**2-1.打开c:\ecnu\_ks\root\fillblank2-1.py程序源文件，按下列要求对文件中的下划线进行填空（用填空内容代替下划线），调试并运行，完成程序。将程序源代码另存为“学号-姓名-填空2-1.py”,存放在c:\ecnu\_ks\root中。**

要求：某次选举活动中有5个候选人，其代号分别用1~5表示；假设有若干选民，每个选民只能选一个候选人，即每张选票上出现的数字只能是1~5间的某一个数字。

每张选票上所投候选人的代号由键盘输入，当输入完所有选票后用-1作为终止数据输入的标志。要求统计输出每个候选人的得票数。

**2-2 打开c:\ecnu\_ks\root\fillblank2-2.py程序源文件，按下列要求对文件中的下划线进行填空（用填空内容代替下划线），调试并运行，完成程序。将程序源代码另存为“学号-姓名-填空2-2.py”,存放在c:\ecnu\_ks\root中。**

要求：字典中已记录学号001-005的同学分数，完善程序，实现按学号修改分数、按学号查看分数、计算总分的功能。

**五处下划线部分如下所示**。

dic={'001':80,'002':90,'003':100,'004':70,'005':60}  
while True:  
  print("请选择功能：\n1: 修改分数\n2：查看分数\n3：计算总分\n0:退出")  
  c=input()  
  if \_\_\_\_\_\_\_\_\_\_:  
    stuno=input("请输入要修改分数的学号（按回车结束）：")  
    if stuno in dic:  
      score=int(input("请输入分数："))  
      dic[stuno]=score  
      print("该学生分数已修改。")  
    else:  
      print("库中未找到这个学生")  
  elif c=="2":  
    stuno=input("请输入要查询的学号（按回车结束）：")  
    if stuno in dic:  
      print("%s的分数是%d"%(\_\_\_\_\_\_\_\_\_\_\_))  
    else:  
      print("库中未找到这个学生")  
  elif c=="3":  
    s=\_\_\_\_\_\_\_  
    for x in dic.keys():  
      s=s+dic[x]  
    print(s)  
  elif c=="0":  
    \_\_\_\_\_\_\_\_\_  
  \_\_\_\_\_\_\_\_\_:  
    print("输入有误！")

**程序运行示例如下**：  
  
>>>  
请选择功能：  
1: 修改分数  
2：查看分数  
3：计算总分  
0:退出  
1  
请输入要修改分数的学号（按回车结束）：002  
请输入分数：100  
该学生分数已修改。  
请选择功能：  
1: 修改分数  
2：查看分数  
3：计算总分  
0:退出  
2  
请输入要查询的学号（按回车结束）：002  
002的分数是100  
请选择功能：  
1: 修改分数  
2：查看分数  
3：计算总分  
0:退出  
3  
410  
请选择功能：  
1: 修改分数  
2：查看分数  
3：计算总分  
0:退出  
0  
>>>

3 在Python环境下打开c:\ecnu\_ks\root\**fillblank2-3.py**程序源文件，并将其另存为“**ans23-学号-姓名.py**”，存放在**c:\ecnu\_ks\root**中。 按下列要求对文件中的下划线进行填空（用填空内容代替下划线，除此之外，不能改动其它任何地方），调试并运行之。

1. 程序功能： 求1000以内所有完全数，并输出。
2. 完全数，又称完美数或完备数，是一些特殊的自然数。它所有的真因子（即除了自身以外的约数）的和，  
   恰好等于它本身。如果一个数恰好等于它的因子之和，则称该数为“完全数”。例如 6 = 1 + 2 + 3，则6是完全数。
3. 原始程序如下图所示：

|  |
| --- |
| def IsProfNum(\_\_\_\_):  lst = []  for x in range(1,num):  if num % x \_\_\_\_0:  lst.append(x)  if sum(lst) == num :  \_\_\_\_\_\_  return False  proflst = []  for n in range(1,1000):  if \_\_\_\_\_\_\_\_:  proflst.\_\_\_\_\_\_\_(n)  print ('1000以内的所有完全数有：', proflst) |

1. 程序运行结果示例如下图所示：

|  |
| --- |
| >>>  1000以内的所有完全数有： [6, 28, 496]  >>> |

4 在Python环境下打开c:\ecnu\_ks\root\**fillblank2-4.py**程序源文件，并将其另存为“**ans24-学号-姓名.py**”，存放在**c:\ecnu\_ks\root**中。 按下列要求对文件中的下划线进行填空（用填空内容代替下划线，除此之外，不能改动其它任何地方），调试并运行之。

1. 程序的功能：统计列表中2020年不同年龄的人数，按年龄升序排列输出。。
2. ‬
3. 主函数**main**中存放了一组身份证号码的列表对象**IDS**，调用函数**CountAge**，获取统计结果列表，显示并输出结果。
4. 原始程序如下图所示：

|  |
| --- |
| def CountAge(\_\_\_\_\_\_\_\_):  d={}  for sid in idCards:  age=2020-\_\_\_\_\_\_\_\_\_\_  if age in d:  d[age]+=1  else:  d[age]=1  L=list(\_\_\_\_\_\_\_\_\_\_)  L.sort(key=lambda x:x[0])  return \_\_\_\_\_\_  def main():  IDS=['310107199401231242','31010719940502913X','310107199702012190','310107199104016896',  '310107199909029846','310107199608013285','310107199612122345','310107199908017384',  '31010719930302774X','310107199703030489','310107199306030821','310107199703018594']  Lcount=\_\_\_\_\_\_\_\_\_\_\_\_  for age,count in Lcount:  print("{}岁\t{}人".format(age,count))  main() |

1. 程序运行结果如下图所示：

|  |
| --- |
| >>>  21岁 2人  23岁 3人  24岁 2人  26岁 2人  27岁 2人  29岁 1人  >>> |

1. 在Python环境下打开c:\ecnu\_ks\root\**fillblank2-5.py**程序源文件，并将其另存为“**ans25-学号-姓名.py**”，存放在**c:\ecnu\_ks\root**中。 按下列要求对文件中的下划线进行填空（用填空内容代替下划线，除此之外，不能改动其它任何地方），调试并运行之。
   1. 程序功能：求10000以内所有玫瑰花数，并输出。
   2. 玫瑰花数是一个四位数，若该数等于它每一位数字的4次方之和，则该四位数为玫瑰花数。 例如：14+64+34+44= 1634，则1634为玫瑰花数。
   3. 函数**IsRoseNum(num)**用来判断一个自然数是否是玫瑰花数，若是返回True，否则返回False。
   4. 原始程序如下图所示：

|  |
| --- |
| def IsRoseNum(num):  sNum = str(num)  if len(sNum)!=4:return \_\_\_\_\_\_\_\_  summ = 0  for x in sNum:  summ += int(x) \_\_\_\_\_ 4  if summ == num :  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  return False  result = []  for n in range(0,10000):  if \_\_\_\_\_\_\_\_\_\_:  result.append(\_\_\_\_\_)  print ('10000以内的所有玫瑰花数有：', result) |

* 1. 程序运行结果示例如下图所示：

|  |
| --- |
| >>>  10000以内的所有玫瑰花数有： [1634, 8208, 9474]  >>> |

1. 在Python环境下打开c:\ecnu\_ks\root\**fillblank2-6.py**程序源文件，并将其另存为“**ans26-学号-姓名.py**”，存放在**c:\ecnu\_ks\root**中。 按下列要求对文件中的下划线进行填空（用填空内容代替下划线，除此之外，不能改动其它任何地方），调试并运行之。
   1. 程序的功能：统计身份证号列表中不同月份的人数，按月份升序排列输出。
   2. 编写函数**CountMonth**,参数为身份证号码列表，根据每个人的身份证号码获得这个人的出生月份，统计不同月份的人数，按月份从小到大排序，返回统计列表，格式为**[(月份,人数),(月份,人数),......]**。
   3. 主函数main中存放了一组身份证号码的列表对象IDS，调用函数**CountMonth**，获取统计结果列表，显示并输出结果。
   4. 原始程序如下图所示：

|  |
| --- |
| def CountMonth(\_\_\_\_\_\_\_\_\_):  d={}  for sid in idCards:  month=\_\_\_\_\_\_\_\_\_\_\_  if month in d:  d[month]+=1  else:  d[month]=1  L=list(\_\_\_\_\_\_\_\_\_\_\_\_)  L.sort(key=lambda x:x[0])  return \_\_\_\_\_\_\_\_\_  def main():  IDS=['310107199401231242','31010719940502913X','310107199702012190','310107199104016896',  '310107199909029846','310107199608013285','310107199612122345','310107199908017384',  '31010719930302774X','310107199703030489','310107199306030821','310107199703018594']  Lcount=\_\_\_\_\_\_\_\_\_\_\_\_\_  for month,count in Lcount:  print("{}月\t{}人".format(month,count))  main() |

* 1. 程序运行结果如下图所示：

|  |
| --- |
| >>>  1月 1人  2月 1人  3月 3人  4月 1人  5月 1人  6月 1人  8月 2人  9月 1人  12月 1人  >>> |