全国青少年信息素养大赛-2026-小高组-算法-Py-星火-复赛-模拟题-01-答案与解析

1	2	3	4	5	6	7	8	9	10
A	C	A	B	C	A,B,C	A,B,D	B,C	A,C,D	A,B,C,D

一、单选题

（1）炮兵连将 17 发炮弹平分给 5 个战斗班。执行以下代码后，每班分到几发、剩几发、精确值是多少？（ ）

```
shells = 17
teams = 5
print(shells // teams, shells % teams)
print(shells / teams)
```

A. `3 2` 然后 `3.4`

B. `3.4 2` 然后 `3.4`

C. `3 3` 然后 `3.4`

D. `3 2` 然后 `3`

【正确答案】

A

【题目解析】

✅ 正确答案

A（`3 2` 然后 `3.4`）

💡 核心考点

整除 `//`、取余 `%` 与真除 `/` 的区别。

🔎 执行追踪

$17//5=3$，$17\%5=2$，$17/5=3.4$。

⚠️ 易错提醒

`/` 总是返回浮点数，`//` 才是向下取整。

（2）后勤班清点了一批物资重量数据，需要按从重到轻排序后找出第二重的物资。执行以下代码后，输出结果为（ ）。

```
data = [5, 2, 8, 1, 9]
data.sort(reverse=True)
print(data[1])
```

A. `1`
B. `2`
C. `8`
D. `9`

【正确答案】
C

【题目解析】
✅ 正确答案
C（`8`）

💡 核心考点
列表降序排序与索引访问。

🔎 执行追踪
排序后列表为 $[9,8,5,2,1]$，`data[1]` 即第二个元素 8。

⚠️ 易错提醒
索引从 0 开始，`data[1]` 是第二重而非最重。

（3）情报科截获了一串密文 `'ABCDEFG'`，需要分别用"倒序读取"和"跳位读取"两种方式解码。执行以下代码后，输出结果为（ ）。

```
s = 'ABCDEFG'
print(s[::-1])
print(s[::2])
```

A. `GFEDCBA` 和 `ACEG`
B. `G` 和 `ACEG`

C. `GFEDCBA` 和 `ABCDEFG`

D. `ABCDEFG` 和 `ACEG`

【正确答案】

A

【题目解析】

✅ 正确答案

A（`GFEDCBA` 和 `ACEG`）

💡 核心考点

字符串切片 `[::step]` 的步长用法。

🔎 执行追踪

`s[::-1]` 步长为 -1，倒序输出 `GFEDCBA`；`s[::2]` 步长为 2，从索引 0 起隔位取字符，得 `ACEG`。

⚠️ 易错提醒

负步长表示反向读取，不是只取最后一个字符。

（4）指挥部派出 3 个侦察组，每组连续执行 4 天侦察任务。执行以下代码后，总共执行了多少次侦察任务？（）

```
count = 0
for i in range(3):
    for j in range(4):
        count += 1
print(count)
```

A. `7`

B. `12`

C. `3`

D. `4`

【正确答案】

B

【题目解析】

✅ 正确答案

B（`12`）

💡 核心考点

双重 `for` 循环次数计算。

🔎 执行追踪

外层执行 3 次，内层执行 4 次，共 $3 \times 4 = 12$ 次。

⚠️ 易错提醒

嵌套循环总次数是相乘而非相加。

（5）情报员尝试将截获的密文片段转换为数字。如果密文中包含非数字字符，转换将失败。执行以下代码后，输出结果为（ ）。

```
try:
    a = int('abc')
    print(a)
except ValueError:
    print('解密失败')
```

A. `abc`

B. `0`

C. `解密失败`

D. 程序报错并终止

【正确答案】

C

【题目解析】

✅ 正确答案

C（`解密失败`）

💡 核心考点

考察 `try-except` 异常捕获机制。

🔎 执行追踪

`int('abc')` 抛出 `ValueError`，被 `except` 捕获，执行 `print('解密失败')`。

⚠️ 易错提醒

异常被捕获后程序不会终止，会继续执行 `except` 块中的语句。

二、多选题

（6）发报员用函数格式化电报内容。执行以下代码，结果正确的有（ ）。

```
def report(name, msg='安全抵达'):
    return f'{msg}, {name}! '

a = report('一排')
b = report('二排', '发现敌情')
c = report(msg='急需弹药', name='三排')
```

A. `a` 的值是 `'安全抵达，一排！'`

B. `b` 的值是 `'发现敌情，二排！'`

C. `c` 的值是 `'急需弹药，三排！'`

D. `report('四排', msg='一切正常')` 会报错（位置参数不能在关键字参数之后）

【正确答案】

A,B,C

【题目解析】

✅ 正确答案

A、B、C

💡 核心考点

函数默认参数、关键字参数调用规则。

🔎 执行追踪

A 使用默认值得到 `'安全抵达，一排！'`；B 覆盖默认值得到 `'发现敌情，二排！'`；C 使用关键字传参得到 `'急需弹药，三排！'`；D 中位置参数 `'四排'` 在关键字参数之前，合法不报错。

⚠️ 易错提醒

只要位置参数在关键字参数之前就合法，D 项描述错误。

（7）关于军需物资清单（列表）和部队编制表（元组），以下说法正确的有（ ）。

A. 列表是可变的，可以修改、增加、删除元素

B. 元组是不可变的，创建后不能修改其中的元素

C. 元组可以使用 `append()` 方法添加元素

D. 列表和元组都支持索引和切片操作

【正确答案】

A,B,D

【题目解析】

✅ 正确答案

A、B、D

💡 核心考点

列表与元组的特性区别。

🔎 执行追踪

列表（list）可变，支持增删改；元组（tuple）不可变，创建后不能修改；两者都是序列类型，均支持索引和切片。

⚠️ 易错提醒

元组没有 `append()` 方法，只有列表才有，故 C 错误。

（8）指挥部档案管理中，已知 `files = {'a': 1, 'b': 2}`，以下操作会导致错误的有（ ）。

A. `files['c'] = 3`

B. `files[[1]] = 'a'`

C. `print(files['d'])`

D. `files['a'] = 10`

【正确答案】

B,C

【题目解析】

✅ 正确答案

B、C

💡 核心考点

字典的键类型限制与键不存在的访问异常。

🔎 执行追踪

A 新增键值对合法；B 列表 `[1]` 不可哈希，报 `TypeError`；C 键 `'d'` 不存在，报 `KeyError`；D 修改已有键值合法。

⚠️ 易错提醒

字典的键必须是不可变（可哈希）类型，列表不能作为键；访问不存在的键会抛出 `KeyError`，需用 `get()` 避免。

（9）关于各连队内部使用的兵力统计函数（作用域），以下说法正确的有（ ）。

A. 函数内部定义的变量默认是局部变量，函数外部无法访问
B. 在函数外部定义的变量在函数内部可以直接修改
C. 使用 `global` 关键字可以在函数内部修改全局变量
D. 不同函数内部可以有同名的局部变量，互不影响

【正确答案】
A,C,D

【题目解析】
✅ 正确答案
A、C、D

💡 核心考点
Python 函数作用域（LEGB 规则）与 `global` 关键字。

🔎 执行追踪
A 正确，函数内变量默认为局部变量；B 错误，函数外的变量在函数内只能读取，不能直接赋值修改（赋值会创建新的局部变量）；C 正确，`global` 声明后可在函数内修改全局变量；D 正确，不同函数的局部变量作用域相互独立。

⚠️ 易错提醒
注意区分"访问"和"修改"——不使用 `global` 时，函数内可以读取全局变量的值，但赋值操作会被解释为创建同名局部变量。

（10）突击连在评估突袭的可行性，运行以下战术条件判断代码：

```
x, y = 10, 3
res1 = x > 5 and y < 5
res2 = not (x == y)
res3 = x % y == 1 and x // y == 3
```

结果正确的有（ ）。

A. `res1` 的值是 `True`
B. `res2` 的值是 `True`
C. `res3` 的值是 `True`
D. `x >= 10 or y > 5` 的值是 `True`

【正确答案】

A,B,C,D

【题目解析】

✅ 正确答案

A、B、C、D

💡 核心考点

考察布尔运算符 `and` 、 `or` 、 `not` 与算术运算符的组合判断。

🔎 执行追踪

`res1` ： $10 > 5$ 且 $3 < 5$ ，为 `True` ； `res2` ： $10e3$ ， `not False` 为 `True` ； `res3` ： $10\%3 = 1$ 且 $10//3 = 3$ ，为 `True` ；D 项 $x \geq 10$ 成立， `or` 短路为 `True` 。

⚠️ 易错提醒

注意 `and` 要求两边同真， `or` 只需一边真； `//` 是整除， `%` 是取余，别混淆。

三、编程题

（11）Y3682 发放军饷

【题目描述】

后勤处按以下规则给新兵营发放军饷（铜板）：

- 第 1 天，每人发 1 个铜板；
- 第 2 天和第 3 天，每人每天发 2 个铜板；
- 第 4 、 5 、 6 天，每人每天发 3 个铜板；
- 第 7 、 8 、 9 、 10 天，每人每天发 4 个铜板……
- 以此类推：当连续 N 天每人每天发 N 个铜板后，接下来的连续 $N+1$ 天里，每人每天发 $N+1$ 个铜板。

  请你编写程序：输入总天数 K ，计算前 K 天里每人一共领到了多少个铜板。

【输入描述】

一个正整数 K （ $1 \leq K \leq 10000$ ），表示发放军饷的天数。

【输出描述】

一个正整数，表示每人累计领到的铜板总数。

【输入样例 1】

```
6
```

【输入样例 2】

```
1000
```

【输出样例 1】

```
14
```

【输出样例 2】

```
29820
```

（12）Y3683 军需物资编号校验

【题目描述】

每一批运往前线的军需物资都有一个标准编号，格式为：`x-xxx-xxxxx-x`，其中符号 `-` 是分隔符，最后一位是校验码。编号共包含 9 位数字、1 位校验码和 3 个分隔符。

校验码的计算方法如下：将编号中的 9 位数字（忽略分隔符）从左到右依次乘以 $1, 2, 3, \cdots, 9$，求和后用结果对 11 取余。如果余数为 10，则校验码为大写字母 `X`；否则校验码为余数本身。

请你编写程序：输入一个物资编号，判断其校验码是否正确。如果正确，输出 `Right`；如果错误，输出修正后的正确编号。

【输入描述】

一个字符串，表示物资编号（保证格式为 `x-xxx-xxxxx-x`）。

【输出描述】

如果校验码正确，输出 `Right`；否则输出正确的完整编号（含分隔符 `-`）。

【输入样例 1】

```
0-670-82162-4
```

【输入样例 2】

```
0-670-82162-0
```

【输出样例 1】

```
Right
```

【输出样例 2】

```
0-670-82162-4
```

（13）Y3684 敌占区物价连续下跌监测

【题目描述】

后勤部侦察员记录了敌占区连续 n 天的粮食采购价格。由于敌军加强封锁，市场价格持续波动，如果某天的价格比前一天更低，则记为“下跌”。

请你编写程序，找出这 n 天中最长的一段**连续下跌**的天数（即连续多天，每一天的价格都比前一天低），帮助指挥部判断敌军封锁力度是否在持续加大。

【输入描述】

第一行，一个整数 n （ $1 \leq n \leq 100$ ），表示记录的天数。
第二行， n 个整数（空格分隔），表示每天的价格。

【输出描述】

一行，一个整数，表示最长连续下跌的天数。

【输入样例 1】

```
9
68 65 62 64 59 56 53 58 52
```

【输出样例 1】

```
4
```

（14）Y3685 龙虎斗

【题目描述】

前线指挥部正在进行一场 龙虎斗 兵棋推演。推演棋盘是一条线段，线段上有 n 个兵营（自左至右编号 $1 \sim n$ ），相邻编号的兵营之间相隔 1 厘米。第 i 号兵营里有 c_i 位工兵。

红军指挥员在左侧，代表 `龙`；蓝军指挥员在右侧，代表 `虎`。他们以 m 号兵营作为分界，靠左的工兵属于龙势力，靠右的工兵属于虎势力，而第 m 号兵营中的工兵很纠结，他们不属于任何一方。

一个兵营的气势为：该兵营中的工兵数 $\times$ 该兵营到 m 号兵营的距离；参与推演一方的势力定义为：属于这一方所有兵营的气势之和。

推演过程中，某一刻天降援兵，共有 s_1 位工兵突然出现在了 p_1 号兵营。作为指挥部的参谋，你知道如果龙虎双方气势差距太悬殊，推演就无法继续。为了让推演继续，你需要选择一个兵营 p_2，并将你手里的 s_2 位援兵全部派往兵营 p_2，使得双方气势差距尽可能小。

注意：你手中的援兵落在哪个兵营，就和该兵营中其他工兵有相同的势力归属（如果落在 m 号兵营，则不属于任何势力）。

【输入描述】

第一行，一个正整数 n，代表兵营的数量。

第二行，n 个正整数，相邻两数之间以一个空格分隔，第 i 个正整数代表编号为 i 的兵营中起始时的工兵数量 c_i。

第三行，四个正整数 m、p_1、s_1、s_2，相邻两数间以一个空格分隔。

【输出描述】

一行，一个正整数 p_2，表示你选择的援兵投放兵营编号。如果存在多个编号同时满足最优，取最小的编号。

【输入样例 1】

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

【输入样例 2】

```
6
1 1 1 1 1 16
5 4 1 1
```

【输出样例 1】

```
2
```

【输出样例 2】

```
1
```