



问题解答

双链表和探索迷宫递归算法

分形树绘制、曼德勃罗集

课堂练习：递归黄金螺线

# 数据结构与算法 ( Python ) -09/0324

陈斌 gischen@pku.edu.cn 北京大学地球与空间科学学院

# 目录

- › 问题解答
- › 双链表和递归内容回顾
- › turtle和分形树绘制、曼德勃罗集
- › 课堂练习



# 问题解答

- › 为什么这门课这么**卷**？
- › 能再讲解一下h3双链表的选做题吗？
- › 再整理一下探索迷宫的算法思路
- › 关于【C1】创意作品征集
  - 这周C作业是选做吗？
  - 介绍一下turtle库的功能
  - 如何画出逼真的树叶和树干？
  - 分形树上面怎么画花？
  - 想再详细的介绍一下较为复杂的分形树作图。



# 问题解答

- › **MOOC第八题mandelbrot为什么不能用递归算法求解呢？**
- › **OJ题求讲**  
四柱汉诺塔
- › **什么是动态规划？**
- › **PyLn平台上的turtle和本机的Python不一样？**

# 为什么这门课这么卷？

为什么编程dalao云集本课？（不是

快速变化的时代

计算思维正在成为基本素质

信息素养差距加大，未来5年间会更大



<b>第 1 单元 数据及其结构</b> <ul style="list-style-type: none"><li>1.1 数据及其价值</li><li>1.2 数据的组织结构</li><li>1.3 认识数据抽象</li></ul> 单元学习评价 单元学习总结	<b>第 4 单元 队列及其应用</b> <ul style="list-style-type: none"><li>4.1 队列结构及其实现</li><li>4.2 基数排序</li><li>4.3 排队取号模拟系统</li></ul> 单元学习评价 单元学习总结
<b>第 2 单元 线性表及其应用</b> <ul style="list-style-type: none"><li>2.1 线性表结构及其实现</li><li>2.2 随机抽取问题</li><li>2.3 字符串应用</li></ul> 单元学习评价 单元学习总结	<b>第 5 单元 栈及其应用</b> <ul style="list-style-type: none"><li>5.1 栈结构及其实现</li><li>5.2 符号匹配问题</li><li>5.3 算术表达式求值</li></ul> 单元学习评价 单元学习总结
<b>第 3 单元 数据的排序与查找</b> <ul style="list-style-type: none"><li>3.1 迭代与递归</li><li>3.2 数据的排序</li><li>3.3 数据的查找</li></ul> 单元学习评价 单元学习总结	<b>第 6 单元 树及其应用</b> <ul style="list-style-type: none"><li>6.1 树结构及其实现</li><li>6.2 用二叉树排序</li><li>6.3 画出二叉树</li></ul> 单元学习评价 单元学习总结

# 为什么这门课这么卷？

## 一些高中信息技术俱乐部的选题

作文的风格仿写、根据描述和图片自动生成海报

根据Python习题执行出错给出修改建议、生命游戏研究疫情扩散

## 一个来自初中一年级课余小组的作品（并非信息奥赛）

第一阶段：  
获得数据库（爬虫、网上下载、做成SQL）  
词语替换算法  
拆分句子结构  
具体的替换

第二阶段：  
语法检测  
句子结构具体拆分  
研究中文语法  
具体实现语法修改

第三阶段：  
文章风格化  
仔细研究中文语法  
具体实现a模仿b文章的功能

```
pycoder.py x
1 """PyCoder - A Python File Editor
2
3 Author: Zhang Haoran; Zhou Yangqi; Su Chuyao
4 Datetime: 2020/3/9
5 Version: 4.3.2"""
6
7 __version__ = '4.3.2'
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2
```

# 为什么这门课这么卷？

- › 将来你会发现，学编程、数据结构、算法可能会是用途最广的工具
- › 无论你是哪个理工专业，甚至你是文史社科专业
- › 为了缩小与**同代人**的信息素养差距
- › 也值得在本科期间加大投入，学习和实践信息技术，包括数据分析、人工智能
- › 大学教师也面临挑战

# PyLn平台上的turtle为什么与本机不一样？

- › 实际上PyLn平台上的Python代码是在浏览器里执行的，并非像OJ那样在后端服务器执行
- › 有几个在浏览器网页里执行Python代码的开源项目



# Brython : <https://brython.info/>

**Brython** Tutorial Demo Documentation Console Editor Gallery Download Development Community

A Python 3 implementation for client-side web programming

Without a doubt, you've seen a clock like this in demos of HTML5

```
1 <html>
2
3 <head>
4   <script src="/brython.js"></script>
5 </head>
6
7 <body onload="brython()">
8   <script type="text/python">
9     from browser import document, alert
10
11     # bind event 'click' on button to function echo
12
13     def echo(ev):
14       alert(document["zone"].value)
15
16     document["mybutton"].bind("click", echo)
17   </script>
18   <input id="zone">
19   <button id="mybutton">click !</button>
20 </body>
21
22 </html>
```



of type "text/python".

or the Web. As such,

gh a web console),

to the DOM objects

document elements

ing Brython.

s.

# skulpt : <http://skulpt.org/>

SKULPT

[Home](#) [Project Gallery](#) [Using Skulpt](#) [Contributing](#) [Docs](#)

## Python. Client Side.

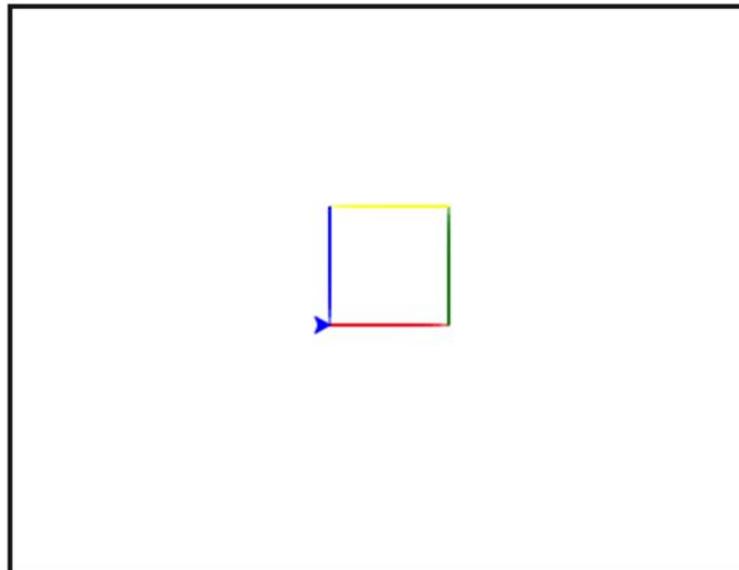
Skulpt is an *entirely in-browser* implementation of Python.

No preprocessing, plugins, or server-side support required, just write Python and reload.

```
1 import turtle
2
3 t = turtle.Turtle()
4
5 for c in ['red', 'green', 'yellow', 'blue']:
6     t.color(c)
7     t.forward(75)
8     t.left(90)
9
```

Run

Help, or examples: 1 2 3 4 5 6 7 8. Ctrl-Enter to



# 【H3】双链表选做部分

## › 从可迭代对象中初始化双链无序表

```
34 class DoublyLinkedList():
35     # 请在此编写你的代码 (可删除pass语句)
36     def __init__(self, it= None):
37         self.head = None # 表头作为index(0)
38         self.tail = None # 表尾作为index(-1)
39         self.length = 0
40         if it is not None:
41             for d in it:
42                 self.append(d)
```

```
>>> l = DoublyLinkedList([23, 45, 67, 89])
```

```
>>> l
[23, 45, 67, 89]
```

```
>>> l = DoublyLinkedList(range(8))
```

```
>>> l
[0, 1, 2, 3, 4, 5, 6, 7]
```

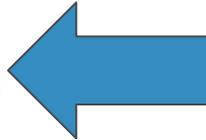
# 【H3】双链表选做部分

```
>>> for i in zip([1,2,3,4], ["a", True, 1.23, 435]):  
        print(i)
```

```
(1, 'a')  
(2, True)  
(3, 1.23)  
(4, 435)
```

## > 相等判断的实现 ( \_\_eq\_\_ )

```
191     def __eq__(self, other):  
192         if other is None or not isinstance(other, DoublyLinkedList):  
193             return False  
194         if len(self) != len(other):  
195             return False  
196         for s, o in zip(self, other):  
197             if s != o:  
198                 return False  
199         else:  
200             return True
```



```
>>> l = DoublyLinkedList(range(8))  
>>> l  
[0, 1, 2, 3, 4, 5, 6, 7]  
>>> k = DoublyLinkedList([0,1,2,3,4,5,6,7])  
>>> k  
[0, 1, 2, 3, 4, 5, 6, 7]  
>>> l == k  
True
```

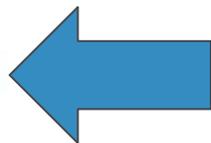
# 【H3】双链表选做部分

## › 实现为可迭代对象 ( `__iter__` )

两种方案:

`__len__` + `__getitem__`, 或者, `__iter__` 利用 `yield` 返回

```
47     def size(self):
48         return self.length
49     __len__ = size
159    def __getitem__(self, key):
160        if isinstance(key, int):
161            # 按照下标
162            current, i = self.head, 0
163            while i < key:
164                current = current.getNext()
165                i += 1
166            if current is not None:
167                return current.getData()
168            else:
169                raise StopIteration
```



# 探索迷宫的算法思路

## › 回顾一下递归解决问题的特征

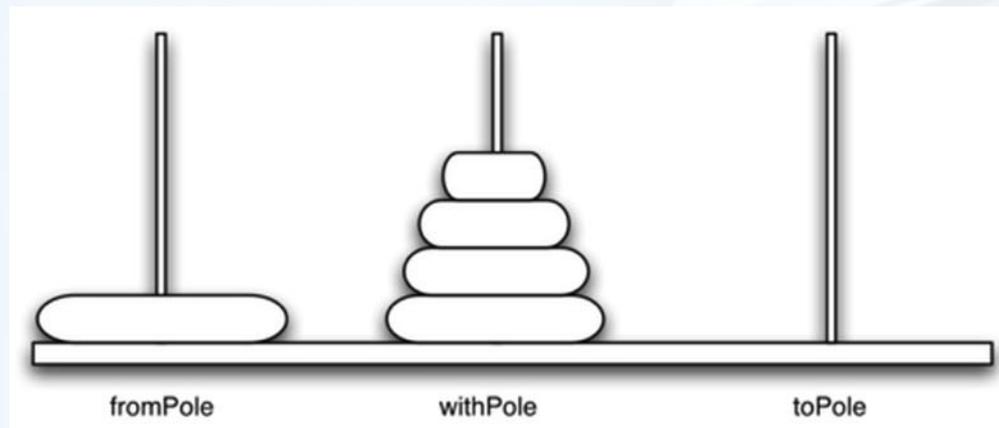
一个问题可以分解为：在解决更小规模的相同问题后，只需要再做个简单操作即可

## › 递归解决数列求和问题

把数列中的**第1个数**，加上，**余下数列**的和。

## › 递归解决汉诺塔问题

把上面 **$n-1$ 个盘子**从A移动到B，  
把**最大盘子**从A移动到C，  
把B上的 **$n-1$ 个盘子**从B移动到C



# 探索迷宫的算法思路

## 递归解决探索迷宫问题

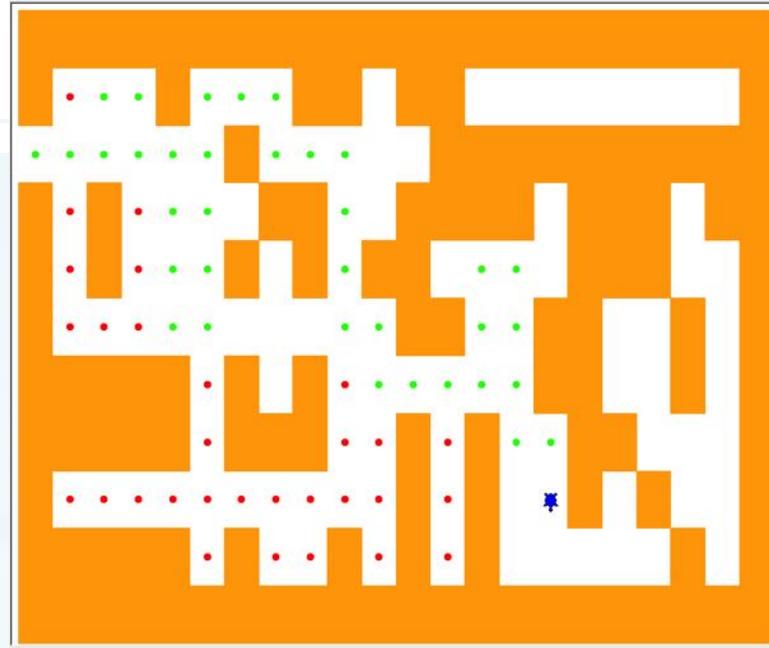
从当前位置，

向北一步，继续探索迷宫；

向南一步，继续探索迷宫；

向西一步，继续探索迷宫；

向东一步，继续探索迷宫。



```
# Otherwise, use logical short circuiting to try each direction
# in turn (if needed)
found = searchFrom(maze, startRow-1, startColumn) or \
        searchFrom(maze, startRow+1, startColumn) or \
        searchFrom(maze, startRow, startColumn-1) or \
        searchFrom(maze, startRow, startColumn+1)
if found:
    maze.updatePosition(startRow, startColumn, PART_OF_PATH)
else:
    maze.updatePosition(startRow, startColumn, DEAD_END)
return found
```

北、南、西、东  
依次尝试

由于or的短路算法，  
只要有一个方向返回  
成功，则后面的尝试  
就不需要做了，直接  
返回成功！

成功，标注

失败，标注

# 探索迷宫的算法思路

## › 接下来是两个重要的细节

基本结束条件.....找到出口、碰到墙、碰到走过的路

防止原地打转.....每步的出发点标记面包屑

```
# base Case return values:  
# 1. We have run into an obstacle, return false  
maze.updatePosition(startRow, startColumn)  
if maze[startRow][startColumn] == OBSTACLE :  
    return False  
# 2. We have found a square that has already been explored  
if maze[startRow][startColumn] == TRIED or maze[startRow][startColumn] == DEAD_END:  
    return False  
# 3. We have found an outside edge not occupied by an obstacle  
if maze.isExit(startRow, startColumn):  
    maze.updatePosition(startRow, startColumn, PART_OF_PATH)  
    return True  
maze.updatePosition(startRow, startColumn, TRIED)
```

墙, 失败

已过, 失败

出口, 成功

标注洒“面包屑”

# 探索迷宫：算法代码

```
def searchFrom(maze, startRow, startColumn):  
    # try each of four directions from this point until we find a way out.  
    # base Case return values:  
    # 1. We have run into an obstacle, return false  
    maze.updatePosition(startRow, startColumn)  
    if maze[startRow][startColumn] == OBSTACLE :  
        return False  
    # 2. We have found a square that has already been explored  
    if maze[startRow][startColumn] == TRIED or maze[startRow][startColumn] == DEAD_END:  
        return False  
    # 3. We have found an outside edge not occupied by an obstacle  
    if maze.isExit(startRow, startColumn):  
        maze.updatePosition(startRow, startColumn, PART_OF_PATH)  
        return True  
    maze.updatePosition(startRow, startColumn, TRIED)  
    # Otherwise, use logical short circuiting to try each direction  
    # in turn (if needed)  
    found = searchFrom(maze, startRow-1, startColumn) or \  
            searchFrom(maze, startRow+1, startColumn) or \  
            searchFrom(maze, startRow, startColumn-1) or \  
            searchFrom(maze, startRow, startColumn+1)  
    if found:  
        maze.updatePosition(startRow, startColumn, PART_OF_PATH)  
    else:  
        maze.updatePosition(startRow, startColumn, DEAD_END)  
    return found
```

墙，失败

已过，失败

出口，成功

标注洒“面包屑”

北、南、西、东  
依次尝试

由于or的短路算法，  
只要有一个方向返回  
成功，则后面的尝试  
就不需要做了，直接  
返回成功！

成功，标注

失败，标注

# 海龟作图turtle模块

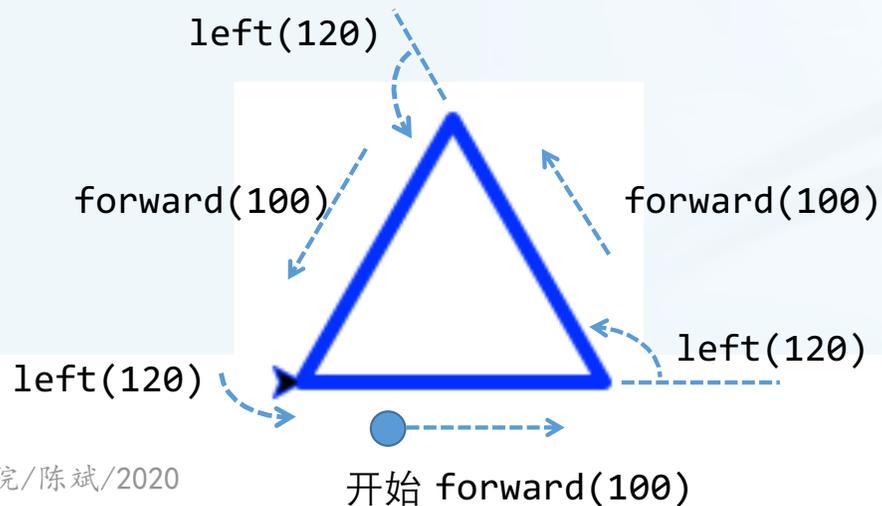
› 【慕课】Python语言基础与应用 ( P42 )

<https://www.bilibili.com/video/BV1V741147QH?p=42>

› 模拟海龟在沙滩上爬行所描绘的轨迹，从LOGO语言借鉴而来  
前进forward(n); 后退backward(n); 左转left(d); 右转right(d)

› 画笔：抬起落下、颜色、粗细

抬起penup(); 落下pendown(); 笔色color(); 笔粗细pensize(n)



# 让海龟直接去到某个位置goto(x,y)

- › 最开始海龟在  $(0,0)$
- › 可以用goto让海龟去到任何地方
- › 用position()获取海龟当前位置
- › 如果不希望留下轨迹
- › 则需要先在goto之前调用penup()



# 作图程序模版

› 首先，导入turtle模块

› 然后，生成一只海龟

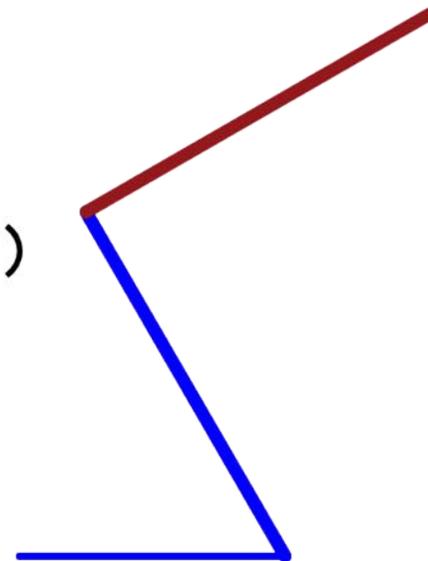
可以做一些初始化设定

› 程序主体：用作图语句绘图

› 最后结束作图

可选隐藏海龟：t.hideturtle()

```
1 # 1. 导入海龟模块
2 import turtle
3
4 # 2. 生成一只海龟，做一些设定
5 t = turtle.Turtle()
6 t.color("blue")
7 t.pensize(3)
8
9 # 3. 用海龟作图
10 t.forward(100)
11 t.right(60)
12 t.pensize(5)
13 t.backward(150)
14 t.left(90)
15 t.color("brown")
16 t.forward(150)
17
18 # 4. 结束作图
19 t.hideturtle()
20 turtle.done()
```



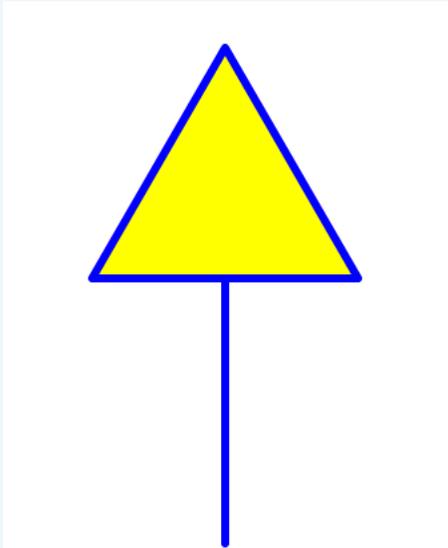
# 样例：警示牌

## › 填充fill

设定填充颜色fillcolor

开始填充begin\_fill

结束填充end\_fill



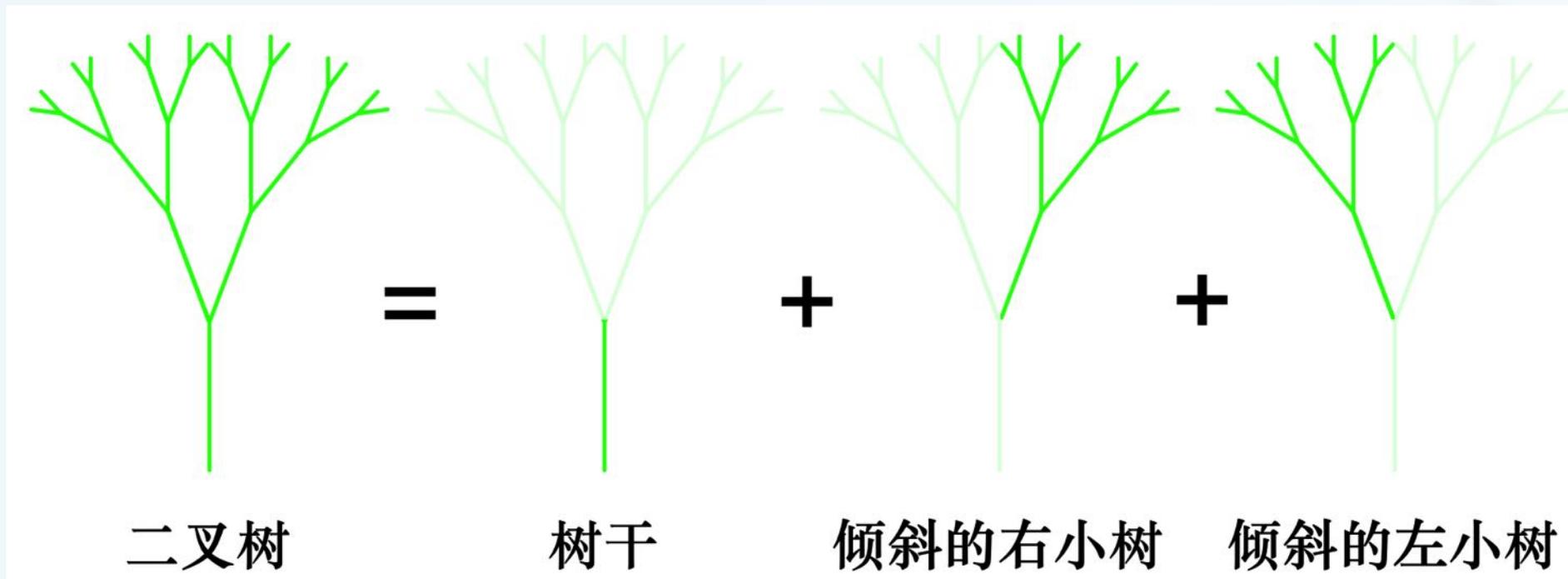
```
4 # 2. 生成一只海龟，做一些设定
5 t = turtle.Turtle()
6 t.pencolor("blue")
7 t.pensize(3)
8 t.fillcolor("yellow")
9
10 # 3. 用海龟作图
11 t.lt(90)
12 t.fd(100)
13 t.rt(90)
14 t.begin_fill()
15 t.fd(50)
16 t.left(120)
17 t.fd(100)
18 t.left(120)
19 t.fd(100)
20 t.left(120)
21 t.fd(50)
22 t.end_fill()
```

# 海龟函数的小结

<https://docs.python.org/zh-cn/3/library/turtle.html>

- › **前进**`forward(n)`
- › **后退**`backward(n)`  
缩写：`fd(n)`、`bk(n)`
- › **左转**`left(n)`、**右转**`right(n)`  
缩写：`lt(n)`、`rt(n)`
- › **画笔**  
笔画颜色`pencolor(颜色名称)`  
笔画粗细`pensize(n)`
- › **抬笔**`penup()`、**落笔**`pendown()`  
缩写`pu()`、`pd()`
- › **画圆**：`circle(半径, 角度)`
- › **画点**：`dot(大小, 颜色)`
- › **填充**  
填充颜色`fillcolor(颜色名称)`  
填充开始`begin_fill()`  
填充结束`end_fill()`
- › **坐标控制**  
直接到达`goto(x,y)`  
获取坐标`position()`  
计算距离`distance(x,y)`

# 二叉树的递归分解



# 海龟作图分形树模版

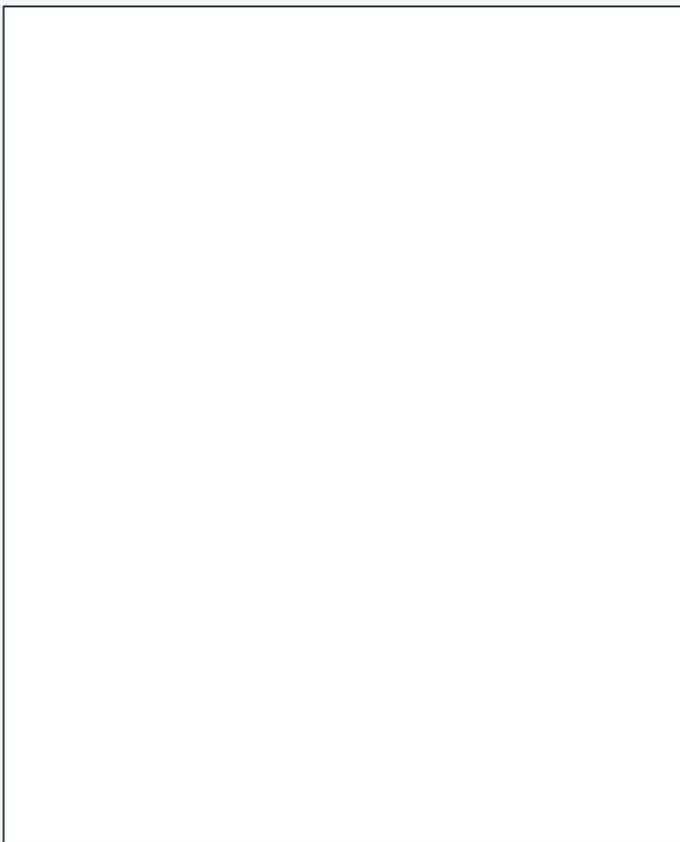


```
2 # 1. 导入海龟模块
3 import turtle
4
5 #turtle.tracer(0) # 去掉注释, 海龟暴走
6
7 # 海龟颜色表 (有些在PyLn中没有效果的可以参照下面链接的RGB值)
8 # http://gis4g.pku.edu.cn/python-turtle-color-names/
9 # https://www.tcl.tk/man/tcl8.6/TkCmd/colors.htm
10
11 # 2. 生成一只海龟, 做一些设定
12 t = turtle.Turtle()
13 t.pencolor("green") # 画笔颜色
14 t.pensize(3) # 画笔粗细
15
16
17 # 3. 用海龟作图
18 def tree(branch_len):
19     if branch_len > 5: #树干太短不画, 即递归结束条件
20         t.fd(branch_len) #画树干
21         t.rt(20) #右倾斜20度
22         tree(branch_len-15) #递归调用, 画右边的小树, 树干减15
23         t.lt(40) #向左回40度, 即向左倾斜20度
24         tree(branch_len-15) #递归调用, 画左边的小树, 树干减15
25         t.rt(20) #向右回20度, 即回正
26         t.backward(branch_len) #海龟退回原来位置
```

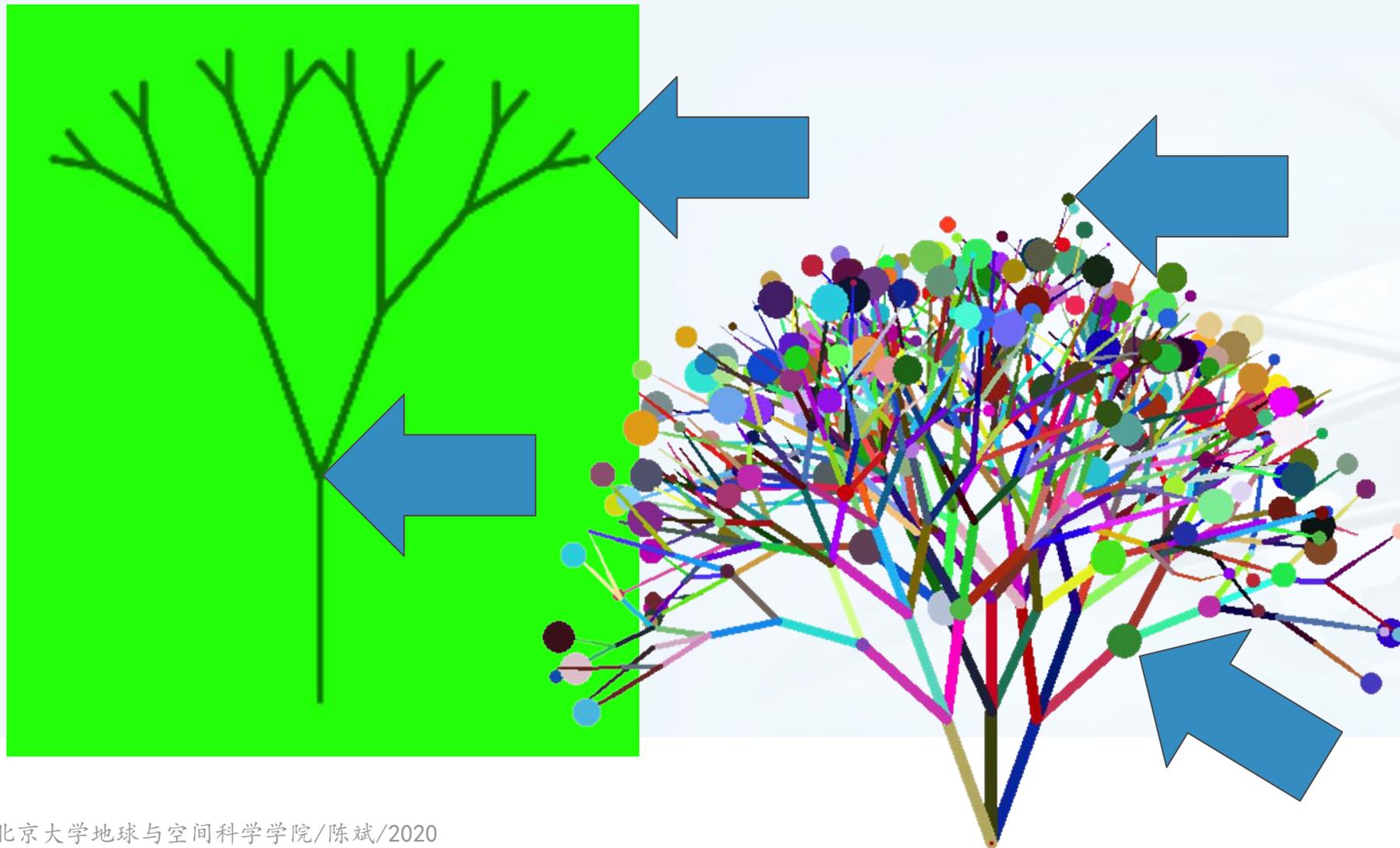
# 海龟作图分形树模版



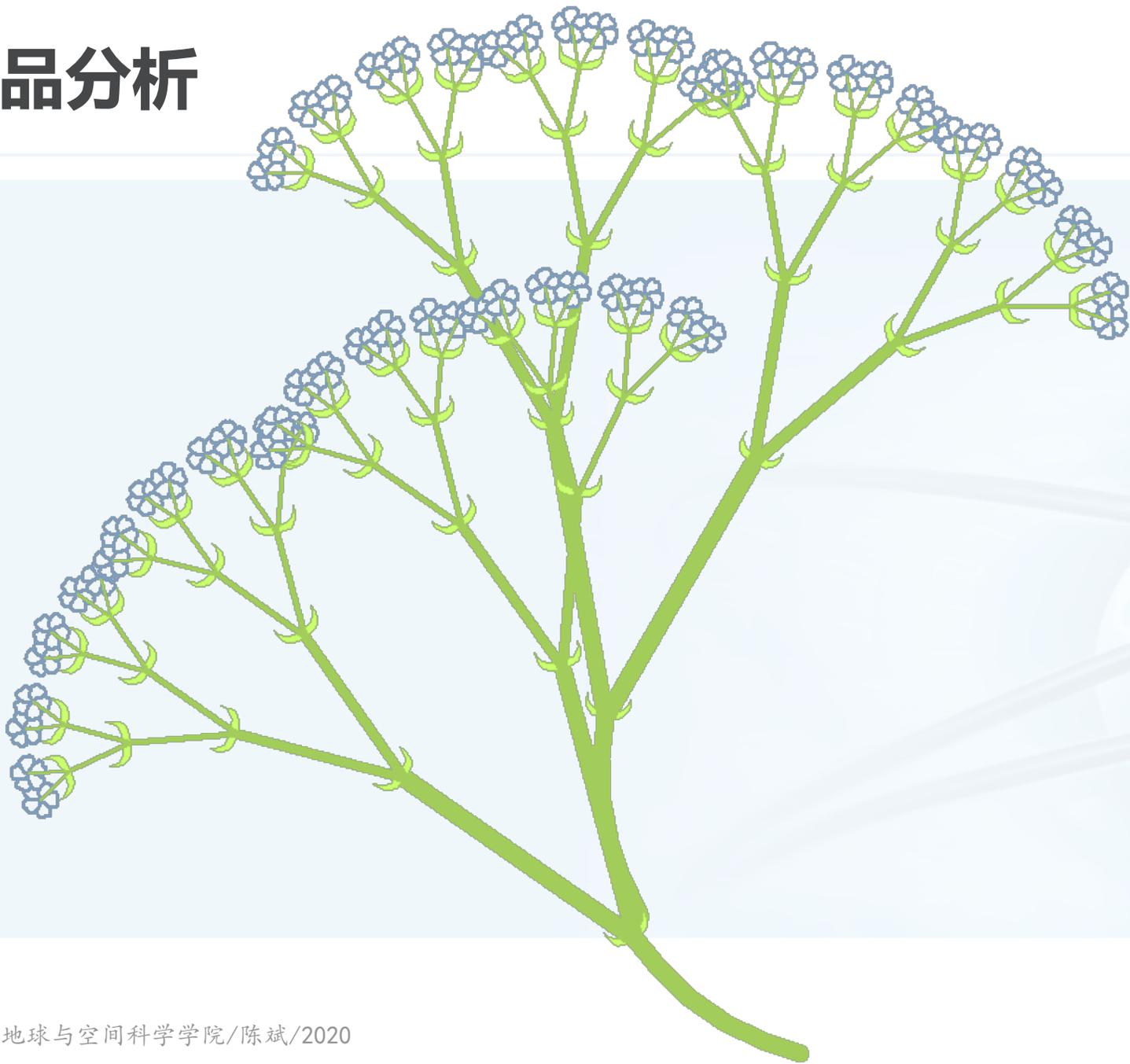
```
28 t.lt(90)
29 t.penup()
30 t.backward(100)
31 t.pendown()
32 t.pencolor('green')
33 t.pensize(2)
34 tree(75)
35
36 # 4. 结束作图
37 t.hideturtle()
38 turtle.update()
39
40 # 5. 设置背景, 收尾
41 ts = turtle.getscreen()
42 ts.bgcolor("lime")
43
44 turtle.done()
```



# 树叶、花、果的绘制时机



# 作品分析

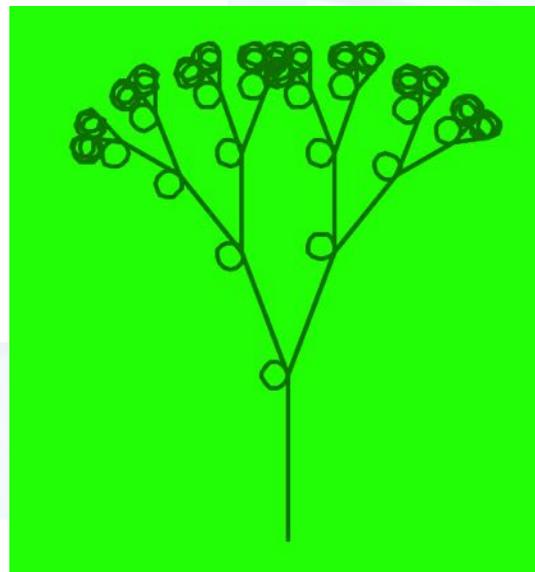


# 作品分析

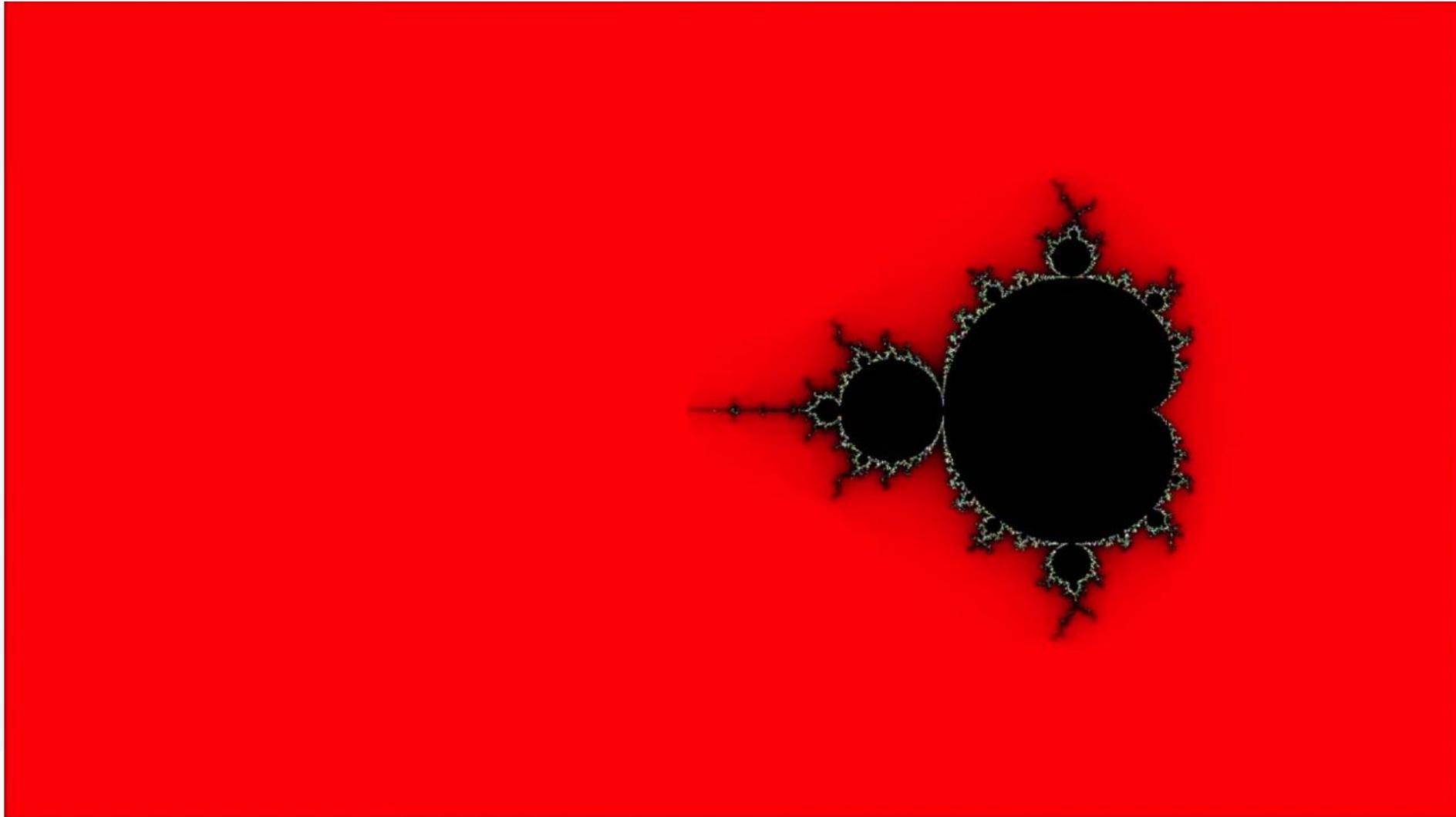


# 树叶、花、果的绘制时机

```
17 # 3. 用海龟作图
18 def tree(branch_len):
19     if branch_len > 5: #树干太短不画, 即递归结束条件
20         t.fd(branch_len) #画树干
21         t.rt(20) #右倾斜20度
22         tree(branch_len-15) #递归调用, 画右边的小树, 树干减15
23         t.lt(40) #向左回40度, 即向左倾斜20度
24         tree(branch_len-15) #递归调用, 画左边的小树, 树干减15
25         t.rt(20) #向右回20度, 即回正
26
27         # 在分叉上画树叶/花/果
28         t.circle(6)
29
30     t.backward(branch_len) #海龟退回原来位置
31 else:
32     # 树干太短, 到树梢了, 可以绘制树叶/花/果
33     t.circle(5)
```



# 曼德勃罗集：奇幻旅程



# 曼德勃罗集绘制

- › 对平面上的每个点（复数） $c=x+yi$  进行检验：  
从  $z=0$  开始， $f(z)=z^2+c$   
序列  $0, f(0), f(f(0)), f(f(f(0))) \dots$
- › 如果序列收敛在有限区域，那么  $c$  属于集合（设为黑色），否则不属于集合（保持背景白色）  
一般在最大迭代次数 `max_repeats` 内，如果这个复数点没有超出逃逸圆（半径为2），则属于集合
- › 但实际上，边界的复杂情况造成了不确定性  
是否属于集合无法判别，随着最大迭代次数变化，结论可能会变化
- › 可视化方案：多少次迭代会超出逃逸圆？次数  $\rightarrow$  颜色

# 一个示例代码

## › 迭代函数

```
1 import numpy as np
2 from matplotlib import cm, pyplot as pl
3
4
5 def iter_point(c):
6     z = c
7     for i in range(100): # 最多迭代100次
8         if abs(z) > 2:
9             break # 半径大于2则认为逃逸
10        z = z**2 + c
11    return i # 返回迭代次数, 对应颜色
```

# 一个示例代码

## > 绘制函数

```
14 def draw_mandelbrot(cx, cy, d):
15     """
16     绘制点(cx, cy)附近正负d的范围的Mandelbrot
17     """
18     x0, x1, y0, y1 = cx - d, cx + d, cy - d, cy + d
19     y, x = np.ogrid[y0:y1:200j, x0:x1:200j] # 复数步长表示s:e之间分为几份
20     c = x + y * 1j # 200*200的矩阵
21     # np.frompyfunc让iter_point处理矩阵, 1是参数个数, 1是返回值个数
22     mandelbrot = np.frompyfunc(iter_point, 1, 1)(c).astype(np.float)
23     pl.imshow(mandelbrot, cmap=cm.jet, extent=[x0, x1, y0, y1])
24     pl.gca().set_axis_off()
```

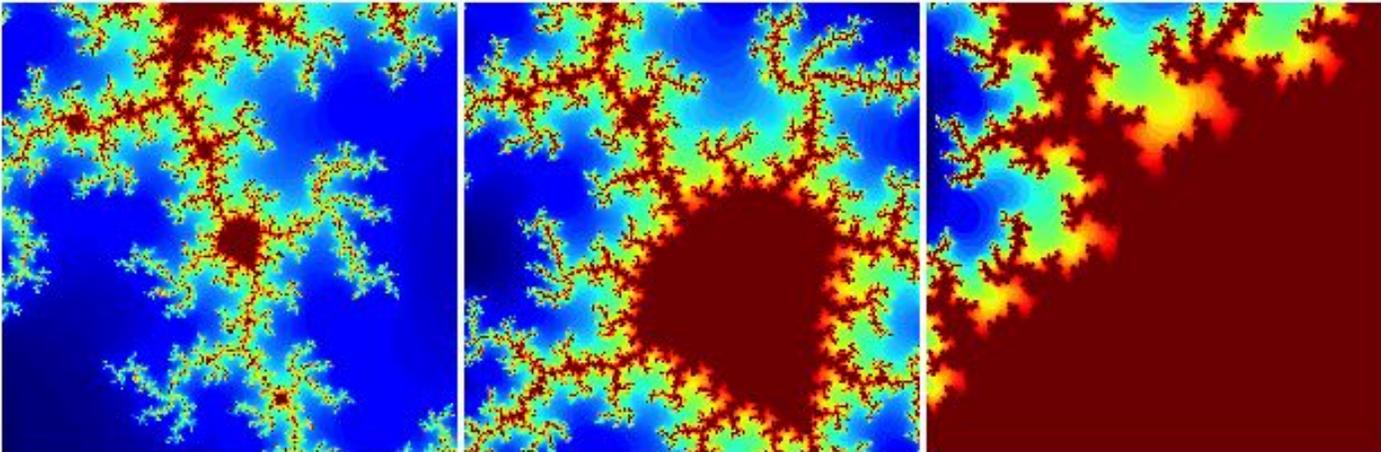
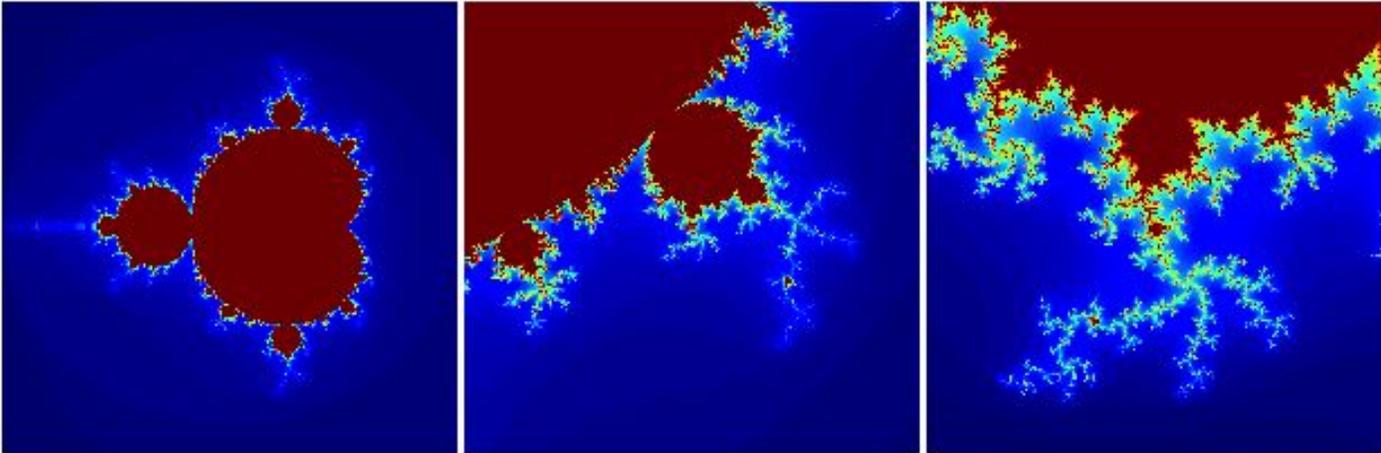
<https://matplotlib.org/3.2.0/tutorials/colors/colormaps.html>

# 一个示例代码

## › 选取范围绘制6个局部

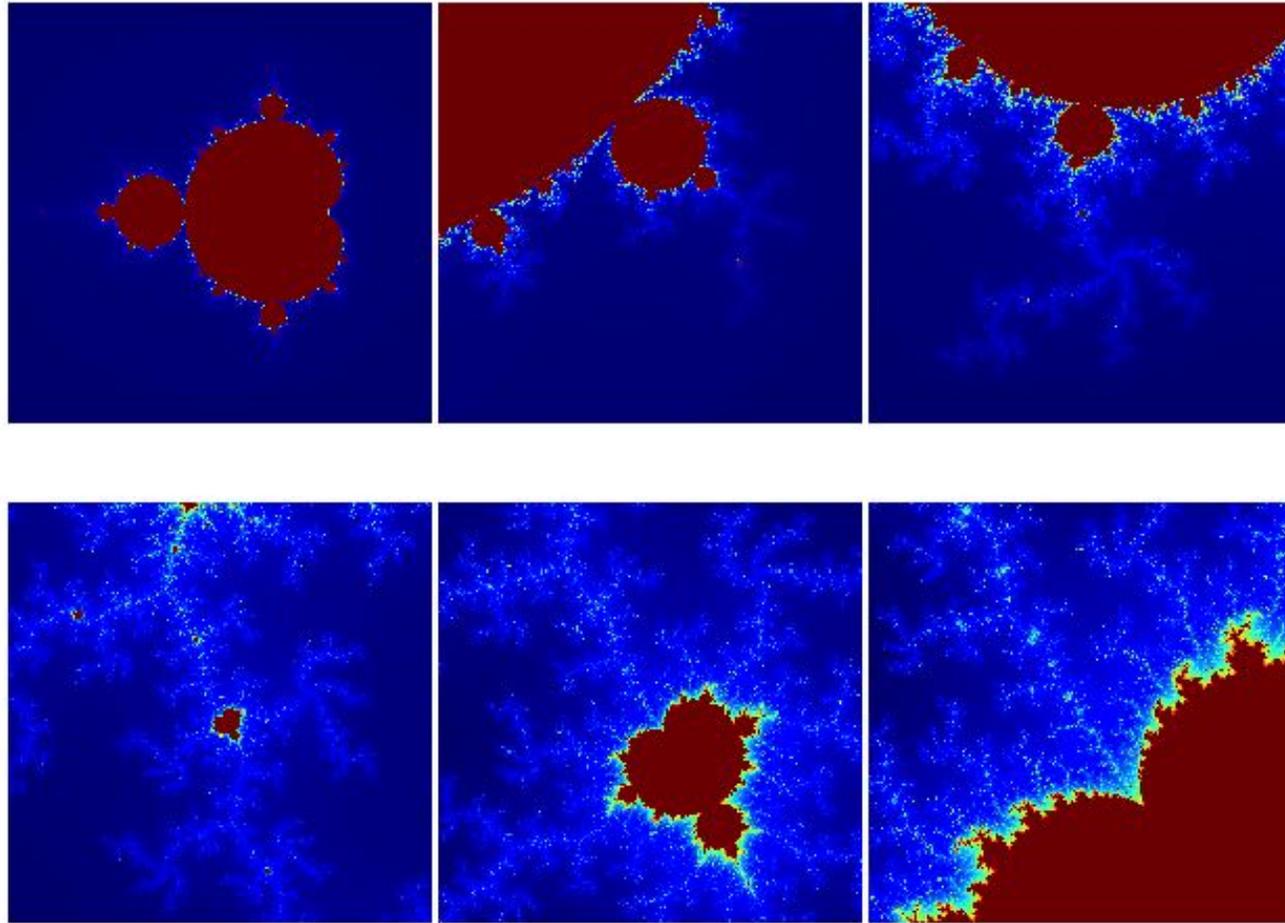
```
27 x, y = 0.27322626, 0.595153338
28
29 plt.subplot(2, 3, 1) # 2行3列里的第1个
30 draw_mandelbrot(-0.5, 0, 1.5)
31 for i in range(2, 7):
32     plt.subplot(2, 3, i)
33     draw_mandelbrot(x, y, 0.2**(i - 1))
34 plt.subplots_adjust(0.02, 0, 0.98, 1, 0.02, 0)
35 plt.show()
```

# 运行结果：最大迭代次数100

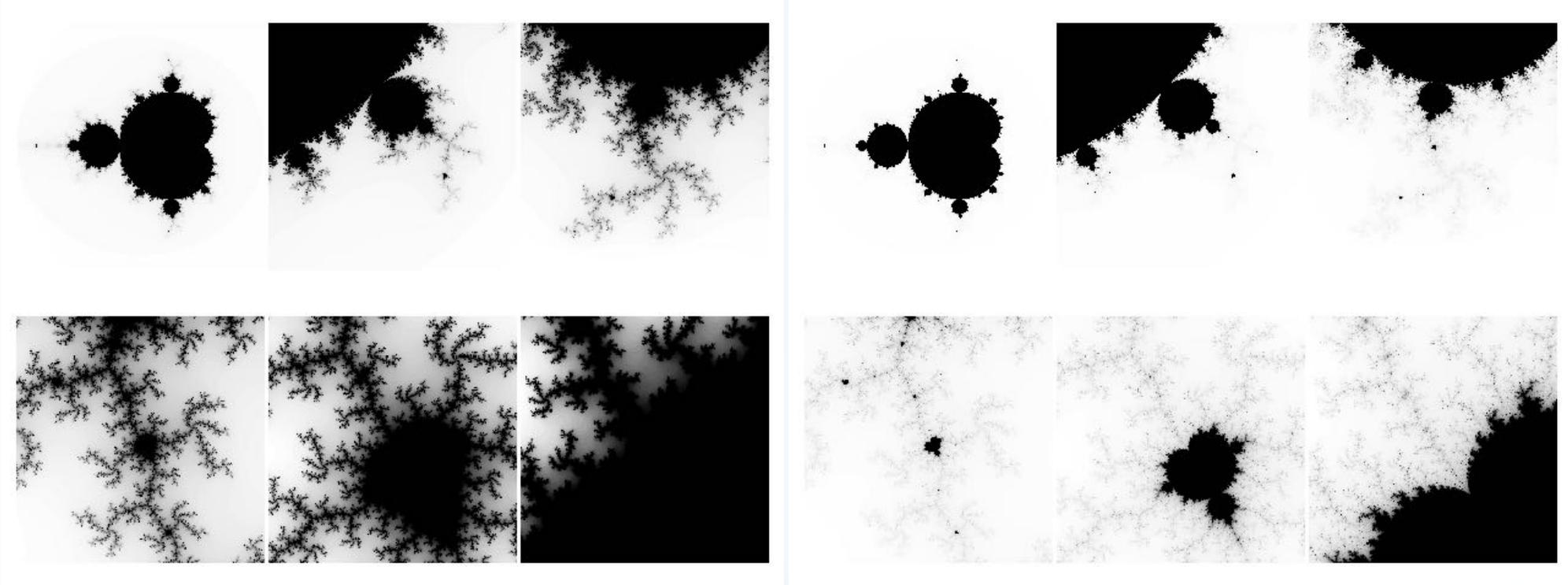


# 运行结果

› 改变最大迭代次数500，则又不同

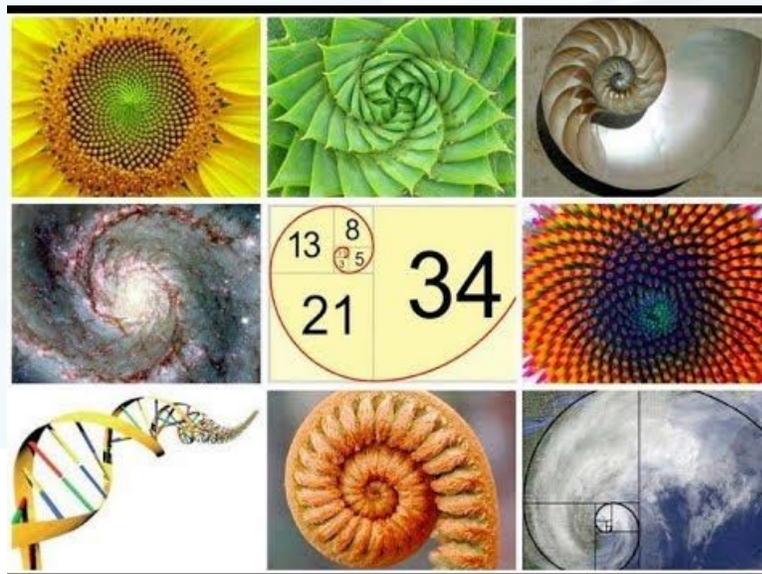
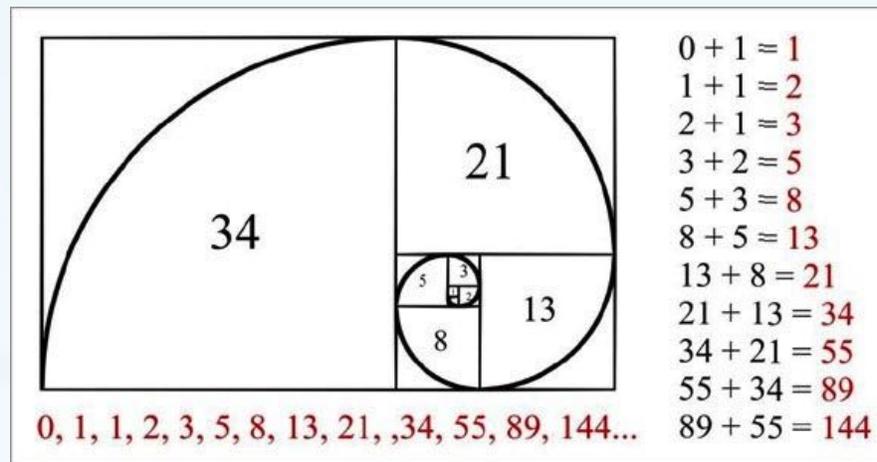


# 运行结果：cmap=Greys 100次vs500次



# 典型例子：斐波那契数列

- 斐波那契数列的发现者，是意大利数学家列昂纳多·斐波那契（Leonardo Fibonacci）
- 从第3项开始，每一项都等于前两项之和
- 相邻两项之比，无限趋近于“黄金分割数” 0.618....
- 在自然界中有很多实例
- 黄金分割比例广泛应用于美术



# 艺术中的黄金分割Golden Ratio

## > 建筑艺术



## > 摄影构图



# 递归函数：fibonacci

- › 观察fibonacci数列的定义
- › 一个典型的递归定义
- › 可以写出递归函数fibonacci
- › 这里，**递归结束条件是什么？**
- › **请分析fibonacci(5)的调用？**

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}, n > 1$$

```
1 def fibonacci(n):
2     if n == 1:
3         return 0
4     elif n == 2:
5         return 1
6     else:
7         return fibonacci(n - 1) + fibonacci(n - 2)
8
9
10 for i in range(1, 12):
11     print(i, fibonacci(i))
```

fibonacci.py

# 【课堂练习】画出这条“黄金螺线”

## 海龟作图circle函数

`circle(<半径>, <角度>)`

## 右图是10阶黄金螺线

由半径为`fibonacci(10)`的1/4圆弧加上9阶黄金螺线构成

## 对应递归函数`golden_spiral(n)` :

`circle(fibonacci(10), 90)`

`golden_spiral(n-1)`

