

## 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，哪种数据类型不能作为字典的键？

- A. 列表
- B. 字符串
- C. 浮点数
- D. 元组

正确答案：A

解析：字典的键必须是可哈希（不可变）的数据类型，列表不满足。

2. 以下选项中，哪个选项创建了四维列表？

- A. `list4d = [[[[1, 2], [3, 4]], [[5, 6], [7, 8]]]]`
- B. `list4d = [[[1, 2], [3, 4]], [5, 6]]`
- C. `list4d = {1: {2: [3, 4]}, 5: {6: [7, 8]}}`
- D. `list4d = ([1, 2], [3, 4], [5, 6], [7, 8])`

正确答案：A

解析：选项 A 创建了一个四维列表，即列表的列表的列表的列表。

3. 以下哪个 Python 内置函数可以作为高阶函数使用？

- A. `sorted`
- B. `len`
- C. `type`
- D. `print`

正确答案：A

解析：`sorted` 函数是一个高阶函数，因为它可以接受一个函数作为参数进行排序。

4. 关于 `lambda` 函数，以下描述不正确的是？

- A. `lambda` 函数可以返回一个值
- B. `lambda` 函数可以有多个参数
- C. `lambda` 函数是匿名的
- D. `lambda` 函数必须有 `return` 语句

正确答案：D

解析：`lambda` 函数不需要 `return` 语句，表达式的结果即为返回值。

5. 以下哪个 Python 表达式正确地使用了高阶函数和匿名函数来筛选并返回偶数？

- A. `result = list(filter(lambda x: x % 2 == 0, [1, 2, 3, 4]))`
- B. `result = map(lambda x: x % 2 == 0, [1, 2, 3, 4])`
- C. `result = [x for x in [1, 2, 3, 4] if x % 2 == 0]`
- D. `result = filter(2, [1, 2, 3, 4])`

正确答案：A

解析：选项 A 正确地使用了 `filter` 高阶函数和匿名函数来筛选偶数。

6. 以下哪种方法不是 Pillow 库用于图像操作的函数？

- A. `resize()`
- B. `rotate()`
- C. `filter()`
- D. `flip()`

正确答案：C

解析：`filter()` 是 Python 内置函数，而非 Pillow 库中的图像操作函数。

7. 使用 Pillow 库将图像从 JPEG 格式转换为 PNG 格式的函数是？

- A. `save()`
- B. `convert()`
- C. `format()`
- D. `transform()`

正确答案：A

解析：`save()` 函数可用于将图像保存为不同的格式，例如从 JPEG 转换为 PNG。

8. 在 Pandas 中，哪个方法用于筛选数据？

- A. `filter()`
- B. `select()`
- C. `choose()`
- D. `extract()`

正确答案：A

解析：`filter()` 方法用于筛选 `DataFrame` 中的数据。

9. 在 Pandas 中，以下哪个选项创建了一个包含两个元素的 Series？

- A. `pd.Series([1, 2])`
- B. `pd.Series({1: 2})`
- C. `pd.Series([1])`
- D. `pd.Series((1, 2, 3))`

正确答案：A

解析：选项 A 创建了一个包含两个元素的 Series。

10. 在 Pandas 中，以下哪个选项准确描述了 Series？

- A. Series 是一个一维数组
- B. Series 是一个二维数组
- C. Series 可以包含多种数据类型
- D. Series 的大小不可变

正确答案：A

解析：Series 是 Pandas 中的一维数据结构，类似于 Python 中的列表。

11. 以下哪个 HTTP 方法用于向服务器发送数据而不影响已有资源？

- A. GET
- B. POST

- C. PATCH
- D. DELETE

正确答案：C

解析：PATCH 方法用于更新资源的一部分内容，而不影响其他部分。

12. 以下哪种排序算法的时间复杂度为  $O(n\log n)$ ？

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

正确答案：C

解析：归并排序的时间复杂度为  $O(n\log n)$ ，在所有情况下表现稳定。

13. 在算法分析中， $O(\log n)$  表示什么意思？

- A. 算法的复杂度与数据的对数成正比
- B. 算法的复杂度与数据的大小成正比
- C. 算法的复杂度与数据的平方成正比
- D. 算法的复杂度与数据的立方成正比

正确答案：A

解析： $O(\log n)$  表示算法的复杂度与输入数据的对数成正比，通常用于描述二分查找等算法。

14. 以下哪种排序算法可能会改变相同元素的相对位置？

- A. 归并排序
- B. 冒泡排序
- C. 选择排序
- D. 插入排序

正确答案：C

解析：选择排序不是稳定的排序算法，可能会改变相同元素的相对位置。

15. 在以下关于时间复杂度的描述中，正确的是？

- A. 时间复杂度为  $O(1)$  的算法在任何情况下执行时间都是常数，不依赖于输入数据的大小。
- B. 时间复杂度为  $O(n)$  的算法总是比时间复杂度为  $O(\log n)$  的算法快
- C. 时间复杂度为  $O(2^n)$  的算法适用于处理非常大规模的数据集
- D. 时间复杂度为  $O(n!)$  的算法在某些情况下可能比时间复杂度为  $O(n^2)$  的算法更快

正确答案：A

解析：这些描述中只有 A 正确。B 选项忽视了数据集很小的情况，C 选项错误，D 选项前者必然不可能更快。

## 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. 关于 Python 列表的说法中，哪些是正确的？

- A. 列表是有序的序列
- B. 列表中的元素可以是不同类型
- C. 列表的大小是固定的，创建后不能改变

D. 可以使用索引访问列表中的元素

正确答案: A, B, D

解析: A 正确: 列表是有序的序列, 每个元素都有一个确定的位置。B 正确: 列表中的元素可以是不同类型。C 错误: 列表的大小是动态的, 可以添加或删除元素。D 正确: 可以通过索引访问列表中的元素。

2. 关于 Python 集合 (set) 的操作, 哪些是正确的?

A. 集合中的元素是无序的

B. 集合可以进行交集、并集和差集操作

C. 集合支持索引操作

D. 可以使用 add() 方法向集合中添加元素

正确答案: A, B, D

解析: A 正确: 集合中的元素是无序的。B 正确: 集合支持交集、并集和差集操作。C 错误: 集合不支持索引操作。D 正确: 可以使用 add() 方法向集合中添加元素。

3. 观察以下代码, 选择正确的描述:

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

```
result = [x**2 for x in numbers if x % 2 == 0]
```

A. 代码使用了列表推导式。

B. result 列表将包含所有 numbers 列表中的元素。

C. 列表推导式中的 if 条件用于过滤偶数。

D. result 列表将包含元素 [4, 16]。

正确答案: A, C, D

解析: A 正确: 代码使用了列表推导式。B 错误: result 列表只包含满足条件的元素。C 正确: if 条件用于筛选出偶数。D 正确: result 列表将包含偶数的平方 [4, 16]。

4. 使用 Pandas 库进行数据处理时, 以下操作哪些是有效的?

A. 使用 dropna() 方法删除 DataFrame 中的缺失值

B. 使用 fillna() 方法填充 DataFrame 中的缺失值

C. 使用 drop() 方法删除指定的行或列

D. 使用 loc[] 方法根据标签筛选数据

正确答案: A, B, C, D

解析: A 正确: dropna() 方法可以删除 DataFrame 中的缺失值。B 正确: fillna() 方法可以填充缺失值。C 正确: drop() 方法用于删除指定的行或列。D 正确: loc[] 方法用于根据行标签或列标签筛选数据。

5. 以下关于 Python 高阶函数的描述, 哪些是正确的?

A. map() 函数可以对序列中的每个元素应用一个函数

B. filter() 函数可以过滤序列中的元素

C. map() 和 filter() 函数返回的是一个列表

D. sorted() 函数可以接受一个函数作为参数进行自定义排序。

正确答案: A, B, D

解析: A 正确: map() 函数对序列中的每个元素应用一个函数。B 正确: filter() 函数可以过

滤序列中的元素。C 错误：函数返回的是一个生成器，而非列表。D 正确：`sorted()` 函数可以接受一个函数作为参数进行自定义排序。

### 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）

某公司正在进行一个为期  $T$  天的产品推广活动。为了监测推广效果，团队每天记录了网站的访问量。现在，推广团队希望找到一个连续的  $Q$  天时间段，使得这个时间段内的平均访问量最大。

输入描述

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

第 2 行有  $T$  个用空格隔开的整数，依次表示接下来  $T$  天中的每日访问量。

输出描述

一个浮点数，表示连续  $Q$  天内的最大平均访问量。结果保留两位小数。

样例输入

7 3

10 20 30 40 50 60 70

样例输出

60.00

示例题解程序：

```
def max_average_visits(t, q, visits):
    max_average = float('-inf')

    for i in range(t - q + 1):
        current_sum = sum(visits[i:i+q])
        current_average = current_sum / q
        max_average = max(max_average, current_average)

    return max_average

# 读取第一行的 t 和 q
t, q = map(int, input().split())
# 读取第二行的访问量
visits = list(map(int, input().split()))

# 计算并输出结果，保留两位小数
result = max_average_visits(t, q, visits)
print(f"{result:.2f}")
```

2. （本题共 5 组测试数据，每个测试点 2 分，共 10 分）

你是一名城市垃圾回收车队的调度员，负责管理垃圾车每天的运行路线。垃圾车每天按照预定的路线从垃圾处理厂出发，经过多个居民区收集垃圾后返回垃圾处理厂。垃圾处理厂和居民区都位于东西走向的主干道旁，垃圾处理厂在最西侧。每辆垃圾车都有一定的容量限制，超出容量时必须返回垃圾处理厂倾倒垃圾，然后再继续未完成的路线。

具体要求如下：

1. 垃圾车从垃圾处理厂出发，按顺序经过每个居民区收集垃圾。
2. 每个居民区有一定量的垃圾等待回收，垃圾车在该居民区停留的时间与垃圾量成正比，每吨垃圾需要花费 1 分钟时间来装车。
3. 垃圾车的额定容量为 100 吨，在某个居民区装载后，当车上的垃圾量达到或超过额定容量时，必须立即返回垃圾处理厂倾倒垃圾（行驶时间为 1 公里需要 2 分钟）。倾倒完毕后，垃圾车继续从第一个未完成的居民区开始收集。
4. 在所有居民区完成收集后，垃圾车需要返回处理厂，倾倒后任务结束。
5. 每次倾倒垃圾需要额外花费 5 分钟。

你的任务是计算垃圾车完成一整天的路线所需的总时间，并判断垃圾车返回处理厂的次数。

输入描述：

第 1 行包含一个整数  $n$ ，表示垃圾车经过的居民区数量（至少为 1 个）。

第 2 行包含  $n$  个整数，依次表示自西向东每个居民区产生的垃圾量（单位为吨）。

第 3 行包含  $n$  个整数，依次表示自西向东每个居民区与垃圾处理厂的距离（单位为公里）。

输出描述：

第 1 行，输出一个整数，表示垃圾车完成整天路线所需的总时间（单位为分钟）。

第 2 行，输出垃圾车返回处理厂的次数（包括最后一次）。

样例输入：

```
4
30 50 20 40
5 8 10 16
```

样例输出：

```
254
2
```

题目分析：

这是一道对生活问题用程序模拟的问题，要求计算电梯根据一系列楼层指令移动的总时间。电梯从1层开始，每层楼的移动时间不同，且每次到达指定楼层后需要停留5秒钟。关键点梳理：

1. 移动时间：每上升1层需要6秒；每下降1层需要4秒。需要在指令过程中判断每次移动是上升还是下降。
2. 停留时间：每次指令后必然停留5秒（不算最后一次）。
3. 初始位置：电梯初始位置在第1层。
4. 特殊情况：楼层指令中可能有负数，表示地下楼层，且-1和1之间的间隔也是一层楼。因此，我们在计算相邻两层的移动时间时，需要区分前后两层是否一正一负。

示例题解程序：

```
def cal_movement_time(start_floor, end_floor):
    if start_floor >= end_floor:
        if end_floor < 0 < start_floor:
            # 从正楼层下降到负楼层
            return 4 * (start_floor - end_floor - 1) + 5
        else:
            # 从高楼层下降到低楼层
            return 4 * (start_floor - end_floor) + 5
    else:
        if start_floor < 0 < end_floor:
            # 从负楼层上升到正楼层
            return 6 * (end_floor - start_floor - 1) + 5
        else:
            # 从低楼层上升到高楼层
            return 6 * (end_floor - start_floor) + 5

# 读取指令序列
floor_sequence = list(map(int, input().split()))
# 总时间变量初始化
total_time = 0
# 将初始楼层1插入到指令序列的开头
floor_sequence = [1] + floor_sequence
# 从第二个楼层开始遍历指令序列
for i in range(1, len(floor_sequence)):
    # 计算从上一个楼层到当前楼层的移动时间，并累加到总时间
    total_time += cal_movement_time(floor_sequence[i-1], floor_sequence[i])
# 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）

小明在一家糖果店工作，店里有  $n$  种不同口味的糖果。为了迎合顾客的喜好，他需要制作一个糖果袋。每个糖果袋必须包含至少一种糖果，但同种口味的糖果在一个袋子里不能出现多次。小明想知道，他可以制作多少种不同组合的糖果袋。

例如，假设糖果店里有 3 种口味的糖果（草莓味、巧克力味、橙子味），那么他可以制作的糖果袋组合有：

1. 草莓味
2. 巧克力味
3. 橙子味
4. 草莓味 + 巧克力味
5. 草莓味 + 橙子味
6. 巧克力味 + 橙子味
7. 草莓味 + 巧克力味 + 橙子味

总共有 7 种不同的组合。

输入描述：

一个正整数  $n$ ，表示糖果的种类数量， $1 \leq n \leq 15$ 。

输出描述：

一个正整数，表示不同组合的糖果袋数量。

样例输入：

3

样例输出：

7

示例题解程序（基本递归解法）：

```
def count_combinations(n):  
    if n == 0:  
        return 0  
    if n == 1:  
        return 1  
    # 考虑包含第 n 种糖果的组合和不包含的组合  
    return 1 + 2 * count_combinations(n - 1)  
  
# 输入糖果种类数量  
n = int(input())  
# 输出不同组合的糖果袋数量  
print(count_combinations(n))
```

4. (本题共 10 组测试数据, 每个测试点 2 分, 共 20 分)

在一个走格子的游戏中, 小明从第 1 格开始, 每一步可以向前前进 1 到 6 格的距离。如果某一步走到了第  $k$  格 (炸弹格), 游戏会立刻结束, 小明需要重新开始。小明的目标是顺利走到第  $n$  格, 请问共有多少种不同的走法可以顺利到达第  $n$  格?

例如, 要走到第 5 格, 其中第 3 格是炸弹格, 则可以按照以下 4 种方案来走:

1.  $1 \rightarrow 2 \rightarrow 1$
2.  $1 \rightarrow 3$
3.  $3 \rightarrow 1$
4. 4

输入描述:

两个用空格隔开的正整数  $n$  和  $k$ , 分别表示目标格数和炸弹格的位置。 $2 \leq n \leq 20, 2 \leq k < n$ 。

输出描述:

一个整数, 表示不同走法的数量。

样例输入:

5 3

样例输出:

4

示例题解程序（递归解法）：

```
def count_ways_to_reach(n, k):  
    if n == 1:  
        return 1  
    if n < 1 or n == k:  
        return 0  
  
    total_ways = 0  
    for step in range(1, 7):  
        total_ways += count_ways_to_reach(n - step, k)  
  
    return total_ways  
  
# 输入目标格数和炸弹格的位置  
n, k = map(int, input().split())  
# 输出不同走法的数量  
print(count_ways_to_reach(n, k))
```