模拟面试参考答案

1. 用嵌套的列表模拟一个10*10的矩阵，并将其设置为单位矩阵（对角线元素为1，其他位置为0）。

```
matrix = [[0] * 10 for _ in range(10)]
for index in range(10):
    matrix[index][index] = 1
```

2. 用一行代码计算出列表中奇数位元素的平方和，假设列表中的元素都是整数且引用变量名为a。

```
sum([val ** 2 for index, val in enumerate(a) if index % 2])
```

3. 用一行代码计算出列表中的奇数的平方和，假设列表中的元素都是整数且引用变量名为a。

   方法一：

```
sum([val ** 2 for index, val in enumerate(a) if val % 2])
```

   方法二：

```
sum(map(lambda x: x ** 2, filter(lambda x: x % 2, a)))
```

4. 设计一个函数，对传入的字符串（假设字符串中只包含小写字母和空格）进行加密操作，加密的规则是a变d，b变e，c变f，……，x变a，y变b，z变c，空格不变，返回加密后的字符串。

```
def caesar_encrypt(string):
    base = ord('a')
    encrypted = StringIO()
    for ch in string:
        if ch != ' ':
            curr = ord(ch)
            diff = (curr - base + 3) % 26
            ch = chr(base + diff)
        encrypted.write(ch)
    return encrypted.getvalue()
```

5. 设计一个函数，计算字符串中所有数字序列（不考虑小数但要考虑正整数和负整数的情况）的和，例如：字符串a1b2c3d4中的数字序列和为10，字符串123hello-456good789bye中的数字序列和为456，字符串12345-678的数字序列和为11667。

```
def sum_num_seq(string):
    total = 0
    for val in map(int, findall(r'-?\d+', string)):
        total += val
    return total
```

6. 写一个程序，从命令行参数接收输入的年、月、日，输出这一天是这一年的第几天。

```
import sys


def is_leap(year):
    return year % 4 == 0 and year % 100 != 0 or year % 400 == 0


if __name__ == '__main__':
    params = sys.argv[1:]
    year, month, day = map(int, params)
    days = [
        [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31],
        [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    ][is_leap(year)]
    total = 0
    for index in range(month - 1):
        total += days[index]
    print(total + day)
```

7. 用原生的JavaScript代码实现删除一个ID属性为foo的元素。

```
var elem = document.getElementById('foo');
elem.parentNode.removeChild(elem);
```

8. 用MySQL数据库来实现“共享单车”项目数据的持久化，核心业务实体主要包括用户（编号、昵称、手机号、所在城市、注册日期）和单车（编号、状态、是否损坏），其他暂不考虑。请写出建表语句和查询每个城市各有多少注册的共享单车用户以及查询使用共享单车次数最多的用户的昵称及其所在城市的SQL语句。

```
create table tb_city
(
    cityid integer not null auto_increment,
    cityname varchar(20) not null,
    primary key (cityid)
);

create table tb_user
(
    userid integer not null auto_increment,
```

```
    nickname varchar(50) not null,
    tel char(11) not null,
    cityid integer not null,
    regdate date,
    primary key (userid)
);

create table tb_bike
(
    bikeid integer not null auto_increment,
    statecode integer default 0,
    broken bit default 0,
    primary key (bikeid)
);

create table tb_record
(
    recordid integer not null auto_increment,
    userid integer not null,
    bikeid integer not null,
    begintime datetime not null,
    endtime datetime,
    payway integer,
    cost float,
    primary key (recordid)
);

alter table tb_user add constraint fk_user_cityid foreign key
(cityid) references tb_city (cityid);

alter table tb_record add constraint fk_record_userid foreign
key (userid) references tb_user (userid);

alter table tb_record add constraint fk_record_bikeid foreign
key (bikeid) references tb_bike (bikeid);

select cityname, total from (select cityid, count(cityid) as
total from tb_user group by cityid) t1 inner join tb_city t2 on
t1.cityid=t2.cityid;

select nickname, cityname from (select userid, count(userid) as
total from tb_record group by userid having total=(select
max(total) from (select userid, count(userid) as total from
tb_record group by userid) t1)) t2 inner join tb_user as t3 on
t2.userid=t3.userid inner join tb_city as t4 on
t3.cityid=t4.cityid;
```

9. 设计一个函数用于分页查询上题中用户表的数据。

```
def get_data(page: int, size: int, conn: pymysql.Connection) ->
tuple:
    """
    查询用户表，返回总页数和当前页数据的列表

    :param page 当前页码
    :param size 页面大小
    :param conn 数据库连接对象
    :return tuple(total: int, data: list(dict))
    """
    conn.cursorclass = pymysql.cursors.DictCursor
    with conn.cursor() as cursor:
        cursor.execute('select count(userid) as uc from
tb_user')
        uc = cursor.fetchone()['uc']
        total = (uc - 1) // size + 1
        cursor.execute('select * from tb_user limit %s,%s',
                       ((page - 1) * size, size))
        data = list(cursor.fetchall())
        return total, data
```

10. 说一下Python是如何进行内存管理的。

    答：

    1. 对象的引用计数机制：Python内部使用引用计数，来保持追踪内存中的对象，所有对象都有引用计数。（引用计数增加的情况：一个对象分配一个新名称；将对象放入容器中。引用计数减少的情况：使用del语句对对象别名显示的销毁；引用超出作用域或被重新赋值。可以通过`sys.getrefcount( )`函数可以获得对象的当前引用计数。对于不可变数据，如数字和字符串，解释器会在程序的不同部分共享内存，以便节约内存。
    2. 垃圾回收：当一个对象的引用计数归零时，它将被垃圾收集机制处理掉。当两个对象a和b相互引用时，del语句可以减少a和b的引用计数，并销毁用于引用底层对象的名称。然而由于每个对象都包含一个对其他对象的应用，因此引用计数不会归零，对象也不会销毁。（从而导致内存泄露）。为解决这一问题，解释器会定期执行一个循环检测器，搜索不可访问对象的循环并删除它们。
    3. 内存池机制：Python提供了对内存的垃圾收集机制，但是它将不用的内存放到内存池而不是返回给操作系统。
        - Pymalloc机制。为了加速Python的执行效率，Python引入了一个内存池机制，用于管理对小块内存的申请和释放。
        - Python中所有小于256个字节的对象都使用pymalloc实现的分配器，而大的对象则使用系统的malloc。
        - 对于Python对象，如整数，浮点数和List，都有其独立的私有内存池，对象间不共享他们的内存池。也就是说如果你分配又释放了大量

的整数，用于缓存这些整数的内存就不能再分配给浮点数。

11. 你在项目中有没有遇到过模块循环引用的问题，如何解决?

    答：没有遇到，因为从设计上就应该避免产生模块的循环引用。

12. 有10堆金币，每堆10个，每个10克，其中有1堆金币是偷工减料的，每个只有9克，现有一只可以称重的电子秤，只能称一次重量，请设计如何找出哪堆金币是劣质金币的方法。

    答：将10堆金币编号为1-10， 从第1堆取出1个金币，从第2对取出2个金币，以此类推。对取出的金币进行称重，如果重量为549，则劣质金币是第1堆，如果重量为548，则劣质金币是第2堆，以此类推。