

题号	1	2	3	4	5	6	7	8	9	10	11	12
答案	C	C	D	A	C	B	B	A	B	D	C	D

高三技术练习

考生须知：

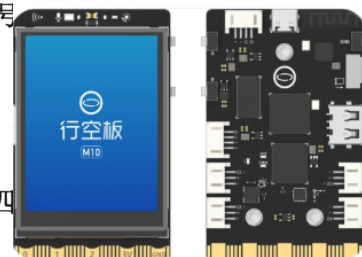
1. 本卷满分 100 分，练习时间 90 分钟；
2. 答题前，在答题卷指定区域填写学校、班级、姓名、试场号、座位号；
3. 所有答案必须写在答题卷上，写在试卷上无效；
4. 练习结束后，只需上交答题卷。

第一部分 信息技术

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 7 题：

AI 跳绳系统以 AI 视觉识别技术为核心，搭载行空板（集成处理器、图像采集模块、网络通信模块）完成跳绳计数模型训练。系统通过摄像头实时采集多人跳绳动作视频数据，经 AI 神经网络算法完成动作识别、精准计数后传输到服务器存储，并在显示屏以柱形图实时显示排名。用户可通过校园小程序输入学号查询个人历史记录。



- Micro:bit板、arduino板、掌控板、行空板...都属于类Micro:bit板的智能终端
1. 下列关于该系统中数据的说法，正确的是 C
A. 过早的历史记录数据没有价值
B. 跳绳计数数据仅存储在行空板 可能存储在智能终端或服务器
C. 大屏展示成绩有助于跳绳信息共享
D. 跳绳视频和个人计数数据的编码方式相同 视频、音频、图片、文本的编码方式各不相同
 2. 关于该系统的信息安全与社会责任，下列行为符合规范的是 C
A. 导出所有学生的成绩数据并分享至班级群
B. 借用他人学号查询其历史成绩以对比训练效果
C. 对传输的成绩数据进行加密
D. 将学生跳绳视频发布至社交平台
 3. 该系统实现“AI 实时计数与现场排名可视化”的过程中，没有用到的技术是 D
如图像、视频识别 联结主义
A. AI 视觉识别 B. 神经网络算法 C. 数据可视化 D. 强化学习 属于行为主义
 4. 该系统需对 20 名学生进行编码，若将该编码存储为字节，则单个学生编码所需的最小存储字节数是 A 20种信息， $2^5=32$ 至少需要5位 → 1字节
A. 1 B. 5 C. 8 D. 20
 5. 下列关于该系统中硬件与网络的说法，正确的是 C 行空板通过摄像头采集视频，直接识别、计数后传输到服务器
A. 行空板中的处理器只负责数据存储，不参与 AI 计数运算
B. 摄像头分辨率越高，AI 动作识别一定越准确 与采集的分辨率、算法模型等因素有关
C. 服务器与小程序之间传输数据需遵循网络协议
D. 大屏与服务器必须通过有线连接展示成绩 可以通过有线、无线连接
 6. 将跳绳视频存储为未经压缩的 AVI 格式，下列说法不正确的是 B
A. 摄像头的采样帧频会影响视频的流畅度 ✓ 帧频越高视频越流畅
B. 跳绳动作的幅度大小会直接影响视频文件的存储容量 × 未经压缩的avi格式，
C. 为节省服务器存储空间，可将 AVI 格式转换为 MP4 格式 ✓ 视频内容与存储容量无关
D. 将分辨率从 1920×1080 调整为 960×540，其他参数不变，则存储容量会变为原来的 1/4. ✓

若视频采用压缩格式，因为视频前后帧有相关性，图像内容会影响存储空间



模型训练完成后，会得到模型参数，实际识别时，只要模型参数就可以，不需要原始训练数据，只需要加载训练好的模型参数，不需要再读取原始训练数据。

神经网络模型是数据驱动的，要准确识别跳绳动作，必须用大量、多样的跳绳样本数据（不同身高、姿势、环境）来训练，才能让模型学习到特征规律。

7.关于该系统的 AI 神经网络算法, 下列说法不正确的是 **B**

- A. 训练模型时需要大量跳绳动作样本数据 ✓
 B. 仅增加同身高样本即可有效优化模型 模型优化需要多样化数据，需要不同身高、不同场景的跳绳动作
 C. 受光线、遮挡等影响，识别结果可能存在误差
 D. 实际识别过程中无需调用原始训练数据即可完成计数

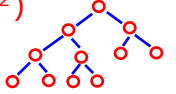
8.一棵高度为 k ($k > 2$) 的完全二叉树, 其左子树的节点个数比右子树节点个数多出 n 个, 则 n 的值不可能为 **A** $k-1$ 层已满, 差值在第 k 层产生, 差值范围 $0 < n \leq 2^{k-1}/2$ (2^{k-2})

- A. $2^{k-2}+1$ B. 2^{k-2} C. $2^{k-2}-1$ D. 0

9. 栈容量为④ 若出栈顺序依次为 ABCDE, 则入栈顺序不可能为 B

- A.ABCDE B.EDCBA C.DCBAE D.BADCE

10.对于任意正整数 n ，甲、乙程序段输出结果相同，则乙程序段划线处的正确代码为 **D**



差值最大

<pre> k = 1 for i in range(n % 2, n, 2): k = k * (i + 2) print(k) </pre> n为偶数, 求 $(2 * 4 * 6 * \dots * n)$ n为奇数, 求 $(3 * 5 * 7 * \dots * n)$ 甲程序段	<pre> def f(n): if n <= 2: return n else: return n * f(n - 2) print(f(n)) </pre> 返回值必须是1、2 $n * f(n - 2)$ 乙程序段
--	---

- A. ① $n < 2$ ② $n * f(n + 2)$ B. ① $n < 2$ ② $n * f(n - 2)$
C. ① $n \leq 2$ ② $n * f(n + 2)$ D. ① $n \leq 2$ ② $n * f(n - 2)$

11. 一个非递减的列表 d 长度为 10, $d[3] \sim d[7]$ 的数据如图所示, 其中 $d[0] \sim d[9]$ 均为 100 以内的正整数。执行如下程序段, 下列说法正确的是 C

```
key = int(input())
```

$$i, j = 0, \text{len}(d) - 1$$

```
while i <= j:
```

$$m = (i + j) // 2$$

```
if d[m] > key:
```

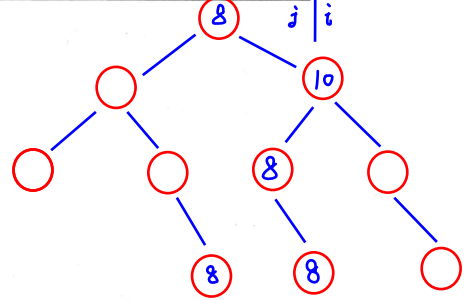
$$j = m - 1$$

else:

$$i = m + 1$$

```
print(j)
```

i	0	1	2	3	4	5	6	7	8	9
d[i]				8	8	8	8	10		/80



- A. 若输入的 key 为 8, 则输出 j 的值为 7. ⁶ 8
 B. 输入 key 为 6 时一定比输入 8 时查找的次数少 key=8: 4次 key=6: 3或4次
 C. 若输入 key 的值为 100, 则输出 j 的值一定为 9 ✓
 D. 无论输入 key 的值为多少, 至少查找 4 次 ✗ 3-4次

12.有如下 Python 程序段:

```
ans = []
```

$$s = []$$

p = 0

idx = -1

#读取数据存入 data, 代码略

ans = []

s = []

p = 0 head开始

idx = -1

#读取数据存入 data, 代码略

while p != -1:

idx += 1 遍历链表节点对应的序号

ans.append(0) #在列表 ans 末尾添加元素 0

while s != [] and s[len(s) - 1][0] < data[p][0]: s: 单调递减栈

ans[s[len(s) - 1][1]] = data[p][0]

s.pop() #将列表 s 末尾元素删除

s.append([data[p][0], idx])

p = data[p][1]

print(ans)

若输出 ans 的值为[7, 0, 4, 5, 0], 则读取 data 的值可能为 D

A. [[5, 4], [9, -1], [5, 3], [5, 1], [7, 2]]

B. [[7, 2], [3, 4], [4, 1], [2, -1], [5, 3]]

C. [[4, 3], [5, 2], [4, -1], [7, 4], [7, 1]]

D. [[2, 4], [3, 2], [4, 3], [5, -1], [7, 1]] 2 7 3 4 5

二、非选择题 (本大题共 3 小题, 其中第 13 小题 10 分, 第 14 小题 7 分, 第 15 小题 9 分, 共 26 分)

13. 某兴趣小组搭建了一个智能空气监测系统, 进行特定区域内空气质量参数及环境变化的监测, 如 PM2.5、温湿度等。该系统的智能终端获取传感器数据, 并通过无线通信方式将数据传输到 Web 服务器, 服务器根据数据判断出异常情况, 通过智能终端控制执行器发出预警信号, 并启动相关调节设备。请回答下列问题:

(1) 该智能空气监测系统中, 可作为调节设备的是 C。

(单选, 填字母: A. Wi-Fi 模块 / B. 湿度计 / C. 排气扇)

(2) 下列功能需要在 Web 服务器端实现的是 C (单选, 填字母: A. 采集温湿度传感器上的数据 / B. 直接控制新风系统的开关 / C. 处理浏览器的访问请求)。

(3) 针对该系统的搭建过程, 下列描述正确的有 AD (多选, 填字母)。

A. 对服务器进行定期检查和维修可以提高系统的稳定性✓

B. 确定该系统网络应用软件的实现架构为 B/S 架构属于需求分析 开发模式选择

C. 管理员通过浏览器查看历史数据时需要知道数据库名 不需要知晓数据库名称 (更安全)

D. 该系统可以同时为多个用户提供数据查询服务✓

E. 智能终端与服务器必须连接同一个 Wi-Fi 可以在不同的wifi环境中, 只要能访问服务器就可

(4) 为提升该智能空气监测系统中 Web 服务器的运行性能, 若从硬件层面进行优化, 可采取哪些具体改进措施? (请列举两项, 每项正确得 1 分)

(5) 该兴趣小组收集了某天校园内不同监测点的 PM2.5 浓度值保存于“data.csv”中, 部分数据如图 a 所示, 现要统计当天监测点为“教室”在每一小时内 PM2.5 的平均浓度情况, 并绘制如图 b 所示的柱形图。

日期	时间	监测点	温度	PM2.5
2025/10/15	07:05:30	教室	28	36
2025/10/15	07:05:31	走廊	24	22
2025/10/15	07:05:33	机房	33	53
2025/10/15	07:05:33	操场	25	25
.....				
2025/10/15	16:50:21	食堂	26	29
2025/10/15	16:50:21	教室	25	31

图 a

实现上述功能的部分 Python 程序如下:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read_csv("data.csv")

df = df[_____]

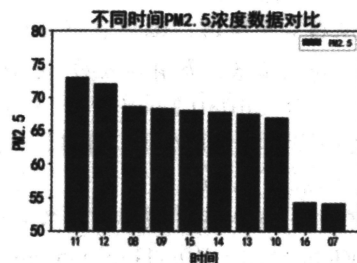


图 b

更换新能更强的处理器/cpu

采用主频更高的处理器/cpu

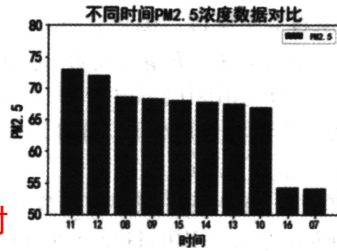
选用核心数更多的处理器/cpu

增加内存 (RAM) 容量

将机械硬盘 (HDD) 替换为固态硬盘 (SSD)

增加cpu数量

df.监测点=="教室" 或 df["监测点"]=="教室"



从图表中看出最终数据为降序

for i in df.index:

df.at[i, "时间"] = df.at[i, "时间"][0:2]

DACB

时间列数据变为小时

#设置绘图参数，代码略

①请在程序划线处填入合适的代码。

②程序方框中应填入的代码顺序为_____（填写字母，顺序完全正确得分，否则不得分）。

A.df = df.sort_values("PM2.5", ascending=False)

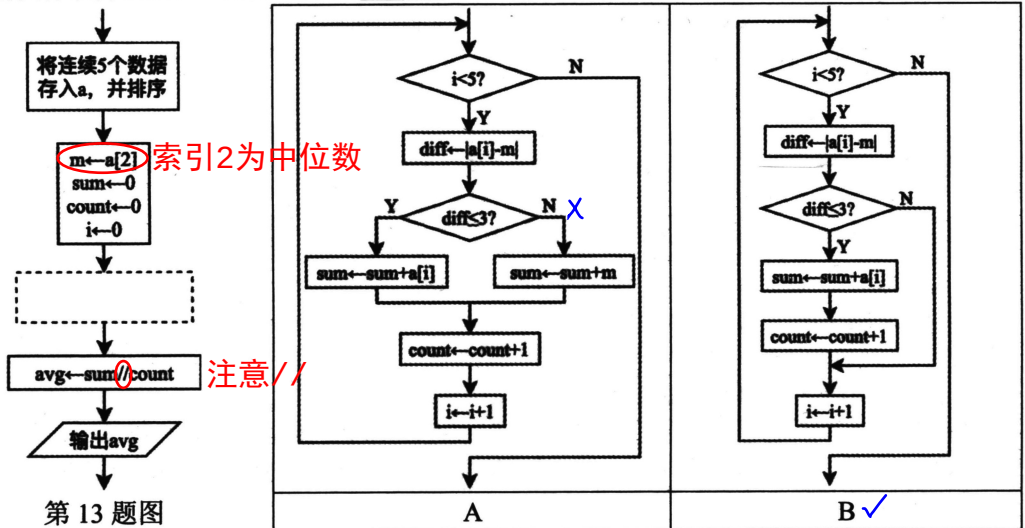
B.plt.bar(df["时间"], df["PM2.5"])

C.df = df.head(10)

D.df = df.groupby("时间", as_index=False).mean()

14.某校搭建“图书馆智能座位管理系统”，采集馆内各阅览区的座位使用数据，进行动态引导。智能终端每隔一定时间通过红外传感器采集数据，连续采集 5 次后，将 5 个数据经过初步处理得到的“有效人数”通过 Wi-Fi 模块上传至服务器。服务器根据各区域座位总数和当前人数计算使用率，并按照一定策略进行人员引导。请回答下列问题：

(1)智能终端计算“有效人数”的算法流程如下图所示，将连续采集的 5 个数据排序后取中位数，筛选出与中位数的差值不超过 3 的数据，然后对这些数据取平均值。则图中虚线框处缺失的流程图可能是 B（单选，填字母）。若某区域连续采集的 5 个数据为：42、43、46、50、55，则执行该算法后，“有效人数”为 44。（43+46）//2=44



(2) 该馆有 n 个区域，编号 0 至 $n-1$ 。每个区域 i 的容量为 $total[i]$ ，当前人数为 $cur[i]$ 。若使用率 $> 80\%$ 为拥挤区，需引导人数 $= \lfloor cur[i] - total[i] \times 0.8 \rfloor$ ($\lfloor \rfloor$ 表示向下取整)；若 $< 20\%$ 为闲置区，可接收人数 $= \lceil total[i] \times 0.2 - cur[i] \rceil$ ($\lceil \rceil$ 表示向上取整)。调度策略：选需引导人数最多的拥挤区（引导人数相同取编号最小），再将其合适数量人员引导至最近的闲置区。实现该功能（处理一遍引导）的部分 Python 程序如下，请在划线处填入合适的代码。

#导入模块，输入各区域座位总数 $total$ 和当前人数 cur ，代码略

$n = \text{len}(total)$; $stat = [[0, -1] \text{ for } i \text{ in range}(n)]$

for i in range(n):

$r = cur[i] / total[i]$

if $r > 0.8$:

$stat[i] = [\text{int}(cur[i] - total[i] * 0.8), (1)]$ 拥挤区，待引导人数

elif (1): $r < 0.2$

$stat[i] = [\text{ceil}(total[i] * 0.2 - cur[i]), (0)]$ # ceil 函数用于向上取整 闲置区，可接收人数

src:source源
dst:destination目的地

```
src = -1;maxm = 0
for i in range(n): 找到引导人数最多的拥挤区
    if ②: stat[i][1] == 1 and stat[i][0] > maxm
        src = i
        maxm = stat[i][0]
#计算距离 src 最近的闲置区 dst, 当没有闲置区时 dst 的值为-1, 代码略
if src != -1 and dst != -1:
    ③ dnum = stat[dst][0]
    num = min(dnum,maxm)
    print("从",src,"区引导",num,"人到",dst,"区")
else:
    print("无需引导")
```

15.某市火车站周围有 m 个人口较为密集的城区。火车站在出站口设置“出租车拼车点”，并通过“出租车拼车系统”为所有前往同一城区的乘客提供“拼车”服务。系统接收乘客的出行请求，每个请求包含乘客编号、请求上车时间、目的地城区，根据乘客的请求上车时间先后依次为乘客安排车辆。假设每辆出租车最多载客 w 人，乘客在请求上车时间一定可以上车，且上车后最多等待 t 分钟，期间一旦满载或超时，则立即发车。编写程序实现，输入出行请求，输出每辆车发车时刻及发车时车内的乘客编号。

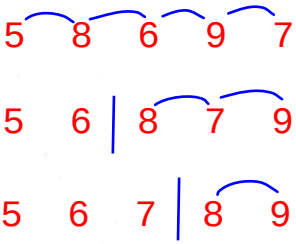
例如：若 m=2、w=4、t=5，8 位乘客出行请求如下表所示，编号 1、2、3、4 乘客的发车时间是 08:15，编号 5、6 乘客的发车时间是 08:21，编号 7、8 乘客的发车时间是 08:23。

乘客编号	1	2	3	4	5	6	7	8
请求上车时间	08:10	08:11	08:11	08:15	08:16	08:17	08:18	08:22
目的地城区	0	0	0	0	1	1	0	0

→等5分钟后出发

- (1) 若将编号为 4 的乘客的请求上车时间更改为 08:16，则编号为 8 的乘客的发车时间是 08:27 / 8:27
- (2) 定义 sort(data)函数，参数 data 中每个元素包含 3 个数据项：乘客编号、请求上车时间、目的地城区。函数功能为按请求上车时间对 data 做升序排序。

```
def sort(data):
    i = 0
    n = len(data)
    flag = True
    while i < n - 1 and flag:
        flag = False
        k = i
        for j in range(n-1,i,-1):
            if data[j][1] < data[j-1][1]: #A 处
                data[j],data[j-1] = data[j-1],data[j]
                flag = True
                k = j
        i = k
```



- ①实现上述功能的代码如上，若 data=[[1,550,0], [2,580,0], [3,560,1], [4,590,1], [5,570,2]]，则 A 处代码执行次数为 7。
- ②若 data=[[1,490,0], [2,491,0], [3,491,0], [4,496,0], [5,496,1], [6,497,1], [7,498,0], [8,502,0]]，删除加框处代码 k=i，对程序运行是否有影响 A。（单选，填字母 A.有影响/B.无影响）
删除 k=i：程序没有任何交换发生，k没有赋值！

(3) 程序运行界面如下图所示。

发车时刻:	495	乘客:	[1, 2, 3]
发车时刻:	501	乘客:	[4, 7]
发车时刻:	501	乘客:	[5, 6]
发车时刻:	507	乘客:	[8]

实现该功能的 Python 程序如下，请在划线处填入合适的代码。

'''

读取 n 个出行请求存入列表 data，每个元素包含乘客编号、请求上车时间、目的地城区 3 个数据项，并将所有的上车时间从“HH:MM”的格式转换为与“00:00”的时间差。例如：“01:10”经转换得到 70，代码略。

读取目的地城区的数量 m，最多等待分钟数 t，每辆出租车最多载客数 w。

'''

sort(data)

ct = 0 **current time 当前时间**

i = 0 **遍历data的索引**

n = len(data)

que = [[1] for i in range(m)] **每个地区对应1个队列**

flag = False

while i < n or flag == True:

while i < n and data[i][1] <= ct: **将当前时间 ct 及之前到达的乘客加入对应城区队列**

k = data[i][2] **目的地**

que[k].append(i)

i += 1

k = 0

flag = False

while k < m:

head = que[k][0] **#①**

tail = len(que[k]) **说明tail指向的对象为虚位**

if head != tail:

idx = que[k][head]

if ct - data[idx][1] >= t or tail - head >= w: **1. 当前时间 - 乘客上车时间 >= 最大等待时间t
2. 队列中等待人数 >= 每辆车最大载客数w**

if head + w < tail:

tail = head + w **如果剩余乘客数超过 w 辆，则本次只载 w 人**

pa = que[k][head:tail] **#② 取出真正乘车的人员**

pa = [data[idx][0] for idx in pa] **#将乘客索引转化为乘客编号**

print('发车时刻: ', ct, '乘客: ', pa)

que[k][0] = tail **tail的位置成为最新的head**

k -= 1

flag = True **处理了非空队列，必须进行下一次外循环**

k += 1

if flag == False and i < n:

ct = data[i][1] **更新当前时间**

else:

ct += 1 **#③**