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

Python 四级

2026 年 09 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	A	C	D	C	D	A	B	C	D	A	B	A

第 1 题 第二届世界人形机器人运动会开幕式上，机器人方阵通过控制程序来完成整场表演。表演前，工程师们使用集成开发环境（IDE）编写并调试方阵的控制程序。下列关于 IDE 功能的描述，错误的是（ ）。

- A. 可以提供代码自动补全，提高编程效率
- B. 可以提供辅助信息，帮助排查程序逻辑错误
- C. 可以直接控制机器人电机转动
- D. 支持语法高亮，使代码更易阅读

第 2 题 计算机 CPU 中的高速缓冲存储器（Cache）主要为提升系统性能而设计。下列关于 Cache 的说法中正确的是（ ）。

- A. Cache 的容量比主存大，速度比主存快
- B. Cache 通常用于暂存 CPU 频繁访问的数据
- C. Cache 中的数据断电后不会丢失，属于非易失性存储器
- D. Cache 的访问速度比 CPU 寄存器还要快

第 3 题 执行下面 Python 代码后，输出的结果是？（ ）

```
original = [{'key': 'value'}, [4, 5, 6]]
new = original.copy()
new[0]['key'] = 'new_value'
new[1].append(7)
print(original[0]['key'], original[1][-1])
```

- A. `value 6`
- B. `new_value 7`
- C. `value 7`
- D. `new_value 6`

第 4 题 执行下面 Python 代码后，输出的结果是？（ ）

```
nested = [[1, 2], [3, 4], [5, 6]]
flat = [num for sublist in nested for num in sublist if num % 2 == 0]
print(flat)
```

- A. `[2, 4, 6]`

- [ ] **B.** `[1, 3, 5]`
- [ ] **C.** `[1, 2, 3, 4, 5, 6]`
- [ ] **D.** `[2, 4]`

第 5 题 执行下面 Python 代码后，输出的结果是？（　　）

```python
def func(d):
    d = {'new': 'value'}
    return d


data = {'old': 'data'}
func(data)
print(data)
```

- [ ] **A.** `{}`
- [ ] **B.** `{'new': 'value'}`
- [ ] **C.** `{'old': 'data'}`
- [ ] **D.** 报错

第 6 题 关于以下代码，说法正确的是？（　　）

```python
def process(a: int, b: str = "default") -> str:
    return b * a
```

- [ ] **A.** 调用函数时必须为形参 `a` 传递一个整数值
- [ ] **B.** 调用函数时不能省略对形参 `b` 的传参
- [ ] **C.** 函数返回值必须是字符串类型
- [ ] **D.** 参数类型注解在运行时不会强制检查类型

第 7 题 执行下面 Python 代码后，输入 `150 165 135 133` 输出的结果是？（　　）

```python
H1, H2, H3, H4 = map(int, input().split())
print(min([H2, H3, H4], key=lambda x: (abs(x - H1), x)))
```

- [ ] **A.** `150`
- [ ] **B.** `165`
- [ ] **C.** `135`
- [ ] **D.** `133`

第 8 题 以下哪种文件打开模式允许同时读写文件，且文件指针从文件开头开始？（　　）

- [ ] **A.** `'r'`
- [ ] **B.** `'w'`
- [ ] **C.** `'a'`
- [ ] **D.** `'r+'`

第 9 题 以下代码的时间复杂度是？（　　）

```python
n = int(input())
f = [0] * (n + 1)
f[0] = 0
f[1] = 1
for i in range(2, n + 1):
    f[i] = f[i - 1] + f[i - 2]
```

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

第 10 题 关于 `try-except-else-finally` 结构，以下说法正确的是？（　　）

- [ ] **A.** 如果 `try` 块中有 `return` 语句，`finally` 块将不会执行
- [ ] **B.** `finally` 块通常用于释放资源，如关闭文件
- [ ] **C.** 一个 `try` 块只能对应一个 `except` 块
- [ ] **D.** 如果 `try` 块中发生了异常，那么 `else` 块也会执行

第 11 题 执行下面 Python 代码后，输出的结果是？（　　）

```python
a = [1, 1]
for i in range(2, 10):
    a.append(a[i - 1] + a[i - 2])
print(a[8])
```

- [ ] **A.** `89`
- [ ] **B.** `55`
- [ ] **C.** `34`
- [ ] **D.** `21`

第 12 题 执行下面 Python 代码后，输出的结果是？（　　）

```python
students = [("Alice", "B", 25), ("Bob", "A", 23), ("Cathy", "A", 22)]
sorted_students = sorted(students, key=lambda x: (x[1], x[2]))
print([s[0] for s in sorted_students])
```

- [ ] **A.** `["Cathy", "Alice", "Bob"]`
- [ ] **B.** `["Bob", "Cathy", "Alice"]`
- [ ] **C.** `["Alice", "Bob", "Cathy"]`
- [ ] **D.** `["Cathy", "Bob", "Alice"]`

第 13 题 以下代码实现了根据交通灯颜色返回动作（停、等、行）的功能，以下哪个选项是正确的？（　　）

```python
def traffic_action(color):
    actions = {"红": "停", "黄": "等", "绿": "行"}
    if color in actions:
        return _____①_____
    else:
        return "无效颜色"


color = input("输入交通灯颜色: ")
action = traffic_action(color)
print(f"颜色 {color} 对应的动作是: {_____②_____}")
```

- [ ] **A.** ① `actions[color]` ② `action`
- [ ] **B.** ① `actions[color]` ② `color`
- [ ] **C.** ① `actions.get(color)` ② `actions`
- [ ] **D.** ① `actions.get(color, "无效颜色")` ② `color`

第 14 题 文件 `text.txt` 的内容如下：

```
Python is a programming language. Python is easy to learn.
Many people use Python for data analysis and web development.
Python has a large community and many libraries.
```

以下代码实现了统计文件中单词 `Python` 出现的次数的功能，以下哪个选项是正确的？（　　）。

```python
count = 0
with open("text.txt", "r") as f:
    content = f._____①_____
    words = content._____②_____
    for word in words:
        if word == "Python":
            count += 1
print(f"Python出现次数: {count}")
```

- [ ] **A.** ① `readlines()` ② `split()`
- [ ] **B.** ① `read()` ② `split()`
- [ ] **C.** ① `read()` ② `strip()`
- [ ] **D.** ① `readlines()` ② `strip()`

第 15 题 以下代码实现了给定一个 $n \times m$ 的整数矩阵，能找出所有 2×2 子矩阵中，元素最大平均值的功能，以下哪个选项是正确的？（　　）

```python
n, m = map(int, input().split())
matrix = [list(map(int, input().split())) for _ in range(n)]

max_avg = float("-inf")
for i in range(_____①_____):
    for j in range(_____②_____):
        # 计算当前2×2子矩阵的和
        total = matrix[i][j] + matrix[i][j + 1] + matrix[i + 1][j] + matrix[i + 1][j + 1]
        avg = total / 4.0
        if avg > max_avg:
            max_avg = avg

print(f"{max_avg:.2f}")
```

- [ ] **A.** ① `n-1` ，② `m-1`

- [ ] **B.** ① `n-2` ， ② `m-2`
- [ ] **C.** ① `n` ， ② `m`
- [ ] **D.** ① `n-1` ， ② `m-2`

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 Python 程序执行 `input()` 函数时通常是使用键盘等输入设备来输入某个变量的值。

第 2 题 执行下面 Python 代码后，输出结果为 `[1, 2]` 。

```
a = [1, 2]
b = (a,)
a.append(3)
print(b[0])
```

第 3 题 执行下面 Python 代码后，输出结果为 `[0,1,4] [0,1,4]` 。

```
gen = (i ** 2 for i in range(3))
print(list(gen), list(gen))
```

第 4 题 执行下面 Python 代码后，输出结果为 `20` 。

```
num = 5

def test():
    num = 10
    global num
    num = 20

test()
print(num)
```

第 5 题 在 Python 中，如果一个函数内部定义了一个嵌套函数，那么嵌套函数只能在外部函数内部被调用，不能在外部函数之外直接调用。但是嵌套函数若被作为返回值、赋给变量等，外部函数之外也可以通过该引用调用。

第 6 题 执行下面 Python 代码后， `oldest` 的值为 `{'name': 'Alice', 'age': 30}` 。

```
data = [{'name': 'Alice', 'age': 30}, {'name': 'Bob', 'age': 25}]
oldest = max(data, key=lambda d: d['age'])
```

第 7 题 在 Python 中， `f.readlines()` 与 `list(f)` 的作用完全相同，都会将文件的所有行读入一个列表，并保留每行已有的换行符，即下列两段代码的输出结果相同。

```
with open("test.txt", "r") as f:
    print(f.readlines())
```

```
with open("test.txt", "r") as f:
    print(list(f))
```

第 8 题 执行下面 Python 代码会抛出 `KeyError` 异常。

```
print({}.get('key'))
```

第 9 题 对一组数据 `[5, 2, 6, 4, 8, 1, 7, 3]` 使用选择排序的方法按从小到大的顺序进行排序，则第 2 轮排序过后的结果是 `[1, 2, 6, 4, 8, 5, 7, 3]`。

第 10 题 以下 Python 代码使用递推算法实现了计算正整数 `n` 的二进制表示中包含 1 个数的功能。

```
def popcount(n):
    cnt = [0] * (n + 1)
    for i in range(1, n + 1):
        cnt[i] = cnt[i >> 1] + (i & 1)
    return cnt[n]
```

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：新汉诺塔
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

汉诺塔问题是最经典的递推问题之一：

有三个可以放圆盘柱子，编号为 A、B 和 C。
开始时柱子 A 上套着 n 个圆盘，它们从上到下按照从小到大的顺序排列。
我们的任务是要把这 n 个圆盘移到柱子 C 上，并保持它们的原有顺序不变。
在移动圆盘的过程中，需要遵守以下规则：

1. 圆盘只能从一根柱子顶部拿出，从另一根柱子顶部放入。
2. 每次只能移动一个圆盘。
3. 小圆盘必须时刻位于大圆盘之上。

小杨在学习了汉诺塔问题后，决定添加一个新规则：

4. 每一次移动，圆盘只能从 A 移动到 B，从 B 移动到 C，或者从 C 移动到 A；其它移动是不允许的。

在新规则下，给定圆盘数量 n，试问最少移动步数是多少？

3.1.2 输入格式

输入一个正整数 n，表示圆盘的数量。

3.1.3 输出格式

输出一个整数，表示在新规则下将 n 个圆盘从 A 移动到 C 所需的最少移动步数。

3.1.4 样例

3.1.5 输入样例 1

```
2
```

3.1.6 输出样例 1

```
7
```

3.1.7 样例解释 1

以下步骤是最佳的（编号为 1 的是小盘，为 2 的是大盘）：

1. 将 1 从 A 移动到 B;
2. 将 1 从 B 移动到 C;
3. 将 2 从 A 移动到 B;
4. 将 1 从 C 移动到 A;
5. 将 2 从 B 移动到 C;
6. 将 1 从 A 移动到 B;
7. 将 1 从 B 移动到 C。

可以证明没有更少步骤可以完成这个任务。

3.1.8 输入样例 2

```
3
```

3.1.9 输出样例 2

```
21
```

3.1.10 数据范围

对于所有数据，$n \leq 20$

3.1.11 参考程序

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

# f[i]: 把i个盘子往右挪一格的操作数量
# g[i]: 把i个盘子往左挪一格的操作数量
f = [0] * (n + 1)
g = [0] * (n + 1)

for i in range(1, n + 1):
    # 顶上的i-1个盘子往左挪一格，把最大盘子往右，然后再把剩下的盘子往左
    f[i] = 2 * g[i - 1] + 1
    # 顶上的i-1个盘子往左挪一格(g[i-1])，最大的盘子挪到B(1)，顶上i-1个盘子挪回A(f(i-1)),最大盘子挪到C(1)，顶上i-1个盘子再挪到C(g[i-1])
    g[i] = 2 * g[i - 1] + f[i - 1] + 2

print(g[n])
```

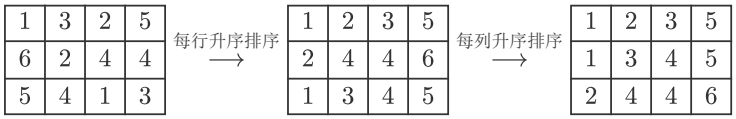
3.2 编程题 2

- **试题名称**：有序网格
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

小 A 有一个 n 行 m 列格子组成的二维网格，从上到下依次是第 1 行到第 n 行，从左到右依次是第 1 列到第 m 列。每个格子里有一个数字，第 i 行第 j 列的格子里的数字是 $a_{i,j}$。

小 A 想让二维网格变得有序，因此他先对每一行从左到右按升序排序，再对每一列从上到下按升序排序。以下是一个先完成行排序再完成列排序的例子：



小 A 想知道二维网格经过以上排序后的结果。你能编写程序帮助他吗?

3.2.2 输入格式

第一行，两个正整数 n, m，分别二维网格的行数与列数。

接下来 n 行，每行 m 个整数 $a_{i,1}, \ldots, a_{i,m}$，表示二维网格种的数字。

3.2.3 输出格式

输出 n 行，每行 m 个整数，表示二维网格先完成行排序再完成列排序后的结果。

3.2.4 样例

3.2.5 输入样例 1

```
3 2
6 5
4 3
2 1
```

3.2.6 输出样例 1

```
1 2
3 4
5 6
```

3.2.7 输入样例 2

```
3 4
1 3 2 5
6 2 4 4
5 4 1 3
```

3.2.8 输出样例 2

```
1 2 3 5
1 3 4 5
2 4 4 6
```

3.2.9 数据范围

对于所有测试点，保证 $2 \leq n \leq 10$，$2 \leq m \leq 10$，$1 \leq a_{i,j} \leq 100$。

3.2.10 参考程序

```python
n, m = map(int, input().split())
a = [list(map(int, input().split())) for _ in range(n)]
# 每行进行冒泡排序
def sort_row(i):
    for k in range(m - 1):
        for j in range(m - 1 - k):
            if a[i][j] > a[i][j + 1]:
                a[i][j], a[i][j + 1] = a[i][j + 1], a[i][j]
# 每列进行冒泡排序
def sort_col(j):
    for k in range(n - 1):
        for i in range(n - 1 - k):
            if a[i][j] > a[i + 1][j]:
                a[i][j], a[i + 1][j] = a[i + 1][j], a[i][j]

for i in range(n):
    sort_row(i)
for j in range(m):
    sort_col(j)
for row in a:
    print(*row)
```