

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，哪种数据类型不允许重复元素？

- A. 列表
- B. 元组
- C. 集合
- D. 字典

正确答案：C

解析：集合（set）是无序且不包含重复元素的容器，自动去除重复项。

2. 以下选项中，哪个选项创建了二维元组？

- A. `tuple2d = ((1, 2), (3, 4))`
- B. `tuple2d = [[1, 2], [3, 4]]`
- C. `tuple2d = {(1, 2), (3, 4)}`
- D. `tuple2d = {1: (2, 3), 4: (5, 6)}`

正确答案：A

解析：选项 A 是一个二维元组，包含了两个元素，每个元素本身又是一个元组。

3. 以下哪个 Python 内置函数不能作为高阶函数使用？

- A. `sorted`
- B. `map`
- C. `filter`
- D. `sum`

正确答案：D

解析：`sum` 函数仅用于计算序列的总和，而 `sorted`、`map` 和 `filter` 都可以接受函数作为参数。

4. 关于 `lambda` 函数，以下描述正确的是？

- A. `lambda` 函数只能有两个参数
- B. `lambda` 函数无法与 `filter` 函数一起使用
- C. `lambda` 函数可以返回多个值
- D. `lambda` 函数可以用于简单的表达式

正确答案：D

解析：`lambda` 函数用于创建简单的匿名函数，通常包含一个表达式。

5. 以下哪个表达式正确使用了 `map` 函数和 `lambda` 函数来将列表中的每个元素平方？

- A. `result = map(lambda x: x2, [1, 2, 3, 4])`

- B. `result = filter(lambda x: x2, [1, 2, 3, 4])`
- C. `result = [x2 for x in [1, 2, 3, 4]]`
- D. `result = map(x2, [1, 2, 3, 4])`

正确答案: A

解析: `map` 函数和 `lambda` 函数配合使用, 可以将列表中的每个元素平方。

6. 以下哪种方法不属于 Pillow 库中用于图像变换的函数?

- A. `transpose()`
- B. `rotate()`
- C. `adjust()`
- D. `crop()`

正确答案: C

解析: `adjust()` 不是 Pillow 库中的图像变换函数, 其他选项均为有效的图像操作。

7. 使用 Pillow 库保存图像的函数是?

- A. `save()`
- B. `store()`
- C. `write()`
- D. `export()`

正确答案: A

解析: `save()` 是 Pillow 库中用于保存图像的方法。

8. 在 Pandas 中, 哪个方法不适用于向 DataFrame 添加数据?

- A. `append`
- B. `insert`
- C. `add`
- D. `concat`

正确答案: C

解析: `add` 用于执行算术操作, 而 `append`、`insert` 和 `concat` 均可用于添加数据。

9. 在 Pandas 中, 如何创建一个包含自定义索引的 Series?

- A. `pd.Series([1, 2, 3], index=['a', 'b', 'c'])`
- B. `pd.Series([1, 2, 3], columns=['a', 'b', 'c'])`
- C. `pd.DataFrame([1, 2, 3], index=['a', 'b', 'c'])`
- D. `pd.DataFrame([1, 2, 3], columns=['a', 'b', 'c'])`

正确答案: A

解析: `Series` 可以通过 `index` 参数定义自定义索引。

10. 在 Pandas 中, 以下哪个说法准确描述了 DataFrame 的特点?

- A. DataFrame 是一个一维数据结构
- B. DataFrame 中的每一列可以包含不同的数据类型
- C. DataFrame 无法进行数据的添加或删除
- D. DataFrame 只能存储数值型数据

正确答案：B

解析：DataFrame 是一种二维数据结构，每列可以包含不同的数据类型，如数值型、字符串型等。

11. 哪个 HTTP 方法通常用于向服务器更新现有资源？

- A. GET
- B. POST
- C. DELETE
- D. PUT

正确答案：D

解析：PUT 方法通常用于更新服务器上的现有资源。

12. 以下哪种排序算法的最佳时间复杂度是 $O(n \log n)$ ？

- A. 选择排序
- B. 快速排序
- C. 冒泡排序
- D. 插入排序

正确答案：B

解析：快速排序的最佳时间复杂度为 $O(n \log n)$ ，在很多情况下表现良好。

13. $O(n^2)$ 时间复杂度表示什么？

- A. 算法的执行时间随输入数据规模的平方增长
- B. 算法的执行时间与输入数据的对数成正比
- C. 算法的执行时间恒定，不受输入数据影响
- D. 算法的执行时间随着数据规模呈线性增长

正确答案：A

解析： $O(n^2)$ 表示算法的执行时间随着输入数据规模的平方而增长，通常是一些简单排序算法的时间复杂度。

14. 以下哪种排序算法是稳定的？

- A. 堆排序
- B. 希尔排序
- C. 归并排序
- D. 快速排序

正确答案：C

解析：归并排序是稳定的排序算法，而堆排序和快速排序通常不是稳定的。

15. 在二分查找算法中，哪种情况会导致查找失败？

- A. 目标值大于数据集中最大值
- B. 数据集未排序
- C. 目标值等于中间元素
- D. 目标值小于数据集中最小值

正确答案：B

解析：二分查找算法要求数据集必须是排序好的，如果未排序，将导致查找失败。

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. 在使用 Requests 库处理 HTTP 请求时，哪些方法可用于发送请求？

- A. `get()`
- B. `post()`
- C. `put()`
- D. `delete()`

正确答案：A, B, C, D

解析：Requests 库提供了多种方法来发送 HTTP 请求，包括 `get()` 用于发送 GET 请求，`post()` 用于发送 POST 请求，`put()` 用于发送 PUT 请求，`delete()` 用于发送 DELETE 请求。

2. 在递归函数中，哪些是必备的元素？

- A. 初始状态
- B. 递归调用
- C. 循环迭代
- D. 中止条件

正确答案：B, D

解析：递归函数必须包含递归调用和基础条件（即最基本的情况），以防止无限递归。中止条件则通常用于明确停止递归的条件。

3. 观察下面的代码段，选择描述正确的选项：

```
items = [2, 3, 4, 5, 6]
result = list(filter(lambda x: x % 2 != 0, map(lambda x: x + 1, items)))
```

- A. 代码段使用了两个高阶函数。
- B. `result` 列表将包含所有 `items` 列表中的元素。
- C. `map` 函数用于将 `items` 列表中的每个元素加 1。
- D. `result` 列表将包含元素 `[3, 5, 7]`。

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，`result` 列表中元素经过了 `map` 和 `filter` 处理，不会包含原始列表的所有元素；C 正确，`map` 函数用 `lambda` 表达式为列表中每个元素加 1；D 正确，`result` 列表将包含 `[3, 5, 7]`，因为这些是经过操作后的奇数。

4. 观察以下代码，选择正确的描述：

```
import pandas as pd
data = {'Product': ['A', 'B', 'C', 'D'],
        'Price': [100, 150, 200, 250],
        'Stock': [10, 20, 30, 40]}
df = pd.DataFrame(data)
product_price = df.loc[df['Price'] > 150, 'Product']
stock_result = df.iloc[1:3]
```

- A. DataFrame 是通过字典创建的，其中键作为列名。
- B. product_price 是一个 Series 对象，包含价格大于 150 的产品名称。
- C. stock_result 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，字典键作为列名；B 正确，product_price 是通过条件选择得到的 Series 对象；C 正确，stock_result 通过 iloc[1:3] 获取，它选择了 DataFrame 的第二和第三行，并保持 DataFrame 对象的结构；D 正确，loc 和 iloc 都可以用于选择 DataFrame 中的行和列，只是用法稍有不同。

5. 使用 Pillow 库处理图像时，下列哪些操作是有效的？

- A. 将图像转换为 RGBA 模式
- B. 裁剪图像的特定区域
- C. 调整图像的大小
- D. 添加水印到图像上

正确答案：A, B, C

解析：A 正确，Pillow 支持将图像转换为 RGBA 模式；B 正确，可以使用 crop() 方法裁剪图像的特定区域；C 正确，可以使用 resize() 方法调整图像大小；D 错误，Pillow 库本身不直接支持添加水印，但可以通过合并图像的方式实现。

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）

小明的家里有一个漂亮的花园，里面种满了鲜花。为了让花园保持美丽，小明每天都要浇水。由于天气的变化，每天的水量需求不同。小明希望找出一个连续的 P 天时间段，使得在这个时间段内花园的水量需求之和与目标值 T 最接近。

输入描述

第 1 行有两个用空格隔开的整数，分别表示 N 和 P ($1 < P < N < 100$)。

第 2 行有 N 个用空格隔开的整数，依次表示接下来 N 天中每天的水量需求。

第 3 行有一个整数 T，表示目标水量需求。

输出描述

一个整数，表示连续 P 天内水量需求与目标值 T 最接近的差值（绝对值）。

样例输入

```
7 3
4 2 5 1 3 6 2
10
```

样例输出

```
0
```

示例题解程序：

```
def closest_water_demand(n, p, demands, target):
    closest_diff = float('inf')

    for i in range(n - p + 1):
        current_sum = sum(demands[i:i+p])
        current_diff = abs(current_sum - target)
        closest_diff = min(closest_diff, current_diff)

    return closest_diff

# 读取第一行的 n 和 p
n, p = map(int, input().split())
# 读取第二行的水量需求
demands = list(map(int, input().split()))
# 读取目标值 T
target = int(input())

# 计算并输出结果
result = closest_water_demand(n, p, demands, target)
print(result)
```

2. (本题共 5 组测试数据, 每个测试点 2 分, 共 10 分)

你是一名智能城市交通系统的工程师, 负责管理一个城市的交通信号灯系统。为了优化交通流量, 你需要根据一天中的不同时间段和交通流量数据, 动态调整交通信号灯的绿灯时长。你的任务是模拟一天内不同路口的信号灯调整过程, 并计算出总的等待时间和绿灯开启的总时长。

具体要求如下:

1. 每个路口都有一个初始的绿灯时长 (单位为秒)。
2. 每个时间段都有不同的交通流量 (单位为辆), 交通流量越大, 绿灯时长应该相应增加, 每增加 50 辆车, 绿灯时长增加 10 秒, 但不得超过 120 秒。
3. 每个路口的红灯时长固定为 60 秒。
4. 你的任务是计算一天内每个路口的绿灯开启总时长和所有车辆的总等待时间 (红灯时长为路口每辆车的等待时间)。

输入描述:

第 1 行包含用空格隔开的整数 n 和整数 m , 分别表示路口的数量和时间段数量 (均至少为 1 个)。

第 2 行包含 n 个用空格隔开的整数, 按顺序表示每个路口的初始绿灯时长 (单位为秒)。

接下来的 m 行, 每行包含 n 个整数, 依次表示每个时间段内每个路口的交通流量 (单位为辆)。

输出描述:

第 1 行, 输出 n 个整数, 依次表示每个路口的绿灯总开启时长 (单位为秒)。

第 2 行, 输出一个整数, 表示所有车辆的总等待时间 (单位为秒)。

样例输入:

```
2 3
30 40
100 150
200 100
50 250
```

样例输出:

```
160 220
51000
```


示例题解程序：

```
def simulate_traffic_lights(n, initial_green_times, m, traffic_data):
    green_totals = [0] * n
    total_wait_time = 0

    for period in range(m):
        for i in range(n):
            traffic = traffic_data[period][i]
            green_time = initial_green_times[i] + (traffic // 50) * 10
            green_time = min(green_time, 120) # 绿灯时间不能超过 120 秒
            red_time = 60
            green_totals[i] += green_time

            # 计算该时间段的总等待时间
            wait_time = traffic * red_time
            total_wait_time += wait_time

    # 输出每个路口的绿灯总开启时长
    print(" ".join(map(str, green_totals)))
    # 输出所有车辆的总等待时间
    print(total_wait_time)

# 读取输入
n, m = map(int, input().split())
initial_green_times = list(map(int, input().split()))
traffic_data = [list(map(int, input().split())) for _ in range(m)]

# 模拟交通信号灯过程
simulate_traffic_lights(n, initial_green_times, m, traffic_data)
```

3. （本题共 5 组测试数据，每个测试点 3 分，共 15 分）

小明最近在研究一个怪异的数列。这种数列的生成规则如下：

数列的第一个元素是一个固定的整数 a 。

数列的第二个元素是一个固定的整数 b 。

从第三个元素开始，每一个元素都是前两个元素之和再减去一个常数 c 。

例如，假设 $a = 2$ ， $b = 3$ ， $c = 1$ ，那么这个数列的前几项为：2、3、4、6、9。

小明想知道，这个数列的第 n 项是什么。

输入描述：

三个整数 a ， b ， c 分别表示数列的第一个元素、第二个元素和常数 c ，以及一个正整数 n 表示数列的第 n 项， $1 \leq n \leq 20$ 。

输出描述：

一个整数，表示数列的第 n 项的值。

样例输入：

2 3 1 5

样例输出：

9

示例题解程序（基本递归解法）：

```
def func(n, a, b, c):  
    if n == 1:  
        return a  
    if n == 2:  
        return b  
    return func(n - 1, a, b, c) + func(n - 2, a, b, c) - c  
  
# 输入数列的初始元素 a, b, 常数 c, 以及 n 的值  
a, b, c, n = map(int, input().split())  
# 输出数列的第 n 项  
print(func(n, a, b, c))
```

4. （本题共 10 组测试数据，每个测试点 2 分，共 20 分）

小明正在帮爸爸整理家里的旧书。家里有很多堆叠在一起的书籍，小明想把这些书重新整理成一些新的堆叠方式。整理的规则是，每一堆书都必须保持比上一堆少或相等数量的书。

例如，假设有 3 本书，小明可以按以下方式进行整理：

1. 第一堆放 3 本
2. 第一堆放 2 本，第二堆放 1 本
3. 第一堆放 1 本，第二堆放 1 本，第三堆放 1 本

小明想知道，他可以有多少种不同的堆叠方式来整理书籍。

输入描述：

一个正整数 n ，表示书籍的数量， $1 \leq n \leq 20$ 。

输出描述：

一个整数，表示小明可以整理书籍的不同堆叠方式数量。

样例输入：

3

样例输出：

3

示例题解程序（递归解法）：

```
def ways_to_stack_books(n, max_books):  
    if n <= 1:  
        return 1  
    total_ways = 0  
    for i in range(1, max_books + 1):  
        total_ways += ways_to_stack_books(n - i, i)  
    return total_ways  
  
# 输入书籍数量  
n = int(input())  
# 输出不同堆叠方式的数量  
print(ways_to_stack_books(n, n))
```