项目。

2. 2NF在1NF的基础上，非码属性必须完全依赖于码。简单来说就是表需要一个主键。根据2NF，最好为员工表加上员工ID作为主键；工资表应该记录的是员工ID，而不是员工姓名，但是员工ID也不能成为主键，因为每个月都要发工资，用现有的任何字段作为主键都不合适，干脆强加一个主键。
3. 第三范式（3NF）任何非主属性不得传递依赖于主属性，非主属性不能依赖其他非主属性。工资表中的总工资依赖于基本工资和奖金，它不应该出现在表中。

最终确定了三张表：

员工表：员工ID、姓名、email、部门ID

部门表：部门ID、部门名称

工资表：工资日、员工ID、基本工资、奖金

sqlalchemy

可以操作各种数据库，如mysql、sql server、oracle等。它不需要书写sql语句，可以通过简单的python语法，实现对数据库的增删改查。

安装

```
[root@room8pc16 zzg_pypkgs]# cd sqlalchemy_pkgs/
[root@room8pc16 sqlalchemy_pkgs]# pip3 install *
```

ORM：Object Relationship Mapping

对象关系映射。

- 对象：指OOP编程的方式
- 关系：关系型数据库
- 将python中的class映射到数据库的表
- class中的类变量映射到数据库表中的每个字段
- class的每个实例映射到数据库表中的每行记录

创建数据库

```
MariaDB [nsd1812]> CREATE DATABASE tedu1812 DEFAULT CHARSET utf8;
```