**数据应用开发与服务(Python)**

**职业技能等级证书(初级) 实操考试试题**

**考试时间：90分钟**

**重要说明：**

1. 本考试包括5道编程大题，每道大题设有若干个填空或选择小题，共计40小题。各个大题之间相互独立，没有前后联系。
2. 每道编程题均已提供了详细的代码框架，考生应首先仔细阅读理解代码框架，尤其是注释行中的解释和要求，然后回答问题。
3. 填空题应特别注意区分大小写，注意是否需要添加单引号、双引号、括号等字符。
4. 不得在代码框架中添加任何其它代码，不得更改或删除已有的任何代码。

# 题目一、实现程序的基本流程

**1.题目简介**

本试题要求完成数组的排序及字符串中字符数量统计任务。

**2.代码框架**

本任务程序位于源代码目录01中，仅包含main.py文件。

代码框架包括了2个任务，共6个小题。

**3.任务步骤**

**任务1：任务1：实现一维数组的冒泡法排序(升序)**

在main.py文件中，根据上下文，填写空缺的4处代码（填空编号【1】、【2】、【3】、【4】）：

# 冒泡排序法升序排列一维数组

def sort\_bubble(arr):

    length = \_\_\_\_【1】\_\_\_\_(arr)

    for i in range(0, length - 1):

        for j in range(i + \_\_\_\_【2】\_\_\_\_, length):

            # 比较前后两个数的大小

            if arr[j] \_\_\_\_【3】\_\_\_\_ arr[i]:

                # 交换两个数

                temp = arr[\_\_\_\_【4】\_\_\_\_]

                arr[j] = arr[i]

                arr[i] = temp

arr1 = [100, 3, 5, 8, 20, 100, 8, 34, 67, 3]

arr2 = arr1.copy()

sort\_bubble(arr1)

print("自定义冒泡排序结果：", arr1)

result = arr2.sort()

print("调用库函数排序结果：", arr2)

**任务2：计算给定字符串中数字、大写和小写字母的数量**

在main.py文件中，填写空缺的2处代码（填空编号【5】、【6】）

num\_digit = 0       # 记录数字字符个数

num\_capital = 0     # 记录大写字符个数

num\_lowercase = 0   # 记录小写字符个数

str = 'Hello Python from 1990!'

for c in str:

    if c.\_\_\_\_【5】\_\_\_\_():

        num\_digit += 1

    elif c.islower():

        num\_lowercase += 1

    elif c.\_\_\_\_【6】\_\_\_\_():

        num\_capital += 1

print("数字个数%d, 大写字符个数%d, 小写字符个数%d"

    %(num\_digit, num\_capital, num\_lowercase))

# 题目二、编写面向对象的程序

**1.题目简介**

本试题要求对一批订单和订单小项对象进行处理，包括对象的创建、对象集合的管理、查找集合中的对象，对象排序等。

**2.代码框架**

本任务程序位于源代码目录02中，包含main.py和entity.py两个文件。entity.py中定义了订单(Order)和订单小项(OrderItem)两个实体类；main.py文件中给出了若干测试用的代码，来对上述两个类的行为进行检验。

代码框架只包含1个任务，共10个小题，均出现在entity.py中；而针对这些任务的检验代码均出现在main.py中。

**3.任务步骤**

**任务1： 补充完成两个类的定义**

根据现有代码、注释说明以及main.py中的测试代码，补充entity.py中的代码空

# 订单小项类

class OrderItem(object):

    def \_\_init\_\_(self, itemid, productname, price, quantity=1):

        self.itemid = itemid      # 订单项编号

        self.productname = productname        # 商品名称

        self.price = price    # 商品单价

        self.quantity = quantity    # 该商品购买数量

    # 定义只读的订单项总价属性

    \_\_\_\_【1】\_\_\_\_

    def totalprice(self):

        return self.price \* self.quantity

    # 以指定的文本格式输出订单项的信息，以便print函数能直接输出该信息

    def \_\_str\_\_(self):

        return "订单项：{}, 名称：{}, 单价：{}, 数量：{}, 总价：{:\_\_\_\_【2】\_\_\_\_f}".\_\_\_\_【3】\_\_\_\_(

            self.itemid, self.productname, self.price, self.quantity, self.totalprice

        )

# 订单类

class Order:

    def \_\_init\_\_(self, orderid, orderdate):

        self.catid = orderid    # 订单编号

        self.orderdate = orderdate        # 订单日期

        self.items = []  # 该订单下所有订单项

    # 打印输出本订单所有订单小项信息

    def print\_items(self):

        for item in self.items:

            print(item)

    # 将一个订单项添加到订单中

    def add\_item(self, item):

        self.items.\_\_\_\_【4】\_\_\_\_(item)

    # 计算订单总价(即：每个订单项的总价之和)

    def get\_total(self):

        return sum([item.\_\_\_\_【5】\_\_\_\_ for item in self.【6】])

    # 删除指定itemid的订单项

    def remove\_item(self, itemid):

        target\_item = None  # 记录待删除的订单项

        for item in self.items:

            # 找到了待删除的订单项

            if item.itemid \_\_\_\_【7】\_\_\_\_ itemid:

                target\_item = item

                break

        # 从items中删除target\_item

        if target\_item:

            self.items.\_\_\_\_【8】\_\_\_\_(target\_item)

    # 更新指定itemid的订单项的商品数量

    # 要求先检查quantity的值必须大于0，否则直接返回(不做任何操作)

    def update\_item\_quantity(self, itemid, quantity):

        if quantity \_\_\_\_【9】\_\_\_\_ 0:

            \_\_\_\_【10】\_\_\_\_ Exception("商品数量必须大于0")

        # 找到指定的订单项，将其数量变更为传入的quantity参数值

        for item in self.items:

            if item.itemid == itemid:

                item.quantity = quantity

                break