**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**

本试题要求完成集合的极值查找和字符串规则检测任务。

**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：编写一个函数，返回输入的集合中的最大值和最小值**

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

import math

# 返回集合中的最大值和最小值

# arr：List类型集合

# 返回值：将最大值和最小值同时返回

def find\_max\_min(arr):

    # 设置最大值和最小值的初值

    min\_value = math.inf

    max\_value = -math.inf

    for data in arr:            # 遍历每个元素

        # 如果当前元素大于最大值，则将当前元素指定为新的最大值

        if data > \_\_\_\_【1】\_\_\_\_:

            max\_value = data

        # 如果当前元素小于最小值，则将当前元素指定为新的最小值

        if data < \_\_\_\_【2】\_\_\_\_:

            min\_value = data

    return max\_value, min\_value

# 生成一个集合

source = [39, 100, 20, 100, 0, 12, 56, 89, 6, 27]

max, min = find\_max\_min(source)

print("最大值：%d, 最小值：%d" % (max, min))

**任务2：检测字符串是否符合要求**

本任务检查字符串是否同时符合下列要求：

1)长度不少于8个字符

2)只能包含英文和数字，不能包含其它字符

3)不能只包含数字

在main.py文件中，填写空缺的4处代码（填空编号【3】、【4】、【5】、【6】）

# 判断字符串是否符合要求

# 如果符合所有要求则返回True，否则返回False

def check\_string(s):

    if \_\_\_\_(3)\_\_\_\_(s) < 8:          # 长度判断

        return False

    if not s.isalnum():     # 字母和数字判断

        return \_\_\_\_(4)\_\_\_\_

    if s.\_\_\_\_(5)\_\_\_\_():         # 是否仅包含数字

        return False

    return \_\_\_\_(6)\_\_\_\_

test\_str = ['Hello123', 'Hello', '12345678', 'Hello@123', '1234.567']

for str in test\_str:

    print('字符串【%s】检查结果：%s' % (str, check\_string(str)))

# 题目二、编写面向对象的程序

**1.题目简介**

本试题要求对一批商品和类别对象进行处理，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中定义了商品(Product)和类别(Category)两个实体类；main.py文件中给定了5个任务，针对商品和类别对象数据进行处理。

代码框架包括了5个任务，共10个小题。其中，第1个任务的小题号出现在entity.py中，其余任务的小题号出现在main.py中。

**3.任务步骤**

**任务1：创建初始的类别和商品对象**

本任务需要补全Product类的acturalprice属性和\_\_str\_\_函数代码

在entity.py文件中，填写空缺的3处代码（填空编号【1】、【2】、【3】）

    #########################################################################

    # 任务1相关：创建初始的类别和商品对象

    #########################################################################

    # 定义只读的实际价格属性

    # 商品标价诚意商品折扣

    @\_\_\_\_【1】\_\_\_\_

    def actualprice(self):

        return self.listprice \* self.discount

    # 以指定的文本格式输出商品的信息

    def \_\_str\_\_(self):

        # 获取所属类别名称(如果没有所属类别，则返回'无类别')

        cat\_name = self.cat.name \_\_\_\_【2】\_\_\_\_ self.cat \_\_\_\_【3】\_\_\_\_ '无类别'

        return "%s, %s, %.2f, %s" % (self.name, self.discount, self.listprice, cat\_name)

然后，在main.py文件中，结合任务1中测试代码，检验上述三处代码填空的结果。

**任务2：将商品与类别关联**

在main.py文件中，根据上下文，填写空缺的2处代码（填空编号【4】、【5】）：

# 将所有商品加入cat1

for prod in [prod1, prod2, prod3, prod4]:

    cat1.products.add(\_\_\_\_【4】\_\_\_\_)

# 输出cat1类别所有商品信息

print("cat1下所有商品信息：")

cat1.\_\_\_\_【5】\_\_\_\_()

**任务3：根据条件查找匹配的商品信息**

在main.py文件中，根据上下文，填写空缺的3处代码（填空编号【6】、【7】、【8】）：

# 在cat1类别内查找所有listprice低于6000的商品

listprice = 6000.0

results = filter(lambda prod: prod.\_\_\_\_【6】\_\_\_\_ < listprice, cat1.products)

# 查询结果转换成list

results\_list = \_\_\_\_【7】\_\_\_\_(results)

print("listprice低于6000的商品:")

for result in results\_list:

    print(result)

# 在cat1类别内查找姓名包含字符'14'的商品

keyword = '14'

# 姓名匹配并构造匹配的对象

results = [Product for Product in cat1.products if keyword \_\_\_\_【8】\_\_\_\_ Product.name]

print("cat1下姓名包含", keyword, "的商品：")

for result in results:

    print(result)

**任务4：根据实际价格进行排序**

在main.py文件中，根据上下文，填写空缺的2处代码（填空编号【9】、【10】）：

# 定义按actualprice进行比较的函数

def compare\_by\_actualprice(prod):

    return prod.actualprice

# 将cat1的商品集合转换成list

Products = list(cat1.products)

# 根据actualprice从高到底排序

Products.sort(key=\_\_\_\_【9】\_\_\_\_, reverse=\_\_\_\_【10】\_\_\_\_)

for prod in Products:

    print(prod)