**ACM模式输入输出总结**

1. **输入字符串**

|  |
| --- |
| Python abc123 |

|  |
| --- |
| Python s = input() |

2. **输入数字**

|  |
| --- |
| Python 5 |

|  |
| --- |
| Python num = int(input()) |

3. **输入字符串，转换为字符串列表**

用空格隔开

|  |
| --- |
| Python aaa bbb ccc |

|  |
| --- |
| Python lst = input().split() # ["aaa", "bbb", "ccc"] |

用逗号隔开

|  |
| --- |
| Python aaa,bbb,ccc |

|  |
| --- |
| Python lst = input().split(",") # ["aaa", "bbb", "ccc"] |

4. **输入字符串，转换为数字列表**

用空格隔开

|  |
| --- |
| Python 0 1 2 |

|  |
| --- |
| Python nums = list(map(int, input().split())) # input().split()会得到["0", "1", "2"] # 用map()可以把["0", "1", "2"]中的每一个元素映射地使用int()， # 结果就可以得到[0, 1, 2] |

用逗号隔开

|  |
| --- |
| Python 0,1,2 |

|  |
| --- |
| Python lst = list(map(int, input().split(","))) |

5. **输入二维列表**

字符串类型

|  |
| --- |
| Python 5 4 0000 0001 1114 2223 3344 |

|  |
| --- |
| Python n, m = map(int, input().split()) # 先输入 5行4列 grid = list() #  for i in range(n): # 遍历n行  grid.append(list(input())) # 然后输入每行  # 或者grid.append(input()) |

数字类型

|  |
| --- |
| Python 5 4 0 0 0 0  0 0 0 1  1 1 1 2 2 2 2 3  3 3 4 4 |

|  |
| --- |
| Python n, m = map(int, input().split(",")) grid = list() for i in range(n): # 遍历n行  grid.append(list(map(int, input().split()))) |

6. **输入次数未知（极少使用）**

|  |
| --- |
| Python 0 1 2 3 |

|  |
| --- |
| Python # try-except异常处理语句可以用来解决输入次数未知的问题 # 当try中的语句块没有出现错误、可以正常运行时时，执行try中的语句块，否则执行except下的语句块 while(True):  try:  n = int(input())  except:  break |

7. **输出**

一般来说直接print()即可，不可以写return。

|  |
| --- |
| Python print(ans) |

8. **Main函数**

|  |
| --- |
| Python #定义节点 class ListNode():  def \_\_init\_\_(self, x):  self.val = x  self.next = None #将传入的数组转化为链表 def create\_linked\_list(arr):  head = ListNode(arr[0])  cur = head  for i in range(1, len(arr)):  cur.next = ListNode(arr[i])  cur = cur.next  return head #传入链表头节点，以数组形式返回 def print\_linked\_list(head):  cur = head  res = []  while cur:  res.append(cur.val)  cur = cur.next  return res class Solution():  def mergeTwoLists(self, l1, l2):  pre = ListNode(0)  head = pre  while l1 and l2:  if l1.val >= l2.val:  pre.next = l2  l2 = l2.next  else:  pre.next = l1  l1 = l1.next  pre = pre.next  pre.next = l1 if l1 else l2  return head.next if \_\_name\_\_ == "\_\_main\_\_":  head1 = create\_linked\_list([1, 2, 4])  head2 = create\_linked\_list([1, 3, 4])  solution = Solution()  sorted\_lists = solution.mergeTwoLists(head1, head2)  print(print\_linked\_list(sorted\_lists)) #输出：[1, 1, 2, 3, 4, 4] |