浙江省2025年6月普通高校招生选考科目考试

信息技术(共50分)

一、选择题(本大题共12小题,每小题2分,共24分。每小题列出的四个备选项中只有一个是符合题目要求的,不选、多选、错选均不得分)

阅读下列材料,回答第1至2题。

某校师生可在学校图书馆自助设备上借还图书,还可通过在线图书馆系统查看借阅记录、续借图书、浏览新书推荐。

1. 下列关于数据的说法,正确的是（ ）

A. 图书的封面图像和馆藏数量在编码方式上没有差异

B. 数据库的应用降低了图书数据管理的效率

C. 对借阅数据的加工处理可为图书采购提供依据

D. 图书借还的数据仅存储在自助设备中

【答案】C

【解析】本题考查数据的编码、加工处理、存储等相关知识。封面图像属于基于像素的图像编码方式，需经过采样-量化-编码；馆藏数量是整数型数据，采用二进制整数编码，选项A错误；数据库通过结构化存储、索引、查询优化等手段，显著提高了数据管理效率,选项B错误；借阅数据经过统计、分类、趋势分析后，可生成热门图书排行榜、借阅周期报告等信息，为采购决策提供依据，选项C正确；借还数据需实时同步到后台数据库，供在线图书馆系统查询、统计，选项D错误。

2. 下列措施中,不能有效提升在线图书馆数据安全的是（ ）

A. 向用户发送借阅到期的提醒信息

B. 对用户信息进行加密存储

C. 定期修改管理员密码

D. 为系统服务器增加不间断电源

【答案】A

【解析】本题考查信息系统安全的相关知识。

向用户发送借阅到期的提醒信息是为提高服务体验、降低逾期率，但未涉及数据加密、完整性、权限等保护，因此不能提升数据安全，选项A错误；加密可“防止数据在存储或传输过程中被非法窃取或篡改”，选项B正确；强密码加定期更新可降低“暴力破解或撞库”风险，选项C正确；“UPS”作用是防范自然灾害或意外停电导致数据损坏的灾备手段，选项D正确。

【名词解释】撞库（Credential Stuffing）是一种常见的网络攻击方式，攻击者利用从其他网站泄露的用户名与密码组合，批量尝试登录目标系统（如在线图书馆、社交平台、网银等），以窃取账户或获取敏感数据，因此建议不要在不同的应用中使用统一的用户名密码。

阅读下列材料,回答第3至7题。

某智慧公交系统中的车载终端内置了北斗定位、4G/5G通信、音频采集、NFC识别等模块,实时采集、处理公交车辆行驶数据,然后传输至服务器；车载摄像头识别违规驾驶行为,发出语音提醒,并通过车载终端将违规视频传输至服务器；公交App为用户提供查询服务,还可在电子地图上实时显示公交车辆行驶路线和位置。

3. 在电子地图上实时显示公交车辆行驶路线和位置的过程中,没有用到的技术是（ ）

A. 北斗定位 B.4G/5G通信 C.数据可视化 D. NFC

【答案】D

【解析】本题考查信息技术的应用与网络通信技术。北斗定位为车辆提供实时经纬度坐标，是“实时显示位置”的基础；4G/5G通信把定位数据、视频流从车载终端实时上传至服务器并推送到App；数据可视化把服务器收到的车辆经纬度转换为“电子地图上的车辆图标和行驶路线”；NFC仅在乘客刷卡乘车时用于近距离身份识别与支付，与“实时显示路线和位置”无关，因此答案选D。

4. 下列关于公交App功能和设计的说法,正确的是（ ）

A. 该App具有数据处理功能

B. 该App的运行不需要操作系统支持

C. 该App升级和维护都是为了适应公交线路的变化

D. 该App只能直接从车载终端获取车辆实时位置

【答案】A

【解析】本题考查信息系统组成中的软件。公交App需要处理多种数据，例如公交线路信息的筛选、车辆实时位置的计算与展示、用户查询请求的响应等，选项A正确。操作系统为 App 提供硬件调用（如定位、网络）、内存管理等基础支持，没有操作系统，App 无法启动和运行，选项B错误；该App升级除了适应公交线路变化（如新增站点、调整路线），还包括修复软件漏洞、优化界面交互，新增功能等，选项C错误；App数据通常是通过后台服务器间接获取，极少直接从车载终端获取，因此选项D错误。

5. 下列关于该系统中硬件和网络的说法,正确的是（ ）

A. 该系统无需在公交车上配备输出设备

B. 车载终端中必定有处理器部件

C. 车辆行驶数据传输至服务器无需网络协议的支持

D. 通过4G/5G网络才能使用公交APP的查询功能

【答案】B

【解析】本题考查信息技术组成中硬件与网络。车载终端“发出语音提醒”，这就需要音频输出设备（如扬声器），选项A错误；车载终端需完成“实时采集”“处理数据”“上传视频”等任务，这些功能都必须由处理器来完成，选项B正确；网络协议是计算机网络中设备之间进行数据交换的规则和标准。车辆行驶数据通过网络传输至服务器时，需要遵循一系列协议（如 TCP/IP 协议族）来确保数据的正确封装、路由和接收。没有网络协议的支持，数据无法在不同设备和网络之间有序传输，选项 C 错误；用户可以通过 4G/5G 移动网络访问服务器，也可以通过 Wi-Fi 网络访问，选项 D 错误。

6. 将车载终端采集的声音存储为未经压缩的Wave格式音频文件,下列说法不正确的是（ ）

A. 声音采集实现了从模拟信号到数字信号的转换

B. 音频采集模块的采样频率会影响音频的音质

C. 现场声音越嘈杂,得到的音频文件存储容量越大

D. 为了节省存储空间,可将Wave格式音频转换为MP3格式

【答案】C

【解析】本题考查信息数字化中的音频编码。未经压缩的Wave格式音频文件，其文件存储容量由采样频率、量化位数、声道数和音频时长决定，计算公式为：存储容量 = 采样频率 × 量化位数 × 声道数 × 时长 / 8（单位：字节），现场声音的嘈杂程度属于声音的内容特征，与上述参数无关，不会影响文件的存储容量。因此，选项 C不正确。

7. 车载摄像头识别违规驾驶行为是基于神经网络方法实现的,下列说法不正确的是（ ）

A. 识别违规驾驶行为是人工智能技术的应用

B. 训练神经网络模型时需要提供驾驶行为数据

C. 进行违规驾驶行为识别时仍离不开原始训练数据

D. 识别违规驾驶行为的结果并不总是正确的

【答案】C

【解析】本题考查人工智能技术中的神经网络。神经网络模型的使用过程一般包括以下几个阶段：

**1.数据准备：**需收集与任务相关的原始数据（如本题中 “驾驶行为数据”）；

2.**训练阶段：**利用大量标注好的驾驶行为数据对模型进行训练，从中提取关键特征并学习规律，最终优化得到模型权重参数；

**3.使用阶段（识别/预测）：**模型训练完成后，输入新的驾驶行为图像或视频，模型可直接根据学习的特征规律输出识别结果，此时不再依赖原始训练数据；

**4.结果特性：**模型的识别结果可能受数据质量和实际场景复杂度影响，存在一定误差。

因此，C项“进行违规驾驶行为识别时仍离不开原始训练数据”的说法错误。

8. 某二叉树如第8题图所示,E节点在前序遍历序列中的位置记号为x。下列二叉树中,E节点在中序遍历序列中的位置序号也为x的是（ ）



【答案】A

【解析】本题考查二叉树遍历的相关知识。前序遍历的访问顺序为“根节点→左子树→右子树”。第8题图所示，E 节点在前序遍历序列中的位置序号为4。中序遍历的访问顺序为“左子树→根节点→右子树”。A 选项E 节点在中序遍历序列中的位置序号为4,B选项E 节点在中序遍历序列中的位置序号为3,C选项E 节点在中序遍历序列中的位置序号为5,D选项E节点在中序遍历序列中的位置序号为3,所以选A。

【解后反思】本题除了正向比较前序与中序中E节点前的“根节点数”外，也可以采用反向思考法比较E节点之后还剩多少个节点。如第8题图所示，前序遍历后E后元素剩下1个，中序遍历后A项E后剩余元素1个，B项2个，C项0个，D项2个答案为A，这种方法能有效缩短解题时间。

9. 某队列中队首到队尾的元素依次为A,B,C,D,E。元素出队后直接输出或重新入队,若输出次序为B,D,C,E,A,则元素A重新入队的最少次数为（ ）

A. 1 B. 2 C. 3 D. 4

【答案】B

【解析】本题考查队列相关知识。出入队过程如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 步骤 | 当前队列 | 出队元素 | 是否输出 | 操作说明 | A 入队次数 |
| 初始 | A, B, C, D, E | — | — | 初始状态 | 0 |
| ① | A, B, C, D, E | A | 否 | A 重新入队 | 1 |
| ② | B, C, D, E, A | B | 是 | 输出 B | 1 |
| ③ | C, D, E, A | C | 否 | C 重新入队 | 1 |
| ④ | D, E, A, C | D | 是 | 输出 D | 1 |
| ⑤ | E, A, C | E | 否 | E 重新入队 | 1 |
| ⑥ | A, C, E | A | 否 | A 再次入队 | 2 |
| ⑦ | C, E, A | C | 是 | 输出 C | 2 |
| ⑧ | E, A | E | 是 | 输出 E | 2 |
| ⑨ | A | A | 是 | 输出 A | 2 |

最终确定A重新入队2次，选项B正确。

10. 有如下Python程序段：

i,r=n,""

while i<len(s):

r+=s[i]

i+=1

if i%5 ==0:

i+=n

若s为"abcdefghi",n为2,运行该程序段后,r的值为（ ）

A. "abefi" B. "abfg" C. "cdehi" D. "cdhi"

【答案】C

【解析】本题考查字符串处理知识。循环变量初始值为 2，因此变量 r 的首个字符为“c”；接着 i=3，获取字符“d”；当 i=4 时，获取字符“e”，此时已可得出答案 C。随后当 i=5 时，由于满足 i % 5 == 0，执行语句 i += 2，使 i 直接变为 7，因此字符“f”、“g”被跳过，之后继续获取的字符为“h”、“i”。

【解后反思】当 i % 5 == 0 成立时，执行语句 i += 2，此时 i 将跳过5和6，直接变为 7。因此结果中必定包含字符 s[4]，而 s[5] 一定不会出现。四个选项中，只有 C 满足这一条件。

【易错警示】选项A 和选项B 存在一定迷惑性，会误把i的初值考虑为0。

11.有如下Python程序段：

tag=[0]\*len(data)

p=i=0

while i<len(data):

if tag[p]==0 and data[p][1]!=-1:

tag[i]+=1

p=data[p][1]

else:

tag[i]+=tag[p]

i+=1

p=i

若data为[[11,3],[23,-1],[15,0],[26,1],[63,2]],运行该程序段后,tag[4]的值为（ ）

A. 1 B. 2 C. 3 D. 4

【答案】D

【解析】本题考查数组、列表。该程序在链表遍历时每次计算tag[i]的值，若当前tag[p]为0时，tag[i]值加1之后，链表继续向后遍历，直至遍历至tag[p]不为0,即p 节点已经计算出到链表末尾的步数，并将该步数回填至i 节点的步数上，以此计算出i 节点至链表末尾的总步数，避免重复遍历。逻辑可分两种情况：

|  |  |  |  |
| --- | --- | --- | --- |
| **代码分支** | **条件** | **含义** | **操作** |
| if tag[p]==0 and data[p][1]!=-1 | 当前结点 p 还没算过，且不是尾结点 | 继续沿链表向后走 | 令 tag[i]+= 1（给起点 i 先记 1 步），p 跳到下一结点 |
| else | 已到尾结点或遇到已算好的结点 | 进入“回填”阶段 | 把已知的 tag[p] 累加到 tag[i]，然后i++、p=i开始处理下一个起点 |

简言之：

“前半段”一路往后走并给当前起点i计步。一旦撞到“尾”或“已算好”结点，就把那里的步数一次性补给起点i，这样i的总步数就完整了。接着进入下一起点，直至整个数组遍历完毕。

【解后反思】本题本质上考查了链表结构的遍历理解与路径步数的动态计算，利用了“走一遍、记一步，遇已知结果时直接累加”的思想，是链式数据结构和缓存计算的一种综合应用。

【易错提醒】容易只看程序表面流程，逐步模拟，浪费答题时间；容易混淆 i 与 p 的关系，导致逻辑紊乱忽视对tag[p]!= 0的意义理解，这是本题优化的关键，意味着路径步数已知，可直接回填。

12. 定义如下函数,返回key在列表 data中的下标,若key不存在,则返回-1。

def bsearch(data,key):

i,j=0,len(data)-1

while i<=j:

m=(i+j)//2

for k in range(m-1,m+2) :

if i<= k<=j and data[k]==key :

return k

if data[m]<key:

i=m+2

else:

j=m-2

return -1

如果调用函数返回结果不正确,则data可能是（ ）

A. [16,19,18,22,20,29,25] B. [16,19,20,18,22,25,29]

C. [18,16,19,20,22,29,25] D. [18,16,20,19,25,22,29]

【答案】B

【解析】本题考查的是对变形二分查找算法适用条件的理解。题中函数本质上是一种具有容错能力的二分查找算法，其设计允许在数据中最多存在一个位置的小范围扰动，仍能成功定位目标元素。

**算法特点说明：**

每轮查找不仅检查 data[m]，还额外检查 data[m-1] 与 data[m+1]，即在当前中点附近的 3 个元素中查找目标 key，以提高在小范围数据无序情况下的鲁棒性。指针更新方式为：

若 data[m] < key，则执行 i = m + 2；否则执行 j = m - 2。这样可以确保不重复遍历已经检查过的 [m-1, m, m+1] 区间。

**算法前提条件：**为了保证查找的正确性，数据 data 应基本满足整体单调递增，且允许局部元素错位。以本题 m=3 为例，当程序将 data[3] 作为中点时，它会同时比较 data[2]、data[3]、data[4]。若 key < data[3]，下一步查找区域为 [i, m-2]，对应的是 [0,1]；因此要求 data[0]、data[1] 必须小于 data[3]。若 key > data[3]，下一步查找区域为 [m+2, j]，即 [5,6]；因此要求 data[5]、data[6] 必须大于 data[3]，否则程序跳过后续关键元素时可能会漏查，如选项B 若key=19 时，下一步的查找区间缩小到[25,19]出现查找失败，因此答案为B。

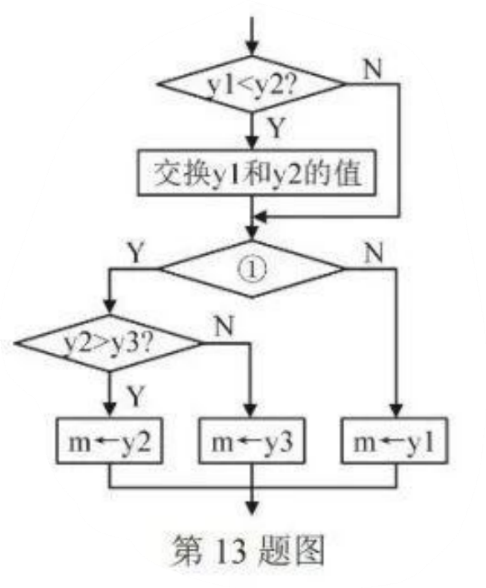
【解后反思】解题关键在于意识到：即使函数设计具备容错能力，但数据仍需“整体近似有序”。尤其当中点为 data[3] 时，前两个元素必须小于它，后两个元素必须大于它，才能保证跳过中点后不会漏查目标值。本题提醒我们不能只看代码逻辑，更要深挖其适用条件与失败边界。今后遇到类似问题，应学会结合“数据”整体分析，而不仅停留在程序语句的逐行分析层面。

二、非选择题(本大题共3小题,其中第13小题8分,第14小题9分,第15小题9分,共26分)

13. 某小组搭建水质监测系统,采集某水域溶解氧和pH的数据,进行水质监测。对于每个传感器,智能终端每小时获取3次数据,将3个数据的中位数(排序后处于中间位置的数)通过5G模块上传至服务器。服务器检测到异常情况时,向管理员发送警示信息,并通过智能终端控制指示灯闪烁。用户通过浏览器可查看系统数据。请回答下列问题：

(1)pH数据从采集到存入数据库的数据流向为（ ）(单选)。

A. 传感器→服务器→智能终端→数据库

B. 传感器→智能终端→服务器→数据库

(2)该系统的数据处理（ ）(单选)。

A. 全部在服务器端完成

B. 全部在智能终端完成

C. 部分在智能终端完成,部分在服务器端完成

(3)若连接在智能终端上的5G模块突发故障不能工作,会引发的问题有（ ）(多选)。

A. 无法通过浏览器访问溶解氧历史数据

B. 智能终端无法传输pH数据至服务器

C. 服务器向智能终端传送控制信号失败

D. 服务器向管理员发送警示信息失败

(4)智能终端每小时获取的3个pH数据分别存入y1、y2和y3,将中位数存入m的部分流程图如第13题图所示。图中①处应填入 。

(5)现需增加水温监测的功能,在智能终端接入温度传感器后,还需对软件部分作多处修改。请用文字描述其中1处修改建议。

【答案】

(1) B (1分)

(2)C (1分)

(3)BC (2分)

(4)yl>y3？或yl>=y3？或其他等价答案 (2分)

(5) 修改智能终端程序，增加向服务器上传温度传感器数据的代码或修改服务器端程序，增加接收温度传感器数据并写入数据库的代码 或其他等价答案 ( 2 分 )

【解析】（1）传感器负责原始数据采集；智能终端对数据进行初步处理(如计算中位数);处理结果通过5G模块上传至服务器；服务器负责接收处理后的数据并存储到数据库，因此选项B“传感器→智能终端→服务器→数据库”正确。

（2）智能终端完成采样、取中位数等初步处理任务；服务器完成数据存储、异常检测、报警推送等后端任务；因此，部分在智能终端完成,部分在服务器端完成，选项C正确。

（3）5G模块是智能终端与服务器通信的唯一通道，5G模块出问题，智能终端与服务器将无法通讯，因此“B.智能终端无法传输pH数据至服务器”“C.服务器向智能终端传送控制信号失败”正确。浏览器访问历史数据是浏览器与服务器之间的通讯，服务器向管理员发送警示，是服务器直连外网操作均与服务器与智能终端通讯无关，因此排除A,D。

（4）流程图要找出y1,y2,y3 的中位数，流程开始的分支结构保证了yl>=y2, 此时若y3>=y1, 则y1 是中位数，而①处条件不成立时m=y1, 故①处应填入 not(y3>=y1)？, 即 y3<y1？。

【易错警示】判断条件填写时“？”容易遗漏，如“y3<y1？”填写为“y3<y1”。

（5）该小题得2分的情况：

①修改智能终端程序，增加从温度传感器获取数据的代码。

②修改服务器端代码，增加从温度传感器获取温度数据并写入数据库的代码。

③在智能终端中，增加代码，向服务器上传从温度传感器获取的数据。

④在数据库中，增加记录温度数据的字段（数据表），并写入温度数据。

得1分的情况：

①在智能终端（服务器）中，增加处理温度数据的代码。

②在智能终端（服务器）中，增加设置阈值。

③在智能终端中，增加设置温度传感器引脚的代码。

④在服务器中，增加记录温度数据的代码。

⑤在数据库中，增加温度记录的字段（数据表）

以上表述没有强调数据流向，扣1分。

【易错警示】本题需特别关注以下两点：“在哪里”（即操作或功能的发生位置）与“干什么”（即实现的功能或作用），需要强调数据的流向。常见失分原因包括：

1.未明确说明操作发生的具体位置或模块；

2.仅描述功能，未涉及具体实现方式或技术手段；

3.未提及“代码”或未在语义中体现代码相关内容；

4.存在错别字或书写不规范；

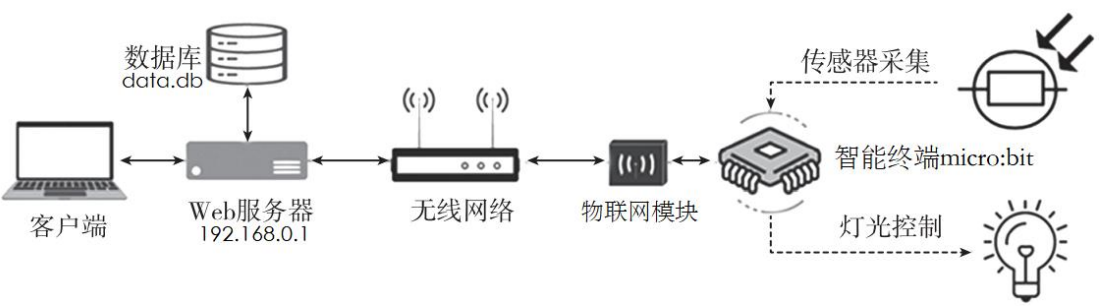
5.术语表达不准确或混淆专业概念。

请注意审题要求，作答时应准确描述“操作位置+实现方式+功能”，并使用规范术语。

【解后反思】为提升此类试题的得分率，建议在日常教学与训练中加强以下几方面：

1.构建信息系统的整体框架意识

需熟悉信息系统的基本组成部分，理解各模块的功能定位，特别是要理清数据的流向与处理路径，形成“数据从哪里来、到哪里去”的清晰逻辑。



2.审题能力是关键

明确题干中是询问解决方法、故障原因还是故障现象，是聚焦于软件层面还是硬件问题，避免答非所问，导致“白写”。

3.提供答题支架，建立表达模式。

使用规范的答题句式，可参考以下模式：

①在哪里+干什么（功能）

②哪个对象+出现什么故障+产生什么结果

③哪个对象+向哪个对象+数据传输或反馈异常

4.规范表达，强化专业术语使用

鼓励学生使用准确、专业的术语，避免错别字或通俗化表达。例如：

①“阈值”不要误写为“阀值”或“域值”；

②数据库相关术语应使用“字段”“数据表”等，而非“列”“部分”等日常用语。

此外，还应注重语言逻辑性与句式完整性，提升表述的准确性和规范性。

14. 水质监测系统已采集了某水域一年的pH数据,该水域pH值的正常范围为6-8。现要对这些数据进行分析,请回答下列问题：

(1)将监测点1的数据导出,存于pHdata.xlsx文件中,如第14题图a所示。现要找出pH均值最高的月份,并统计该月pH大于8的日分布情况,绘制如第14题图b所示的柱形图。实现上述功能的部分Python程序如下,请选择合适的代码填入划线处单选)。

|  |  |
| --- | --- |
|  |  |
| 第14题图a | 第14题图b |

import pandas as pd

import matplotlib.pyplot as plt

df=pd.read\_excel("pHdata.xlsx")

dfl=df.groupby("月",as\_index=False).pH.mean() #分组求平均

df2= ①

#将df2首行的月份存入m, 代码略

print("pH值最高月份序列为:",m)

df\_m= ②

df\_ex=df\_m[df\_m["pH"]>8] #筛选

df\_cnt= ③

#重命名pH列名称为"次数",代码略

plt.bar(df\_cnt["日"], df\_cnt["次数"])

#设置绘图参数,显示如第14题图b所示的柱形图,代码略

程序中①②③处可选的代码有：

A. df[df["月"]==m]

B. df2[df2["月"]==m]

C. df.sort\_values("pH",ascending=False) #降序排序

D. df\_ex.groupby("日",as\_index=False).pH.count() #分组计数

E. df1.sort\_values("pH",ascending=False)

F. df\_ex.groupby ("时",as\_index=False). pH.count()

(2)将7-12月的pH数据存储于列表data中,要求出一个最长连续序列,其中每个pH值均在正常范围内。如果这样的序列有多个,则选择数值总和最小的序列(若仍有多个,选择最早出现的),输出其长度和起始下标。实现上述功能的部分Python程序如下,请在划线处填入合适的代码。

#读入pH数据, 按采集的时间顺序存储于列表data中, 代码略

maxn=start=maxt=0

①

while i<len(data):

if 6<=data[i]<=8:

total=0

k=i

while i<len(data) and 6<= data[i]<=8:

total+= data[i]

i+=1

②

if n>maxn:

maxn=n

start=k

maxt=total

elif ③

star=k

maxt=total

i+=1

#输出最长连续序列的长度maxn和起始下标start,代码略

【答案】

(1) ①E（1分）

②A（1分）

③D（1分）

(2)①i=0（2分）

②n=i-k（2分）

③n==maxn and total<maxt（2分）

【解析】本题考查pandas模块的数据分析与可视化；最长连续序列算法。

(1)①处前一行代码是按“月”分组求“pH”的平均值，存入df1; 而其下一行注释是df2的首月存入变量m; 所以此处填空是要将df1以“pH”列做降序排序，所以答案为E.df1.sort\_values("pH",ascending=False)。

②处需要筛选出月份为m ( 即pH 均值最高的月份)的所有数据，因此使用布尔索引：df[df["月" ] =m],选A,这里特别注意，不能选B,因为df2是已经按月分组后的数据，无法获取每日的详细数据。

③处之前已经完成了最高月份 pH 大于8的数据筛选，此处需要按照“日”列做分组计数统计，所以答案为D.df\_ex=groupby("日",as\_index=False).pH.count()。

【解后反思】本题的三个填空可按代码结构相同、功能相似的原则，将备选代码整理为三组对应模块，以理清代码功能定位。

筛选类代码：

A. df[df["月"]==m]

B. df2[df2["月"]==m]

排序类代码：

C. df.sort\_values("pH",ascending=False)

E. df1.sort\_values("pH",ascending=False)

统计类代码（按组计数）：

D. df\_ex.groupby("日",as\_index=False).pH.count()

F. df\_ex.groupby("时",as\_index=False).pH.count()

接下来依次分析每空应填内容：

空①：排序类代码。空①前执行了语句 df1=df.groupby("月",as\_index=False).pH.mean()，此处已生成df1对象。根据“物尽其用原则”，应尽量使用已生成的数据对象。整段代码中，df1只在该处被使用，其他地方未再引用。因此，应填 E.df1.sort\_values("pH", ascending=False)。

空②：筛选类代码。题干要求“找出 pH 均值最高的月份，并统计该月 pH 大于 8 的日分布情况”，所以接下来要对该月份的数据进行筛选。结合“对应原则”，此处应在原始数据 df 中筛选出对应的月份。因此，应填 A.df[df["月"]==m]。

空③：统计类代码。在剩余的 D 和 F 两项中进行选择。题干提及“统计该月pH> 8的日分布情况”，明确是按“日”分组统计，应填 D.df\_ex.groupby("日", as\_index=False).pH.count()。

通过将备选代码分类归组，再结合题干需求与已有代码逻辑，逐一对应分析，可以有效理清解题思路，提升解题效率。

（2）① 位置：i=0 ——初始化原则：i变量使用前必须初始化；另外根据 i 从头开始扫描索引值为0，填i=0。

②子序列长度计算，n=i- k：结合上下文我们发现，变量 k 存储的是当前满足条件子序列的起始下标。而当某段子序列不再满足条件时，变量 i 已经移动到了该子序列的结束位置的下一个。因此：子序列长度=结束下标（不含）-起始下标，长度n=i-k。

③判断条件：n==maxn and total<maxt。题目要求：“如果满足条件的序列有多个，选择数值总和最小的那个”，因此我们需要第一重判断：当前子序列的长度n是否等于最大长度maxn；

第二重判断：如果一样长，再比较总和total是否比目前记录的最小总和 maxt更小。这正好体现了“对应原则”：程序逻辑要精确对应题目条件，n== maxn and total<maxt。

【易错警示】

1.子序列长度计算：切忌机械套用公式

在本程序中，我们通过语句：

total+=data[i]

i+=1

可以看出，变量i每次循环后已经指向当前子序列的最后一个元素的下一个位置。因此，当我们用 k 表示子序列的起始下标时，长度应为n=i-k。有同学习惯性地使用i- k + 1，这是将“首尾索引都包含”的情况机械套用过来的，忽略了i实际上已经越过了子序列末尾。

2.比较语句书写错误：赋值=与比较==混淆

在程序中，我们需要判断当前子序列是否与最大长度相等，因此应写为n==max，但有同学会误写成n=maxn。

15. 某校实验室有多台相同的实验仪器,每台仪器同一时间仅限一位学生使用。现要开发一个仪器预约系统,开发前先对实验室的学生进出记录进行统计分析。请回答下列问题：

(1)实验室某天09:30前学生进出情况如第15题图a所示,09:00时实验室内的学生数为 。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 学号 | 1010 | 1021 | 3009 | 1010 | 2081 | 3009 | 1021 |
| 时间 | 08:00 | 08:05 | 08:20 | 08:45 | 08:50 | 08:50 | 09:15 |
| 行为 | 进 | 进 | 进 | 出 | 进 | 出 | 出 |

第15题图a

(2)定义如下函数,用于统计在实验室连续停留时间少于5分钟的学生人次。参数b列表中每个元素包含3个数据项,依次为学号、时间(用分钟表示,例如08:05表示为485)和行为(1表示进,-1表示出)。列表b用于存放某天学生进、出实验室的记录,每条进(出)的记录都有一条与之对应的出(进)记录。

def fcount(b):

i=cnt=0

while i<len(b)-1:

if b[i+1][1]-b[i][1]<5:

cnt+=1

i+=2

return cnt

①要实现函数功能,方框处需实现对列表b的操作是（ ）(单选)。

A. 按时间排序：时间相同时,按行为由大到小排序

B. 按学号排序：学号相同时,按时间由小到大排序

C. 按行为排序：行为相同时,按学号由小到大排序

②若函数fcount(b)的功能修改为统计“学生离开实验室后,8分钟内返回”的情况出现的次数,可将函数中划线处代码修改为 and b[i][1]-b[i-1][1]<8。

(3)仪器预约系统中,学生可在线预约仪器使用时段,系统按预约提交顺序逐个处理,如果预约的使用时段有仪器可用,预约成功,否则失败。实现预约处理功能的函数如下,其中用到的部分列表函数与方法如第15题图b所示,请在划线处填入合适的代码。

|  |  |
| --- | --- |
| 函数与方法 | 功能 |
| lst.insert(i,x) | 在列表lst中下标为i的位置插入元素x。若i大于等于len(lst),则在lst末尾添加元素x。 |

第15题图b

'''

函数参数data列表存放已按提交顺序排列的预约信息,每个元素包含4个数据项,依次为学号、起始时间、终止时间、是否成功。其中起始、终止时间都用8位数字字符串表示,如“09280830”表示9月28日8点30分；“是否成功”数据项用于存放每个预约的处理结果。

参数m存放实验室仪器台数。

'''

def proc(data,m):

a=[]

for i in range(len(data)):

①

p1=0

while p1<len(a) and a[p1][0]<=data[i][1]:

p1+=1

if p1>0:

p2=p1-1

else:

p2=0

while p2<len(a) and a[p2][0]<data[i][2]:

if ② :

flag=False

break

p2+=1

if flag:

for j in range(p1,p2):

a[j][1]+=1

a.insert(p2,[data[i][2],0])

a.insert(p1,[data[i][1],1])

if p1>0:

③

a[p2+1][1]=a[p2][1]-1

data[i][3]=flag

【答案】

(1) 2(1分)

(2)①B(1分)

②i>0 and b[i][0]==b[i-1][0](2分)

(3)①flag=True(1分)

②a[p2][1]==m(2分)

③a[p[1]=a[p1-1]+1(2分)

【解析】本题考查了“时间段冲突判断”“区间问题处理”“资源分配模拟”等场景，综合应用了排序相关知识的理解与应用。

（1）09:00之前共有4次进实验室记录、2次出实验室记录，因此此时实验室内应有4-2=2 名学生在场。

（2）①函数fcount(b)统计在实验室连续停留时间少于5分钟的学生人次。参数b是一个列表，列表中的每个元素都是 [学号，时间（分钟），行为]：

·行为 为 1 表示“进实验室”；

·行为 为 -1 表示“出实验室”。

代码逻辑：

程序采用每次读取相邻两个元素（b[i]和 b[i+1]）的方式，来判断同一学生停留时间是否少于5分钟：b[i+1][1]-b[i][1]<5；若满足条件，则将cnt累加。因此，加框处需要对列表b进行排序，以确保：

·同一学生的进出记录紧邻排列（便于配对）；

·同一学生的记录按时间升序排列（确保进在前、出在后）；

故排序规则应为先按学号升序，学号相同时按时间升序。答案选B。

② 将函数功能改为统计“学生离开实验室后，8分钟内返回”的情况次数。要找出所有“出→进”对，且时间差小于8分钟；因为数据仍是排好序的，b[i]是一次“进”，b[i-1]是上一次“出”。需要满足的三个条件：

·i>0：防止访问越界；

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... |
| 学号A | 学号A | 学号B | 学号B | 学号C | 学号C | 学号C | 学号C |  |
| 进 | 出 | 进 | 出 | 进 | 出 | 进 | 出 |  |

·b[i][0]== b[i-1][0]：是否为同一个学生，因为某一学生进出都是成对存在，但i与i-1相邻进出组存在不是同一个学生的可能性，比如

有可能不是同一个学生如2和3，也有可能是同一个学生如6和7

·b[i][1]-b[i-1][1]<8：返回间隔在8分钟内。

因此，判断条件应写作if i>0 and b[i][0]==b[i-1][0] and b[i][1]-b[i-1][1]<8:

【易错警示】此i>0边界保遗漏的同学较多，平时练习的时候应加强代码的严谨与完备性。

(3)函数pro基于实验仪器调度问题。结合代码段，分析核心变量及其功能：

①函数功能说明：本函数模拟实验室仪器的预约过程。

·每个学生提交预约请求（含起止时间），系统按提交顺序依次处理；

·若该时间段内实验室仪器数量（共 m 台）尚未被占满，则预约成功；

·否则，预约失败；

函数更新每条预约记录的处理结果（是否成功字段）。

②参数说明

|  |  |
| --- | --- |
| 参数 | 说明 |
| data | 已按预约提交顺序排列的列表，每个元素是 [学号, 起始时间, 终止时间, 是否成功] |
| m | 实验室仪器的总台数 |

③变量说明

|  |  |
| --- | --- |
| 变量 | 功能 |
| a | 存储时间点列表，每项为 [时间, 当前预约人数]，用于动态维护仪器使用情况 |
| flag | 标记预约是否成功 |
| p1 | 找到当前预约起始时间应插入 a 中的位置 |
| p2 | 用于扫描起始到结束时间段内的预约状态是否已满，结束时间点应插入 a 中的位置 |

参数m存放实验室仪器台数。

'''

def proc(data,m):

a=[] # 时间点列表，记录预约人数变化

for i in range(len(data)):

flag=True # 初始标记为“预约成功”

# ---------- 步骤1：查找起始时间的插入位置 p1 ----------

p1=0

while p1<len(a) and a[p1][0]<=data[i][1]:

p1+=1

# ---------- 步骤2：设置 p2 为最后一个不小于起始时间的位置 ----------

if p1>0:

p2=p1-1

else:

p2=0

# ---------- 步骤3：检查预约区间是否冲突 ----------

while p2<len(a) and a[p2][0]<data[i][2]:

if a[p2][1]==m: #判断该时间段预约人数是否达到m人，若达到，则无法预约。

flag=False #标记预约失败

break

p2+=1

# ---------- 步骤4：若预约成功，更新预约时间段 ----------

if flag: #若预约成功

# 对起始到终止时间段内所有时间点，预约人数+1

for j in range(p1,p2):

a[j][1]+=1

#为避免先插入起始节点导致后续结束节点位置偏移，故上面代码先插入结束节点，再插入起始节点。因结束时间靠后，其插入不会影响起始时间的插入位置。

a.insert(p2,[data[i][2],0])

a.insert(p1,[data[i][1],1])

# 设置插入时间点的预约人数

if p1>0:

a[p1][1]=a[p1-1][1]+1 #起始时间节点为前一个时间节点的预约人数加1

a[p2+1][1]=a[p2][1]-1 #结束时间节点为前一个时间节点预约人数减1

# ---------- 步骤5：记录预约是否成功 ----------

data[i][3]=flag

【解后反思】列表 a 实际上构成了一条时间轴，每个关键时间点记录当前时刻实验室中正在使用仪器的人数。系统通过遍历这条时间轴来判断预约是否冲突：

·判断策略：当一个新的预约请求提交时，只需检查该预约时间段内是否有某个时间点的预约人数已达到仪器总数 m，若有，则该预约失败；

·更新策略：如果预约成功，则需要在时间轴上对应时间段内对人数进行更新处理，模拟该预约对仪器资源的占用。

**解题效率的提升策略**

在程序实现过程中，我们可以通过以下代码设计策略提高效率和准确性：

① 初始化策略（预约标志位）：

·使用变量flag=True表示预约默认成功；

·在判断时间段内某个时刻已达m台仪器时，将flag设为False，中断处理。

②对应策略（插入起始、结束时间点人数更新）

预约结束时间节点的预约人数应为其前一节点人数减1，即a[p2+1][1]=a[p2][1]-1，对应

预约起始时间节点的预约人数应为该时间点前一时刻的预约人数加1：a[p1][1]=a[p1-1][1]+1。