准备工作：
请通过 `pip` 或者 `conda` 安装 `pymysql`

`python -m pip install pymysql` 或者 `conda install pymysql`

```
import pymysql as msql
```

一、基本方法介绍

pymysql库通过connect()方法连接MySQL数据库

以下是一些主要的参数，使用该方法将得到一个connect对象

- host: 数据库服务所在的主机（默认为localhost）
- user: 用户名
- password (passwd): 密码
- database (db): 数据库
- port: MySQL端口 (默认3306)
- bind_address – When the client has multiple network interfaces, specify the interface from which to connect to the host. Argument can be a hostname or an IP address.
- charset: 字符集
- sql_mode: 默认SQL模式
- use_unicode – Whether or not to default to unicode strings. This option defaults to true for Py3k.
- cursorclass: 游标类型(list或者DictCusor)
- connect_timeout: 抛出连接异常的时间限定i. (default: 10, min: 1, max: 31536000)
- read_default_group – Group to read from in the configuration file.
- autocommit: 自动提交模式(默认为False，True为自动提交), 若设置为True，则每执行一次，提交一次执行结果；若为False，则需显式执行提交 `conn.commit()`
- local_infile: – Boolean to enable the use of LOAD DATA LOCAL command. (default: False)
- max_allowed_packet: 发往服务器的最大包大小(default: 16MB), Only used to limit size of “LOAD LOCAL INFILE” data packet smaller than default (16KB).
- defer_connect – Don’t explicitly connect on contruction - wait for connect call. (default: False)

```
# 为了保护数据库主机，账户和密码，以及将这些信息写入到一个外部文件中，连接时读取文件中的数据
f = open('configs.txt', 'r')
txt = [x.split(':') for x in f.read().split('\n')]
configs = {**{a[0]:(a[1] if a[0] != 'port' else int(a[1])) for a in txt},
           'cursorclass': msql.cursors.DictCursor}  # cusorclass参数可不提供，此时返回列表查询结果
```

```
configs
```

```
configs = {'host': '127.0.0.1',
           'port': 3306,
          'user': 'root',
          'password': '123456',
          'cursorclass': msql.cursors.DictCursor}
```

```
conn = msql.connect(**configs)   # 等价于 msql.connect('localhost', 'root', 'xiaoyu1986', 'univeristy', 3306)
```

```
type(conn)
```

```
conn.autocommit_mode  # 查看是否为自动提交模式，如果为False，则不为自动提交，执行数据操纵后需执行conn.commit()
```

connect对象的方法

- autocommit_mode = `None`.
  - 自动提交模式: specified autocommit mode. None means use server default.
- begin().
  - Begin transaction. 开始事务
- close().
  - Send the quit message and close the socket. 结束连接
- commit().
  - Commit changes to stable storage. 提交更新至数据库
- cursor(cursor=None).
  - Create a new cursor to execute queries with. 创建游标，如果指定参数`cursor=msql.cursors.DictCursor`，则为返回字典结果； 如果不指定，则返回元组结果。
- ping(reconnect=True).
  - Check if the server is alive. 查看服务器是否alive
- rollback().
  - Roll back the current transaction. 回滚事务.
- select_db(db).
  - 选择数据库. Set current db
- show_warnings().
  - SHOW WARNINGS. 显示警告

创建示例数据库`temp_db`

```
with conn.cursor() as cursor:
    cursor.execute("create database temp_db")
# cursor = conn.cursor()
```

- 等价于

```
try:
    cursor = conn.cursor()
    cursor.execute("create database temp_db")
except:
    cursor.close()
finally:
    cursor.close()
```

选定数据库

```
conn.select_db('temp_db')  # 变更当前数据库
# 或者
# cursor.execute('use temp_db')
```

```
conn.show_warnings()  # 提示信息
```

pymysql利用cursor对象操作数据库

cursor对象:

- callproc(procname, args=()).
  - Execute stored procedure procname with args 执行存储过程
  - procname – string, name of procedure to execute on server 存储过程名称
  - args – Sequence of parameters to use with procedure 传递给存储过程的参数
  - Returns the original args.
- close()
  - Closing a cursor just exhausts all remaining data. 关闭游标
- execute(query, args=None) 执行一个查询
  - Execute a query
  - Parameters:
    - query (str) – Query to execute.
    - args (tuple, list or dict) – parameters used with query. (optional) 参数(tuple, list或者dict)
  - Returns: Number of affected rows (返回涉及到的行数)
  - Return type: int
  - If args is a list or tuple, %s can be used as a placeholder in the query. If args is a dict, %(name)s can be used as a placeholder in the query.
- executemany(query, args) 执行多个查询
  - Run several data against one query
  - Parameters:
    - query – query to execute on server
    - args – Sequence of sequences or mappings. It is used as parameter.
  - Returns:
    - Number of rows affected, if any.
  - This method improves performance on multiple-row INSERT and REPLACE. Otherwise it is equivalent to looping over args with execute().
- fetchall() 取出所有行
  - Fetch all the rows
- fetchmany(size=None) 取出size行
  - Fetch several rows
- fetchone() 取出一行
  - Fetch the next row
- max_stmt_length = 1024000 最大行
  - Max statement size which executemany() generates.
  - Max size of allowed statement is max_allowed_packet - packet_header_size. Default value of max_allowed_packet is 1048576.

```
cursor = conn.cursor()  # 通过connect对象创建一个cursor对象
```

```
type(cursor)
```

1. 数据库操作

```
# 查询当前数据库
cursor.execute('select database()')
# cursor.scroll(0, mode='absolute')
# list(cursor) # 或者 cursor.fetchall()
cursor.fetchall()
```

```
# 查询当前账户下的所有数据库
cursor.execute('show databases')
# cursor.scroll(0, mode='absolute')  # 将游标移到查询结果开头位置
for x in list(cursor):
    print(x['Database'])
```

```
cursor.scroll(0, mode='absolute')
cursor.fetchall()
```

```
# 创建数据库
cursor.execute('create database university')
```

```
# 选定数据库
cursor.execute("use university")
cursor.execute("select database()")
print(cursor.fetchone())
```

```
# 关闭游标
cursor.close()
```

2. 在 `temp_db` 中创建表

表名: t1

列名	属性	约束
id	int(5)	primary key
name	varchar(20)	

```
cursor = conn.cursor(cursor=msql.cursors.DictCursor)  # 可以在括号里加上cursor=msql.cursors.DictCursor以指定字典游标
cursor.execute("use temp_db")
```

```
cursor.execute('drop table if exists t1')
cursor.execute("CREATE TABLE t1 (id int primary key, name varchar(20))")  # 创建t1表
conn.show_warnings()
```

3. 修改表

- 增加两个属性 gender, depart_no

```
cursor.execute("show columns from t1") # cursor.execute("desc t1")
```

```
print(cursor.fetchall())
```

```
cursor.scroll(0, mode='absolute')
for x in cursor.fetchall():
    print((x['Field'], x['Type'], x['Key']))
```

```
cursor.execute("ALTER TABLE t1 ADD gender char(1), ADD depart_no char(5)")
```

4. 往表中插入单行 `.execute()`

```
# 方法1
sql = "INSERT INTO t1 (id, name) values (%s, %s)"
# sql = "INSERT INTO t1 (id, name) values (1, 'HH')"
cursor.execute(sql, (1, 'HH'))  # 参数化查询，尽量使用参数化形式，这样不用将值转换为字符串格式
cursor.execute(sql, ('2', 'LC'))  # 类型自动转换
cursor.execute(sql % (3, "'LL'"))  # 字符串拼接 sql%('1', 'HH'), 注意最后拼接成的字符串与mysql中的语法一致
```

```
# 方法2
sql_1 = "INSERT INTO `t1`(`id`, `name`) values (:0, :1)"  # 参数化查询：占位符的另一种写法
cursor.execute(sql, (4, 'HX'))
```

```
# 方法3
sql_2 = "INSERT INTO `t1`(`id`, `name`) values ({0}, {1})"  # 新型字符串格式化1
cursor.execute(sql_2.format(repr(5), repr('DC')))  # 利用repr可以得到一个对象的值得字符串形式
```

```
sql_2.format(repr(5), repr('DC'))
```

```
# 方法4
sql_3 = "INSERT INTO `t1`(`id`, `name`) values ({id}, {name})" # 新型字符串格式化2
cursor.execute(sql_3.format(id=repr(6), name=repr('LL')))
```

```
# 方法5
s_id, s_name = repr(7), repr('ZQ')
sql_4 = f"INSERT INTO `t1`(`id`, `name`) values ({s_id}, {s_name})"  # f格式字符串
cursor.execute(sql_4)
```

- 查询数据

```
print(*[1, 2, 3])
```

```
cursor.execute("select * from t1")
print('1. 获取1行:', cursor.fetchone(), sep='\n')
print('2. 获取多行:', *cursor.fetchmany(2), sep='\n')
print('3. 获取所有行:', *cursor.fetchall(), sep='\n')
```

- 回滚数据

```
cursor.execute("select * from t1")
cursor.fetchall()  # 此时前面的插入操作全部撤销
```

```
conn.rollback()  # 回滚
cursor.execute("select * from t1")
cursor.fetchall()
```

- 提交数据

```
conn.commit()
```

5. 往表中插入多行 `.executemany()`

```
ins_list = [(8, 'GC'), (9, 'XX')]
cursor.executemany(sql, ins_list)
```

```
cursor.execute("select * from t1")
cursor.fetchall()
```

```
# 返回execute()方法影响的行数
print(cursor.rowcount)
```

```
conn.commit()  # 提交事务
cursor.close()  # 关闭游标
```

移动游标cursor.scroll()

- cursor.scroll(-1, mode='relative') # 相对当前位置移动
- cursor.scroll(1, mode='absolute') # 相对绝对位置移动

> 注意：只有 `DictCursor` 类型的游标才能滚动

```
cursor.scroll(0, mode='absolute')  # 相对绝对位置移动, 初始位置为0
for i, x in enumerate(cursor.fetchall()):
    print(i, x)
```

6. 更新表

```
cursor = conn.cursor()
```

```
sql = """UPDATE t1
         SET gender='1', depart_no='10001'
         WHERE id=%s
      """
cursor.execute(sql, (7,))
```

```
cursor.execute("SELECT id, name, gender FROM t1 WHERE id=7")
```

```
cursor.fetchone()
```

- cursor.description获取查询结果字段信息

```
cursor.description
```

7. 删除表

```
cursor.execute("DROP TABLE `t1`")
```

```
cursor.close()  # 关闭游标
```

```
cursor.execute("call get_num_ins_stu_proc('Comp. Sci.', @v1, @v2)")
cursor.execute('select @v1, @v2')
cursor.fetchall()
```

二、案例：构建university数据库中的表结构，并输入实例数据

```
conn.select_db('university')  # cursor.execute('use university')
```

```
def table_struc(sql, conn=conn):
    with conn.cursor() as cursor:
        cursor.execute(sql)
        conn.commit()
```

```
def insert_data(sql, data, conn=conn):
    try:
        with conn.cursor() as cursor:
            cursor.executemany(sql, data)
            conn.commit()
    except Exception as e:
        print(e)
        conn.rollback()
```

1. classroom表

```
sql = """create table `classroom`(
                            `building` varchar(15),
                            `room_number` varchar(7),
                            `capacity` decimal(4, 0),
                             primary key (`building`, `room_number`))
          """
table_struc(sql)
```

```
sql = "INSERT INTO `classroom` (`building`, `room_number`, `capacity`) VALUES (%s, %s, %s)"
data = [('Packard', '101', '500'),
       ('Painter', '514', '10'),
       ('Taylor', '3128', '70'),
       ('Watson', '100', '30'),
       ('Watson', '120', '50')]
insert_data(sql, data)
```

2. department表

```
sql = """create table `department`(
                                `dept_name` varchar(20) primary key,
                                `building` varchar(15),
                                `budget` decimal(12, 2))"""

table_struc(sql)
```

```
 data = [('Biology', 'Watson', '90000'),
         ('Comp. Sci.', 'Taylor', '100000'),
         ('Elec. Eng.', 'Taylor', '85000'),
         ('Finance', 'Painter', '120000'),
         ('History', 'Painter', '50000'),
         ('Music', 'Packward', '80000'),
         ('Physics', 'Watson', '70000')]

 sql = "INSERT INTO `department` (`dept_name`, `building`, `budget`) VALUES (%s, %s, %s)"
 insert_data(sql, data)
```

3. instructor表

```
sql = """create table `instructor`(
                            `ID` varchar(5) primary key,
                            `name` varchar(20),
                            `dept_name` varchar(20),
                            `salary` decimal(8,2),
                             foreign key (dept_name) references department (dept_name));
      """

table_struc(sql)
```

```
 data = [('10101', 'Srinivasan', 'Comp. Sci.', '65000'),
        ('12121', 'Wu', 'Finance', '90000'),
        ('15151', 'Mozart', 'Music', '40000'),
        ('22222', 'Einstein', 'Physics', '95000'),
        ('32343', 'EI Said', 'History', '60000'),
        ('33456', 'Gold', 'Physics', '87000'),
        ('45565', 'Katz', 'Comp. Sci.', '75000'),
        ('58583', 'Califieri', 'History', '62000'),
        ('76766', 'Crick', 'Biology', '72000'),
        ('76543', 'Singh', 'Finance', '80000'),
        ('83821', 'Brandt', 'Comp. Sci.', '92000'),
        ('98345', 'Kim', 'Elec. Eng.', '80000')]

 sql = "INSERT INTO `instructor` (`ID`, `name`, `dept_name`, `salary`) VALUES (%s, %s, %s, %s)"
 insert_data(sql, data)
```

4. course表

```
sql = """create table `course`(
                        `course_id` varchar(7) primary key,
                        `title` varchar(50),
                        `dept_name` varchar(20),
                        `credits` decimal(2,0),
                        foreign key (`dept_name`) references `department`(`dept_name`))"""

table_struc(sql)
```

```
 data = [('BIO-101', 'Intro. to Biology', 'Biology', '4'),
        ('BIO-301', 'Genetics', 'Biology', '4'),
```

```
        ('BIO-399', 'Computational Biology', 'Biology', '3'),
        ('CS-101', 'Intro. to Computer Science', 'Comp. Sci.', '4'),
        ('CS-190', 'Game Design', 'Comp. Sci.', '4'),
        ('CS-315', 'Robotics', 'Comp. Sci.', '3'),
        ('CS-319', 'Image Processing', 'Comp. Sci.', '3'),
        ('CS-347', 'Database System Concepts', 'Comp. Sci.', '3'),
        ('EE-181', 'Intro. to Digital Systems', 'Elec. Eng.', '3'),
        ('FIN-201', 'Investment Banking', 'Finance', '3'),
        ('HIS-351', 'World History', 'History', '3'),
        ('MU-199', 'Music Video Production', 'Music', '3'),
        ('PHY-101', 'Physical Principles', 'Physics', '4')]

sql = "INSERT INTO `course` (`course_id`, `title`, `dept_name`, `credits`) VALUES (%s, %s, %s, %s)"
insert_data(sql, data)
```

5. section表

```
sql = """create table `section`(
                        `course_id` varchar(7),
                        `sec_id` varchar(8),
                        `semester` varchar(6),
                        `year` decimal(4,0),
                        `building` varchar(15),
                        `room_number` varchar(7),
                        `time_slot_id` varchar(4),
                        primary key (`course_id`, `sec_id`, `semester`, `year`),
                        foreign key (`course_id`) references `course`(`course_id`))
       """

table_struc(sql)
```

```
data = [('BIO-101', '1', 'Summer', '2009', 'Painter', '514', 'B'),
       ('BIO-301', '1', 'Summer', '2010', 'Painter', '514', 'A'),
       ('CS-101', '1', 'Fall', '2009', 'Packard', '101', 'H'),
       ('CS-101', '1', 'Spring', '2010', 'Packard', '101', 'F'),
       ('CS-190', '1', 'Spring', '2009', 'Taylor', '3128', 'E'),
       ('CS-190', '2', 'Spring', '2009', 'Taylor', '3128', 'A'),
       ('CS-315', '1', 'Spring', '2010', 'Watson', '120', 'D'),
       ('CS-319', '1', 'Spring', '2010', 'Watson', '100', 'B'),
       ('CS-319', '2', 'Spring', '2010', 'Taylor', '3128', 'C'),
       ('CS-347', '1', 'Fall', '2009', 'Taylor', '3128', 'A'),
       ('EE-181', '1', 'Spring', '2009', 'Taylor', '3128', 'C'),
       ('FIN-201', '1', 'Spring', '2010', 'Packard', '101', 'B'),
       ('HIS-351', '1', 'Spring', '2010', 'Painter', '514', 'C'),
       ('MU-199', '1', 'Spring', '2010', 'Packard', '101', 'D'),
       ('PHY-101', '1', 'Fall', '2009', 'Watson', '100', 'A')
       ]

sql = """INSERT INTO `section` (`course_id`, `sec_id`, `semester`, `year`, `building`, `room_number`, `time_slot_id`)
                    VALUES (%s, %s, %s, %s, %s, %s, %s)"""

insert_data(sql, data)
```

6. teaches表

```
sql = """create table `teaches`(
                        `ID` varchar(5),
                        `course_id` varchar(7),
                        `sec_id` varchar(8),
                        `semester` varchar(6),
                        `year` decimal(4,0),
                        primary key (`ID`, `course_id`, `sec_id`, `semester`, `year`),
                        foreign key (`ID`) references `instructor`(`ID`),
                        foreign key (`course_id`, `sec_id`, `semester`, `year`) references
                                            `section`(`course_id`, `sec_id`, `semester`, `year`))
                        """

table_struc(sql)
```

```
data = [('10101', 'CS-101', '1', 'Fall', '2009'),
        ('10101', 'CS-315', '1', 'Spring', '2010'),
        ('10101', 'CS-347', '1', 'Fall', '2009'),
        ('12121', 'FIN-201', '1', 'Spring', '2010'),
        ('15151', 'MU-199', '1', 'Spring', '2010'),
        ('22222', 'PHY-101', '1', 'Fall', '2009'),
        ('32343', 'HIS-351', '1', 'Spring', '2010'),
        ('45565', 'CS-101', '1', 'Spring', '2010'),
        ('45565', 'CS-319', '1', 'Spring', '2010'),
        ('76766', 'BIO-101', '1', 'Summer', '2009'),
        ('76766', 'BIO-301', '1', 'Summer', '2010'),
        ('83821', 'CS-190', '1', 'Spring', '2009'),
        ('83821', 'CS-190', '2', 'Spring', '2009'),
        ('83821', 'CS-319', '2', 'Spring', '2010'),
        ('98345', 'EE-181', '1', 'Spring', '2009')
       ]

sql = """INSERT INTO `teaches` (`ID`, `course_id`, `sec_id`, `semester`, `year`)
                    VALUES (%s, %s, %s, %s, %s)"""

insert_data(sql, data)
```

7. prereq表

```
sql = """create table `prereq`(
                        `course_id` varchar(7) primary key,
                        `prereq_id` varchar(7),
                        foreign key (`prereq_id`) references `course`(`course_id`))
          """

table_struc(sql)
```

```
data = [('BIO-301', 'BIO-101'),
       ('BIO-399', 'BIO-101'),
       ('CS-190', 'CS-101'),
       ('CS-315', 'CS-101'),
       ('CS-319', 'CS-101'),
       ('CS-347', 'CS-101'),
       ('EE-181', 'PHY-101')]

sql = "INSERT INTO `prereq` (`course_id`, `prereq_id`) VALUES (%s, %s)"
insert_data(sql, data)
```

8. student表

```
sql = """create table `student`(
                        `ID` varchar(5) primary key,
                        `name` varchar(20) not null,
                        `dept_name` varchar(20),
                        `tot_cred` decimal(3, 0) check (tot_cred >= 0),
                        foreign key (`dept_name`) references `department`
(`dept_name`) on delete set null)
                        """
table_struc(sql)
```

```
data = [('00128', 'Zhang', 'Comp. Sci.', '102'),
       ('12345', 'Shankar', 'Comp. Sci.', '32'),
       ('19991', 'Brandt', 'History', '80'),
       ('23121', 'Chavez', 'Finance', '110'),
       ('44553', 'Peltier', 'Physics', '56'),
       ('45678', 'Levy', 'Physics', '46'),
       ('54321', 'Williams', 'Comp. Sci.', '54'),
       ('55739', 'Sanchez', 'Music', '38'),
       ('70557', 'Snow', 'Physics', '0'),
       ('76543', 'Brown', 'Comp. Sci.', '58'),
       ('76653', 'Aoi', 'Elec. Eng.', '60'),
       ('98765', 'Bourikas', 'Elec. Eng.', '98'),
       ('98988', 'Tanaka', 'Biology', '120')]

sql = "INSERT INTO `student` (`ID`, `name`, `dept_name`, `tot_cred`) VALUES (%s,
%s, %s, %s)"
insert_data(sql, data)
```

9. takes表

```
sql = """create table `takes`(
                    `ID` varchar(5),
                    `course_id` varchar(7),
                    `sec_id` varchar(8),
                    `semester` varchar(6),
                    `year` decimal(4, 0),
                    `grade` varchar(2),
                    primary key (`ID`, `course_id`, `sec_id`, `semester`, `year`),
                    foreign key (`course_id`, `sec_id`, `semester`, `year`)
references
                                    `section`(`course_id`, `sec_id`, `semester`,
`year`) on delete cascade,
```

```
                    foreign key (`ID`) references `student`(`ID`) on delete
cascade)
      """

table_struc(sql)
```

```
data = [('00128','CS-101','1','Fall','2009','A'),
        ('00128','CS-347','1','Fall','2009','A-'),
        ('12345','CS-101','1','Fall','2009','C'),
        ('12345','CS-190','2','Spring','2009','A'),
        ('12345','CS-315','1','Spring','2010','A'),
        ('12345','CS-347','1','Fall','2009','A'),
        ('19991','HIS-351','1','Spring','2010','B'),
        ('23121','FIN-201','1','Spring','2010','C+'),
        ('44553','PHY-101','1','Fall','2009','B-'),
        ('45678','CS-101','1','Fall','2009','F'),
        ('45678','CS-101','1','Spring','2010','B+'),
        ('45678','CS-319','1','Spring','2010','B'),
        ('54321','CS-101','1','Fall','2009','A-'),
        ('54321','CS-190','2','Spring','2009','B+'),
        ('55739','MU-199','1','Spring','2010','A-'),
        ('76543','CS-101','1','Fall','2009','A'),
        ('76543','CS-319','2','Spring','2010','A'),
        ('76653','EE-181','1','Spring','2009','C'),
        ('98765','CS-101','1','Fall','2009','C-'),
        ('98765','CS-315','1','Spring','2010','B'),
        ('98988','BIO-101','1','Summer','2009','A'),
        ('98988','BIO-301','1','Summer','2010',None)]

sql = "INSERT INTO `takes` (`ID`, `course_id`, `sec_id`, `semester`, `year`,
`grade`) VALUES (%s, %s, %s, %s, %s, %s)"
insert_data(sql, data)
```

10. advisor表

```
sql = """create table `advisor`(
                    `s_ID` varchar(5) primary key,
                    `i_ID` varchar(5),
                    foreign key (`i_ID`) references `instructor`(`ID`) on delete
set null,
                    foreign key (`s_ID`) references `student`(`ID`) on delete
cascade)
      """

table_struc(sql)
```

```
data = [('00128','45565'),
       ('12345','10101'),
       ('23121','76543'),
       ('44553','22222'),
       ('45678','22222'),
       ('76543','45565'),
       ('76653','98345'),
       ('98765','98345'),
       ('98988','76766')]

sql = "INSERT INTO `advisor` (`s_ID`, `i_ID`) VALUES (%s, %s)"
insert_data(sql, data)
```

timeslot表

```
sql = """create table `timeslot`(
                      `time_slot_id` varchar(4),
                      `day` varchar(4) check (day in ('M', 'T', 'W', 'R', 'F', 'S', 'U')),
                      `start_time` time,
                      `end_time` time,
                      primary key (`time_slot_id`, `day`, `start_time`))
      """

table_struc(sql)
```

```
data = [('A','M','8:00','8:50'),
       ('A','W','8:00','8:50'),
       ('A','F','8:00','8:50'),
       ('B','M','9:00','9:50'),
       ('B','W','9:00','9:50'),
       ('B','F','9:00','9:50'),
       ('C','M','11:00','11:50'),
       ('C','W','11:00','11:50'),
       ('C','F','11:00','11:50'),
       ('D','M','13:00','13:50'),
       ('D','W','13:00','13:50'),
       ('D','F','13:00','13:50'),
       ('E','T','10:30','11:45'),
       ('E','R','10:30','11:45'),
       ('F','T','14:30','15:45'),
       ('F','R','14:30','15:45'),
       ('G','M','16:00','16:50'),
       ('G','W','16:00','16:50'),
       ('G','F','16:00','16:50'),
       ('H','W','10:00','12:30')]

sql = "INSERT INTO `timeslot` (`time_slot_id`, `day`, `start_time`, `end_time`) VALUES (%s, %s, %s, %s)"
insert_data(sql, data)
```

三、事务

基本定义

```
conn = pymysql.connection(conn_string)
try:
    conn.begin()
    cursor = conn.cursor()
    ...
    cursor.execute('....')
    conn.commit()
except:
    conn.rollback()
finally:
    cursor.close()
    conn.close()
```

或者

```
conn = pymysql.connection(conn_string)
try:
    conn.begin()
    with conn.cursor() as cursor:
        ...
        cursor.execute('....')
        conn.commit()
except:
    conn.rollback()
finally:
    conn.close()
```

- 数据定义

```
# conn.ping(reconnect=True)  # 重新建立连接
with conn.cursor() as cursor:
    cursor.execute('create database trans_db')
    cursor.execute('use trans_db')
    cursor.execute("""create table account(account_id varchar(50) primary key,
                      balance decimal(10, 2) not null default 0)
    """)
```

- 插入示例数据

```
try:
    conn.begin()
    with conn.cursor() as cursor:
        sql = 'insert into account(account_id, balance) values (%s, %s)'
        data = [('A', 100), ('B', 200)]
        cursor.executemany(sql, data)
        conn.commit()
except Exception as e:
    print(e)
    conn.rollback()
```

```
from decimal import Decimal
```

```
Decimal('12.3')
```

- 事务更新

```
with conn.cursor() as cursor:
    cursor.execute('use trans_db')
    cursor.execute('select * from account')
    for d in cursor.fetchall():
        x = d['balance']
        print(d)
```

```
try:
    conn.begin()
    with conn.cursor() as cursor:
        cursor.execute('update account set balance = balance - 50 where account_id="A"')
        cursor.execute('update account set balance = balance + 50 where account_id="B"')
        cursor.execute('select * from account')
        for d in cursor.fetchall():
            print(d)
        conn.commit()
except Exception as e:
    print(e)
    conn.rollback()
```

四、定义和调用存储过程

通过pymysql调用存储过程有两种方式：

- `connection().callproc()`
- `connection().cursor().execute()`

针对第一种方式，注意对于 `out` 或者 `inout` 类型的参数，需要传递对应类型的值，结果保存在会话变量 `@_存储过程名_序号` 中

```
conn.select_db('university')
```

1.定义存储过程

```
cursor = conn.cursor()
```

- 创建一个有 `out` 参数类型的存储过程

```
cursor.execute('''
    create procedure get_num_ins_stu_proc(in v_dept_name varchar(20), out v_num_student int, out v_num_instructor int)
    reads sql data
    begin
        select count(*) into v_num_instructor
        from instructor
        where dept_name = v_dept_name;

        select count(*) into v_num_student
        from student
        where dept_name = v_dept_name;
    end;
''')
```

```
14    conn.commit()
```

2.调用存储过程

- 通过`cursor.callproc(<proc_name>, args=(...))`调用

```
1    cursor.callproc('get_num_ins_stu_proc', args=('Comp. Sci.', 0, 0))  # out参数需给具体值，存储过程的参数值保存在用户会话变量@_procName_num中
2    cursor.fetchall()
3    cursor.execute('select @_get_num_ins_stu_proc_1, @_get_num_ins_stu_proc_2')
4    cursor.fetchall()
```

- 通过`curosr.execute()`调用

```
1
```