CCF 编程能力等级认证
Grade Examination of Software Programming

Python 四级

2026 年 06 月

1 单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	C	B	B	B	D	A	B	B	B	D	B	B	A	A	B

第 1 题 学校组织同学们到未来农场参观，小明听讲解员介绍。在智能温室中，湿度传感器可以连续检测土壤的湿度，并将检测到的湿度数据实时发送给中央控制器。中央控制器根据这些数据判断是否开启灌溉系统。请问，这里的湿度传感器所发挥的作用，类似于计算机系统中的哪一类组件？（ ）

- A. 处理器
- B. 存储器
- C. 输入设备
- D. 输出设备

第 2 题 小明同学去郊外农场参观回来后，在自己的电脑中安装了一个 3D 农场仿真模拟系统。因为他今年买的这台二手电脑有点老旧，3D 农场仿真模拟系统运行时经常弹出“系统内存不足”的警告，导致系统卡顿严重。他计划通过增加物理内存来解决问题。请问他应该购买以下哪种硬件组件？（ ）

- A. 机械硬盘
- B. 内存条
- C. 图形显卡
- D. 移动硬盘

第 3 题 执行下面 Python 代码后，输出的结果是？（ ）

```python
a = {1: [2], 3: [4]}
b = a.copy()
b[1].append(3)
print(a[1])
```

- A. `[2]`
- B. `[2, 3]`
- C. `{1: [2, 3]}`
- D. 报错

第 4 题 如果想要将一个二维列表 `matrix = [[1, 2], [3, 4], [5, 6]]` 扁平化为 `[1, 2, 3, 4, 5, 6]`，应使用下列哪个列表推导式？（ ）

- A. `[item for item in matrix]`
- B. `[item for row in matrix for item in row]`

- [ ] **C.** `[row for row in matrix for item in row]`
- [ ] **D.** `[row for item in matrix for item in row]`

第 5 题 执行下面 Python 代码后，输出的结果是？（ ）

```python
def func(x, lst=[]):
    lst.append(x)
    return lst


print(func(1), func(2), func(3))
```

- [ ] **A.** `[1] [2] [3]`
- [ ] **B.** `[1] [1, 2] [1, 2, 3]`
- [ ] **C.** `[3] [3] [3]`
- [ ] **D.** `[1, 2, 3] [1, 2, 3] [1, 2, 3]`

第 6 题 执行下面 Python 代码后，输出的结果是？（ ）

```python
def func(x, y, *args, z=10, **kwargs):
    return x + y + z + sum(args) + kwargs.get('m', 0)


result = func(1, 2, 3, 4, z=5, m=6, n=7)
print(result)
```

- [ ] **A.** `21`
- [ ] **B.** `28`
- [ ] **C.** `31`
- [ ] **D.** 程序会报错

第 7 题 与列表推导式 `[i**2 for i in range(10)]` 功能等价的 `map` 函数调用是？（ ）

- [ ] **A.** `list(map(i: i ** 2, range(10)))`
- [ ] **B.** `list(map(lambda i: i ** 2, range(10)))`
- [ ] **C.** `map(lambda i: i ** 2, list(range(10)))`
- [ ] **D.** `map(i ** 2 for i in range(10))`

第 8 题 以下代码的功能是读取一个二进制图片文件 `pic.jpg` 并复制到 `pic_copy.jpg`，空白处应填入？（ ）

```python
with open('pic.jpg', '__①__') as src, open('pic_copy.jpg', '__②__') as dst:
    dst.write(src.read())
```

- [ ] **A.** ① `'r'` ② `'w'`
- [ ] **B.** ① `'rb'` ② `'wb'`
- [ ] **C.** ① `'r+'` ② `'w+'`
- [ ] **D.** ① `'a'` ② `'wb'`

第 9 题 以下函数的时间复杂度是？（ ）

```python
def example(n):
    matrix = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            matrix[i][j] = i * j
    return matrix
```

- [ ] **A.** $O(n)$
- [ ] **B.** $O(n^2)$
- [ ] **C.** $O(n^3)$
- [ ] **D.** $O(1)$

第 10 题 执行下面 Python 代码后，输出的结果是？ （ ）

```python
def test_function():
    try:
        return "From try"
    finally:
        print("From finally", end="#")


print(test_function(), end="#")
```

- [ ] **A.** `From try#`
- [ ] **B.** `From finally#`
- [ ] **C.** `From try#From finally#`
- [ ] **D.** `From finally#From try#`

第 11 题 若输入 `5` ，以下程序的输出结果是？ （ ）

```python
n = int(input())
f = [0] * (n + 1)
f[0] = 1
for i in range(1, n + 1):
    f[i] = f[i - 1] * i
print(f[n])
```

- [ ] **A.** `24`
- [ ] **B.** `120`
- [ ] **C.** `125`
- [ ] **D.** `720`

第 12 题 关于以下代码，说法正确的是？ （ ）

```python
words = ["apple", "Cherry", "Banana"]
sorted_words = sorted(words, key=lambda s: s.lower())
```

- [ ] **A.** 排序结果中 `"Banana"` 会排在 `"apple"` 前面，因为大写字母 `ASCII` 码小于小写字母。
- [ ] **B.** `key=lambda s: s.lower()` 可以替换为 `key=str.lower` ，效果相同。
- [ ] **C.** 此排序会修改原始列表 `words` 中的元素顺序。
- [ ] **D.** `sorted` 函数不能直接对字符串列表进行排序。

第 13 题 某电商平台根据用户的会员等级计算商品最终价格。等级与折扣率的对应关系存储在字典 `discounts` 中。为了保证程序在面对未知等级时不会报错，而是按原价（即不打折）处理，以下哪个选项是正确的？（ ）

```python
def calculate_price(price, level):
    discounts = {"VIP": 0.8, "SVIP": 0.6}
    # 请补全代码
    discount_rate = ______①______
    return ______②______


final_price = calculate_price(100, "VIP")
print(final_price)
```

- ☐ **A.** ① `discounts.get(level, 1.0)` ② `price * discount_rate`
- ☐ **B.** ① `discounts[level]` ② `price * discount_rate`
- ☐ **C.** ① `discounts.get(level)` ② `price / discount_rate`
- ☐ **D.** ① `level in discounts` ② `price * (1 - discount_rate)`

第 14 题 有一个名为 `score.txt` 的文本文件，内容如下（第一行为标题行，后面每行包含姓名和分数，用空格分隔）：

```
姓名 分数
张三 85
李四 42
王五 90
赵六 78
```

编写一段 Python 代码，读取该文件，统计及格人数（分数 ≥ 60 分），并在最后输出统计结果，以下哪个选项是正确的？（ ）

```python
count = 0
with open("score.txt", "r") as f:
    lines = f.readlines()
    for i in range(1, len(lines)):
        data = lines[i].______  # 此处需要填入适当的方法
        score = int(data[1])
        if score >= 60:
            count += 1
print("及格人数: ", count)
```

- ☐ **A.** `split()`
- ☐ **B.** `strip()`
- ☐ **C.** `read()`
- ☐ **D.** `splitlines()`

第 15 题 商品列表已按特定规则排序，每个商品包含三个信息：编号、价格、库存量，对应三元组 `(id, price, stock)`。现有预算 `M` 元，按排好的顺序依次购买，只要当前商品的价格不超过剩余预算并且尚有剩余就购买并扣款，否则立即停止。以下哪个选项是正确的？（ ）

```python
budget = M
products.sort(key=lambda x: (x[1], x[2], x[0]))  # 按规则排序
bought = []
for item in products:
    if ________:  # 此处需要填入适当的代码
        budget -= item[1]
        bought.append(item[0])
    else:
        break
```

- ☐ **A.** `budget > 0 and item[2] != 0`
- ☐ **B.** `budget >= item[1] and item[2] != 0`
- ☐ **C.** `budget > item[1] and item[2] > 0`
- ☐ **D.** `budget <= item[1] and item[2] > 0`

2 判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	×	√	√	√	×	√	√	×	×

第 1 题 又到期末考试周，小明发现这次许多闭卷考试不仅禁止携带手机、平板电脑，还有最近比较时髦的各类 AI 眼镜（也有叫智能眼镜）也同样不允许带入考场。这些 AI 眼镜应该也是内置了操作系统并可能支持 Wi-Fi 或蓝牙连接。

第 2 题 执行 `[[j for j in range(i)] for i in range(3)]` 解析式的结果是 `[[0], [0,1]]` 。

第 3 题 使用 `global` 关键字在函数内部创建的变量，在函数执行结束后依然存在。

第 4 题 执行以下代码后，输出结果为 `(1, 'a')` 。

```python
pairs = [(2, 'b'), (1, 'a'), (3, 'c')]
result = min(pairs, key=lambda p: p[0])
print(result)
```

第 5 题 在 Python 中， `read()` 方法会一次性读取整个文件内容，返回字符串； `readlines()` 方法会一次性读取所有行，返回行列表。

第 6 题 以下函数的空间复杂度为 $O(1)$。

```python
def func(n):
    return [i ** 2 for i in range(n)]
```

第 7 题 在 Python 的异常处理结构中， `try` 块后面可以只跟随 `finally` 块，而不需要包含任何 `except` 块。

第 8 题 已知递推关系式为 $F(n) = F(n-3) + F(n-5)$（其中 $n \geq 5$），为了保证对所有 $n \geq 0$ 的计算都不出现负数下标，必须提供的初始值至少包括 $F(0), F(1), F(2), F(3), F(4)$ 这五个值。

第 9 题 选择排序是稳定的排序算法，因为它在每次选择最小元素时不会改变相等元素的相对顺序。

第 10 题 执行以下代码后，输出结果为 `2` 。

```python
n, m = 3, 4
grid = [[1, 3, 2, 4],
        [5, 7, 6, 8],
        [9, 0, 1, 2]]
dx = [-1, -1, -1, 0, 0, 1, 1, 1]
dy = [-1, 0, 1, -1, 1, -1, 0, 1]
count = 0
for i in range(n):
    for j in range(m):
        lower_neighbors = 0
        for k in range(8):
            x, y = i + dx[k], j + dy[k]
            if 0 <= x < n and 0 <= y < m and grid[x][y] < grid[i][j]:
                lower_neighbors += 1
        if lower_neighbors >= 5:
            count += 1
print(count)
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：扫雷
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

小杨同学正在游玩经典游戏「扫雷」，他想自己生成一个「扫雷」的地图。

小杨同学希望生成的地图大小为 n 行 m 列，一共 $n \times m$ 个区块。
区块行号为 $1, 2, \cdots, n$，列号为 $1, 2, \cdots, m$。
其中一些区块为雷区，其它区块不为雷区。

小杨同学指定了 q 个区块为雷区，而其它区块均不为雷区。小杨同学希望你帮忙计算非雷区的区块，每个区块与多少个雷区相邻?

我们定义区块相邻，当且仅当两个区块至少有一个公共顶点（也就是说对于不在地图边缘的区块，周围 8 个区块均与其相邻）。

3.1.2 输入格式

输入包含 $q+1$ 行。
第一行，三个正整数 n，m 和 q，分别表示地图行数和列数，以及雷区数量。
接下来的 q 行，每行有 2 个整数，分别表示第 i 个雷区的行号和列号。

保证输入的雷区不重复。

3.1.3 输出格式

输出 n 行，每行 m 个 **字符**（使用空格分割），对于第 i 行第 j 列，输出地图对应区块的信息:

1. 如果为雷区，输出 `*`；
2. 如果不是雷区，输出其相邻雷区数量（输出 0 到 8 中的一个数字）。

3.1.4 样例

3.1.5 输入样例 1

```
3 4 4
1 1
1 3
2 4
3 2
```

3.1.6 输出样例 1

```
* 2 * 2
2 3 3 *
1 * 2 1
```

3.1.7 输出解释 1

根据输入，在 3×4 的地图上有 4 个雷区，分别是 $(1,1)$，$(1,3)$，$(2,4)$ 和 $(3,2)$，如输出样例中 `*` 所示，其它非雷区区块的相邻雷区数量可以直观看出。

3.1.8 数据范围

$3 \le n, m \le 500, 1 \le q \le n \cdot m$。

输入的雷区必定在地图内且不重复，注意行号和列号均从 1 开始。

3.1.9 参考程序

```python
n, m, q = map(int, input().split())
# 地图。mp是二维数组，使用二维数组表示地图。mp[i][j]表示地图第i+1行第j+1列的格点
mp = [[0 for _ in range(m)] for _ in range(n)]
# 将地图上x行y列的地方放上地雷
for _ in range(q):
    x, y = map(int, input().split())
    mp[x - 1][y - 1] = '*'
# 遍历整张地图，先从行开始
for i in range(n):
    # 然后是列
    for j in range(m):
        # 如果这里是地雷，那就不管了，只考虑不是地雷的地方
        if mp[i][j] == '*':
            continue
        # 对于不是地雷的格点，判断周边是否有地雷
        # 遍历周边的8个格子，也就是i+di, j+dj
        for di in [-1, 0, 1]:
            for dj in [-1, 0, 1]:
                # 判断这个格子是否存在
                if i + di < 0 or i + di >= n or j + dj < 0 or j + dj >= m:
                    continue
                if mp[i+di][j+dj] == '*':
                    mp[i][j] += 1
# 最后把答案输出即可完成本题
for i in range(n):
    print(' '.join(map(str, mp[i])))
```

3.2 编程题 2

- **试题名称**：身高体重指数
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

一个人的身高体重指数（BMI）等于其体重（千克为单位）除以其身高（米为单位）的平方。
例如，一个体重为 50 kg，身高为 1.6 m 的人的身高体重指数为 $50\ \mathrm{kg}/1.6\ \mathrm{m}/1.6\ \mathrm{m} = 19.53125\ \mathrm{kg/m^2}$。

现在有 n 个小朋友，第 i 个小朋友的编号为 i，体重为 w_i，身高为 h_i。
请按照身高体重指数从高到低为小朋友们排序，数据保证不存在两个小朋友的身高体重指数完全相同。输出排序后小朋友的编号。

3.2.2 输入格式

输入 3 行，
第一行为一个正整数 n，表示小朋友的个数；
第二行为 n 个整数 $w_1, w_2, \cdots, w_n$ 表示小朋友们的体重，单位为 kg；
第三行为 n 个浮点数 $h_1, h_2, \cdots, h_n$ 表示小朋友们的身高，单位为 m。

3.2.3 输出格式

输出一行 n 个数，表示按照身高体重指数从高到低排序后的编号。

3.2.4 样例

3.2.5 输入样例 1

```
3
45 33 39
1.55 1.33 1.44
```

3.2.6 输出样例 1

```
3 1 2
```

3.2.7 样例解释

三个小朋友（编号依次为 `1`，`2`，`3`）的身高体重指数分别为（保留两位小数的结果）：18.73，18.66，18.81，故排序后输出的编号为 `3 1 2`。

3.2.8 数据范围

$1 \le n \le 1000, 10 \le w_i \le 100, 0.8 \le h_i \le 1.9$，$h_i$ 均恰有两位小数。

3.2.9 参考程序 1

```python
# 读取小朋友个数
n = int(input())
# 读取体重和身高
w = list(map(int, input().split()))
h = list(map(float, input().split()))

# 把小朋友们的编号和bmi指数打包在一个tuple放到列表里面
data = []
for i in range(n):
    bmi = w[i] / h[i] / h[i]
    data.append((bmi, i + 1))

# 按 BMI (即每个小列表的第 0 号元素) 降序排序
data.sort(reverse=True)

# 输出排序后的编号
for i in range(n):
    print(data[i][1], end=' ')
print()
```

3.2.10 参考程序 2

```python
# 同学总数
n = int(input())
# 同学们的体重
w = list(map(int, input().split()))
# 同学们的身高
h = list(map(float, input().split()))
ids = list(range(1, n + 1))

# 遍历i表示每一次都会将目前没确定顺序的元素中最小的元素挪到最右侧
for i in range(n):
    # j表示每一次会把没确定顺序的元素中最小的元素挪到坐标n-i
    for j in range(1, n - i):
        # 计算同学i和同学j之间的bmi指数
        bmi1 = float(w[j - 1]) / h[j - 1] / h[j - 1]
        bmi2 = float(w[j]) / h[j] / h[j]
        if bmi1 < bmi2:
            w[j - 1], w[j] = w[j], w[j - 1]
            h[j - 1], h[j] = h[j], h[j - 1]
            ids[j - 1], ids[j] = ids[j], ids[j - 1]

print(*[i for i in ids])
```