

## 青少年人工智能编程水平测试八级试题 4

### 一、单项选择题（每题 2 分，共 15 题，共 30 分）

1. 在 Python 中，类的实例化后可以产生什么？

- A. 一个新的类
- B. 一个对象
- C. 一个模块
- D. 一个方法

正确答案：B

解析：实例化是类生成对象的过程，生成的就是一个对象。

2. 以下哪种方法不能在外部被直接调用？

- A. 类方法
- B. 静态方法
- C. 实例方法
- D. 私有方法

正确答案：B

解析：私有方法不能在类外部被直接调用。

3. 下面的代码定义了一个类，请问对该类描述正确的是？

```
class Vehicle:
    def __init__(self, type):
        self.type = type

    def describe(self):
        print(self.type)
```

- A. 类 Vehicle 没有属性。
- B. 类 Vehicle 含有一个属性和一个方法。
- C. describe 是一个类方法。
- D. \_\_init\_\_ 方法不做任何事情。

正确答案：B

解析：类 Vehicle 包含属性 type 和方法 describe，B 选项正确。

4. 在 Python 中，当 try 块成功执行且无异常时，哪个块将不会被执行？

- A. except
- B. else
- C. finally
- D. 都会被执行

正确答案：A

解析：except 块仅在 try 块发生异常时执行。

5. 在 Python 的异常处理结构中，finally 块的执行时机是什么？

- A. 异常发生时
- B. 异常未发生时
- C. 不论是否发生异常
- D. 无法执行

正确答案: C

解析: `finally` 块无论是否发生异常都会执行。

6. 如何用 NumPy 库创建一个包含 10 到 30 的数组?

- A. `np.arange(10, 30)`
- B. `np.arange(10, 31)`
- C. `np.linspace(10, 30, 21)`
- D. `np.zeros(20)`

正确答案: B

解析: `np.arange(10, 31)` 生成包含 10 到 30 的整数数组。

7. 以下哪种 NumPy 函数用于在指定范围内生成给定数量的数值?

- A. `zeros()`
- B. `arange()`
- C. `linspace()`
- D. `ones()`

正确答案: C

解析: `linspace()` 在给定范围内生成等间距数值。

8. 以下哪种操作可用于计算数组的最大值?

- A. `min()`
- B. `max()`
- C. `mean()`
- D. `sum()`

正确答案: B

解析: `max()` 函数用于计算数组中所有元素的最大值。

9. 在 Pandas 中, 读取 SQL 数据库数据的方法是?

- A. `read_csv()`
- B. `read_sql()`
- C. `to_sql()`
- D. `read_json()`

正确答案: B

解析: `read_sql()` 用于从 SQL 数据库读取数据到 `DataFrame`。

10. 在 Pandas 中, 以下哪个方法用于按列值升序排序?

- A. `sort()`
- B. `sort_values(ascending=True)`
- C. `sort_values(ascending=False)`
- D. `rank()`

正确答案：B

解析：sort\_values() 可以使用 ascending=True 选项来进行升序排序。

11. 如何在 Pandas 中对数据分组后计算每个组的最小值？

- A. df.groupby('column').min()
- B. df.groupby('column').sum()
- C. df.min('column')
- D. df.group\_min('column')

正确答案：A

解析：groupby().min() 用于计算每组数据的最小值。

12. 在 Pandas 中，如何筛选出 DataFrame 中所有 'Height' 列的值大于 160 的行？

- A. df[df['Height'] > 160]
- B. df['Height'] > 160
- C. df.filter('Height > 160')
- D. df.query('Height > 160')

正确答案：A

解析：A 选项通过布尔索引筛选满足条件的行。

13. 贪心算法主要用于以下哪类问题？

- A. 最优子结构问题
- B. 简单线性问题
- C. 回溯问题
- D. 有向图问题

正确答案：A

解析：贪心算法用于求解具有最优子结构的复杂问题。

14. 以下哪种排序算法应用了分治算法思想？

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

正确答案：C

解析：归并排序应用了分治算法的思想。

15. 深度优先搜索（DFS）适合于解决以下哪类问题？

- A. 遍历所有可能的解
- B. 层次遍历
- C. 查找最短路径
- D. 数据筛选

正确答案：A

解析：DFS 适合于遍历所有可能的解决方案和路径探索问题。

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 Pandas 进行数据处理时，以下哪些方法可用于数据转换？

- A. `apply()`
- B. `map()`
- C. `astype()`
- D. `merge()`

答案：A, B, C

解析：`apply()`、`map()`、`astype()` 都用于数据转换，而 `merge()` 用于数据合并。

2. 在面向对象编程中，关于多态性，下列哪些描述是正确的？

- A. 多态允许子类以不同方式实现父类方法
- B. 多态性使代码更灵活
- C. 多态只能通过继承实现
- D. 接口和抽象类都支持多态

答案：A, B, D

解析：多态允许子类不同实现父类方法，提升代码灵活性，接口和抽象类都支持多态；C 不正确，多态不只限于继承。

3. 以下代码展示了 NumPy 和 Pandas 的结合使用，哪些描述是正确的？

```
import numpy as np
import pandas as pd
arr = np.array([1, 2, 3, 4])
df = pd.DataFrame(arr, columns=['Numbers'])
df['Double'] = df['Numbers'] * 2
```

- A. 数组被转换为 DataFrame
- B. 新增列 Double 包含每个原始数据的两倍的值
- C. 数据 arr 是 Series 类型
- D. 数据类型不兼容，会出现报错

答案：A, B

解析：数组被转换为 DataFrame，新增的 Double 列包含元素的两倍；C 不正确，arr 是 numpy 的数组；D 不正确，这段代码兼容数据类型。

4. 以下代码演示了 Python 中的异常处理机制，哪些描述是正确的？

```
try:
    numbers = [1, 2, 3]
    print(numbers[5])
except IndexError as e:
    print('Index Error:', e)
finally:
    print('Execution complete')
```

- A. 代码捕获了 IndexError
- B. except 块未被执行
- C. finally 块始终执行

D. 程序在异常发生时停止执行

答案：A, C

解析：代码捕获了 `IndexError`，`finally` 块始终执行，而 `except` 块在捕获异常时才执行。

5. 关于动态规划算法，下列哪些描述是正确的？

A. 动态规划适合解决重叠子问题

B. 动态规划通常通过表格或数组保存中间结果

C. 动态规划总是解决问题最优的选择

D. 动态规划可用于最短路径问题

答案：A, B, D

解析：动态规划解决重叠子问题，通过表格存储中间结果，可用于最短路径问题；但 C 不正确，动态规划并非总是最优选择。

### 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）

小李是一家电影院的排片经理，他需要安排电影在不同的时间段播放。每部电影都有一个开始时间和结束时间，且在同一个时间段只能播放一部电影。为了让电影院获得最大的收益，小李希望在一天内安排尽可能多的电影播放。请帮助小李设计出一个最优的电影播放安排方案，使得一天内播放的电影数量最多。

输入描述

第一行包含一个整数  $n$ ，表示电影的数量。 $(1 \leq n \leq 100)$

接下来的  $n$  行，每行包含两个整数  $s$  和  $e$ ，分别表示第  $i$  部电影的开始时间和结束时间。 $(0 \leq s < e \leq 24)$

输出描述

输出一天内最多能安排播放的电影数量。

样例输入

```
5
1 4
3 5
0 6
5 7
8 9
```

样例输出

```
3
```

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)

小红和她的朋友们在一个美术课上需要制作一条彩色的串珠项链。项链由  $n$  颗珠子组成, 每颗珠子可以是红色、绿色或蓝色。他们需要遵循以下规则:

1. 相邻的两颗珠子不能是相同的颜色。
2. 项链的首尾珠子不能是相同的颜色。
3. 在项链的任何一段连续三颗珠子中, 必须有三种不同的颜色。

请帮助小红计算在给定的珠子数量下, 有多少种符合要求的项链制作方案。

输入描述

一个正整数  $n$ , 表示项链的珠子数量,  $3 \leq n \leq 12$ 。

输出描述

一个整数, 表示所有符合要求的项链制作方案数量。

样例输入

5

样例输出

6



示例题解程序：

```
from itertools import product
```

```
def count_necklace_combinations(n):
```

```
    # 三种颜色的珠子
```

```
    colors = ['R', 'G', 'B']
```

```
    count = 0
```

```
    # 生成所有可能的珠子颜色组合
```

```
    for necklace in product(colors, repeat=n):
```

```
        # 检查相邻珠子和首尾的颜色是否不同
```

```
        if all(necklace[i] != necklace[i + 1] for i in range(n - 1)) and  
necklace[0] != necklace[-1]:
```

```
            # 检查每三个连续珠子是否都是三种不同的颜色
```

```
            if all(len(set(necklace[i:i+3])) == 3 for i in range(n - 2)):
```

```
                count += 1
```

```
    return count
```

```
# 读取输入
```

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

```
# 计算并输出结果
```

```
print(count_necklace_combinations(n))
```

3. （本题共 5 组测试数据，每个测试点 3 分，共 15 分）

小美是一名考古学家，她正在研究一片古代遗迹。这个遗迹可以看作一个二维地图，其中有一些区域被水淹没了，只有陆地部分是可以行走的。小美希望能够测量每片相连的陆地的面积（格子数）。相连的陆地是指在上下左右方向上相邻的陆地格子，它们形成一个整体。

请帮助小美计算遗迹中所有相连的陆地区域的面积，并输出最大的陆地区域的面积。

输入描述

第一行包含两个整数  $n$  和  $m$ ，分别表示地图的行数和列数， $2 \leq n, m \leq 50$ 。

接下来的  $n$  行，每行包含  $m$  个字符，表示遗迹的地图：

- `.`：表示陆地，可以行走。
- `#`：表示水域，不可行走。

输出描述

输出一个整数，表示遗迹中最大连通陆地区域的面积（格子数）。

样例输入

```
5 5
..#..
.#.#.
..##.
.##..
.....
```

样例输出

```
17
```

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）

小明是一名旅行爱好者，他正在准备一次长途旅行。为了在旅途中保持精力充沛，他决定带上一些零食。小明有一个容量为  $W$  的背包，背包最多能承载一定的重量。小明的家中有  $n$  种不同的零食，每种零食都有一定的重量和美味度，且每种零食最多只能带一份。

为了让这次旅行更加愉快，小明希望背包中的零食总重量不超过  $W$ ，且美味度尽可能高。同时，为了避免挑食，他规定背包中任意一个零食的美味度不能超过总美味度的一半。

请帮助小明计算，他最多可以获得的总美味度是多少。

输入描述

第一行包含两个整数  $n$  和  $W$ ，分别表示零食的种类数量和背包的最大承重量， $1 \leq n \leq 50$ ， $1 \leq W \leq 1000$ 。

接下来的  $n$  行，每行包含两个整数，表示第  $i$  种零食的重量和美味度， $1 \leq \text{重量} \leq 1000$ ，美味度  $\leq 100$ 。

输出描述

输出一个整数，表示小明在满足所有条件的情况下能够获得的最大总美味度。

样例输入

```
4 10
2 5
3 6
4 8
5 7
```

样例输出

```
19
```

示例题解程序：

```
def max_delicious(n, W, snacks):
    # 初始化 DP 数组，dp[j]表示容量为 j 时的最大美味度
    dp = [[0] * (W + 1) for _ in range(n + 1)]

    # 遍历每种零食
    for i in range(1, n + 1):
        weight, delicious = snacks[i - 1]
        for j in range(W + 1):
            # 不选当前零食
            dp[i][j] = dp[i - 1][j]
            # 选当前零食，且背包容量足够
            if j >= weight:
                dp[i][j] = max(dp[i][j], dp[i - 1][j - weight] + delicious)

    max_delicious = 0

    # 遍历 dp 数组找出符合条件的最大美味度
    for i in range(n + 1):
        for j in range(W + 1):
            total_delicious = dp[i][j]
            if total_delicious > 0:
                # 检查是否符合约束条件
                valid = True
                for k in range(n):
                    if snacks[k][1] > total_delicious / 2 and (dp[i][j] -
snacks[k][1]) >= 0:
                        valid = False
                        break
                if valid:
                    max_delicious = max(max_delicious, total_delicious)

    # 返回符合要求的最大美味度
    return max_delicious if max_delicious > 0 else 0

# 读取输入
n, W = map(int, input().split())
snacks = [tuple(map(int, input().split())) for _ in range(n)]

# 输出结果
print(max_delicious(n, W, snacks))
```