1. **import** math, random, statistics
2. **import** numpy as np
4. #1 输入三个数求最大值
5. **def** func1():
6. **print**("输入三个数判断大小")
7. number\_list = []
8. **for** i **in** range(3):
9. number\_list.append(input())
10. max = number\_list[0]
11. **for** i **in** range(2):
12. **if** number\_list[i+1]>max:
13. max = number\_list[i+1]
14. **print**(max)
15. **return** max
17. #2 生成20随机浮点数并输出一系列数据
18. **def** func2():
19. **print**("生成20个随机浮点数")
20. random\_number\_list = []
21. **for** i **in** range(20):
22. random\_number\_list.append(random.uniform(0,100))
23. **print**(random\_number\_list)
24. max = random\_number\_list[0]
25. min = random\_number\_list[0]
26. **for** i **in** range(20):
27. **if** random\_number\_list[i]>max:
28. max = random\_number\_list[i]
29. **if** random\_number\_list[i]<min:
30. min = random\_number\_list[i]
31. **print**("最大值为{}，最小值为{}" .format(max,min))
32. mean = statistics.mean(random\_number\_list)
33. **print**("平均值为{}".format(mean))
34. pvariance = statistics.pvariance(random\_number\_list)
35. **print**("方差为{}".format(pvariance))
36. pstdev = statistics.pstdev(random\_number\_list)
37. **print**("标准差为{}".format(pstdev))
39. #3 判断一个数是否为素数
40. **def** func3():
41. **print**("判断一个数是否为素数,并找到1到1000内所有素数")
42. **def** if\_PrimeNumber(number):
43. k=0
44. **for** i **in** range(2,number):  #从3到number循环寻找因数
45. **if** number%i==0:
46. **break**
47. k+=1
48. **if** k==number-2:   #若无其他因数则说明是素数
49. **return** 1
50. **else**:
51. **return** 0
52. PrimeNumber\_list = []
53. **for** j **in** range(1000):
54. **if** if\_PrimeNumber(j)==1:
55. PrimeNumber\_list.append(j)
56. **print**(PrimeNumber\_list)
57. **print**("共计%d个素数" % len(PrimeNumber\_list))
59. #4 矩阵乘法，直接调用numpy计算
60. **def** func4():
61. **print**("矩阵乘法")
62. matrix\_1 = np.random.random((3, 3))  #随机生成两个3\*3矩阵
63. matrix\_2 = np.random.random((3, 3))
64. result = matrix\_1.dot(matrix\_2)    #点乘
65. **print**("随机生成矩阵1：")
66. **print**(matrix\_1)
67. **print**("随机生成矩阵2：")
68. **print**(matrix\_2)
69. **print**("矩阵乘法结果为：")
70. **print**(result)
72. #5 多无人机任务分配， 对于4飞机3任务，有一架飞机空闲，因此可以将评价矩阵改写
73. **def** func5():
74. **print**("无人机任务分配")
75. #评价矩阵
76. matrix\_PJ = np.array([[0,1.2,2.7,3.0],
77. [0,2.5,2.9,2.4],
78. [0,4.5,5.6,6.5],
79. [0,7.1,8.2,8.8]])
80. **print**("PJ矩阵：")
81. **print**((matrix\_PJ))
82. matrix\_L = np.zeros((24, 4))
83. #全排列函数
84. **def** get\_all\_permu(some\_list):
85. **if** some\_list:
86. all\_p = []
87. **for** x **in** some\_list:  # 第一次遍历：遍历列表中的所有元素
88. temp = some\_list[:]
89. temp.remove(x)  # 得到不包含x的列表
90. **for** y **in** get\_all\_permu(temp):   # 第二次遍历：遍历不包含x的所有元素的组合，开始递归
91. all\_p.append([x] + y)   # 将x与不包含x的其他元素的所有组合放在一起，组成一个原列表的排列
92. **return** all\_p
93. **else**:
94. **return** [[]]  # 如果要排列的列表为空，则返回一个空列表
95. matrix\_P = get\_all\_permu([3, 2, 1, 0])   #对四架飞机进行全排列
96. matrix\_P = np.array(matrix\_P)
97. **print**("P矩阵：")
98. **print**(matrix\_P)
99. **for** i **in** range(24):
100. **for** j **in** range(4):
101. matrix\_L[i][j] = matrix\_PJ[j][matrix\_P[i][j]]
102. **print**("L矩阵：")
103. **print**(matrix\_L)
104. matrix\_sum = np.sum(matrix\_L,axis=1)    #对列求和计算sum矩阵
105. **print**("sum矩阵：")
106. **print**(matrix\_sum)
107. max\_sum = np.max(matrix\_sum)
108. min\_sum = np.min(matrix\_sum)
109. **print**("最大取值为{}，最小取值为{}".format(max\_sum,min\_sum))
110. a = np.where(matrix\_sum == max\_sum)   #返回最大值索引
111. **print**("最大价值分配方案%s"% str(matrix\_P[a][0]))
112. **print**("即第1架飞机执行%d任务\n第2架飞机执行%d任务\n第3架飞机执行%d任务\n第4架飞机执行%d任务" % (matrix\_P[a][0][0],matrix\_P[a][0][1],matrix\_P[a][0][2],matrix\_P[a][0][3]))
113. b = np.where(matrix\_sum == min\_sum)   #返回最小值索引
114. **print**("最小价值分配方案%s"% str(matrix\_P[b][0]))
115. **print**("即第1架飞机执行%d任务\n第2架飞机执行%d任务\n第3架飞机执行%d任务\n第4架飞机执行%d任务" % (matrix\_P[b][0][0], matrix\_P[b][0][1], matrix\_P[b][0][2], matrix\_P[b][0][3]))
117. **if** \_\_name\_\_ == "\_\_main\_\_":
118. **print**("是否测试函数1，是：1，否：0，下同")
119. s1 = input()
120. **if** s1 == '1':
121. func1()
122. **print**("是否测试函数2")
123. s2 = input()
124. **if** s2 == '1':
125. func2()
126. **print**("是否测试函数3")
127. s3 = input()
128. **if** s3 == '1':
129. func3()
130. **print**("是否测试函数4")
131. s4 = input()
132. **if** s4 == '1':
133. func4()
134. **print**("是否测试函数5")
135. s5 = input()
136. **if** s5 == '1':
137. func5()