

青少年人工智能编程水平测试七级试题

一、单项选择题（每题 2 分，共 15 题，共 30 分）

1. 在 Python 中，哪种数据类型不适合用于存储复杂的层次结构数据？

- A. 列表
- B. 字典
- C. 集合
- D. 元组

正确答案：C

解析：集合（set）是无序且不支持嵌套，不适合存储复杂的层次结构数据。

2. 以下哪个选项正确创建了一个包含四个元素的元组？

- A. tuple4 = (1, 2, 3, 4)
- B. tuple4 = [1, 2, 3, 4]
- C. tuple4 = {(1, 2, 3, 4)}
- D. tuple4 = {1: 2, 3: 4}

正确答案：A

解析：选项 A 创建了一个包含四个元素的元组。

3. 以下哪个 Python 内置函数是高阶函数的例子？

- A. map
- B. len
- C. input
- D. print

正确答案：A

解析：map 函数是一个高阶函数，因为它接受另一个函数作为参数来处理序列中的每个元素。

4. 关于 lambda 表达式，以下描述正确的是？

- A. lambda 表达式可以有多个返回值
- B. lambda 表达式可以包含复杂的逻辑
- C. lambda 表达式通常用于定义简短的匿名函数
- D. lambda 表达式需要使用 return 语句

正确答案：C

解析：lambda 表达式通常用于定义简短的匿名函数，且不能包含 return 语句。

5. 以下哪个 Python 表达式正确使用了 map 和 lambda 来计算列表中每个元素的平方？

- A. result = list(map(lambda x: x**2, [1, 2, 3, 4]))
- B. result = map(lambda x: x**2, [1, 2, 3, 4])
- C. result = [x**2 for x in [1, 2, 3, 4]]
- D. result = filter(lambda x: x**2, [1, 2, 3, 4])

正确答案：A

解析：选项 A 正确使用了 map 和 lambda 来计算列表中每个元素的平方。

6. 以下哪种方法不是 Pillow 库中的图像过滤函数？

- A. filter()
- B. blur()
- C. sharpen()
- D. rotate()

正确答案：D

解析：rotate() 是图像旋转函数，而不是用于过滤图像的函数。

7. 使用 Pillow 库翻转图像的函数是？

- A. flip()
- B. transpose()
- C. mirror()
- D. rotate()

正确答案：B

解析：transpose() 方法可用于垂直或水平翻转图像。

8. 在 Pandas 中，哪个方法用于向 DataFrame 添加新的列？

- A. add_column()
- B. insert()
- C. append()
- D. assign()

正确答案：D

解析：assign() 方法用于向 DataFrame 中添加新的列。

9. 在 Pandas 中，以下哪个选项创建了一个包含日期的 Series？

- A. pd.Series(pd.date_range('2024-01-01', periods=5))
- B. pd.Series([1, 2, 3, 4, 5])
- C. pd.Series(['2024-01-01', '2024-01-02'])
- D. pd.Series(pd.to_datetime(['2024-01-01', '2024-01-02']))

正确答案：A

解析：选项 A 使用 pd.date_range 创建了一个包含日期的 Series。

10. 在 Pandas 中，以下哪个选项描述了 DataFrame 的特点？

- A. DataFrame 可以包含多种类型的数据
- B. DataFrame 的行索引必须是唯一的
- C. DataFrame 的列名必须是字符串
- D. DataFrame 的元素可以是多维数组

正确答案：A

解析：DataFrame 是 Pandas 中的二维数据结构，可以包含多种类型的数据。

11. 以下哪个 HTTP 方法用于替换服务器上的现有资源？

- A. GET
- B. POST

- C. PUT
- D. DELETE

正确答案：C

解析：PUT 方法用于在服务器上替换指定的资源。

12. 以下哪种排序算法的最坏时间复杂度为 $O(n^2)$ ？

- A. 归并排序
- B. 堆排序
- C. 冒泡排序
- D. 快速排序

正确答案：C

解析：冒泡排序的最坏时间复杂度为 $O(n^2)$ 。

13. 在算法分析中， $O(1)$ 表示什么意思？

- A. 算法的复杂度与输入数据的大小无关
- B. 算法的复杂度与输入数据的大小成正比
- C. 算法的复杂度与输入数据的对数成正比
- D. 算法的复杂度与输入数据的平方成正比

正确答案：A

解析： $O(1)$ 表示算法的复杂度与输入数据的大小无关，通常用于常数时间复杂度的操作。

14. 以下关于排序算法的说法中，哪一个是正确的？

- A. 快速排序是一种不稳定的排序算法，因为它可能改变相同元素的相对顺序
- B. 归并排序虽然是稳定的排序算法，但在处理大规模数据时效率较低
- C. 选择排序是一种稳定的排序算法，因为它每次都选择最小元素
- D. 冒泡排序是一种不稳定的排序算法，因为相邻元素交换会改变相同元素的相对顺序

正确答案：A

解析：B 归并排序效率较高；C 选择排序并不稳定；D 冒泡排序并不会改变相同元素的顺序。

15. 以下关于时间复杂度的描述中，哪一个是正确的？

- A. 时间复杂度为 $O(n \log n)$ 的算法在所有情况下都比 $O(n)$ 的算法执行得更快
- B. 在时间复杂度为 $O(2^n)$ 的算法中，随着 n 的增大，算法的执行时间呈指数级增长
- C. 如果一个算法的时间复杂度是 $O(n^2)$ ，那么它在任何情况下的执行时间都一定比 $O(n)$ 的算法更长
- D. 时间复杂度为 $O(n!)$ 的算法在处理较大规模的数据时，其执行时间增长得很慢

正确答案：B

解析：A 未考虑数据小的情况；C 混淆了时间复杂度和执行时间的概念；D 完全错误。

二、多项选择题（每题 3 分，共 5 题，共 15 分，多选或少选不得分）

1. 在 Python 中，以下哪些操作可以用于创建或修改列表的内容？

- A. 使用切片语法修改列表中的一部分元素
- B. 使用列表推导式生成一个新的列表
- C. 使用 `extend()` 方法将另一个列表的元素添加到当前列表末尾

D. 使用元组解包的方式将元素添加到列表末尾

正确答案: A, B, C

解析: A 正确: 切片语法可以用于修改列表中的一部分元素。B 正确: 列表推导式是一种生成新列表的方式。C 正确: `extend()` 方法用于将另一个列表的元素添加到当前列表的末尾。

D 错误: 元组解包可用于变量赋值, 但不能直接用于列表添加操作。

2. 以下关于算法的描述中, 哪些是正确的?

A. 二分查找的时间复杂度为 $O(\log n)$, 前提是数据必须是有序的

B. 归并排序的时间复杂度为 $O(n^2)$, 适用于小规模数据的排序

C. 快速排序在最坏情况下的时间复杂度为 $O(n^2)$, 但平均情况下效率更高

D. 递归算法的时间复杂度通常依赖于问题的规模和递归的深度

正确答案: A, C, D

解析: A 正确: 二分查找的时间复杂度为 $O(\log n)$, 且要求数据必须是有序的。B 错误: 归并排序的时间复杂度为 $O(n \log n)$, 它是一个有效的分治算法, 适用于大规模数据的排序。

C 正确: 快速排序的最坏时间复杂度为 $O(n^2)$, 但在大多数情况下, 平均时间复杂度为 $O(n \log n)$, 效率较高。D 正确: 递归算法的时间复杂度通常取决于递归调用的次数和每次调用所需的时间。

3. 观察以下代码段, 选择描述正确的选项:

```
values = [1, 2, 3, 4, 5]
```

```
processed = list(map(lambda x: x ** 2, filter(lambda x: x % 2 != 0, values)))
```

A. 代码段使用了两个高阶函数

B. `processed` 列表将包含所有 `values` 列表中的元素

C. `filter` 函数用于筛选出奇数

D. `processed` 列表将包含元素 `[1, 9, 25]`

正确答案: A, C, D

解析: A 正确: 代码段使用了 `map` 和 `filter` 两个高阶函数。B 错误: `processed` 列表只包含满足条件的元素。C 正确: `filter` 函数用于筛选出奇数。D 正确: `processed` 列表将包含 `[1, 9, 25]`, 因为这些是经过操作后的奇数的平方。

4. 观察以下代码, 选择正确的描述:

```
import pandas as pd
```

```
data = {'Item': ['X', 'Y', 'Z'],
```

```
        'Quantity': [10, 20, 15],
```

```
        'Price': [5, 10, 8]}
```

```
df = pd.DataFrame(data)
```

```
expensive_items = df.loc[df['Price'] > 7, 'Item']
```

```
mid_quantity = df.iloc[1:3]
```

A. `DataFrame` 是通过字典创建的, 其中键作为列名

B. `expensive_items` 是一个 `Series` 对象, 包含价格大于 7 的商品名称

C. `mid_quantity` 是一个 `DataFrame` 对象, 包含第二行和第三行的数据

D. `loc` 和 `iloc` 均可用于选择 `DataFrame` 中的行和列。

正确答案：A, B, C, D

解析：A 正确：DataFrame 可以通过字典创建，字典的键作为列名。B 正确：expensive_items 是通过条件选择得到的 Series 对象。C 正确：mid_quantity 通过 `iloc[1:3]` 获取，它选择了 DataFrame 的第二行和第三行数据。D 正确：loc 和 iloc 都可以用于选择 DataFrame 中的行和列，只是用法不同。

5. 使用 Pillow 库处理图像时，下列哪些操作是可以用库中的特定函数实现的？

- A. 将图像转换为 PNG 格式
- B. 调整图像的对比度
- C. 旋转图像到指定角度
- D. 替换图像中的某种颜色。

正确答案：A, B, C

解析：A 正确：Pillow 支持将图像转换为不同格式，如 PNG。B 正确：Pillow 可以通过增强功能调整图像的对比度。C 正确：可以使用 `rotate()` 方法将图像旋转到指定角度。D 错误：Pillow 不直接支持替换颜色操作，但可以通过像素处理来实现。

三、编程题（共 4 题，共 55 分）

注：试题均不允许在输入函数的括号中写入内容，输出函数仅输出题目要求的信息。所有程序运行时间限制为 3000MS，内存限制为 512MByte。

1. （本题共 5 组测试数据，每个测试点 2 分，共 10 分）

某电影院将在未来的 M 天内上映一部新电影。为了最大化观影人数，电影院希望选择一个连续的 X 天时间段进行重点推广，并让这段时间内的总观影人数最大化。但由于预算限制，每天只能在总观影人数不超过某个最大阈值 C 的情况下进行推广。

输入描述

第 1 行有三个用空格隔开的整数，分别表示 M 、 X 和 C ($1 < X < M < 100$)。

第 2 行有 M 个用空格隔开的整数，依次表示接下来 M 天中每天的观影人数。

输出描述

一个整数，表示符合条件的连续 X 天内的最大观影人数。如果无法找到满足条件的时间段，则输出 0。

样例输入

7 3 80

10 20 30 40 25 35 15

样例输出

75

示例题解程序：

```
def max_audience_with_limit(m, x, c, audiences):
    max_audience_sum = 0

    for i in range(m - x + 1):
        current_sum = sum(audiences[i:i+x])
        if current_sum <= c:
            max_audience_sum = max(max_audience_sum, current_sum)

    return max_audience_sum

# 读取第一行的 m、x 和 c
m, x, c = map(int, input().split())
# 读取第二行的观影人数
audiences = list(map(int, input().split()))

# 计算并输出结果
result = max_audience_with_limit(m, x, c, audiences)
print(result)
```

2. （本题共 5 组测试数据，每个测试点 2 分，共 10 分）

你是一名智能农场的管理员，负责管理一个多功能温室大棚。这个大棚可以根据植物的需要，自动调节温度和湿度。每天早晨，你需要输入当天的温度和湿度要求，并且设定每小时的调整策略。每小时可以调节温度和湿度一次，温度每次调整的幅度是 1 度，湿度每次调整的幅度是 5%。

调整策略如下：

每调高 1 度温度需要耗费 2 秒时间，调低 1 度需要耗费 1 秒时间。

每增加 5%湿度需要耗费 3 秒时间，每降低 5%湿度需要耗费 2 秒时间。

你需要为一天的温室管理任务编写一个程序，计算在给定的一天里，温室自动调整设备所需的总时间。

注意：调整的时间仅计算到达目标值的过程，不包括达到目标后的维持时间。

输入描述：

24 组以空格隔开的整数，表示每小时的温度目标值（范围为 15~30 度）和湿度目标值（范围为 30%~90%），每组分别用两个整数表示，前一个为温度目标值，后一个为湿度目标值。

输出描述：

一个整数，表示一天 24 小时内温室设备调整的总时间（单位为秒）。

样例输入：

20 50 22 60 18 55 21 65 19 50 23 70 24 75 20 65 22 60 25 70 24 80 22 60 19 50 23
65 21 70 20 60 22 75 21 70 19 60 18 55 23 65 24 70 20 60 22 65 21 70

样例输出：

203

示例题解程序：

```
def cal_temperature_time(current_temp, target_temp):
    if current_temp >= target_temp:
        return (current_temp - target_temp) * 1
    else:
        return (target_temp - current_temp) * 2

def cal_humidity_time(current_humidity, target_humidity):
    if current_humidity >= target_humidity:
        return (current_humidity - target_humidity) // 5 * 2
    else:
        return (target_humidity - current_humidity) // 5 * 3

# 读取输入的温度和湿度目标值序列
data = list(map(int, input().split()))

# 初始化当前温度和湿度
current_temp = 20
current_humidity = 50

# 总时间变量初始化
total_time = 0

# 每小时遍历温度和湿度目标值
for i in range(0, len(data), 2):
    target_temp = data[i]
    target_humidity = data[i+1]

    # 计算温度调整所需时间，并累加到总时间
    total_time += cal_temperature_time(current_temp, target_temp)

    # 计算湿度调整所需时间，并累加到总时间
    total_time += cal_humidity_time(current_humidity, target_humidity)

    # 更新当前温度和湿度
    current_temp = target_temp
    current_humidity = target_humidity

# 输出总时间
print(total_time)
```

3. （本题共 5 组测试数据，每个测试点 3 分，共 15 分）

小明有一个喜欢的玩具积木塔，每次 he 可以从塔顶拿走 1 块或 2 块积木，但他想知道有多少种不同的方式可以把这座积木塔完全拆完。

假设塔的高度是 n ，表示有 n 块积木叠在一起。

例如，如果塔高 3 块， he 可以有以下三种方式拆完积木：

1. 拿走 1 块，再拿走 1 块，再拿走 1 块（1+1+1）
2. 拿走 1 块，再拿走 2 块（1+2）
3. 一次性拿走 2 块，再拿走 1 块（2+1）

输入描述：

一个正整数 n ，表示积木塔的高度（即积木块数）， $1 \leq n \leq 20$ 。

输出描述：

一个正整数，表示可以拆完积木塔的不同方式数量。

样例输入：

3

样例输出

3

示例题解程序（基本递归解法）：

```
def count_ways(n):  
    if n == 0:  
        return 1  
    if n < 0:  
        return 0  
    return count_ways(n - 1) + count_ways(n - 2)
```

输入积木塔的高度

```
n = int(input())
```

输出不同的拆塔方式数量

```
print(count_ways(n))
```

4. （本题共 10 组测试数据，每个测试点 2 分，共 20 分）

小红在准备一个花园，她决定在花园里种植一些花。花园里有一个长条形的花坛，小红需要在花坛里按顺序种花。她有 n 种不同的花种，规定每种花种的数量少于上一次种植的数量（即，每一种花种的数量必须小于前一次种植的花种数量）。小红想知道，她有多少种不同的种植方案。

例如，假设小红有 3 朵不同的花，第 1 组种植了 3 朵，则她可以按以下方式进行种植：

1. 第 2 组种植 2 朵，第 3 组种植 1 朵。
2. 第 2 组种植 2 朵，第 3 组不种。
3. 第 2 组种植 1 朵，第 3 组不种。

在这种情况下，一共有 3 种不同的种植方案。

输入描述：

一个正整数 n ($1 \leq n \leq 20$) 和一个正整数 m ，表示花朵的种类和第一组花的数量。

输出描述：

一个整数，表示小红的种植方案数量。

样例输入

3 3

样例输出

3

示例题解程序（递归解法）：

```
def count_planting_ways(flowers_in_group, max_flowers):
    if flowers_in_group == 0:
        return 0
    if max_flowers == 0:
        return 1
    total_ways = 0
    for next_group_flowers in range(0, max_flowers):
        total_ways += count_planting_ways(flowers_in_group - 1,
next_group_flowers)
    return total_ways
```

输入花朵的种类和第一组花的数量

n, m = map(int, input().split())

输出种植方案的数量

print(count_planting_ways(n, m))