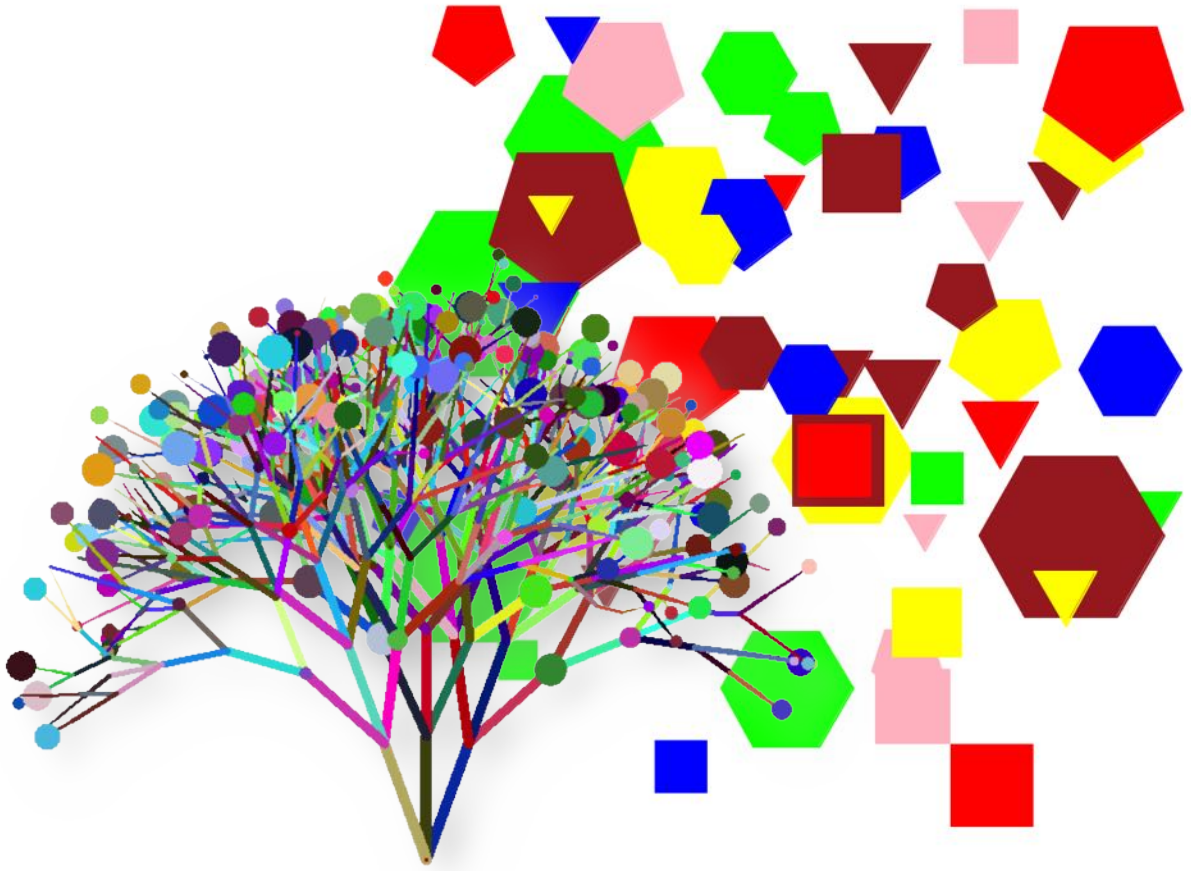
Python语言基础与应用04

北京大学 陈斌

2019.07.04

目录

- 函数的基本概念
- 组合图形练习
- 递归的概念、递归函数
- 绘制简单二叉树
- 分治策略
- 算法分析与评价
- micro:bit开源硬件

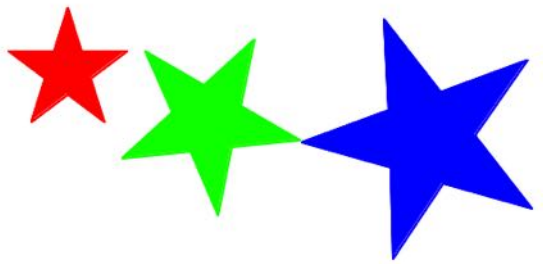


代码复用情境

- 有时候我们需要反复使用某些代码
  - 比如组合图形出现多个五角星
- 如果到处拷贝这些代码，会出现弊端
  - 程序变得冗长，可读性差
  - 一旦需要修改或扩充，要在各处同步改代码，容易出错，可维护性差

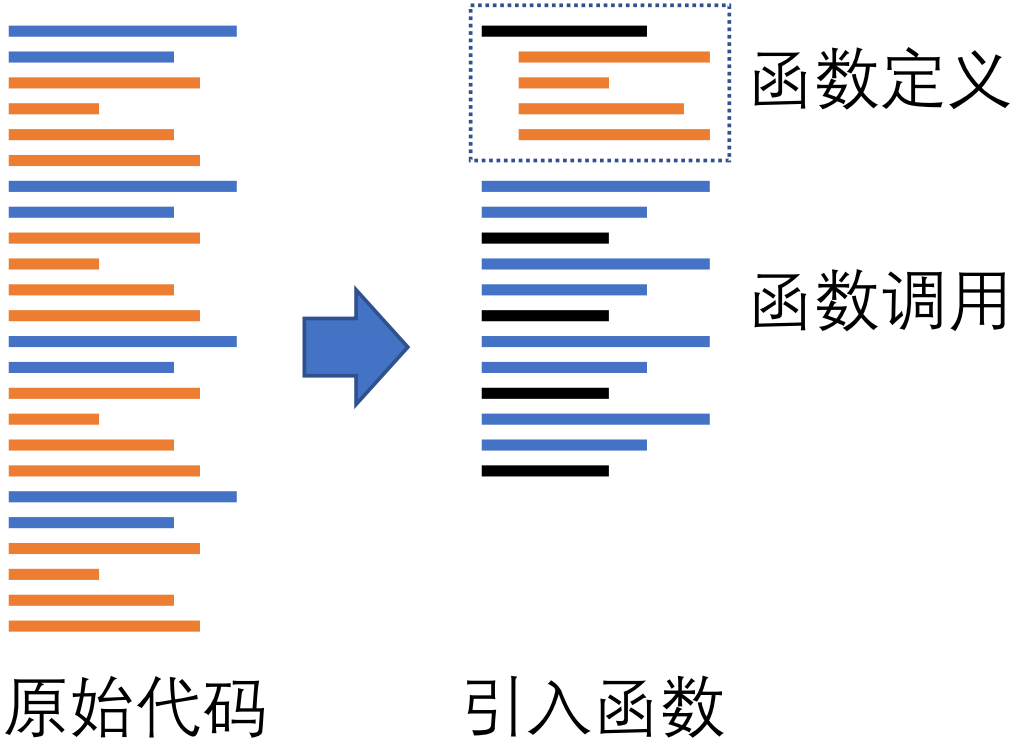
func_star.py

```
1  import turtle
2
3  t = turtle.Turtle()
4
5  t.color('red')
6  t.begin_fill()
7  for i in range(5):
8      t.forward(20)
9      t.left(72)
10     t.forward(20)
11     t.right(144)
12 t.end_fill()
13
14 t.penup()
15 t.forward(60)
16 t.right(30)
17 t.pendown()
18
19 t.color('green')
20 t.begin_fill()
21 for i in range(5):
22     t.forward(30)
23     t.left(72)
24     t.forward(30)
25     t.right(144)
26 t.end_fill()
27
28 t.penup()
29 t.forward(80)
30 t.left(50)
31 t.pendown()
32
33 t.color('blue')
34 t.begin_fill()
35 for i in range(5):
36     t.forward(40)
37     t.left(72)
38     t.forward(40)
39     t.right(144)
40 t.end_fill()
41
42 t.hideturtle()
43 turtle.done()
```



解决方案：函数（functions）

- 我们把这些重复代码单独收集起来，组成一个“函数”对象，并赋予一个名称
- 在需要用到这些代码的时候就通过名称加括号来“呼叫”这些“函数”
- 前者称为函数定义（define）
- 后者称为函数调用（call）



定义函数：def语句

- 函数定义语句def

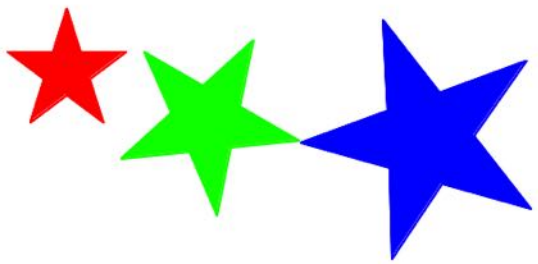
```
def <函数名称>([<参数表>]):
    <语句块>
    [return <返回值>]
```

- 几个要素
  - def关键字
  - 函数名称，后跟一对圆括号
  - （可选的）参数表
  - 语句块
  - （可选的）返回值

```
1  import turtle
2
3
4  def star(size, color):
5      t.color(color)
6      t.begin_fill()
7      for i in range(5):
8          t.forward(size)
9          t.left(72)
10         t.forward(size)
11         t.right(144)
12     t.end_fill()
13
14
15 t = turtle.Turtle()
16
17 star(20, 'red')
18 t.penup()
19 t.forward(60)
20 t.right(30)
21 t.pendown()
22 star(30, 'green')
23 t.penup()
24 t.forward(80)
25 t.left(50)
26 t.pendown()
27 star(40, 'blue')
28
29 t.hideturtle()
30 turtle.done()
```

函数定义

func_star2.py



函数的参数：可调节的选项

- 如果代码块里没有可供调节的选项，可以定义没有参数的简单函数
- 一般函数会带有可供调节的选项参数，参数可以有多个
  - 如画五角星的函数，包含两个参数：大小size和颜色color

func_star2.py

```
4   def star(size, color):
5       t.color(color)
6       t.begin_fill()
7       for i in range(5):
8           t.forward(size)
9           t.left(72)
10          t.forward(size)
11          t.right(144)
12      t.end_fill()
```

函数的返回值：得到一个结论

- 有时候函数会有返回值
  - 如math模块中求平方根的函数math.sqrt(n)返回n的平方根
- return语句负责结束函数执行，并返回值
- return语句可以根据需要，出现在语句块中的任何位置

func_sqrt.py

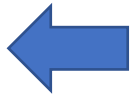
```
1   import math
2
3   s = math.sqrt(120)
4   print(s)
5
6
7   def my_abs(n):
8       if n >= 0:
9           return n
10      else:
11          return -n
12
13
14  s = my_abs(-10)
15  print(s)
```

函数定义中的代码块

- 由于函数定义def语句仅仅是把代码块“打包封装”
- def语句执行的时候，代码块并不会被执行
- 所以，在执行def语句的时候，除非语句块中包含了明显的语法错误
- Python解释器是不会检查语句块中其它错误的。

func_star2.py

```
1   import turtle
2
3
4   def star(size, color):
5       t.color(color)
6       t.begin_fill()
7       for i in range(5):
8           t.forward(size)
9           t.left(72)
10          t.forward(size)
11          t.right(144)
12      t.end_fill()
13
14
15  t = turtle.Turtle()
16
17  star(20, 'red')
```



并不会出现“t未定义”的错误

调用函数：call function

- def定义了函数之后，函数名称仅代表这个“函数对象”
- 如果需要执行语句块代码，需要有如下的要素
  - 函数名称，后加括号
  - 括号内放置参数的具体值
- 没有或者不需要返回值

```
func(a,b,c)  #如调用star
```

- 获取返回值

```
v = func(a,b,c)
```

func_call.py

```
1   def my_abs(n):
2       if n >= 0:
3           return n
4       else:
5           return -n
6
7
8   # 函数对象
9   s = my_abs
10  print("function object:", s)
11
12  # 加括号和参数，调用函数
13  s = my_abs(-10)
14  print("call function:", s)
```

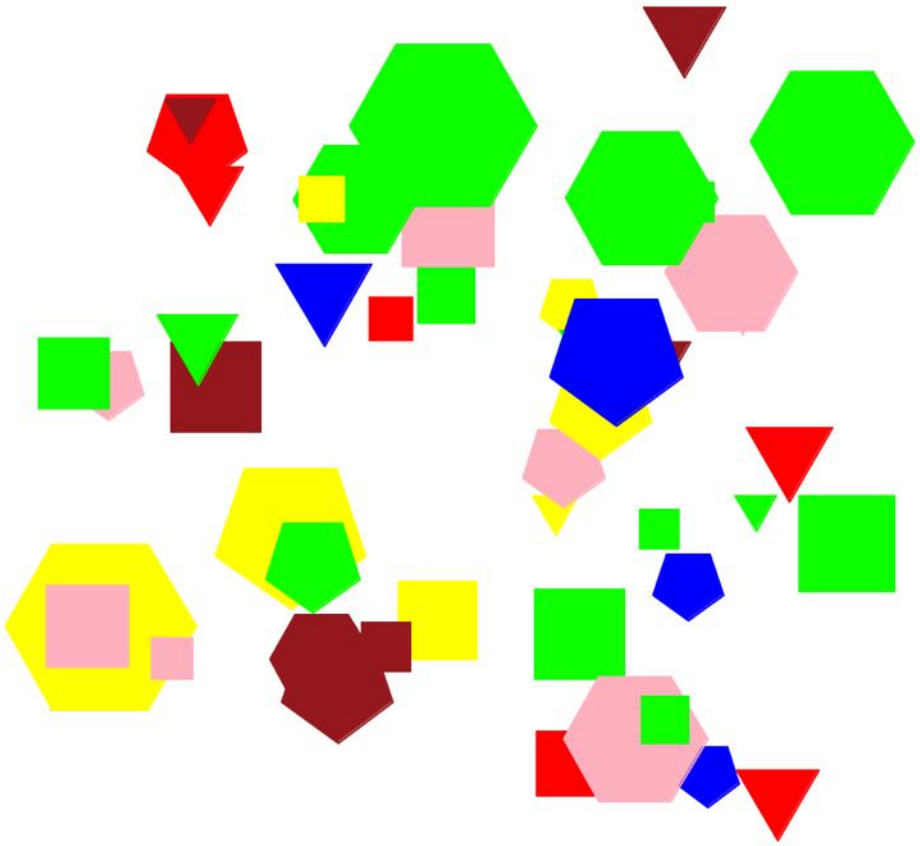
随机数模块random

- 产生一定范围内的随机数
  `random.randint(min,max)`
- 从列表中随机选择
  `random.choice(list)`

```
>>> import random
>>> colors=['red', 'green', 'blue', 'brown', 'pink']
>>> c=random.choice(colors)
>>> c
'green'
>>> n=random.randint(10,20)
>>> n
19
```

随堂作业：组合图形

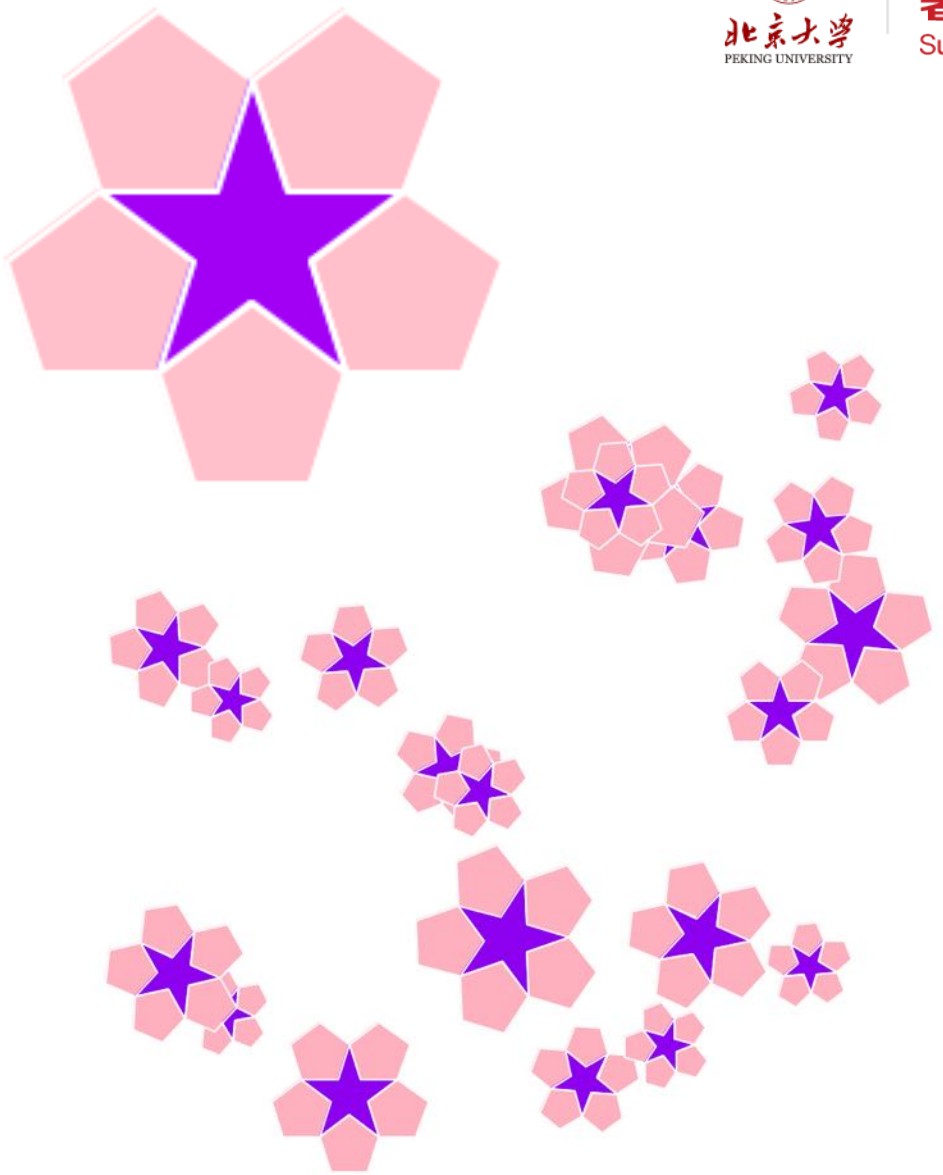
- 定义一个多边形函数
  `def polygon(n,size,color):`
  - 绘制正n边形，边长为size，填充颜色color
- 编写一个程序，绘制现代时尚几何多边形色块抽象装饰画
  - 随机模块random
  - t.goto(x,y)
- 可以进一步修改程序
  - 如：将数学曲线定义为函数，组合进随机图案中来



geom_draw.py

函数调用函数

- 只要def定义过的函数对象都可以被调用
- 也可以从函数里调用另一个函数
- 例如flower函数
  - 有参数size
  - 调用了star画紫色五角星
  - 调用了polygon画粉色五边形



程序： func_call2.py

```
1   import turtle
2   import random
3   import math
4
5
6   def polygon(n, size, color):
7       t.fillcolor(color)
8       t.begin_fill()
9       for i in range(n):
10          t.forward(size)
11          t.right(360 / n)
12      t.end_fill()
13
14
15  def star(size, color):
16      t.fillcolor(color)
17      t.begin_fill()
18      for i in range(5):
19          t.forward(size)
20          t.left(72)
21          t.forward(size)
22          t.right(144)
23      t.end_fill()
```

```
26  def flower(size):
27      for i in range(5):
28          t.forward(2 * size * (1 + math.sin(18 * math.pi / 180)))
29          t.right(36)
30          polygon(5, size, 'pink')
31          t.right(108)
32      star(size, 'purple')
33
34
35  t = turtle.Turtle()
36  t.speed(0)
37  t.pencolor('white')
38  for i in range(20):
39      t.penup()
40      x = random.randint(-200, 200)
41      y = random.randint(-200, 200)
42      t.goto(x, y)
43      t.pendown()
44      head = random.randint(1, 9)
45      t.setheading(head * 40)
46      size = random.randint(10, 25)
47      flower(size)
48  t.hideturtle()
49  turtle.done()
```

递归：函数调用自己?

- 函数可以调用自己吗?
  - 从Python语言的函数定义角度来说
  - def在前，调用在后，所以函数可以调用自己，称为“递归调用”
- 那会不会有什么问题?

recursion.py

```
1  def tell_story():
2      print("从前有座山，山上有座庙，庙里有个老和尚，他在讲：")
3      tell_story()
4
5
6  tell_story()
```

“好”的递归：必须有终止条件

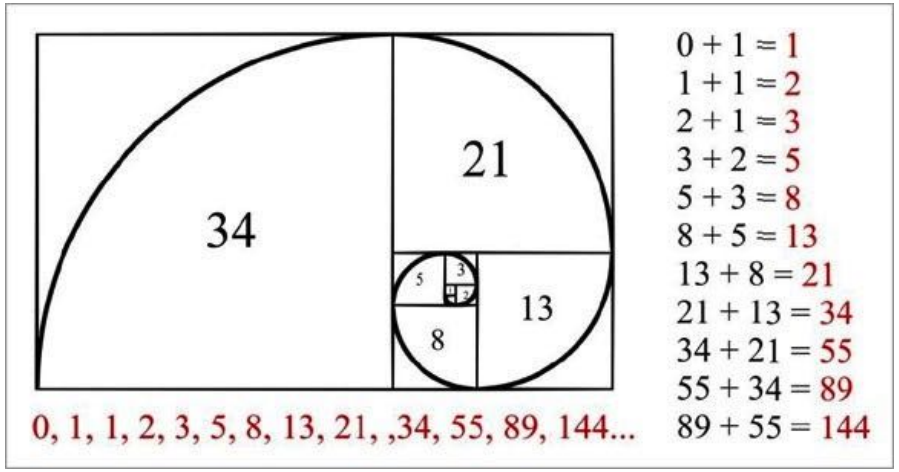
- 递归调用很有用，但要有个终止结束条件，结束时不再调用自身
- 一般是通过参数来控制，递归调用时让参数向终止条件演进
  - 例子：参数n，结束条件是n==0，递归调用时参数为n-1
  - 无论多大的n，总会变为0

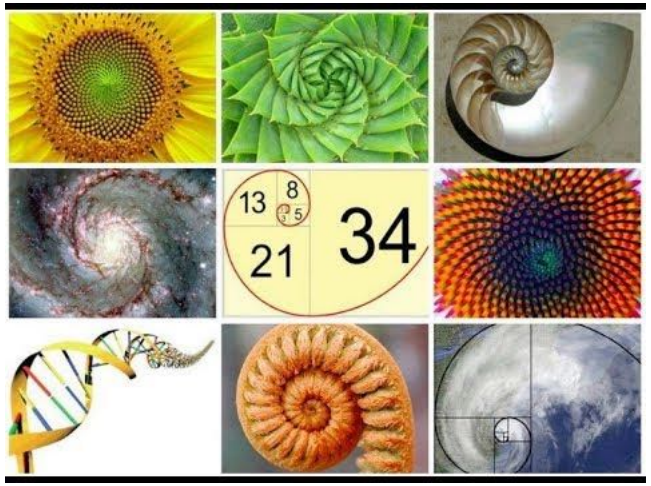
recursion2.py

```
1  def tell_story(n):
2      if n > 0:
3          print("从前有座山，山上有座庙，庙里有个老和尚，他在讲：")
4          tell_story(n - 1)
5      else:
6          print("讲完了！")
7
8
9  tell_story(10)
```

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？

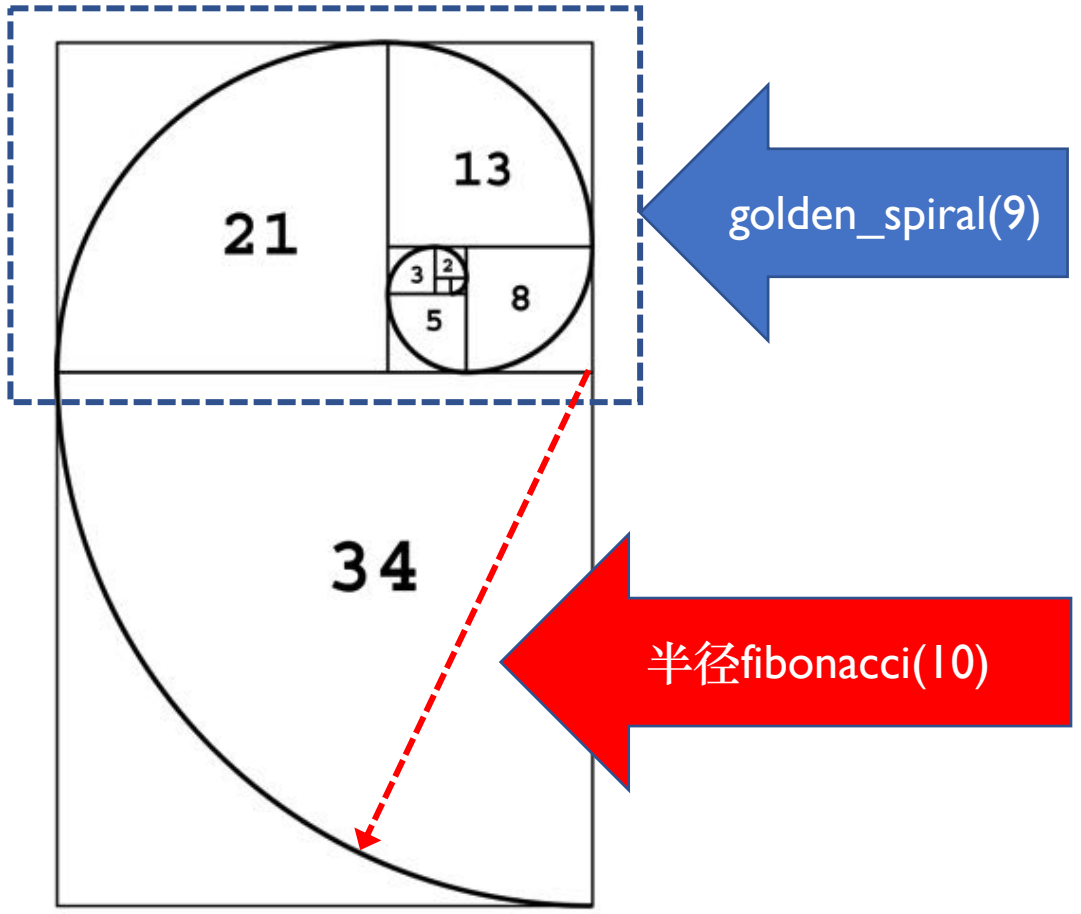
$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}, n > 1$$

fibonacci.py

```
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))
```

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)



程序： fibonacci2.py

- 看看如何写出golden_spiral函数？ （代码已遮挡）
- 注意：递归结束条件

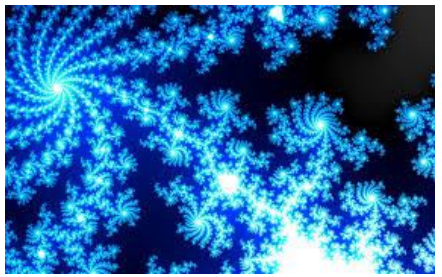
fibonacci2.py

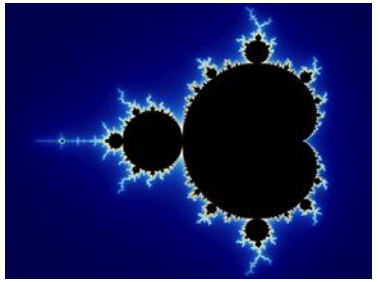
```
1   import turtle
2
3
4   def fibonacci(n):
5       if n == 1:
6           return 0
7       elif n == 2:
8           return 1
9       else:
10          return fibonacci(n - 1) + fibonacci(n - 2)
11
12
13  def golden_spiral(n):
14      
15
16
```

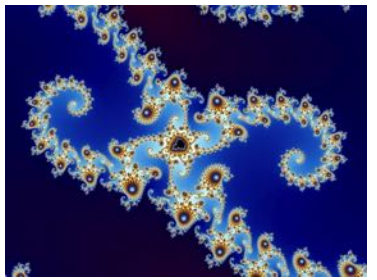
```
19  t = turtle.Turtle()
20
21  golden_spiral(12)
22
23  t.hideturtle()
24  turtle.done()
```

分形树：自相似递归图形

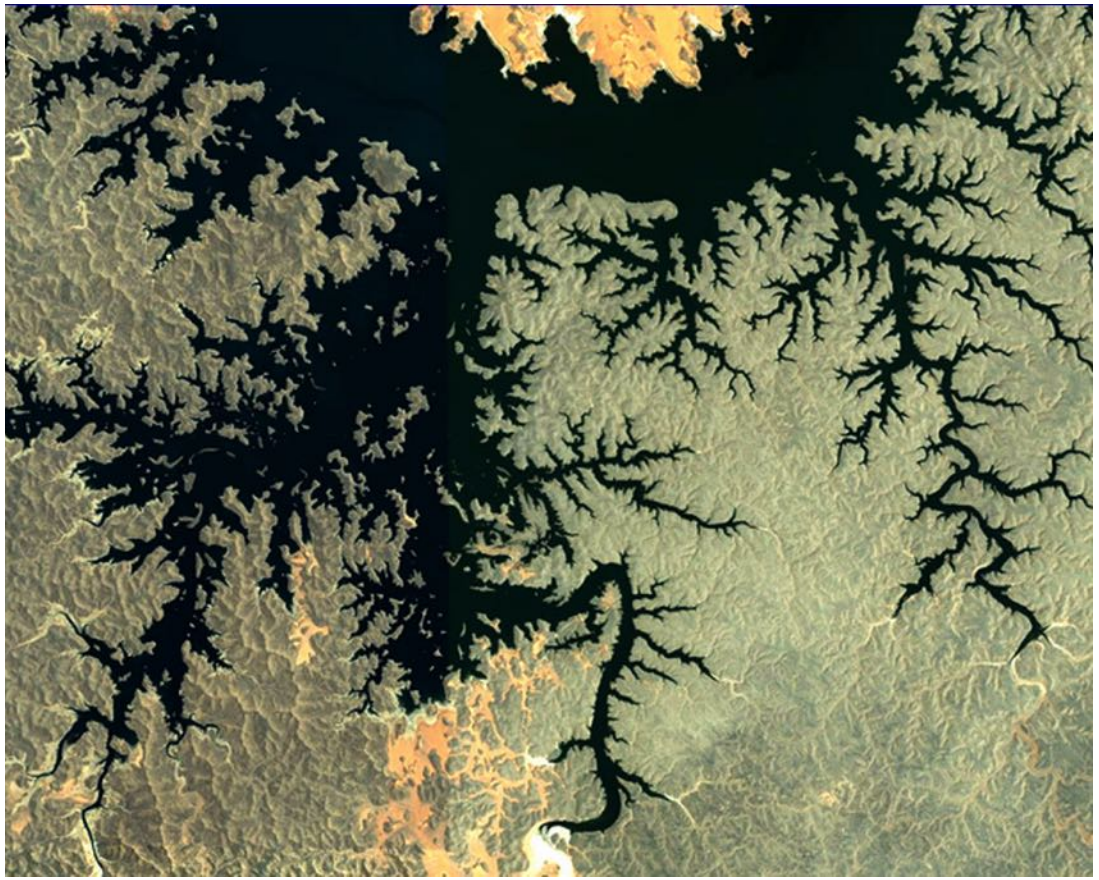
- 分形Fractal，是1975年由Mandelbrot开创的新学科
- 通常被定义为“一个粗糙或零碎的几何形状，可以分成数个部分，且每一部分都（至少近似地）是整体缩小后的形状”，即具有自相似的性质。
- 自然界中能找到众多具有分形性质的物体
  - 海岸线、山脉、闪电、云朵、雪花、树
  - http://paulbourke.net/fractals/googleearth/
  - http://recursivedrawing.com/

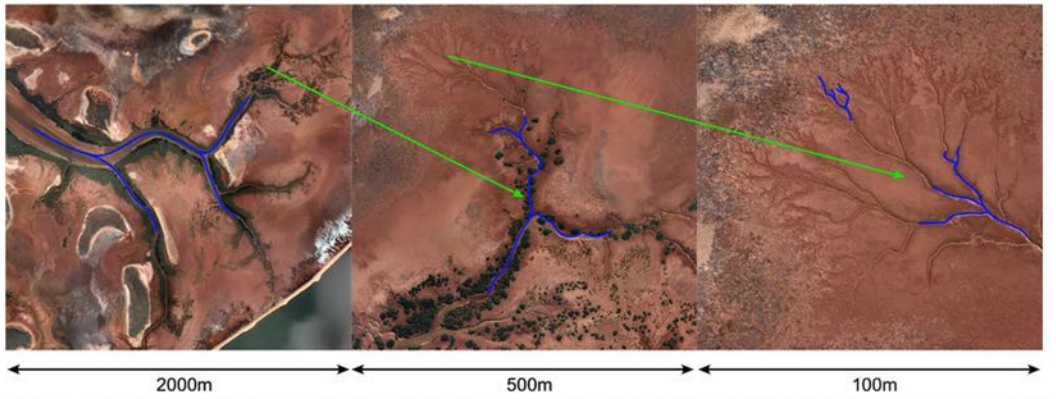






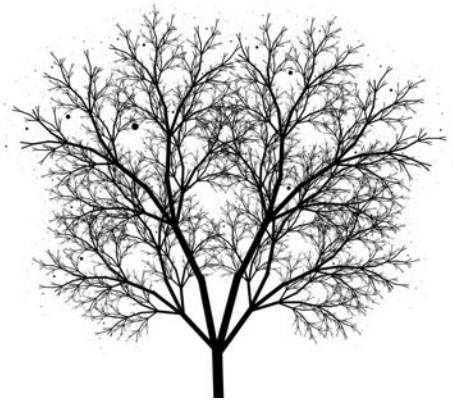
自相似的河流/海岸线





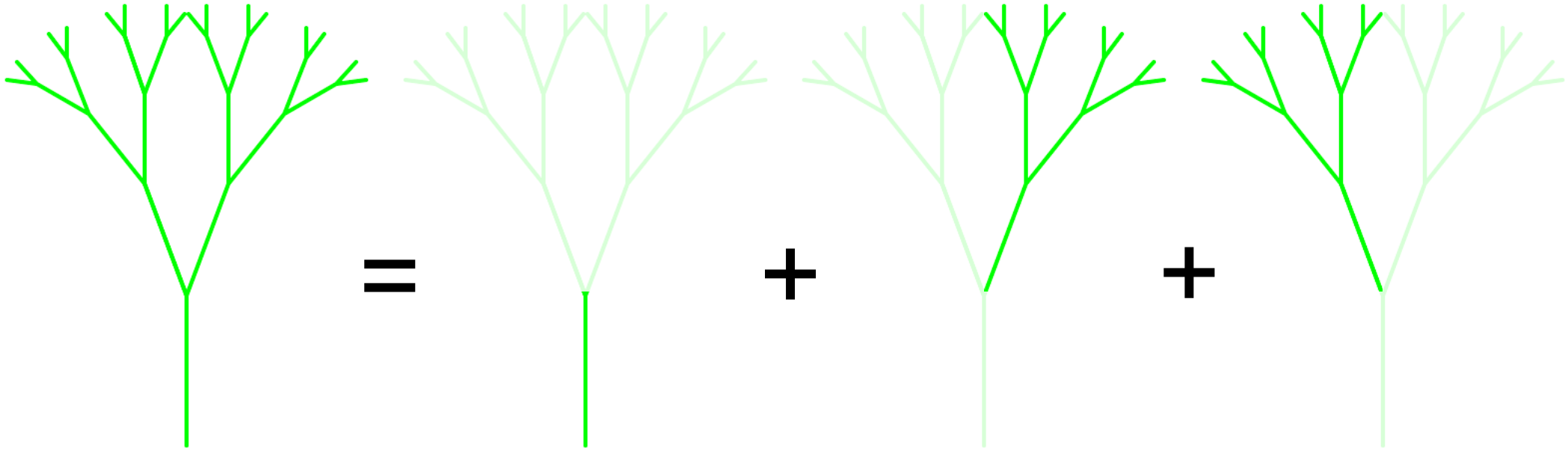
分形树：自相似递归图形

- 自然现象中所具备的分形特性，使得计算机可以通过分形算法生成非常逼真的自然场景，下面我们以树为例做一个粗糙的近似
- 我们发现，一棵树的每个分叉和每条树枝，实际上都具有整棵树的外形特征（也是逐步分叉的）



二叉树的递归分解

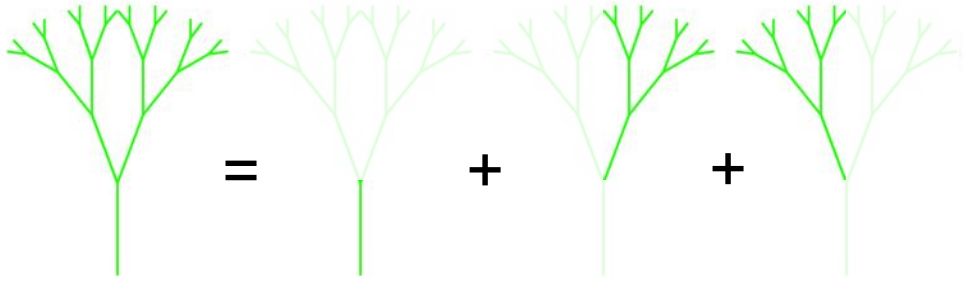
- 可以把二叉树分解为三个部分：树干、左边的小树、右边的小树
  - 这样的分解，正好符合递归的定义：对自身的调用



二叉树　　树干　　倾斜的右小树　　倾斜的左小树

程序： tree.py

```
1   import turtle
2
3
4   def tree(branch_len):
5       if branch_len > 5:   # 树干太短不画，即递归结束条件
6           t.forward(branch_len)   # 画树干
7           t.right(20)   # 右倾斜20度
8           tree(branch_len - 15)   # 递归调用，画右边的小树，树干减15
9           t.left(40)   # 向左回40度，即左倾斜20度
10          tree(branch_len - 15)   # 递归调用，画左边的小树，树干减15
11          t.right(20)   # 向右回20度，即回正
12          t.backward(branch_len)   # 海龟退回原位置
```



二叉树　　树干　　倾斜的右小树　　倾斜的左小树

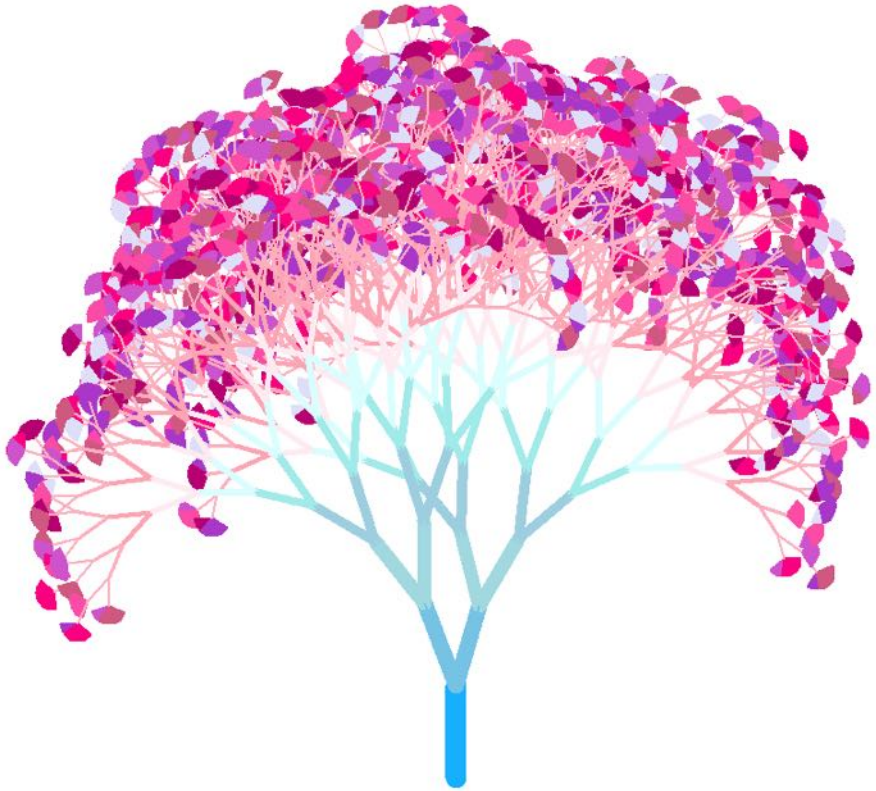
```
15  t = turtle.Turtle()
16  t.left(90)
17  t.penup()
18  t.backward(100)
19  t.pendown()
20  t.pencolor('green')
21  t.pensize(2)
22  tree(75)   # 画树干长度75的二叉树
23  t.hideturtle()
24  turtle.done()
```

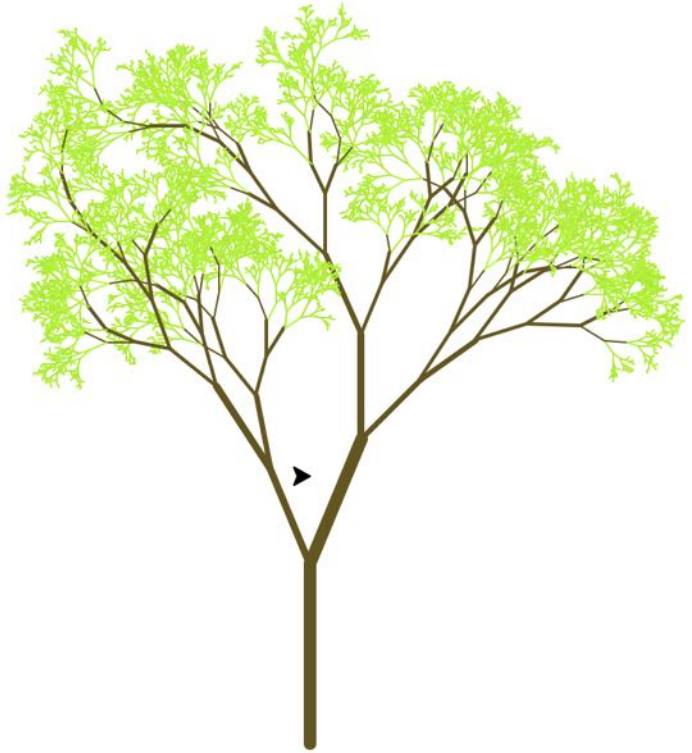
【H7】艺术分形树

- 修改分形树程序，增加如下功能：
  - 树枝的粗细可以变化，随着树枝缩短，也相应变细
  - 树枝的颜色可以变化，当树枝非常短的时候，使之看起来像树叶的颜色，或者花、果；
  - 让树枝倾斜角度在一定范围内随机变化，如15～45度之间，左右倾斜也可不一样，做成你认为最好看的样子
  - 树枝的长短也可以在一定范围内随机变化，使得整棵树看起来更加逼真

学生作业精选：分形树

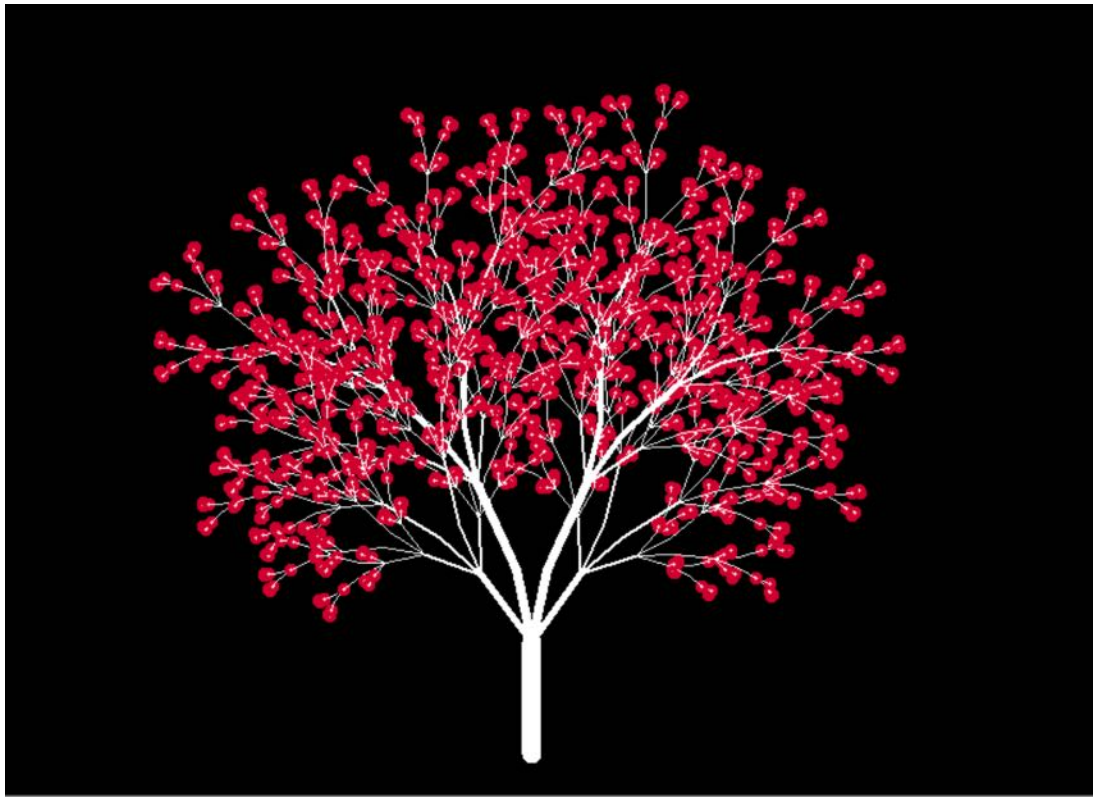
学生作业精选：分形树

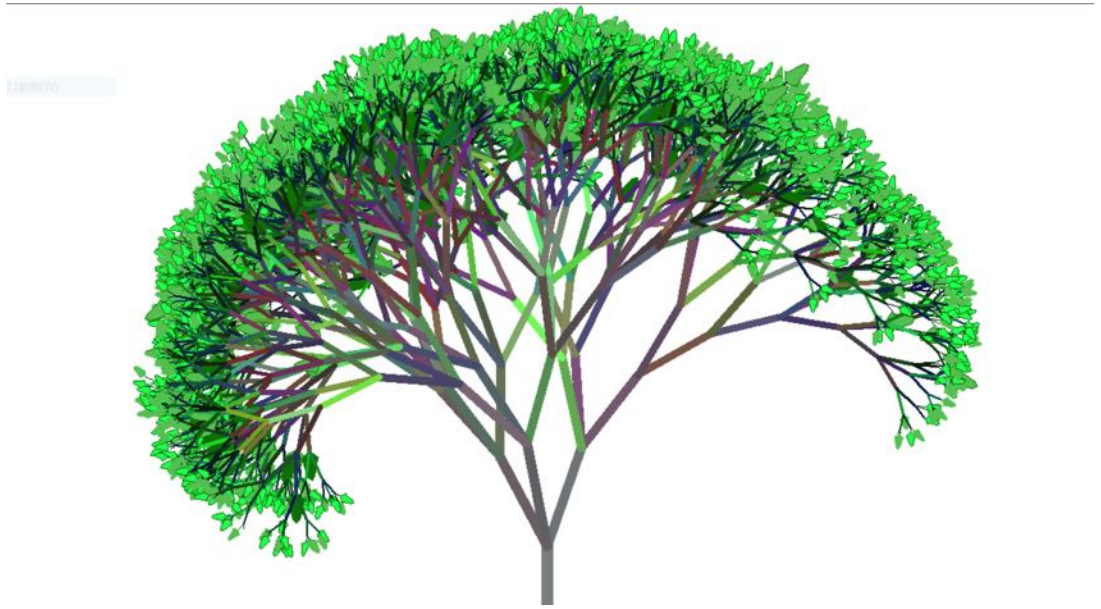




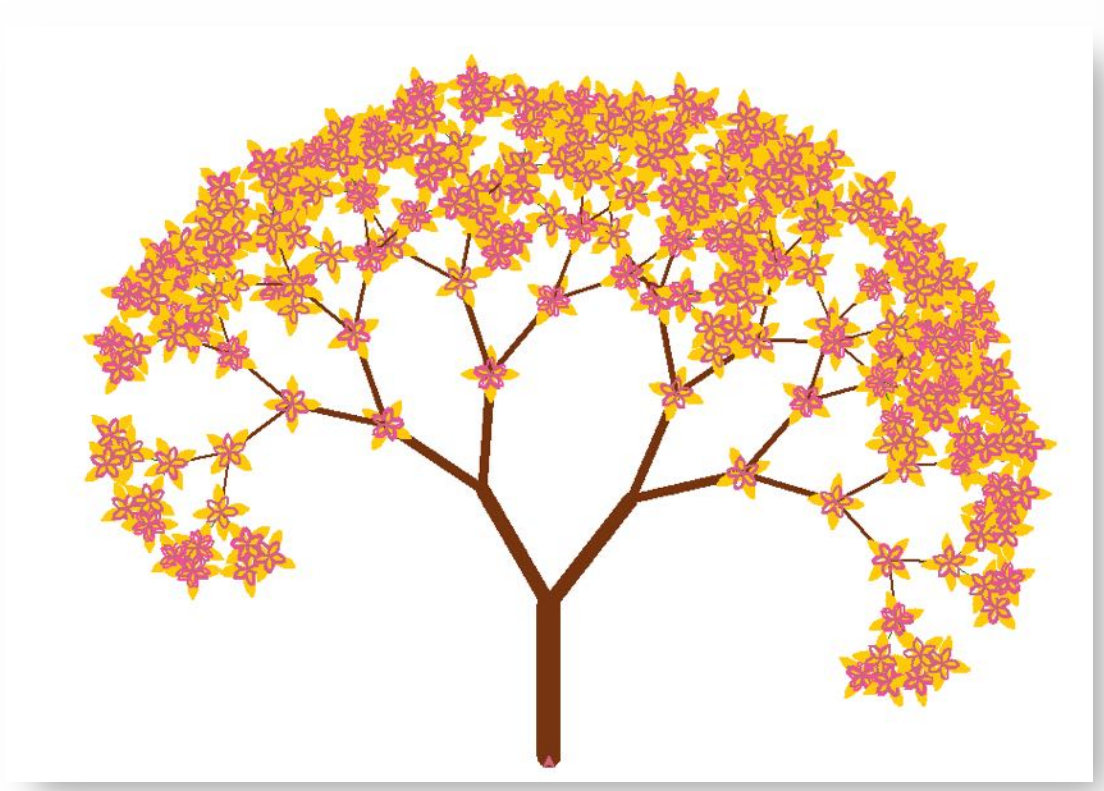
学生作业精选：分形树

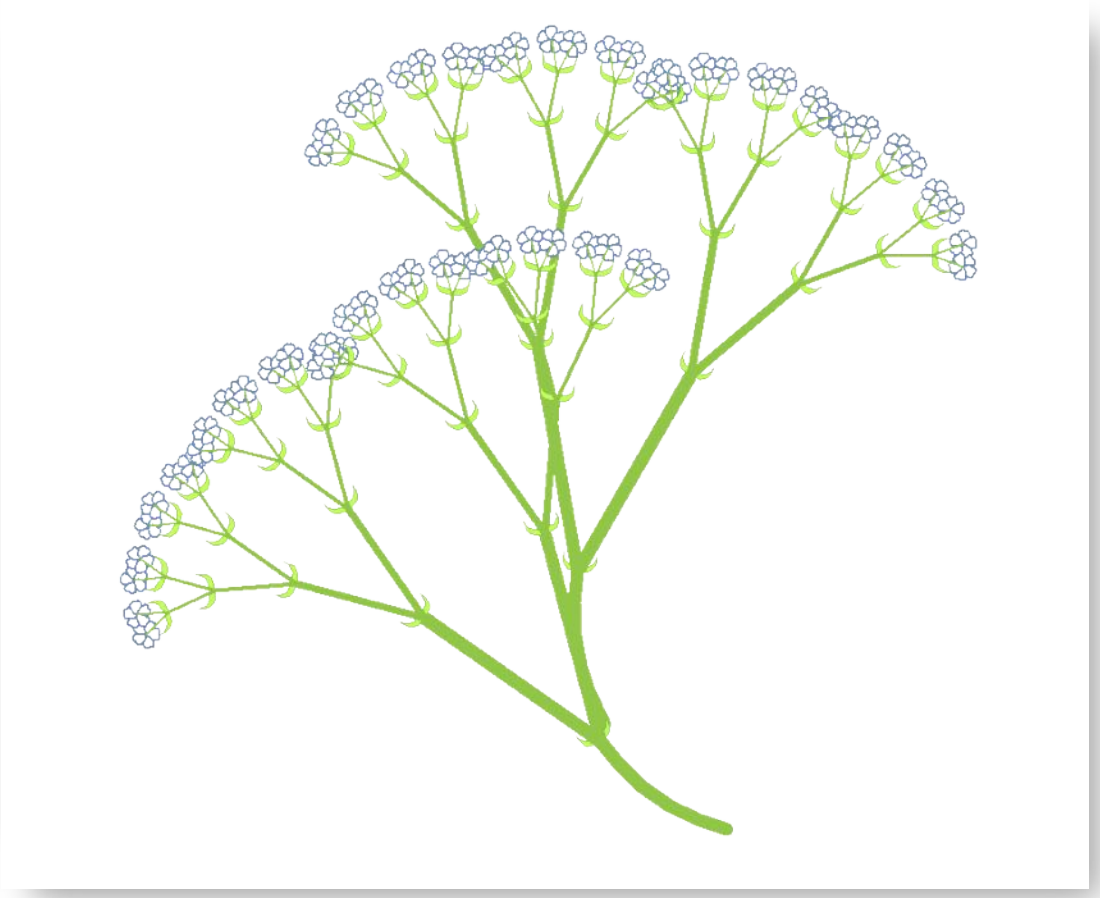
学生作业精选：分形树





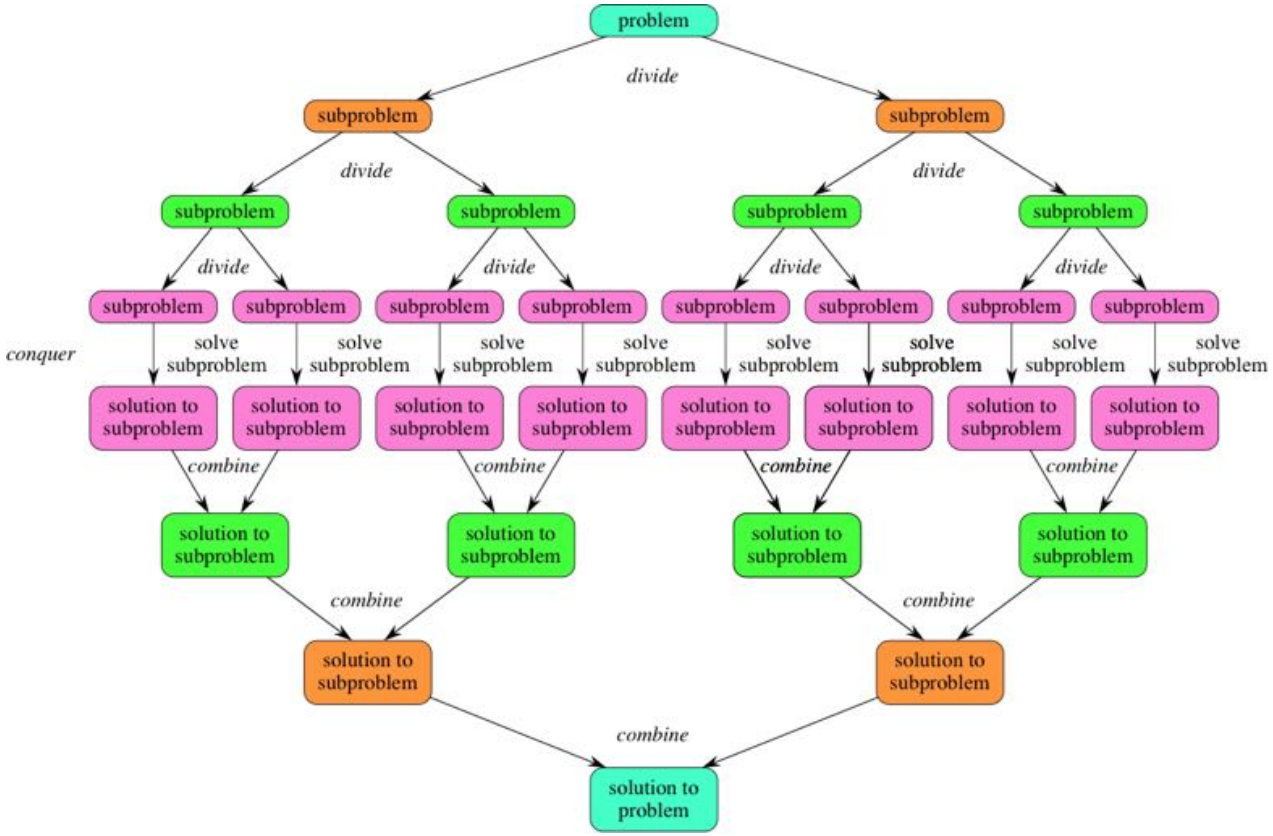
学生作业精选：分形树





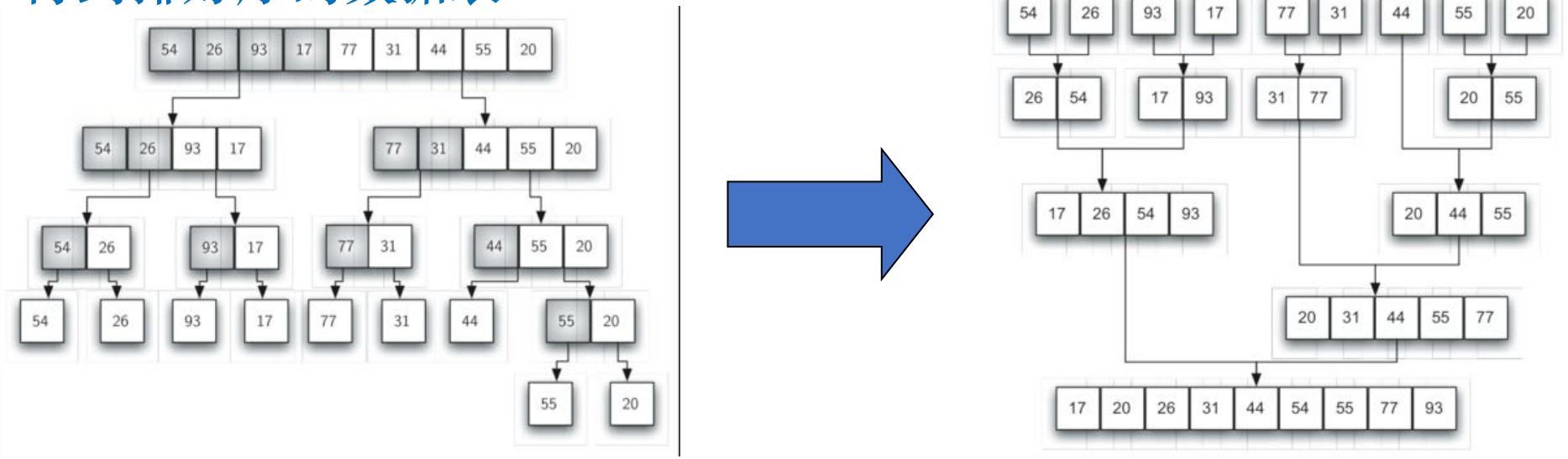
分治策略：分而治之Divide and Conquer

- 将复杂问题分解为几个规模更小问题的组合，是一种直观而又巧妙的问题解决手段
- “大事化小，小事化了”
- 将问题的规模缩到足够小的时候，许多问题变得显而易见，甚至无需解决方案
  - 求和？排序？查找？
- 分解-解决-合并



归并排序Merge Sort

- 归并排序是递归算法，思路是将数据表持续分裂为两半，对两半分别进行归并排序，再把两半进行合并。
  - 递归的基本结束条件是：数据表仅有1个数据项，自然是排好序的；
  - 缩小规模：将数据表分裂为相等的两半，规模减为原来的二分之一；
  - 调用自身：将两半分别调用自身排序，然后将分别排好序的两半进行归并，得到排好序的数据表。



归并排序

```python
# merge sort
# 归并排序

def merge_sort(lst):
    # 递归结束条件
    if len(lst) <= 1:
        return lst

    # 分解问题，并递归调用
    middle = len(lst) // 2
    left = merge_sort(lst[:middle])  # 左半部排好序
    right = merge_sort(lst[middle:])  # 右半部排好序

    # 合并左右半部，完成排序
    merged = []
    while left and right:
        if left[0] <= right[0]:
            merged.append(left.pop(0))
        else:
            merged.append(right.pop(0))

    merged.extend(right if right else left)
    return merged


data_lst = [6, 202, 100, 301, 38, 8, 1]
print(merge_sort(data_lst))
```

算法的分析与评价

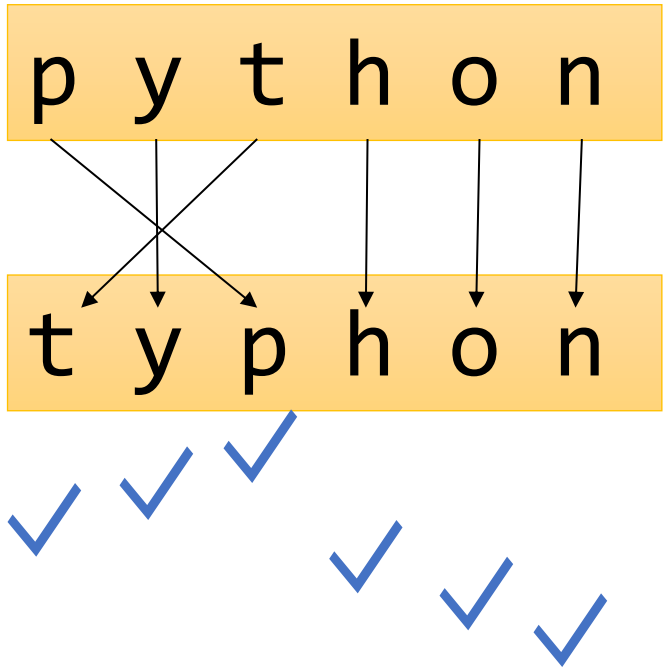
- 一个问题可以有多种解法

“变位词”判断问题

- “变位词”判断问题可以很好地展示不同数量级的算法
- 问题描述
  - 所谓“变位词”是指两个词之间存在组成字母的重新排列关系
  - 如：heart和earth，python和typhon
  - 为了简单起见，假设参与判断的两个词仅由小写字母构成，而且长度相等
- 解题目标：写一个bool函数，以两个词作为参数，返回真假，表示这两个词是否变位词

解法1：逐字检查

- 解法思路为将字符串1中的字符逐个到字符串2中检查是否存在，存在就“打勾”标记（防止重复检查），如果每个字符都能找到，则两个词是变位词，只要有1个字符找不到，就不是变位词
- 程序技巧
  - 实现“打勾”标记：将字符串2对应字符设为None
  - 由于字符串是不可变类型，需要先复制到列表中



```
1   def anagramSolution1(s1, s2):
2       alist = list(s2)
3       pos1 = 0
4       stillOK = True
5       while pos1 < len(s1) and stillOK:
6           pos2 = 0
7           found = False
8           while pos2 < len(alist) and not found:
9               if s1[pos1] == alist[pos2]:
10                  found = True
11              else:
12                  pos2 = pos2 + 1
13          if found:
14              alist[pos2] = None
15          else:
16              stillOK = False
17          pos1 = pos1 + 1
18      return stillOK
19
20
21  print(anagramSolution1('abcd', 'dcba'))
```

复制s2到列表

循环s1的每个字符

在s2列表逐个对比

找到，打勾

未找到，失败

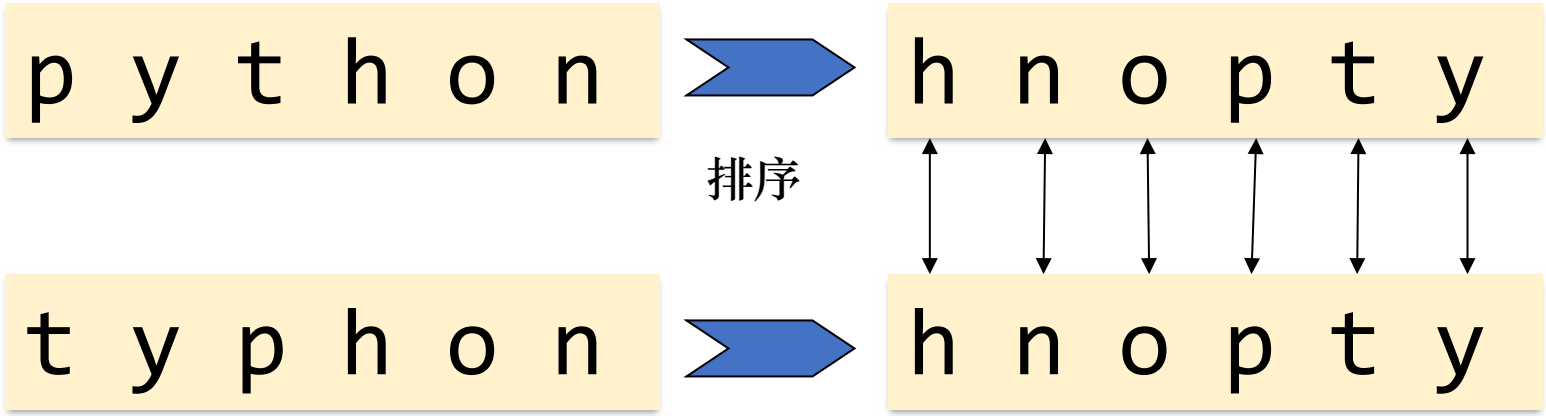
解法1：逐字检查-算法分析

- 问题规模：词中包含的字符个数n
- 主要部分在于两重循环
- 外重循环要遍历字符串1每个字符，将内层循环执行n次
- 而内重循环在字符串2中查找字符，每查找一个字符的操作次数，分别是1、2、3……n中的一个，而且各不相同
- 所以总执行次数是1+2+3+……+n
- 可知其数量级为O(n^2)

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$$

解法2：排序比较

- 解题思路为：将两个字符串都按照字母顺序排好序，再逐个字符对比是否相同，如果相同则是变位词



```python
def anagramSolution2(s1, s2):
    alist1 = list(s1)
    alist2 = list(s2)

    alist1.sort()
    alist2.sort()
    pos = 0
    matches = True
    while pos < len(s1) and matches:
        if alist1[pos] == alist2[pos]:
            pos = pos + 1
        else:
            matches = False
    return matches


print(anagramSolution2('abcde', 'edcba'))
```

分别都排序

逐个对比

解法2：排序比较-算法分析

- 粗看本算法只有一个循环，最多执行n次，数量级应该是O(n)
- 但循环前面的两个sort并不是无代价的
- 如果查询下sort的时间复杂度，会发现排序算法采用不同的解决方案，其运行时间数量级差不多是O(n log n)，大过循环的O(n)
- 所以本算法中其决定性作用的步骤是排序步骤
- 本算法的运行时间数量级就等于排序过程的数量级O(n log n)

解法3：暴力法

- 暴力法解题思路为：穷尽所有可能组合
- 对于变位词问题来说，暴力法具体是，将字符串1中出现的字母进行全排列，再查看字符串2是否出现在全排列列表中
- 这里最大的困难是产生字符串1所有字母的全排列，根据组合数学的结论，如果n个字符进行全排列，其所有可能的字符串个数为n!
- 我们已知n!的增长速度甚至超过2^n
  - 如对于20个字符长的词来说，将产生20!=2,432,902,008,176,640,000 个候选词
  - 如每秒处理一个词，需要77,146,816,596年（百亿）的时间来做完所有的匹配。
- 结论：暴力法恐怕不能算是个好算法

解法4：计数比较

- 最后一个算法解题思路为：对比两个字符串中每个字母出现的次数，如果26个字母出现的次数都相同的话，这两个字符串就一定是变位词
- 具体做法：为每个字符串设置一个26位的计数器，先检查每个字符串，在计数器中设定好每个字母出现的次数
- 计数完成后，进入比较阶段，看两个字符串的计数器是否相同，如果相同则输出是变位词的结论

```
1   def anagramSolution4(s1, s2):
2       c1 = [0] * 26
3       c2 = [0] * 26
4       for i in range(len(s1)):
5           pos = ord(s1[i]) - ord('a')
6           c1[pos] = c1[pos] + 1
7       for i in range(len(s2)):
8           pos = ord(s2[i]) - ord('a')
9           c2[pos] = c2[pos] + 1
10      j = 0
11      stillOK = True
12      while j < 26 and stillOK:
13          if c1[j] == c2[j]:
14              j = j + 1
15          else:
16              stillOK = False
17      return stillOK
18
19
20  print(anagramSolution4('apple', 'pleap'))
```

分别都计数

计数器比较

解法4：计数比较-算法分析

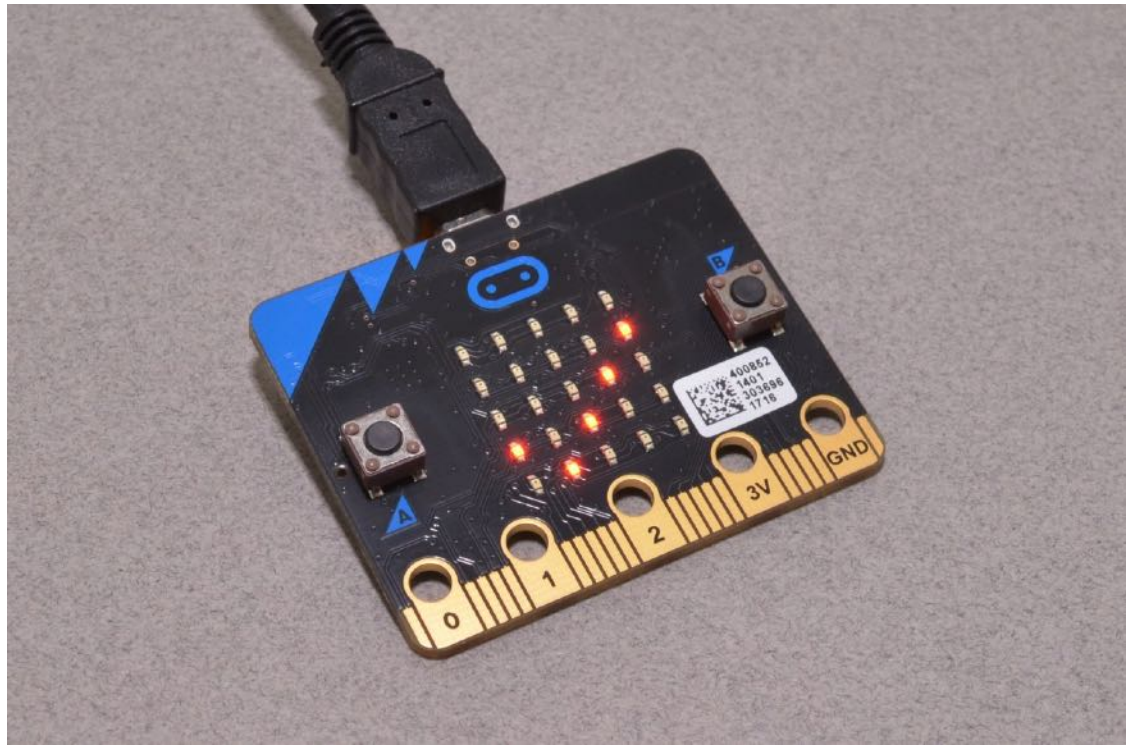
- 计数比较算法中有3个循环迭代，但不象解法1那样存在嵌套循环
  - 前两个循环用于对字符串进行计数，操作次数等于字符串长度n
  - 第3个循环用于计数器比较，操作次数总是26次
- 所以总操作次数T(n)=2n+26，其数量级为O(n)，这是一个线性数量级的算法，是4个变位词判断算法中性能最优的。

解法4：计数比较-算法分析

- 值得注意的是，本算法依赖于两个长度为26的计数器列表，来保存字符计数，这相比前3个算法需要更多的存储空间
  - 由于仅限于26个字母构成的词，本算法对空间额外的需求并不明显
  - 但如果考虑由大字符集构成的词（如中文具有上万不同字符），情况就会有所改变。
- 牺牲存储空间来换取运行时间或者相反，这种在时间空间之间的取舍和权衡，在选择问题解法的过程中经常会出现。

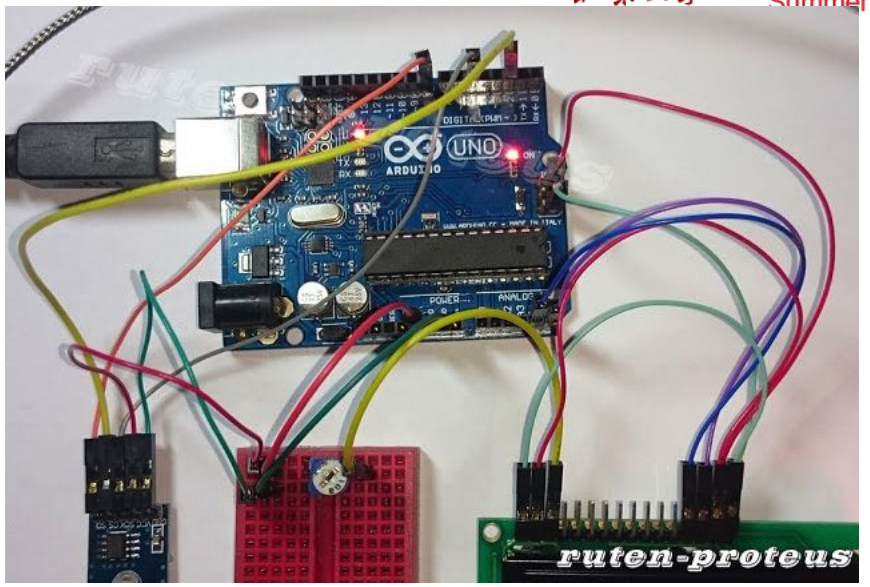
目录：microbit开源硬件

- 开源硬件教学创新
- 初识microbit-micropython
- microbit硬件规格
- microbit基础编程



传统开源硬件教学

- 以arduino、51等单片机为主
- 用C语言编程，门槛较高
- 杜邦线、面包板连接，需要熟悉基本的电子电路知识，容易出错
- 程序逻辑与底层硬件端口访问混合，代码结构不清晰

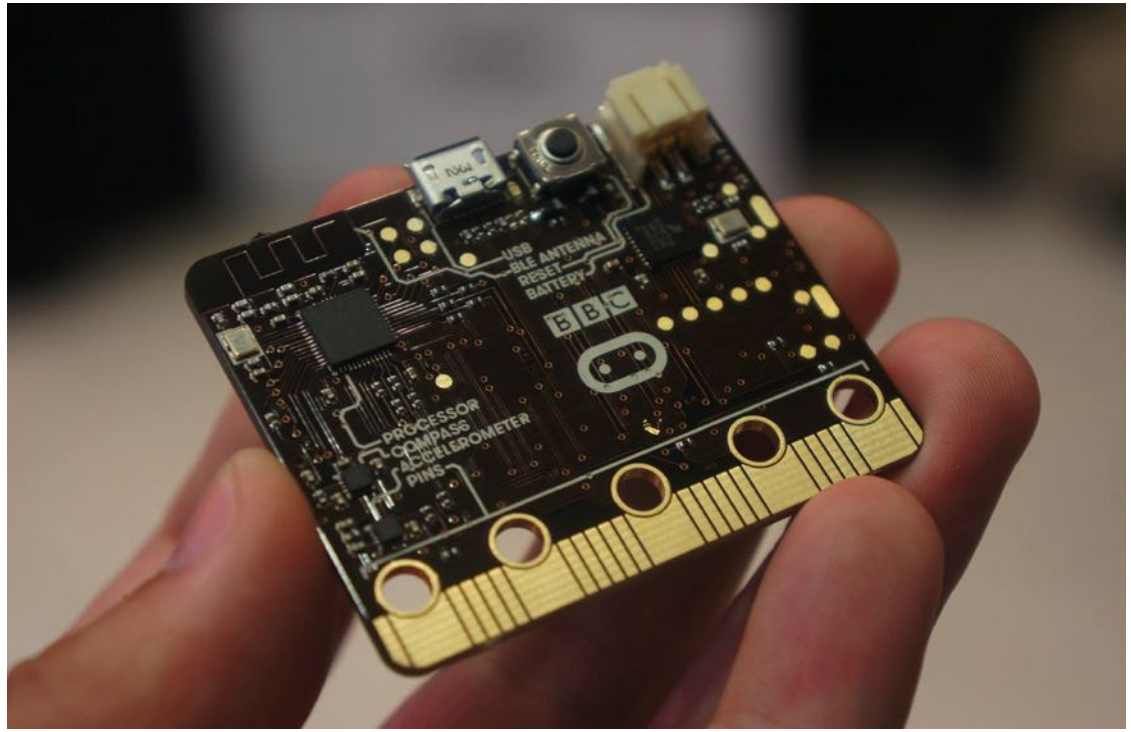



```
165 void loop() {
166   // 確認 TC 是否開路或是故障
167   if( is_tc_open )
168   {
169     static uint8_t tc_check;
170     Serial.println( "TC is Open !");
171     #ifdef USEI2CLCD
172       displayCharOnLCD( 1, 1, "  TC is Open !  ", 16 );
173       displayCharOnLCD( 2, 1, "                ", 16 );
174       char a = 0x35 - (char)tc_check;
175       displayCharOnLCD( 2, 8, &a , 1 );
176     #endif
177
178     // 每 5 秒鐘重新確認 TC 是否已重新連接上
179     if( tc_check ++ > 3 )
180     {
181       max6675_getCelsius();
182       tc_check = 0;
183     }
184   }
185   else
186   {
187     float C = max6675_getCelsius();
188     float F = max6675_getFahrenheit();
189     Serial.print( C );
190     Serial.println(" C");
191     Serial.print( F );
192     Serial.println(" F");
193     #ifdef USEI2CLCD
194       char buf[17];
```

```
39  void clearLCD()
40  {
41
42    Wire.beginTransmission(ADDRI2CLCD);
43
44    Wire.write( 0x80 ); // One command
45    Wire.write( 0x01 ); // Clear display
46
47    Wire.endTransmission();
48  }
49
50  void displayCharOnLCD( int line, int column,
51  {
52    unsigned char i;
53
54    Wire.beginTransmission(ADDRI2CLCD);
55
56    Wire.write( 0x80 );
57    Wire.write( 0x80 + ( line - 1 ) * 0x40 + ( c
58    Wire.write( 0x40 );
59
60    for( i = 0; i < len; i++)
61    {
62      Wire.write( *dp++ );
63    }
64
65    Wire.endTransmission();
66  }
```

通过开源硬件进行编程语言教学

- 学习编程语言，而非学单片机
- 利用硬件的实物特性，声光电多种反馈形式，提高学习兴趣
- 需要消除单片机编程的繁杂
- 建立良好的层次抽象，凸显主要的程序逻辑
- 培养计算思维，编程解决问题
- Programming not coding

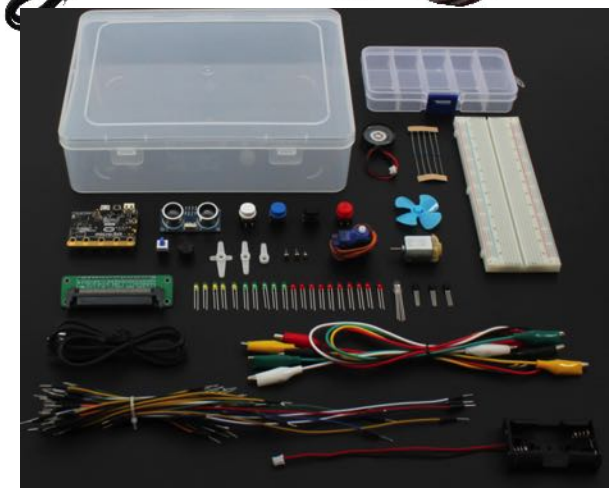


MicroPython： Python语言教学利器


```
1  from microbit import *
2
3  while True:
4      if accelerometer.was_gesture('shake'):
5          display.show(Image.HAPPY)
6          sleep(1000)
7          display.clear()
```

- 在单片机上直接使用Python语言编程
- 面向对象、动态、脚本语言的特性，使得访问硬件端口非常简单
- 有效隔离了底层繁琐的细节，对传感器、效应器等访问也非常便捷
- 大大降低单片机操作门槛，提高编程语言学习兴趣

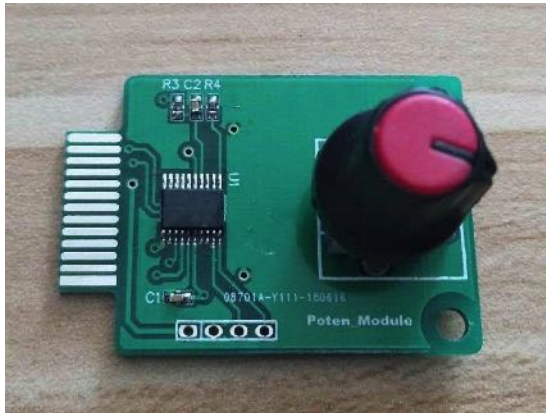
micro:bit的各种扩展板套件



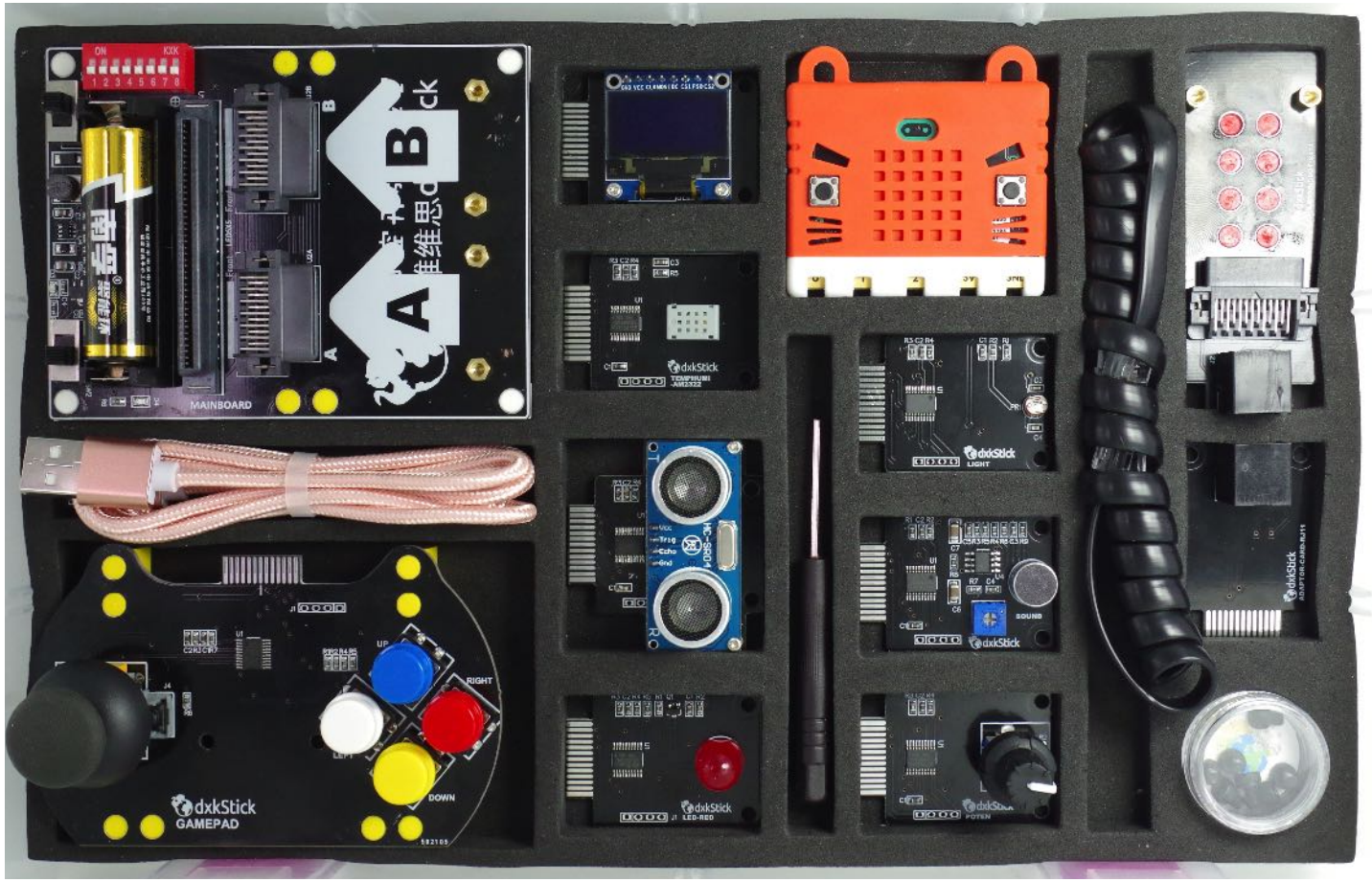
- 多数仍然是杜邦线加面包板
- 杂乱不易打理
- 主要面向图形化编程，对Python语言支持度不高
- 对传感器抽象程度较低，仍然面向引脚读取数值

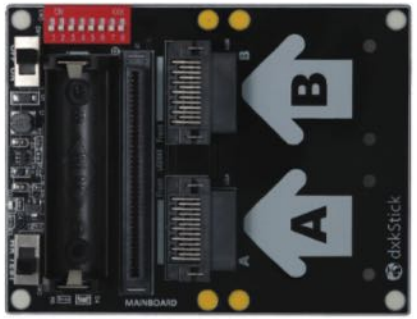
地小空开放实验室dxkStick扩展板套件

- 全插卡式传感器模块，消除杜邦线带来的杂乱困扰
- 完全面向Python语言编程教学
- 通过高层次指令访问传感器，程序逻辑清晰
- 支持多个单片机之间的分组无线通信
- 兼容乐高LEGO构件尺寸
- 开放的指令系统，易于扩展



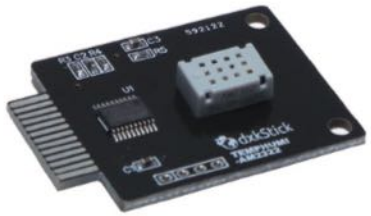
dxkStick Python语言教学套件





DSK000
MAINBOARD
主板
x1

DSK001
DISPLAY-OLED12864
12864OLED显示屏
x1



DSK002
TEMPHUMI-AM2322
温湿度传感器
x1

DSK003
LED-RED
单头LED红色
x1

DSK004
POTEN
模拟电位器
x1

DSK005
LIGHT
光敏电阻
x1

DSK006
SOUND
MIC声音传感器 x1

DSK007
GAMEPAD
游戏手柄 x1

DSK008
ULTRASONIC-SR04
超声波测距 x1

DSK900
MICROBIT
microbit单片机 x1

DSK901
ADAPTOR-CARD-RJ11
转接卡RJ11 x1

DSK902
ADAPTOR-SLOT-RJ11
转接槽RJ11 x1

DSK903
WIRE-20CM-RJ11
20厘米RJ11六芯线 x1

DSK904
WIRE-MICROUSB
microUSB线 x1

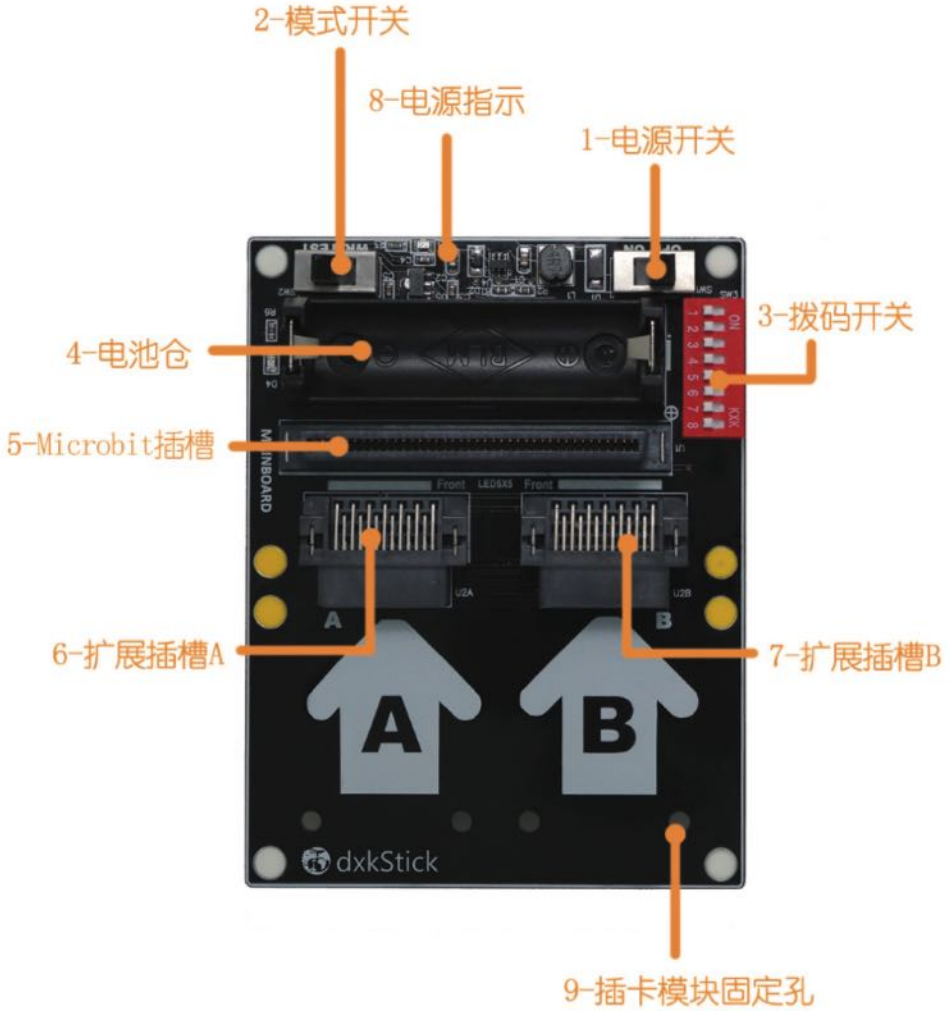
DSK905
SI-CASE
硅胶保护套 x1

DSK906
HEX-SCREWDRV
内六角螺丝刀 x1

DSK907
SCREW-BOX
螺丝盒子 x1

3.0 连接和开启

3.1 主板安装设置



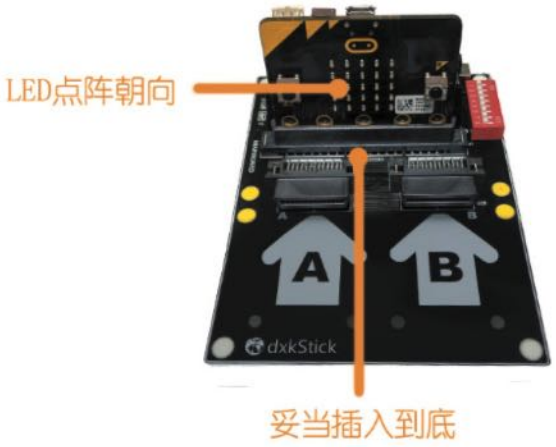
序号-名称	功能
1-电源开关	开启ON-关闭OFF电源
2-模式开关	切换工作WR-测试TEST模式
3-拨码开关	设置（1-5）无线通信频道CH；（6-8）同频道分组GRP 以二进制进行编码，ON的方向为1，小数字为低位。
4-电池仓	安装5号电池，注意正负极标志
5-Microbit插槽	安装microbit单片机，标注LED5X5为点阵的朝向
6-扩展插槽A	可以插入各型号插卡模块
7-扩展插槽B	可以插入各型号插卡模块
8-电源指示	电源开启后会持续发 光
9-插卡模块固定孔	可用螺丝，将插卡模块强力固定在主板上

电池安装供电

本套件主板可以由各种类型的5号电池供电，包括：碳性电池、碱性电池、镍氢电池、锂电池、磷酸铁锂电池。从安全和经济的角度，推荐使用磷酸铁锂电池。

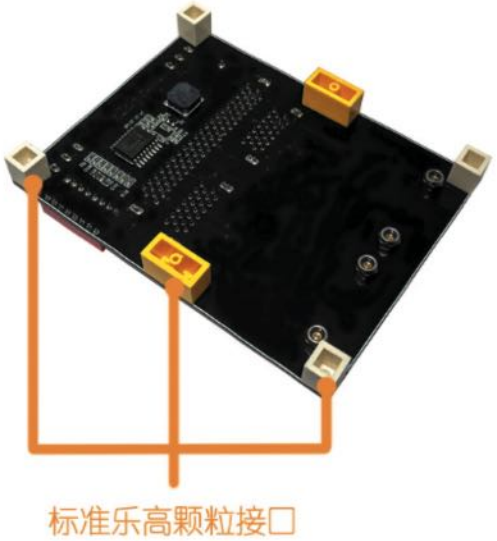
电池在安装到电池仓时，需注意正负极标志，不能反接，并注意电量充足，否则主板将无法正常工作。

Microbit单片机安装



本套件主板的microbit插槽有方向性，但由于microbit单片机没有防呆设计，所以需要辨认单片机上LED点阵的朝向，小心地将microbit单片机妥当插入到插槽底部。

固定到乐高模型



本套件主板的四角和长边中央都已经预装了乐高标准颗粒，用户可以方便地将主板固定到各种乐高拼插积木搭成的模型上，增加学习乐趣，提高套件创造性。

3.2 插卡模块安装

插卡模块安装和固定

本套件主板上有两个插卡插槽，分别标注为A和B，插卡模块可以插入任何一个插槽中。在每个插槽的旁边都有两个固定铜柱，在需要特别加固的情况下，可以用套件附带的内六角螺丝将插卡模块牢牢固定在主板上。

插卡模块的延伸

某些插卡模块的物理尺寸较大，无法插入主板上的插卡插槽，或者需要将模块安装到主板之外的空间时，可以使用模块延伸配件。模块延伸配件包括一个插在主板上的RJ11转接卡，一段RJ11接头的螺旋线，以及一个带有插卡插槽的RJ11转接槽。

由于采用螺旋线延伸插卡模块增加了插卡脱落的风险，建议用内六角螺丝将RJ11转接卡固定在主板上。

固定到乐高模型

本套件主板的四角和长边中央都已经预装了乐高标准颗粒，用户可以方便地将主板固定到各种乐高拼插积木搭成的模型上，增加学习乐趣，提高套件创造性。

3.3 无线联网设置

本套件支持microbit的radio无线编程，并通过自主开发的套件python模块，将主板和各插卡模块的功能融合到microbit无线联网的功能，让初学者能够不需要了解复杂的网络通信知识，就能实现远程通信编程，极大丰富了编程教学的案例和套件的适用范围。

无线联网编程模式需要用到多个dxkStick套件，其中一个套件作为主控节点，其余套件作为受控节点，由主控节点上的程序调用受控节点，获取其连接的传感器信息，或在其连接的显示屏上显示信息。

相同无线频道的节点才可以互相通信。套件主板上的8位拨码开关用于设置无线通信的参数。实际使用中，用户可以将1-5位设置为无线通信的频道，频道号按照二进制设定的整数范围为0—31，在同一个教室中的学生最多可以分为32组做实验而不会互相干扰。

3.3 无线联网设置

同一频道的几个套件还可以进一步分为几个组，主控节点可以单独控制某个组的受控节点。在一般使用中，可以将每个受控节点分成不同的组，这样就能够单独读取某一个受控节点的传感器数值了，比如，分别读取放置在窗外的温度节点，和放置在门口的温度节点。

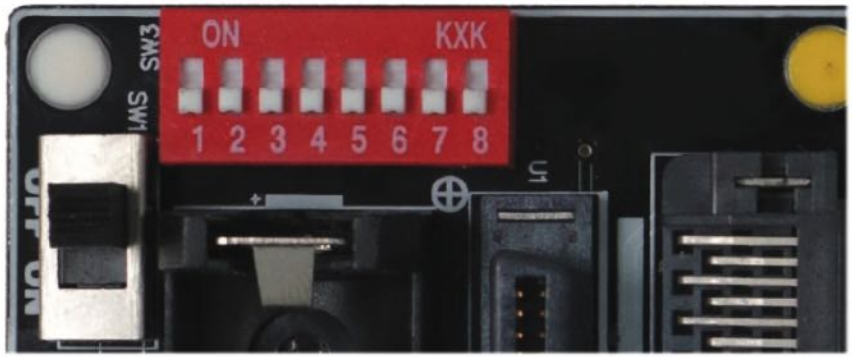
拨码开关编号小的为二进制的低位，拨到ON表示1，否则是0。其中1—5位是频道号，6—8位是组号。所以如下的拨码设置为二进制“10010011”，表示频道号19，和组号4。

频道号：19

5	4	3	2	1
1	0	0	1	1

组号：4

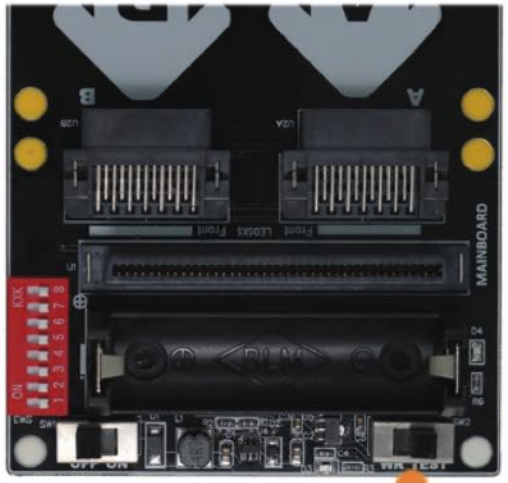
8	7	6
1	0	0

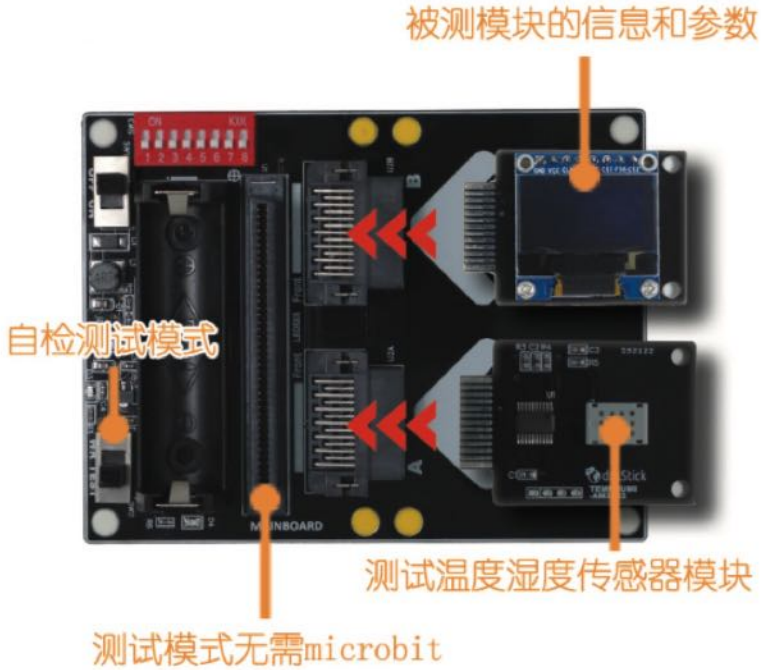


用户设置完拨码开关，即可按照设置的组号进行通信，频道号对用户程序是透明的，无需记录和写入程序，套件底层系统会自动读取频道号来初始化无线通信子系统。

4.0 模块自检

4.1 关于自检模式





自检模式是本套件的特色，用户无需编程就可以检查套件各组件工作是否正常。要启用自检模式，无需安装microbit单片机，但需要将主板上的模式选择开关拨到“TEST”。

OLED显示屏上显示的自检信息分为4行，分别显示套件名称、插卡模块的ID和类型type、模块名称和相应的参数。

自检步骤如下：

1. 将电源开关置于OFF；
2. 安装好电池；
3. 将模式开关置于TEST；
4. 将OLED显示屏模块插入任意一个插槽；
5. 将需要测试的插卡模块插入另一个插槽；
6. 打开电源开关；
7. 观察OLED显示屏上显示的信息。

通过自检测试模式，用户可以确定模块是否完好，同时，也可以抄录插卡模块的ID，用于高级开发编程。

4.2 各模块自检信息

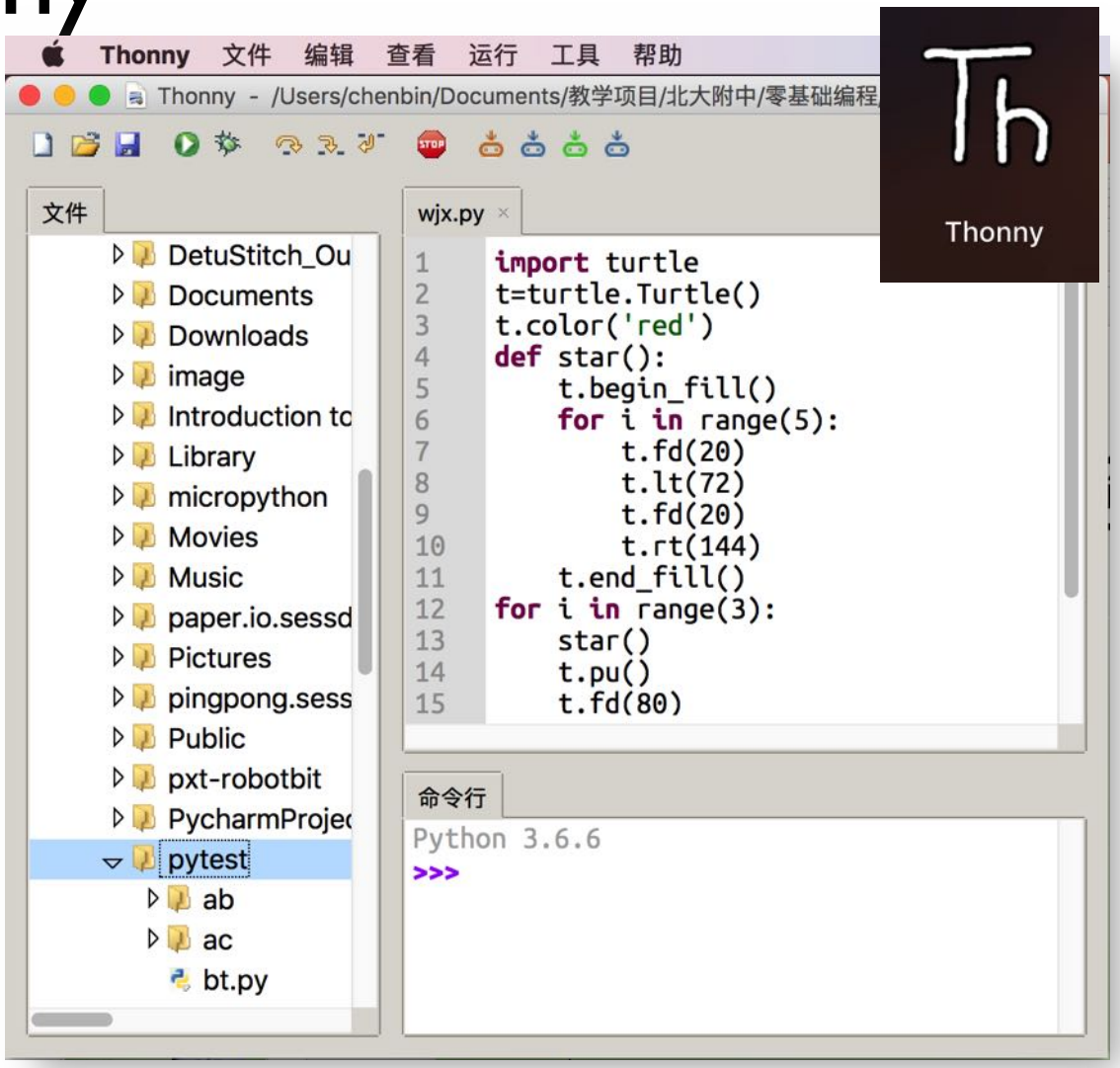
不同的插卡模块会显示不同的自检测试信息，列表说明如下：

代码	模块名称	自检测试信息
DSK001	128640LED 显示屏	（不接其它模块即为自检测试） 显示分辨率，提示内置中文字库
DSK002	温湿度传感器	显示当前的温度值和湿度值
DSK003	单头 LED 红色	显示 ON/OFF，同时闪烁
DSK004	模拟电位器	显示电位器采样值（0-4095），可以旋转旋钮测试
DSK005	光敏电阻	显示当前电阻值，光照越强，电阻值越小
DSK006	MIC 声音传感器	显示当前检测的声音强度
DSK007	游戏手柄	显示每个按钮按下的情况，以及转轴的采样值
DSK008	超声波测距	显示最近障碍的距离（单位 cm）

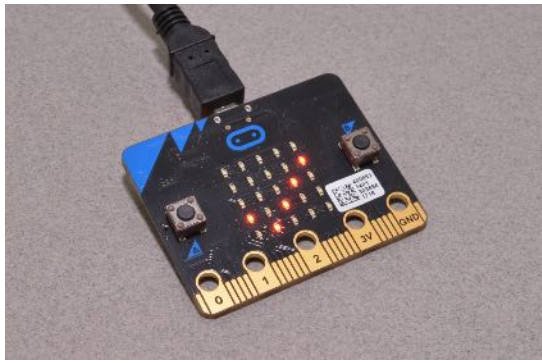
提示：游戏手柄和超声波测距模块可以用模块延伸配件延长测试。

集成开发环境：Thonny

- 跨平台Windows/macOS/Linux
- 自带最新版本Python3，无需安装Python
- 体积小巧，功能齐全
- 安装第三方模块很方便
- 可以连接microbit单片机编程
- 地小空开放实验室汉化版本
  - https://github.com/chbpku/dxkStickIDE/tree/master/Setup



初识microbit-micropython



- USB线连接microbit
- 打开Thonny
- 输入程序，保存为py文件
- 点击“写入运行环境”
  - 稍等一下，闪烁结束
  - 出现成功Done字样
- 点击“写入当前代码”
  - 出现成功Done字样
- 立刻运行，可按RESET重启

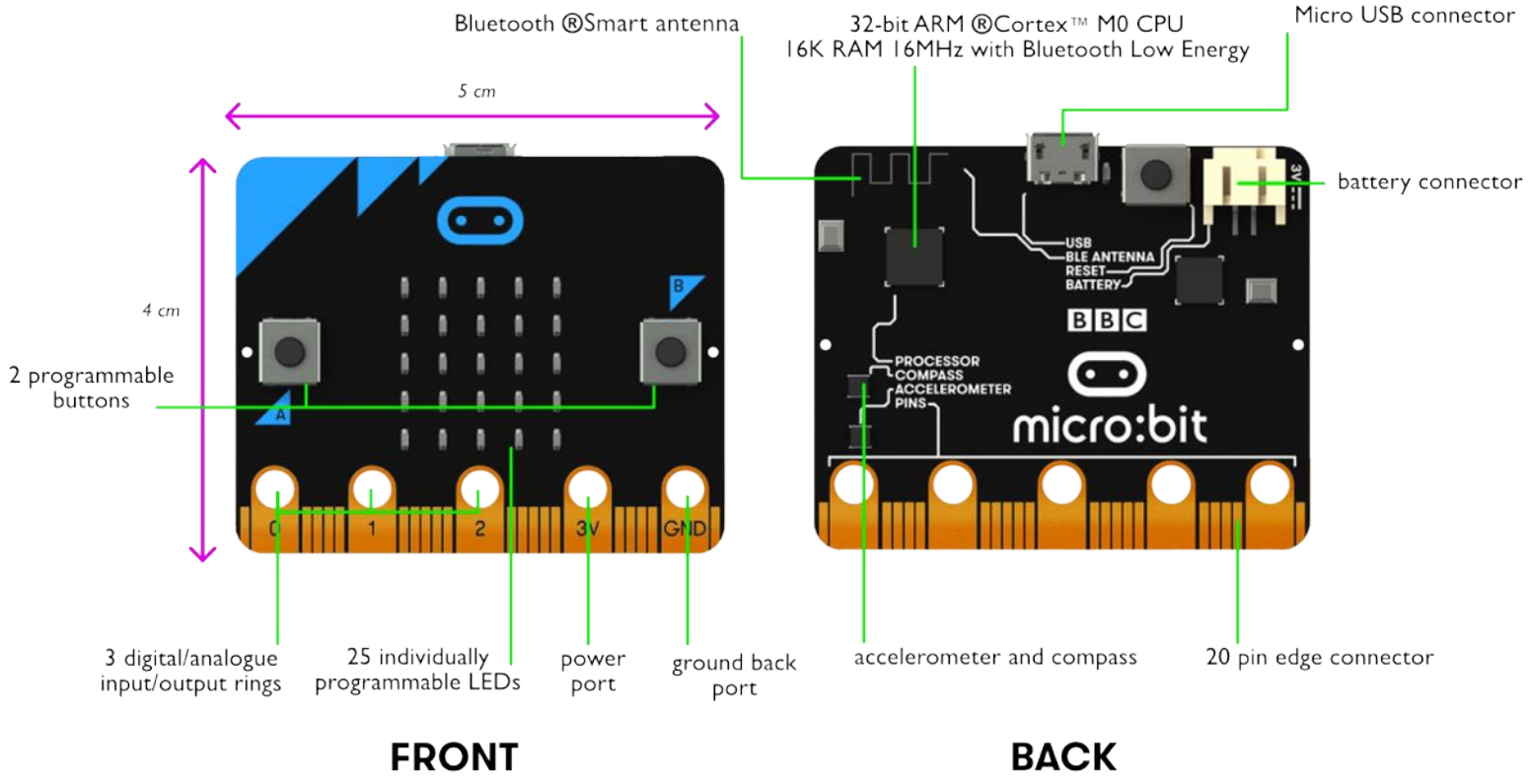
实例1：电子骰子

```
from microbit import *
from random import randint

while True:
    display.show('*')
    if accelerometer.was_gesture('shake'):
        display.clear()
        display.show(str(randint(1,6)))
        sleep(1000)
    sleep(10)
```

实例2：萤火虫

```
from microbit import *
from random import *
import radio

display.show(Image.HAPPY)
f = [Image().invert()*(i/9) for i in range(9, -1, -1)]

radio.on()

while True:
    if button_a.was_pressed():
        radio.send('flash')  # 按键发送呼叫信息
    r = radio.receive()
    if r == 'flash':
        sleep(randrange(1000)+50)
        display.show(f, delay=100, wait=False)
        if randrange(10) <= 0:  #回应的机会0-9，数字越大，机会越大
            sleep(500)
            radio.send('flash')
```

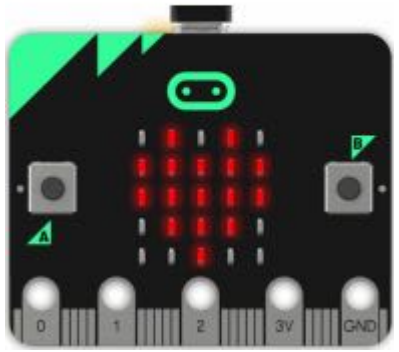
microbit from BBC介绍



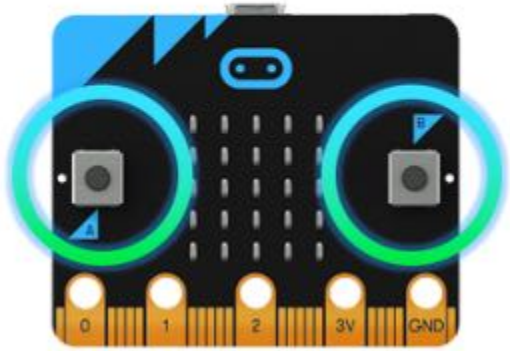
- 25个独立编程的LED
- 2个可编程的按钮
- 1个reset按钮
- microUSB接口
- 3V电源接口
- 光线传感器、温度传感器
- 加速计、电子罗盘
- 无线通信：射频以及蓝牙

microbit概貌

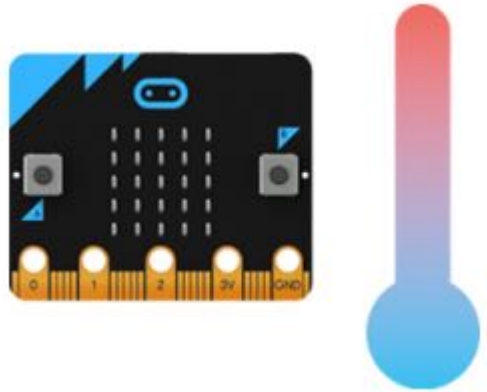
LED



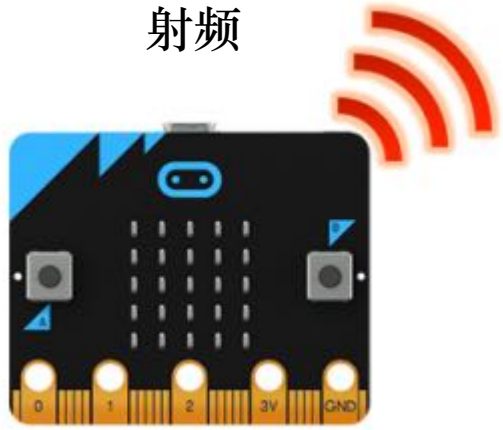
按钮传感器



温度传感器



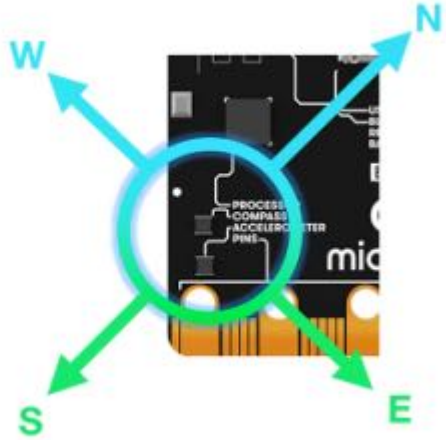
射频



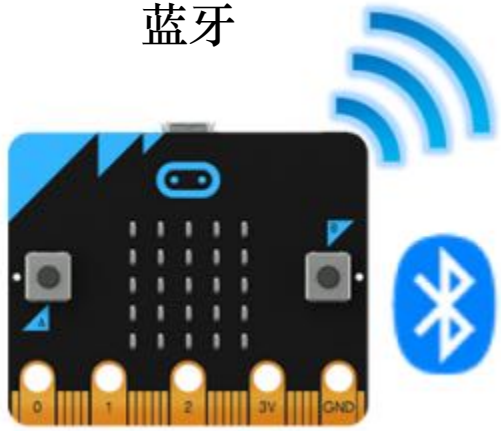
光线传感器

加速度计

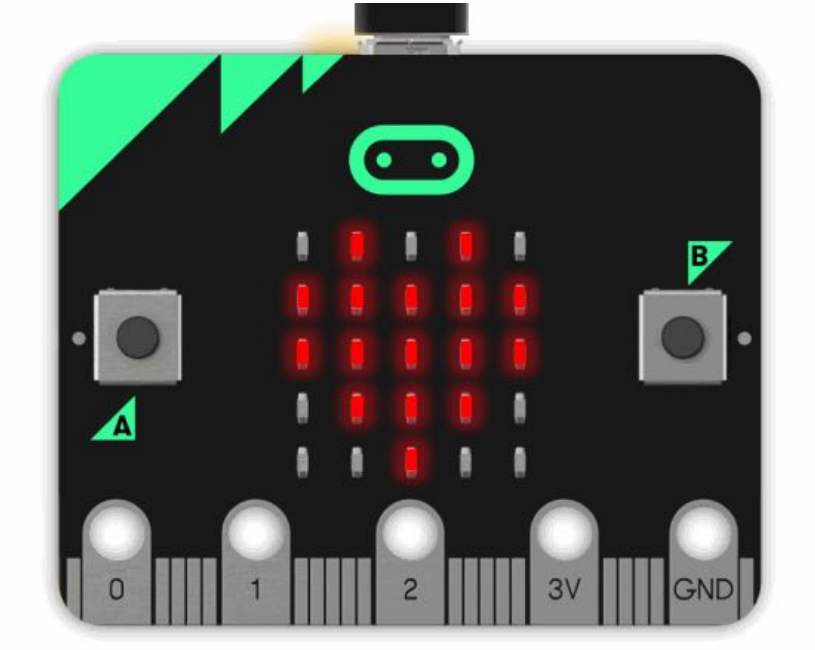
电子罗盘



蓝牙

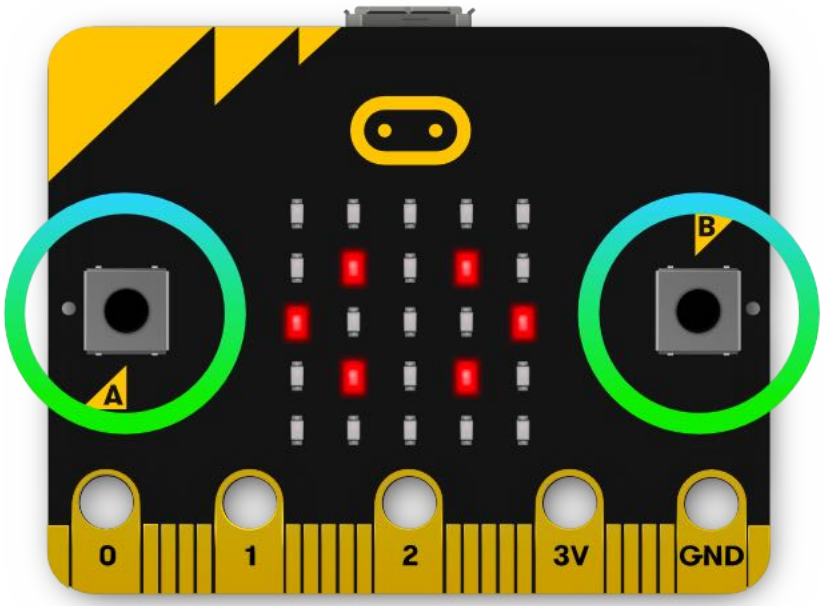


microbit特征： LED



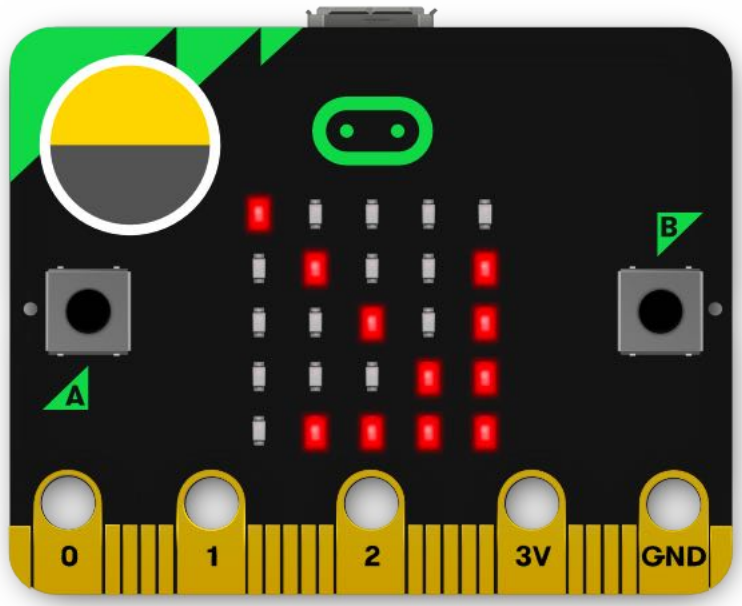
- micro:bit有25颗可独立编程的LED灯
- 可以用来显示文本，数字以及图像

microbit特征：按钮



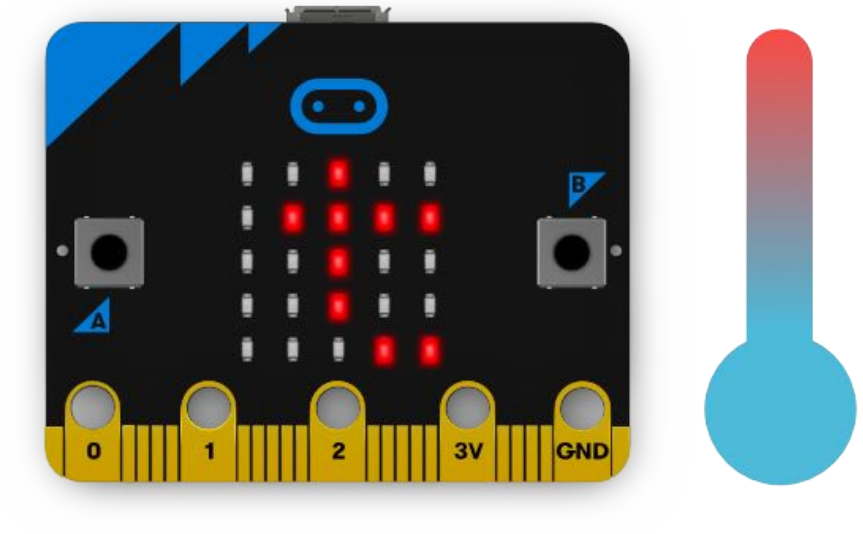
- 在micro:bit板子前面有2个按钮（标记了A和B）
- 可以检测按下这些按钮，运行代码
- 还可以检测这些按钮被按下的时间和次数

microbit特征：光线传感器



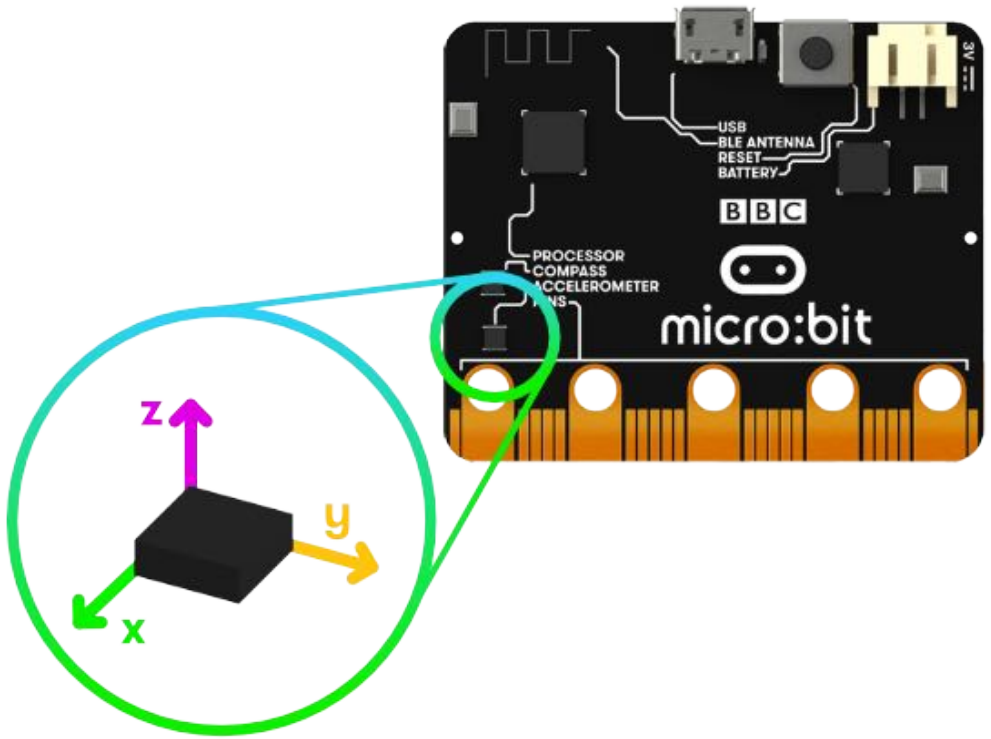
- 通过反转LED屏幕，micro:bit进入输入模式
- LED屏幕起到一个基础的光线传感器的作用
- 可以用来检测周围的光线

microbit特征：温度传感器



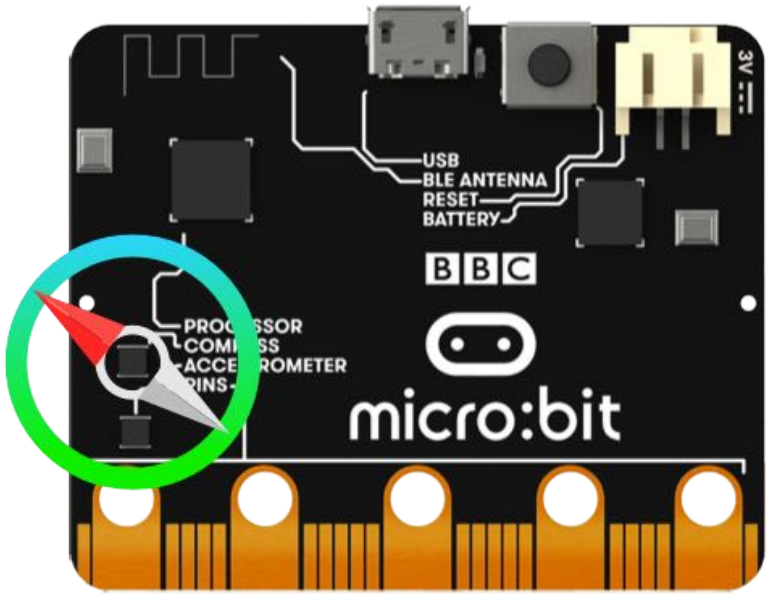
- 温度传感器可以让micro:bit检测当前环境温度(以摄氏度为单位)
- 温度传感器本来是用作检测处理器温度
- 如果处理器计算负载高的话，温度测量值也高

microbit特征：加速度传感器



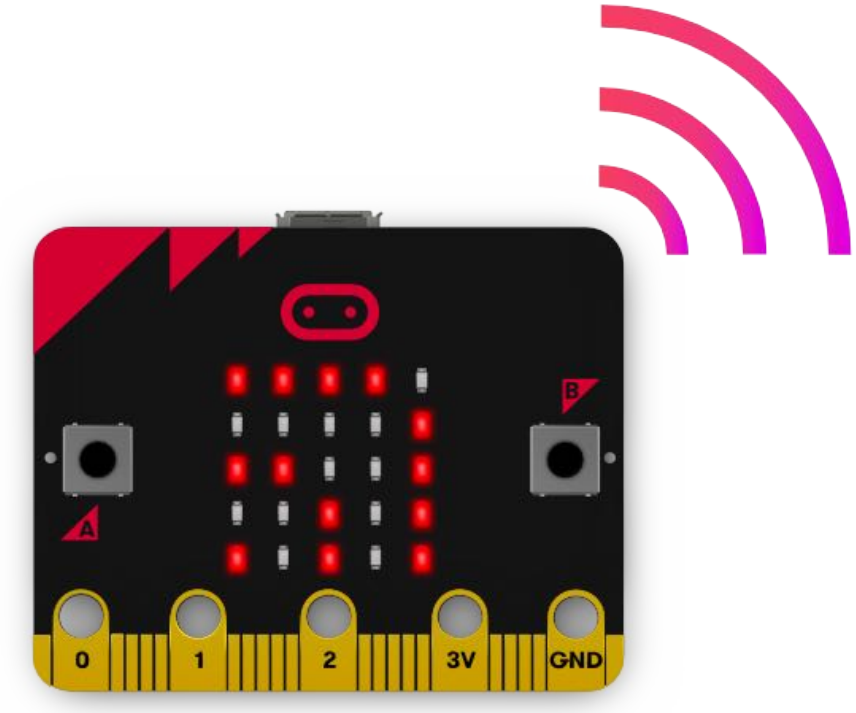
- 加速度传感器可以测量micro:bit的加速度
- 可以检测micro:bit的移动
- 也可以检测其他的动作
- 例如：摇动，倾斜以及自由落体。

microbit特征：指南针



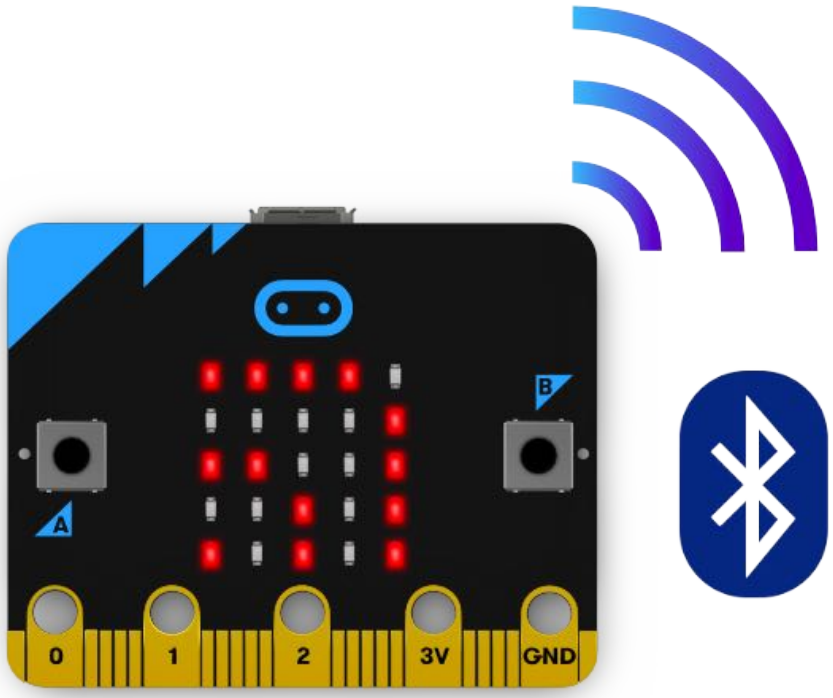
- 指南针用于检测地球磁场，可以探测到micro:bit面对的方向
- 在使用之前，需要校准指南针。
- “校准”是为了确保指南针的结果正确
- 在JavaScript积木块编辑器中，使用 “指南针校准” 积木块
- 在Python中用compass.calibrate()校准指南针

microbit特征：无线电



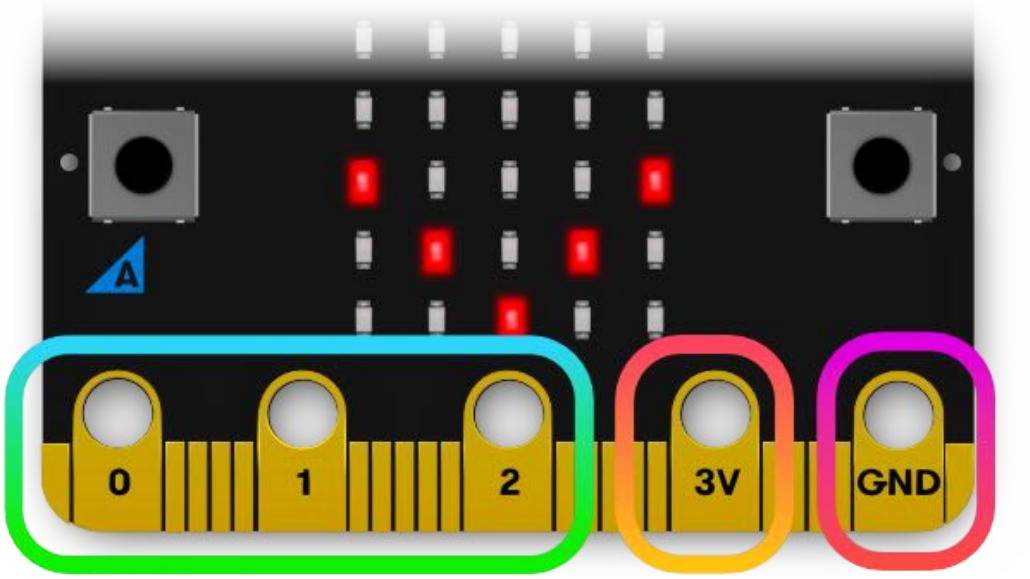
- 可以在2块甚至多块micro:bit板子之间进行无线通讯
- 用无线电发送信息到其他的micro:bit板子上
- 无线电可以有100个频道，调节发射功率，以免互相干扰
- 用于创建多人游戏以及更多有趣的发明！

microbit特征：蓝牙（Python不可用）



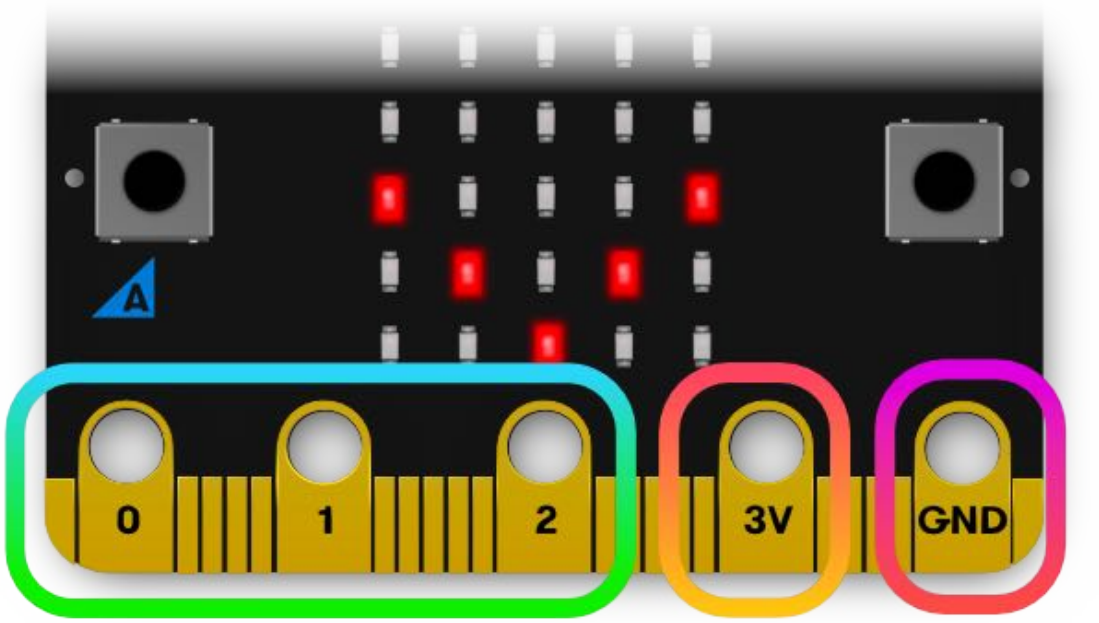
- BLE(蓝牙低能量）天线可以让micro:bit接收蓝牙信息
- 这可以让micro:bit和电脑，手机以及平板进行无线通信
- 可以用micro:bit控制手机
- 或者用手机发送无线代码到设备上

扩展引脚



- 在micro:bit连接器的边缘有25个外部接口
- 这些接口称作“引脚”
- 引脚可以扩展连接电机，LED灯
- 或者其他带引脚的电子元器件编程
- 或者是连接外部传感器控制代码

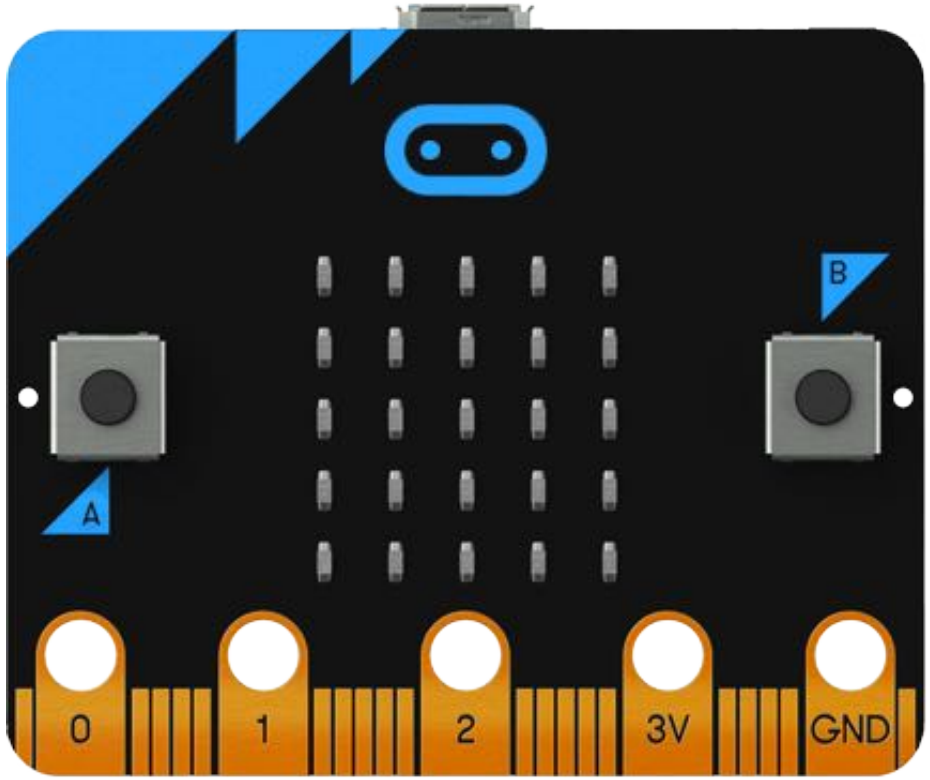
micro:bit的扩展引脚：大引脚



- P0 : GPIO (通用数字输入和输出) 带模拟-数字转换器 (ADC)
- P1: 带ADC的GPIO
- P2: 带ADC的GPIO
- 3V： 电源
- GND： 接地

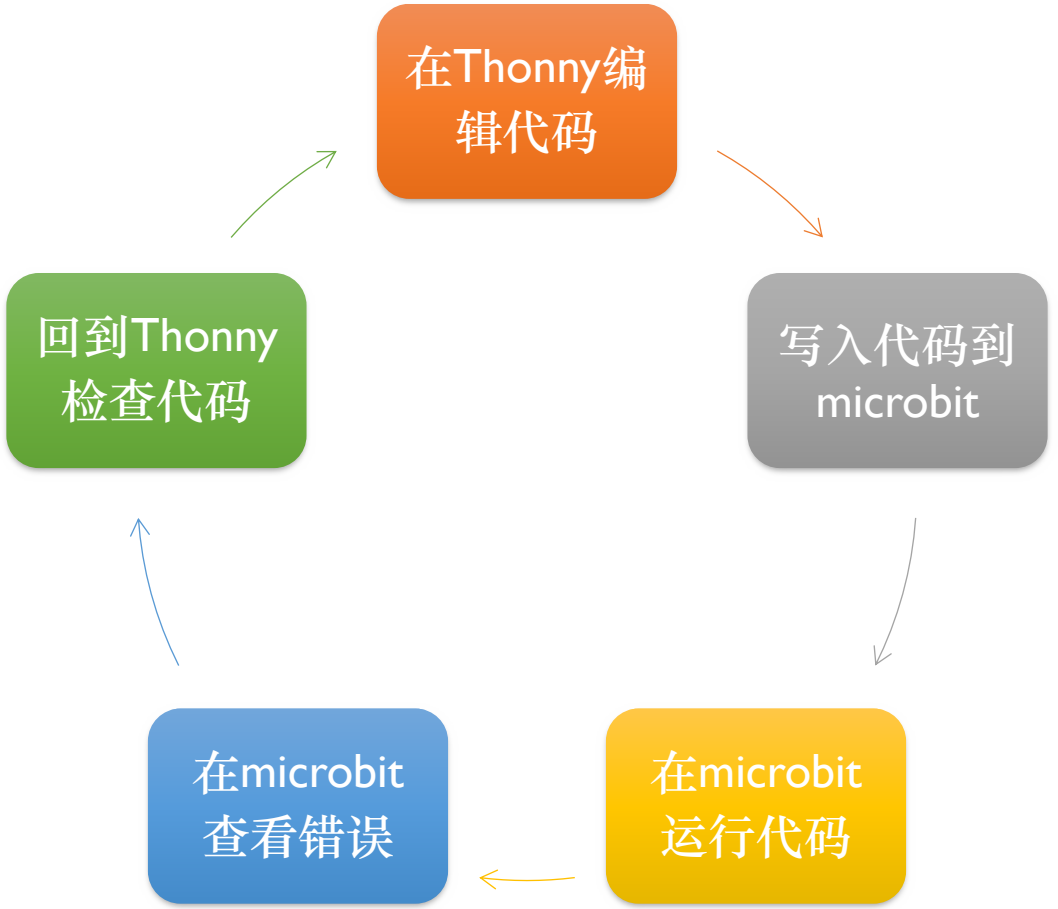
目录：micro:bit基础编程

- 第一个程序Hello World!
- LED图像、按钮
- 输入/输出引脚
- 音乐、语音合成
- 随机数发生器
- 移动检测和姿态识别、指南针
- NeoPixel彩灯
- 无线通信



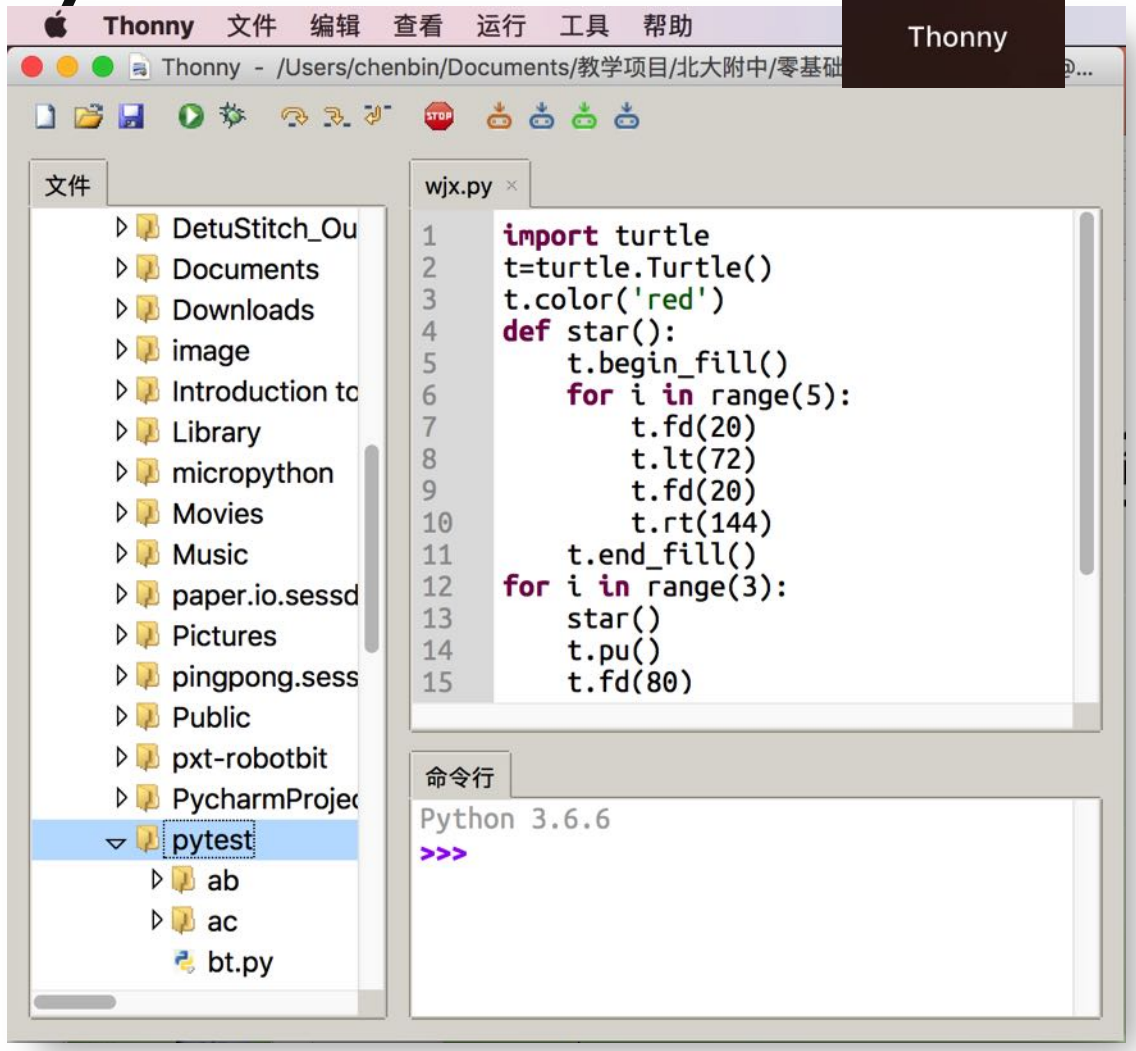
microbit-micropython编程的过程

- 在PC上编写程序
- 下载到microbit运行，并观察运行结果
- microbit作为单片机可以脱离PC自主运行程序，只需要正常供电即可
- 有错误的话再回到PC上修改
- 重复上述过程



集成开发环境：Thonny

- 跨平台Windows/macOS/Linux
- 自带最新版本Python3，无需安装Python
- 体积小巧，功能齐全
- 安装第三方模块很方便
- 可以连接microbit单片机编程
- 地小空开放实验室汉化版本
  - https://github.com/chbpku/dxkStickIDE/tree/master/Setup



初识microbit-micropython

1. USB线连接microbit
2. 打开Thonny
3. 输入程序，保存为py文件
4. 点击“写入运行环境”
   - 稍等一下，闪烁结束
   - 出现成功Done字样
5. 点击“写入当前代码”
   - 出现成功Done字样
6. 立刻运行，可按RESET重启

第一个程序：Hello World!

- microbit基本硬件的访问都在模块microbit中
- 通常，首先导入microbit模块的所有对象
- 我们来写第一个helloworld程序
  - 电源开关打到OFF，模式开关打到WR，连接USB线

hello.py

```
1  from microbit import *
2
3  display.scroll("Hello, World!")
```

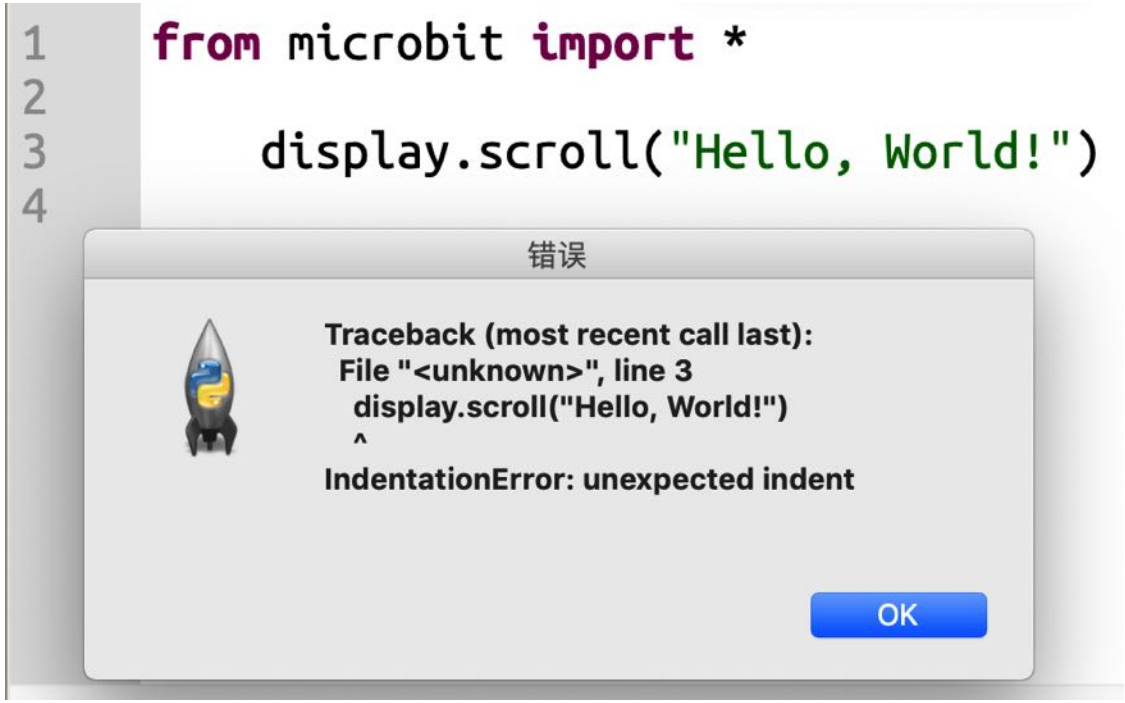
第一个程序：Hello World!

- 可能出现错误
  - 在LED屏滚动显示错误
  - 指示错误代码的行号和错误类型
- 名称错误：NameError
  - 注意大小写
- 语法错误：SyntaxError
  - 注意中文标点等
- 死机?
  - RESET按钮重启

```
1    from microbit import *
2
3    Display.scroll("Hello, World!")
```

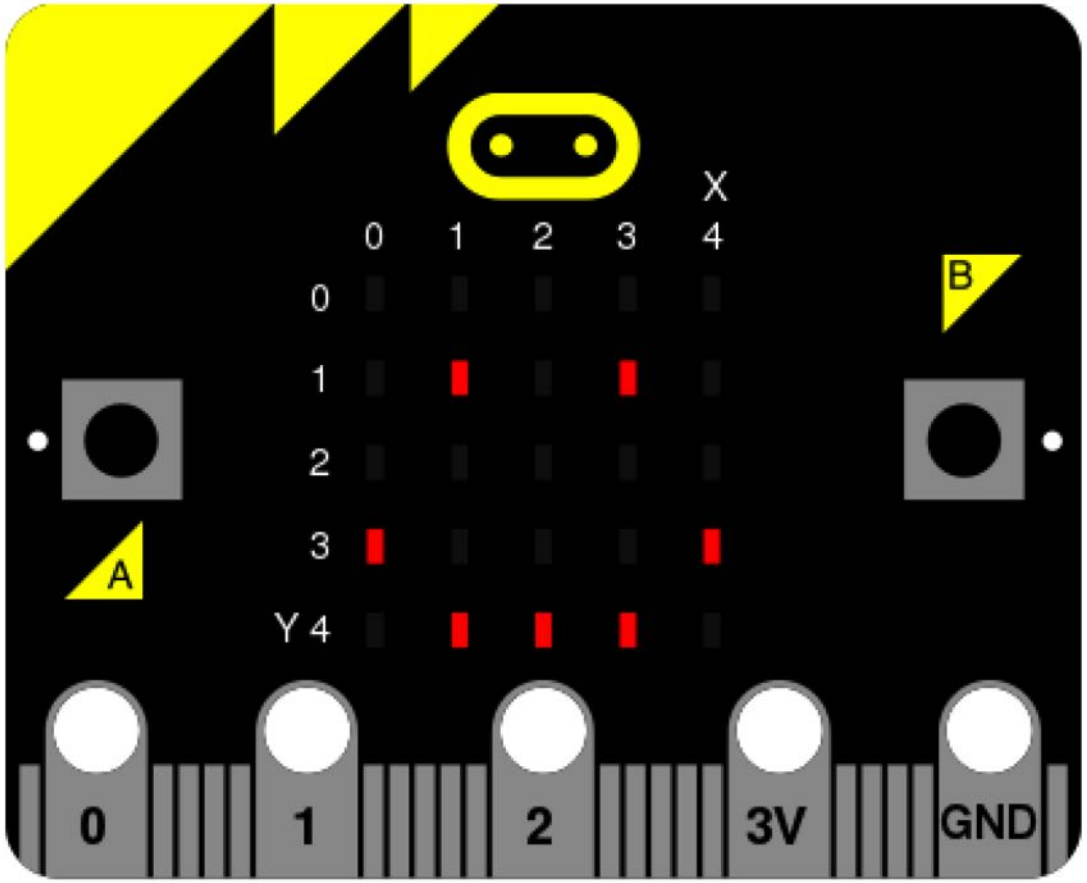
第一个程序：Hello World!

- 在写入代码之前发生的错误
- 常见：IndentationError
  - 缩进错误
- Python语言缩进规则
  - 同一层次一律左对齐
  - 有冒号结尾的语句包含语句块一律缩进
    - if, elif, else, for, while, def, class, with
  - 同一个源代码文件中缩进要一致
  - 一般是4个空格



图像Image

- 5*5 LED点阵可构成图像显示
  - x,y坐标（0,0)~(4,4)
- 每个LED亮度0～9
  - 0=off
- 用display.show显示Image类对象
  - 内置Image对象的图像
  - 自定义的Image对象
  - 动画：Image对象序列



内置Image对象

- microbit模块内置了数十个Image对象，可以直接调用
  - Image.HAPPY
- 分类
  - 表情类
  - 时钟类：CLOCK1~12
  - 方向类：ARROW_N/E/W/S
  - 形状类：TRIANGLE…
  - 动物类：RABBIT…
  - 杂物类：HOUSE/SKULL…

- Image.HEART
- Image.HEART_SMALL
- Image.HAPPY
- Image.SMILE
- Image.SAD
- Image.CONFUSED
- Image.ANGRY
- Image.ASLEEP
- Image.SURPRISED
- Image.SILLY
- Image.FABULOUS
- Image.MEH
- Image.YES
- Image.NO

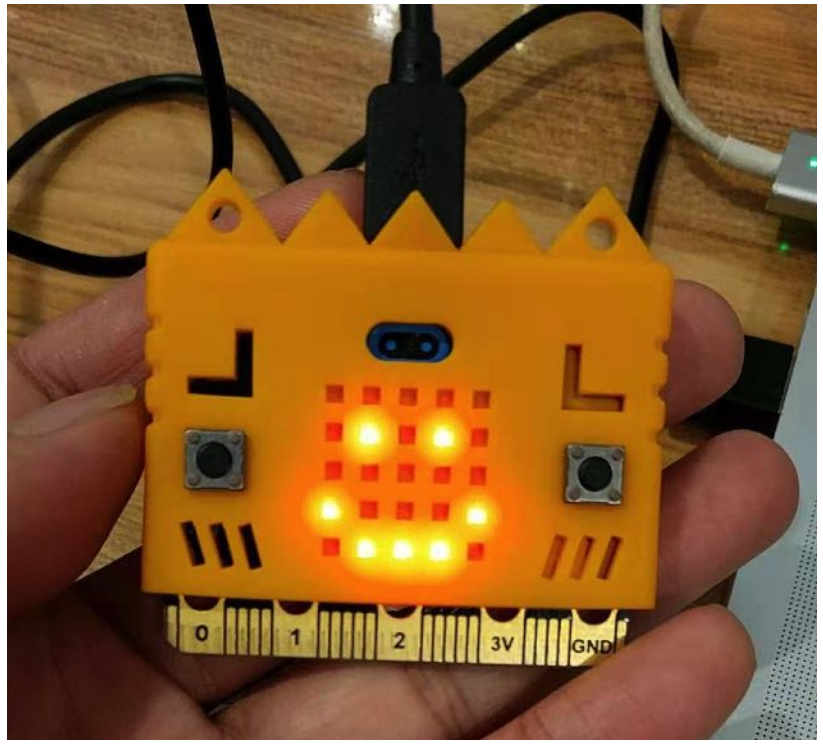


image1.py

```
1  from microbit import *
2
3  display.show(Image.HAPPY)
```

自定义Image图像

- 用Image类来生成
  - Image(str)
- 指定格式字符串
  - 字符0～9对应LED灯亮度
  - :或者\n对应换行
- Python字符串跨行表示
  - 挨在一起的多个引号字符串
  - 表示一个字符串常量
  - "05050:05050:05050:99999:09990"

image2.py

```
1  from microbit import *
2
3  boat = Image("05050:"
4               "05050:"
5               "05050:"
6               "99999:"
7               "09990")
8  display.show(boat)
```

显示动画

- 用图像序列来显示动画
  - display.show(seq,delay=400,wait=True,loop=False,clear=False)
  - 可以是列表或者元组，可以是内置图像或自定义图像
  - 可以指定帧间隔时间delay，是否动画结束再执行下一条语句wait
  - 是否不停循环动画loop，是否动画结束后清除显示clear
- 内置的两个图像列表
  - Image.CLOCKS, Image.ARROWS

image3.py

```
1  from microbit import *
2
3  display.show(Image.ALL_CLOCKS, loop=True, delay=100)
```

动画：沉船 image4.py

```
from microbit import *

boat1 = Image("05050:"
              "05050:"
              "05050:"
              "99999:"
              "09990")
boat2 = Image("00000:"
              "05050:"
              "05050:"
              "05050:"
              "99999")
boat3 = Image("00000:"
              "00000:"
              "05050:"
              "05050:"
              "05050")
boat4 = Image("00000:"
              "00000:"
              "00000:"
              "05050:"
              "05050")
boat5 = Image("00000:"
              "00000:"
              "00000:"
              "00000:"
              "05050")
boat6 = Image("00000:"
              "00000:"
              "00000:"
              "00000:"
              "00000")
all_boats = [boat1, boat2, boat3, boat4, boat5, boat6]
display.show(all_boats, delay=200)
```

LED点阵屏控制display

- 开关显示屏
  - display.on(), off()
- 显示字符串或者图像
  - display.show(s)
- 滚动显示字符串
  - display.scroll(s)
- 清除显示
  - display.clear()
- 点亮一个像素(b=0~9)
  - display.set_pixel(x,y,b)

image5.py

```
1   from microbit import *
2
3   display.show('HELLO')
4   sleep(1000)
5   display.scroll('WORLD')
6   sleep(1000)
7   display.clear()
8   for i in range(5):
9       display.set_pixel(i, i, 9)
10      sleep(200)
11  display.clear()
12  display.scroll('BYE!')
13
```

按钮控制button_a / button_b

- 有两个对象和三个方法
  - button_a, button_b
- 一直按着按钮
  - is_pressed()
- 按下-放开按钮
  - was_pressed()
  - 调用将清除状态
- 按过按钮的次数（距上次调用）
  - get_presses()

button1.py

```
1  from microbit import *
2
3  display.show(Image.ALL_CLOCKS, wait=True)
4  display.scroll(str(button_a.get_presses()))
```

条件循环while

- 检测条件是否成立
  - 成立则执行语句块
  - 执行完毕则再次检测条件
  - 直到条件不成立，执行else
  - while语句结束
- 系统函数running_time
  - 返回启动开始的毫秒数
  - running_time()
- microbit没有时钟硬件模块，掉电就丢失时间

button2.py

```
1  from microbit import *
2
3  while running_time() < 5000:
4      display.show(Image.ASLEEP)
5  else:
6      display.show(Image.SURPRISED)
```

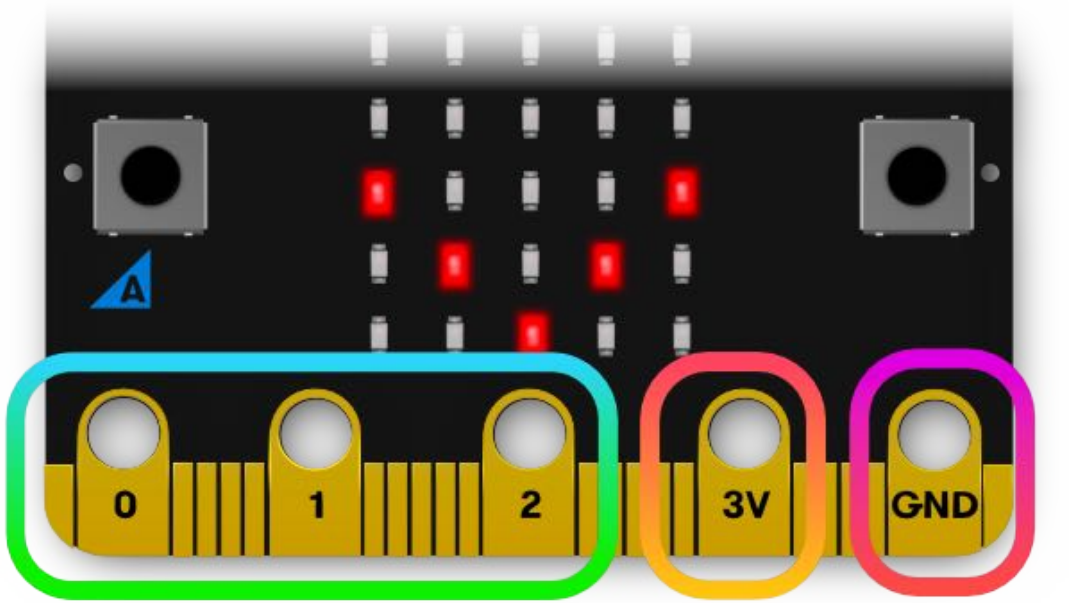
事件循环和处理

- 如果是检测按钮动作，一般需要无限循环来等待事件发生
  - while True:
  - 判断is_pressed()是否True
- 可以用逻辑运算符连接条件
  - 同时成立and
  - 任一成立or
  - 不成立not
- 两个按钮同时按下?

button3.py

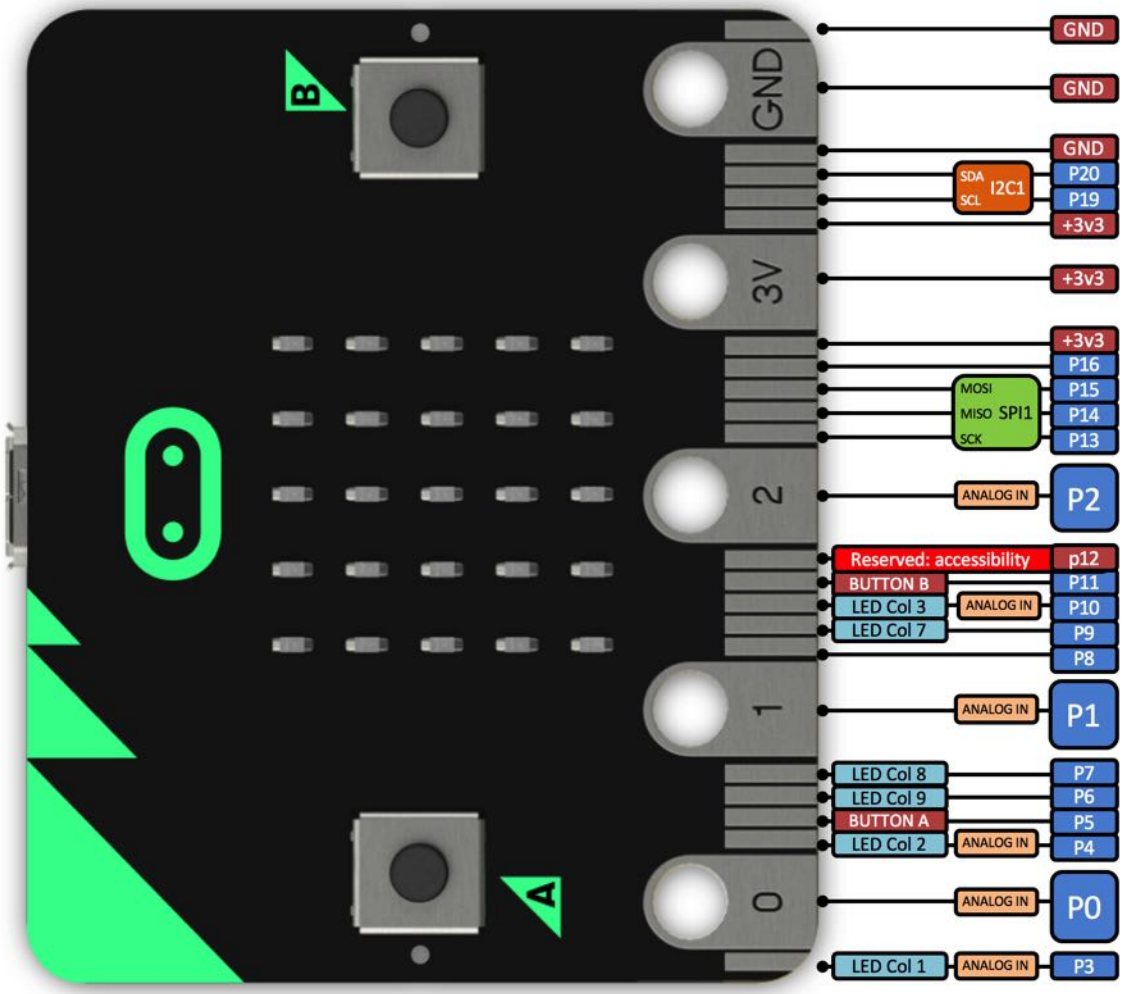
```
1   from microbit import *
2
3   while True:
4       if button_a.is_pressed():
5           display.show(Image.HAPPY)
6       elif button_b.is_pressed():
7           break
8       else:
9           display.show(Image.SAD)
10  display.clear()
```

输入/输出引脚： pin0~pin20



- microbit模块中的内置对象
  - pin0~pin16; pin19, pin20
- 分为3个不同的类
  - MicroBitTouchPin: pin0~2
  - MicroBitAnalogDigitalPin: pin3,4,10
  - MicroBitDigitalPin: pin5~9,pin11~16,pin19~20

输入/输出引脚：三种类



序号	名称	类型	作用
1	P0	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
2	P1	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
3	P2	带ADC的GPIO	空闲的模拟量输入ADC-GPIO（0-1023）
4	P3	LED Col1/ADC-GPIO	如果关闭LED，可以当作ADC-GPIO用
5	P4	LED Col2/ADC-GPIO	如果关闭LED，可以当作ADC-GPIO用
6	P5	Button-A	按钮A
7	P6	LED Col9/D-GPIO	如果关闭LED，可以当作D-GPIO用
8	P7	LED Col8/D-GPIO	如果关闭LED，可以当作D-GPIO用
9	P8	D-GPIO	空闲的数字输入输出D-GPIO
10	P9	LED Col7/D-GPIO	如果关闭LED，可以当作D-GPIO用
11	P10	LED Col3/D-GPIO	如果关闭LED，可以当作ADC-GPIO用
12	P11	Button-B	按钮B
13	P12	D-GPIO	空闲的数字输入输出D-GPIO
14	P13	D-GPIO/SPI-SCK	缺省的SPI-SCK信号
15	P14	D-GPIO/SPI-MISO	缺省的SPI-MISO信号
16	P15	D-GPIO/SPI-MOSI	缺省的SPI-MOSI信号
17	P16	D-GPIO/SPI-NSS(CS)	空闲的数字输入输出D-GPIO，可用作SPI芯片选择
18	P17	3v3	电源输入
19	P18	3v3	电源输入
20	P19	I2C-SCL	I2C总线的时钟信号（内置加速计/指南针）
21	P20	I2C-SDA	I2C总线的数据线（内置加速计/指南针）
22	P21	GND	接地
23	P22	GND	接地

引脚输入：MicroBitTouchPin类

- 大引脚pin0~2的触摸操作
  - 一手接触GND
  - 一手接触P0～2
  - 从pin0.is_touched()读出
- 触摸操作会有诸多因素影响，可能会不触发事件
- 通常可以给一个延迟时间
  - sleep(50)

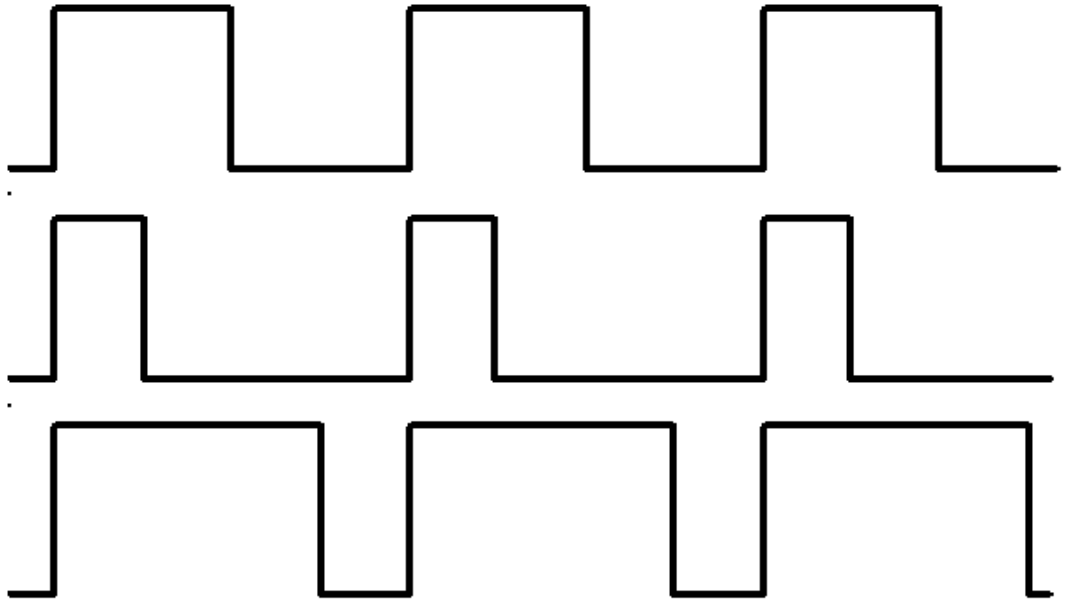
pin1.py

```
1  from microbit import *
2
3  while True:
4      if pin0.is_touched():
5          display.show(Image.HAPPY)
6      else:
7          display.show(Image.SAD)
8      sleep(50)
```

引脚输入输出：模拟数字转换

- 数字信号Digital
  - 0V等于0；3.3V等于1
- 模拟电压信号Analog
  - 0V～3.3V之间
- 模拟转数字（采样）ADC
  - MicroBitAnalogDigitalPin
  - MicroBitTouchPin
  - 1024级：0V=0，3.3V=1023
- 数字转模拟（PWM）
  - 所有的pin都可以0～1023输出

write_analog(511)
write_analog(255)
write_analog(767)



脉冲转换为声音（套件P0接了扬声器）

pin2.py

```
1    from microbit import *
2
3    while True:
4        pin0.write_digital(1)
5        sleep(20)
6        pin0.write_digital(0)
7        sleep(480)
```

pin3.py

```
1    from microbit import *
2
3    pin0.write_analog(511)
```

音乐和语音合成

- music模块可以从引脚输出音乐，由喇叭播放
  - 内置音乐乐曲
  - 由音符编写乐曲
  - 发出指定频率声音
- speech模块输出语音合成
  - say单词
  - pronounce音节
  - sing用指定频率说音节
  - 试验性的模块

music1.py

```
1  import music
2
3  music.play(music.BIRTHDAY, wait=False)
```

内置乐曲

- microbit内置了一些乐曲
  - DADADADUM, ENTERTAINER
  - PRELUDE, ODE, NYAN
  - RINGTONE, FUNK, BLUES
  - BIRTHDAY, WEDDING
  - FUNERAL, PUNCHLINE
  - PYTHON, BADDY, CHASE
  - BA_DING, WAWAWAWAA
  - JUMP_UP, JUMP_DOWN
  - POWER_UP, POWER_DOWN

音符和组成乐曲

- 音符的格式
  - 音符[八度][:时长]
  - 音符：CDEFGAB, #, b, R
  - 八度：0～8，4是中音
  - 时长：整数，tick的数量
- music.play
  - 单个音符，或者音符的序列
  - pin=pin0：播放的引脚
  - wait=True：等待播放结束
  - loop=False：无限循环

music2.py

```
1  import music
2
3  tune = ["C4:4", "D4:4", "E4:4", "C4:4",
4          "C4:4", "D4:4", "E4:4", "C4:4",
5          "E4:4", "F4:4", "G4:8",
6          "E4:4", "F4:4", "G4:8"]
7  music.play(tune)
```

指定频率发声

- music.pitch()
  - 指定频率（Hz）
  - duration=-1，无限或指定毫秒
  - pin=pin0
  - wait=True
- music.stop()
- music.reset()

music3.py

```
1   import music
2
3   while True:
4       for freq in range(880, 1760, 16):
5           music.pitch(freq, 6)
6       for freq in range(1760, 880, -16):
7           music.pitch(freq, 6)
```

综合练习：反应力小游戏

- 在指定时间内需要同时按下AB键可以升级
  - 时间从1000ms开始每级减100ms
- 显示级别（1～10）
- 显示Image.ASLEEP/音乐music.POWER_UP
- 显示3、2、1、Image.TARGET
- 等待指定时间
- 成功按钮则显示Image.HAPPY/音乐music.JUMP_UP
  - 升级，减时间，10级祝贺音乐，退出游戏
- 失败则显示Image.SKULL/音乐music.WAWAWAWAA
  - 回到显示级别，重来

随机数发生器

- random模块是伪随机数发生器
  - seed(整数)
  - 二进制整数getrandbits(位数)
  - randint(start, end)
  - randrange(start, end)
  - 浮点数： random()
  - 浮点数： uniform(start, end)
  - 序列选择： choice(序列)
- 伪随机数pseudo random
  - 根据算法生成的数值序列，均匀分布，每个seed对应一个序列

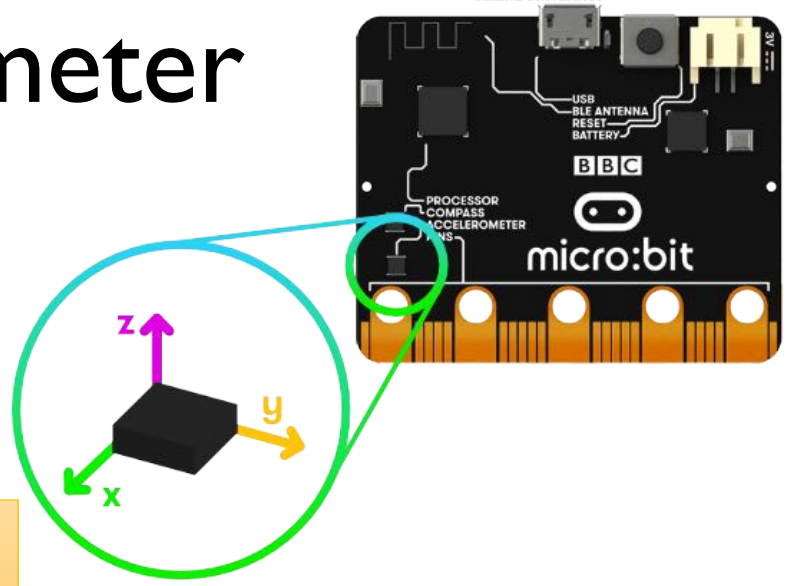
修改代码为按钮显示随机图像

random1.py

```
1  from microbit import *
2  import random
3
4  random.seed(1337)
5  while True:
6      if button_a.was_pressed():
7          display.show(str(random.randint(1, 6)))
8          sleep(500)
9          display.clear()
```

移动检测：加速计accelerometer

- 检测3DoF（三自由度）的移动
  - 绕x/y/z轴旋转运动
- 基本方法
  - 每个自由度返回正负值
  - get_x()
  - get_y()
  - get_z()
  - get_values()



accel.py

```
1   from microbit import *
2
3   while True:
4       reading = accelerometer.get_x()
5       if reading > 20:
6           display.show("R")
7       elif reading < -20:
8           display.show("L")
9       else:
10          display.show("-")
```

魔音盒：控制发声频率

- 倾斜的角度对应发声频率
  - accelerometer.get_y()
  - music.pitch(freq,10)
- 修改为控制一个像素点?
  - accelerometer.get_x()
  - accelerometer.get_y()
  - display.set_pixel(x,y,b)

acce2.py

```
1  from microbit import *
2  import music
3
4  while True:
5      music.pitch(accelerometer.get_y(), 10)
```

姿态识别gesture

- 综合运动的姿态识别
  - current_gesture()
  - was_gesture()
- 常用的姿态
  - 上up/下down/左left/右right
  - 面朝上face up/面朝下face down
  - 自由落体freefall
  - 加速度运动3g/6g/8g
  - 摇shake

写一个沙漏　程序

acce3.py

```
1  from microbit import *
2  import random
3
4  while True:
5      display.show("*")
6      if accelerometer.was_gesture("shake"):
7          display.show(str(random.randint(1, 6)))
8          sleep(1000)
```

指南针

- 内置的指南针传感器compass
  - is_calibrated()：是否校准
  - calibrate()：进行校准
  - clear_calibrate()：清除
  - heading():方向0～359度
- 指北针
  - 利用Image.ALL_CLOCKS序
  - 1～12来指示正北

compass1.py

```
1  from microbit import *
2
3  compass.calibrate()
4
5  while True:
6      sleep(100)
7      needle = ((15 - compass.heading()) // 30) % 12
8      display.show(Image.ALL_CLOCKS[needle])
```

指南针：金属探测器

- 磁场强度
  - get_field_strength()
- 距离金属越近磁场强度越大

compass2.py

```
1  from microbit import *
2  import music
3
4  # 金属探测器
5  while True:
6      fs = compass.get_field_strength()
7      if fs > 60000:
8          music.pitch(fs // 100, 10)
9      sleep(200)
```

彩灯NeoPixel模块

- 创建neopixel对象
  - from microbit import *
  - import neopixel
  - np = neopixel.NeoPixel(pin16,12)
- 设置彩灯（0～11）
  - np[0]=(255,0,0)
  - np[-1]=(0,255,0)
- 点亮/清除彩灯
  - np.show()
  - np.clear()

写一个秒针效果的程序

np1.py

```
1   from microbit import *
2   import neopixel
3
4   np = neopixel.NeoPixel(pin16, 12)
5   i = 0
6   while True:
7       np.clear()
8       np[i] = (30, 0, 0)
9       np[(i + 6) % 12] = (0, 30, 0)
10      np.show()
11      i = (i + 1) % 12
12      sleep(200)
```

无线通信radio

- 通过频道发送和接收消息
  - config(channel=7)：设置
  - on()：打开无线
  - send()：发送字符串
  - receive()：接收字符串
  - off()：关闭无线
- 关于频道
  - 缺省值为7
  - 可以是0～83之间的一个频道
  - 不同频道之间无法通信

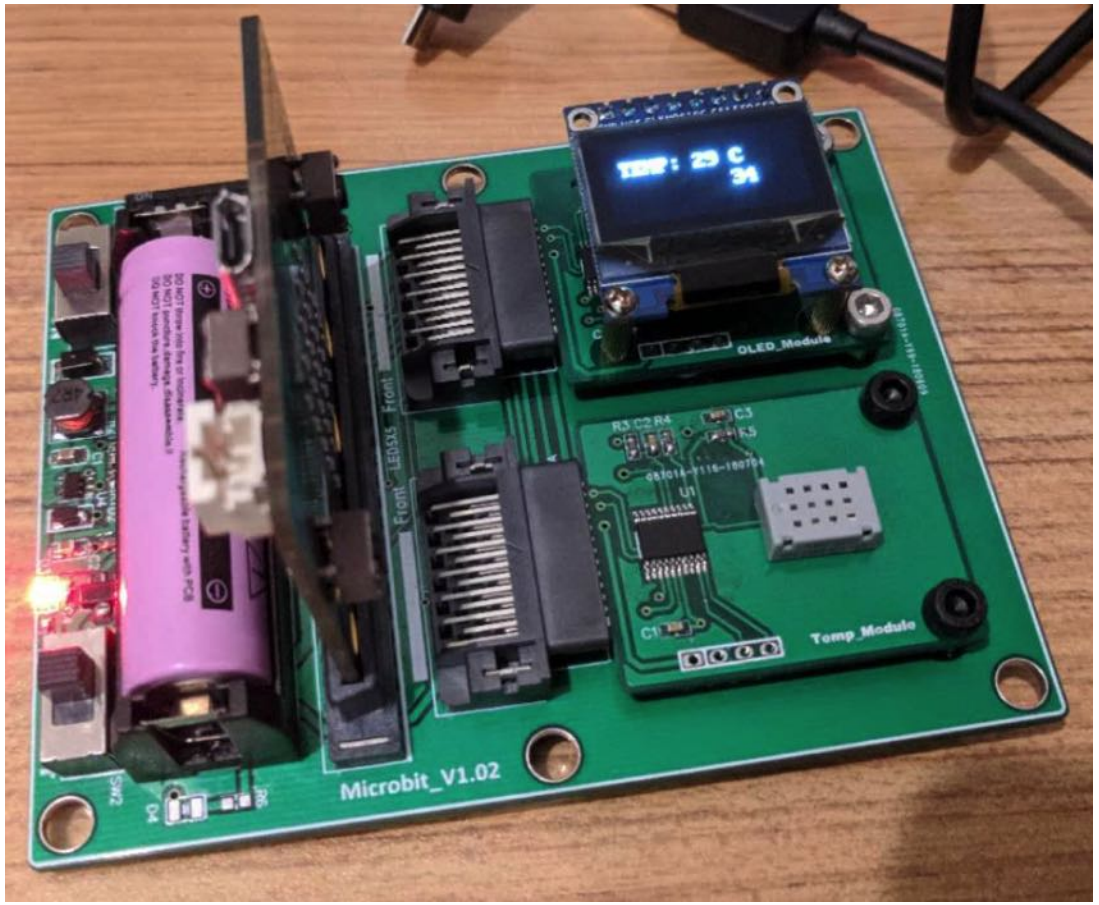
radio1.py

```
1   from microbit import *
2   import radio
3   import neopixel
4
5   np = neopixel.NeoPixel(pin16, 12)
6
7   # 自己的颜色
8   my = "60,0,0"
9
10  # 记录收到的颜色队列
11  clst = [(0, 0, 0) for i in range(12)]
12  radio.on()
13  while True:
14      # 按钮控制自己发送颜色
15      if button_a.was_pressed():
16          radio.send(my)
17      # 接收空中的颜色
18      r = radio.receive()
19      if r is not None:
20          display.show(Image.YES)
21          clst.pop(0)
22          clst.append(tuple(map(int, r.split(","))))
23      else:
24          display.show(Image.ASLEEP)
25      # 显示颜色队列
26      np.clear()
27      for i in range(12):
28          np[i] = clst[i]
29      np.show()
30      # 停留一会儿
31      sleep(100)
```

程序汇总和教学要点

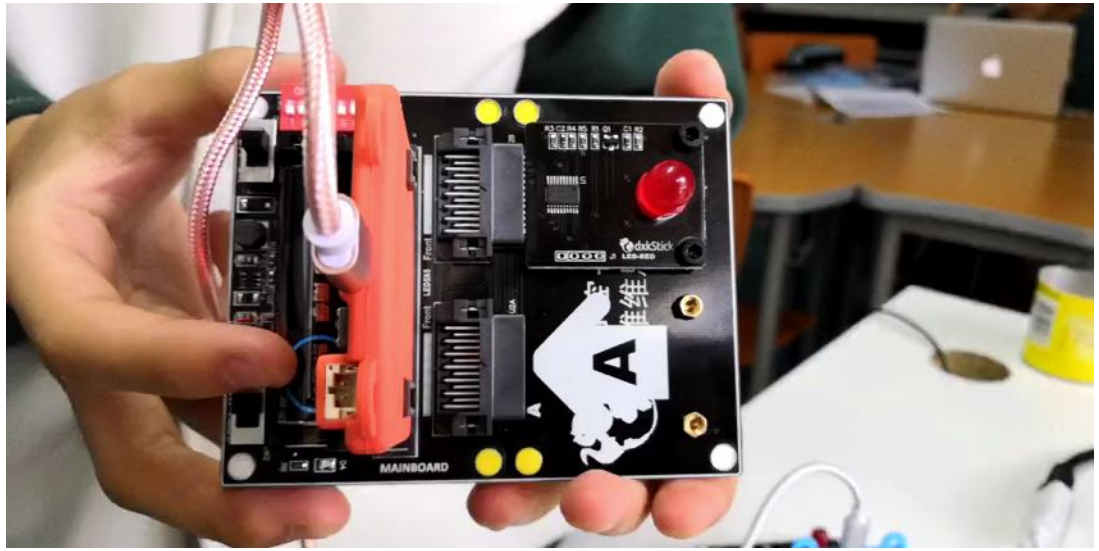
- 第一行程序hello.py
  - 观察和修正错误
- 图像image1~5.py
  - 对象和赋值
  - 字符串表示
  - 序列和列表的应用
  - 函数的关键字参数
- 按钮button1~3.py
  - 嵌套的函数调用
  - 条件循环while
  - 事件循环和处理
- 输入输出引脚pin1~3.py
  - 采样和PWM
- 音乐、语音合成music1~3.py
  - 迭代循环for
- 随机数发生器random1.py
  - 从序列中随机选择元素
  - 伪随机数
- 移动和姿态识别acce1~3.py
- 指南针compass1~2.py
- 彩灯np1.py
  - 取余数的应用

实例：读取温度并显示（microbit编程）




```
from microbit import *
port_A = 0x16
port_B = 0x17

def get_temp(slot):
    i2c.write(slot, b'get_temp')
    b= i2c.read(slot, 1)
    v= int.from_bytes(b, 'big')
    return v

def show(slot, r, c, s):
    v= b'DisplayGB2312,%d,%d,%s    ' % (r, c, s)
    i2c.write(slot, v)
    sleep(50)
    return

c = 0
while True:
    t = get_temp(port_A)
    show(port_B, 2, 10, 'TEMP: %s C' % t)
    show(port_B, 4, 80, c)
    sleep(2000)
    c+= 1
```

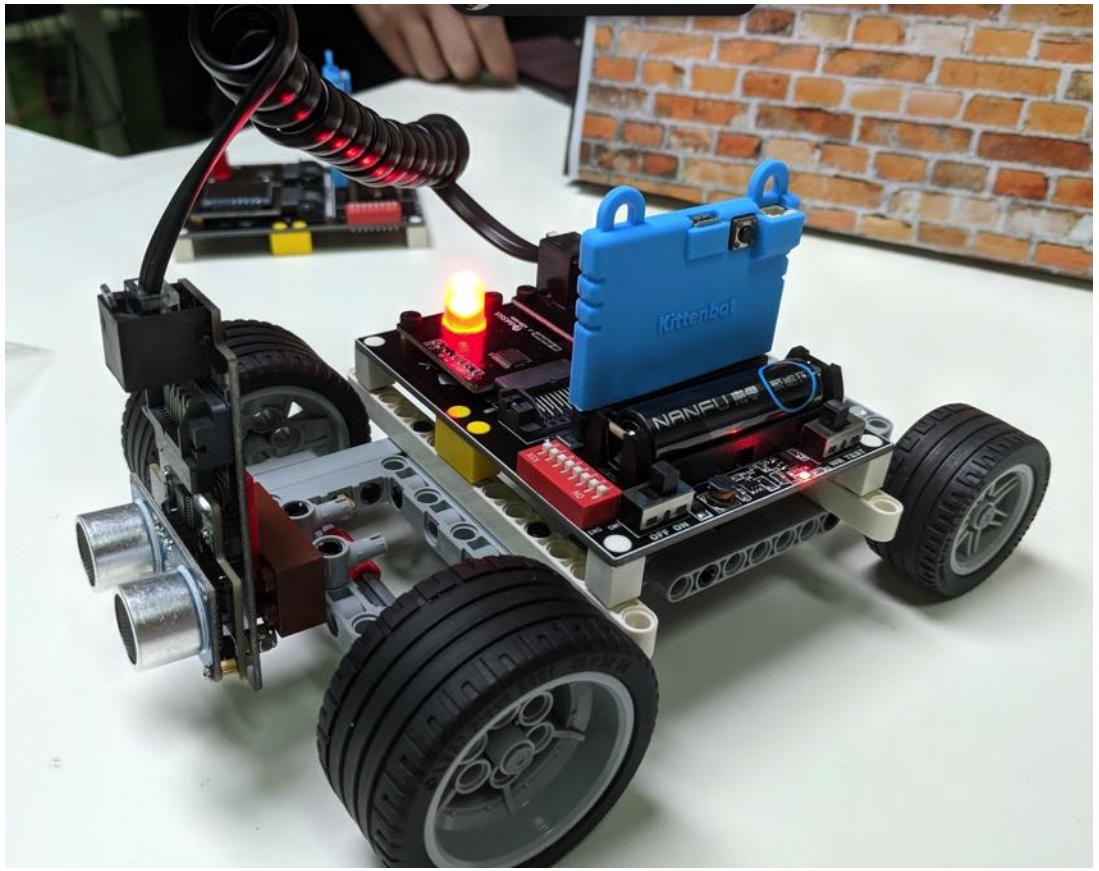
教学实例：SOS莫尔斯码（扩展库编程）




```
29  while True:
30      s()  # 字符s的莫尔斯码
31      sleep(150)  # 字符与字符之间等待150毫秒
32      o()  # 字符o的莫尔斯码
33      sleep(150)  # 字符与字符之间等待150毫秒
34      s()  # 字符s的莫尔斯码
35      sleep(150)  # 字符与字符之间等待150毫秒
36      sleep(350)  # 单词之间等待350毫秒
```

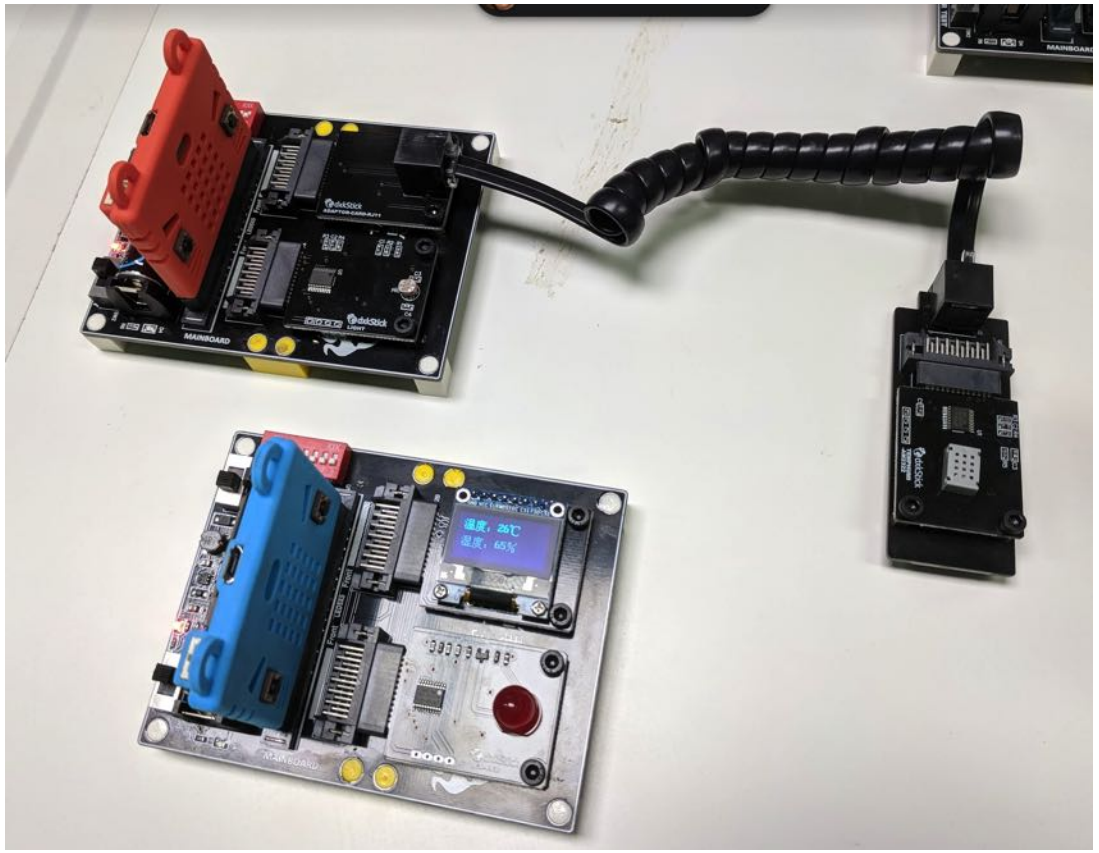
```
13  def s():  # 字符s的莫尔斯码表示
14      for i in range(3):  # 滴循环三次
15          music.pitch(1000, 50, wait=0)  # 蜂鸣 频率 持续时间 等待时间
16          led.on()  # 点亮led
17          sleep(50)  # 延迟50毫秒
18          led.off()  # 关闭led
```

实例：倒车雷达




```python
# 倒车雷达

# 简介:
# 模拟倒车雷达功能，通过超声测距模块测量与障碍物的距离
# 并根据距离远近发出不同频率的蜂鸣声与LED灯闪烁提示。

# 硬件模块:
# micro:bit×1；主板×1；延长插槽×1
# 模块×2：超声测距、LED灯泡

from microbit import *
import music
import ultrasonic, led  # 模块控制库

# 初始化LED灯闪烁系统
light_on = 0
ltimer = 0
led.off()

# 初始化测距记录系统
dist = ultrasonic.value()
update_timer = 0
```

实例：无线传输温度值



```
# 远程测量温湿度、亮度

# 简介:
# 主节点读取子节点上传感器的读数并将结果显示在自身的元件上:
# 温湿度传感器的读数将直接显示于OLED显示屏内，光敏电阻则会在光强低于一定阈值

# 硬件模块:
# micro:bit×2; 主板×2
# 模块×4: （主节点）OLED显示屏、LED灯泡; （子节点）温湿度传感器、光敏电阻

import mb,oled,light,led,temp_humi # 主板、模块控制库
from microbit import *
import music

# 初始化
mb.remote_on()# 启动远程模式
led_on=0
oled.clear()
led.off()
timer=0
light_read=0
tmp,hum='--','--'
```

dxkStick开源仓库（开发环境/案例）

- https://github.com/chbpku/dxkStickIDE

- 国内： https://gitee.com/chbpku/dxkStickIDE

【分组作业】microbit创意作品

- 每组4人，协作一个硬件作品
- 要求用microbit和Python语言
- 作业提交资料
  - 创意作品Python源代码；
  - 实习报告（DOCX+PDF）；
  - 作品Poster：单页PPT（16:9格式）
  - 作品照片：10张以内照片；
  - 运行和介绍视频：2分钟以内的MP4视频（1920*1080）
- 实习报告要求
  - 标题、作者、摘要
  - 一、选题及创意介绍
  - 二、设计方案和硬件连接
  - 三、实现方案及代码分析
  - 四、后续工作展望
  - 五、小组分工合作（含讨论照片）
  - 字数建议在2000字左右
- 评分：设计30%，编程40%，报告30%

2018 microbit创意作品人气榜的15个作品

① 多功能搬运车
② microbit模拟器
③ paper.io多人对抗游戏
④ 俄罗斯方块
⑤ 数算课的生死时速
⑥ 微钢琴
⑦ 遥控向日葵花车
⑧ microbit超级马里奥
⑨ 捕鱼达人
⑩ 宇宙飞船游戏系列
⑪ 音乐游戏MusicBlocks
⑫ 捣蛋机器人
⑬ 电子歌姬
⑭ 疯狂炸弹人
⑮ 音乐编辑器。

https://space.bilibili.com/275008758/#/favlist?fid=1702213

【sessdsa18】第07组：点球大战

收藏于5-7

【sessdsa18】第21组：奔跑计步器

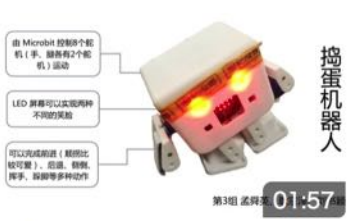
收藏于5-7

【sessdsa18】第12组：手速比拼

收藏于5-7

【sessdsa18】第09组：多功能测量仪

收藏于5-7



【sessdsa18】第03组：捣蛋机器人

收藏于5-7

【sessdsa18】第27组 音乐合奏—天空之城

收藏于5-7

【sessdsa18】第16组：paper.io多人对抗游戏

收藏于5-7

【sessdsa18】第25组：双人射击对战游戏

收藏于5-7

【sessdsa18】第04组：中文语音合成播报

收藏于5-7

【sessdsa18】第31组：微钢琴

收藏于5-7

【sessdsa18】第26组：俄罗斯方块

【sessdsa18】第20组：野外探险工具箱

【sessdsa18】第24组：皮坎旋律（音乐游戏）

【sessdsa18】第06组：简易版坦克大战

【sessdsa18】第10组：默契大考验之你们凉凉了么

【sessdsa18】第13组：宇宙飞船系列游戏

收藏于5-7

【sessdsa18】第22组：疯狂炸弹人（超长3分钟）

收藏于5-7

【sessdsa18】第17组：迷宫

收藏于5-7

【sessdsa18】第18组：复古小游戏机

收藏于5-7

【sessdsa18】第28组：Micro:musicbox

收藏于5-7

【sessdsa18】第33组：多功能搬运车

收藏于5-7

【sessdsa18】第11组：音乐游戏MusicBlocks

收藏于5-7

【sessdsa18】第08组：音乐编辑器

收藏于5-7

【sessdsa18】第15组：深蹲计数&猜旋律

收藏于5-7

【sessdsa18】第34组：Microbit 实现超级马里奥

收藏于5-7

【sessdsa18】第30组：无敌乒乓球

【sessdsa18】第02组：模拟电报机

【sessdsa18】第29组：数算课的生死时速

【sessdsa18】第35组：micro:bit模拟器

【sessdsa18】第14组：遥控向日葵花车

2019 microbit创意作品人气榜的19个作品

① 进击的陈老师
② 飞行模拟器
③ 决战数算之巅
④ 创新加速度计GC游戏手柄
⑤ 造血干细胞保卫战
⑥ 简易测谎仪
⑦ Arcflight
⑧ 神庙逃亡
⑨ 体感声控燃脂游戏
⑩ 节奏大师
⑪ 火柴人乱打
⑫ 彩虹屁
⑬ 盆栽拯救者+经典游戏机
⑭ 雷神的减肥梦
⑮ 泡泡堂
⑯ 守望先锋
⑰ 寝室管家
⑱ 泡泡堂2

菜刀电子钢琴

【sessdsa19】第10组：贪吃蛇大作战

收藏于： 5-7

【sessdsa19】第03组：飞行模拟器

收藏于： 5-7

【sessdsa19】第31组：跳一跳

收藏于： 5-7

【sessdsa19】第39组：菜刀电子钢琴

收藏于： 5-7

【sessdsa19】第43组：彩球滑梯+云影浮踪

收藏于： 5-7

【sessdsa19】第42组：守望先锋

收藏于： 5-7

【sessdsa19】第27组：尖叫大师

收藏于： 5-7

【sessdsa19】第44组：坦克大战

收藏于： 5-7

【sessdsa19】第5组：雷神的减肥梦

收藏于： 5-7

【sessdsa19】第7组：接苹果游戏

收藏于： 5-7

【sessdsa19】第01组：ARCFLIGHT

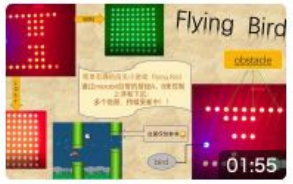
收藏于： 5-7

【sessdsa19】第13组：创新加速度计GC游戏手柄

收藏于： 5-7

【sessdsa19】第28组：造血干细胞保卫战

收藏于： 5-6



【sessdsa19】第46组：Flying Bird

收藏于： 5-6

【sessdsa19】第24组：进击的陈老师

收藏于： 5-6

【sessdsa19】第41组：光标人盗墓

收藏于： 5-6

【sessdsa19】第45组：火柴人乱打

收藏于： 5-6

【sessdsa19】第40组：泡泡堂

收藏于： 5-6

【sessdsa19】第12组：神庙逃亡

收藏于： 5-6

【sessdsa19】第04组：格斗游戏

收藏于： 5-6

【sessdsa19】第09组：贪吃蛇

收藏于： 5-6

【sessdsa19】第02组：超级马里奥

收藏于： 5-6

【sessdsa19】第11组：抵抗外星人

收藏于： 5-6

【sessdsa19】第14组：迷你贪吃蛇

收藏于： 5-6

【sessdsa19】第15组：数算课的Flappy Bird

收藏于： 5-6

【sessdsa19】第19组：简易测谎仪

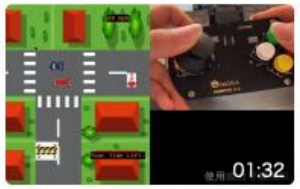
收藏于： 5-6

【sessdsa19】第18组：体感声控燃脂游戏

收藏于： 5-6

【sessdsa19】第17组：DJ游戏

收藏于： 5-6

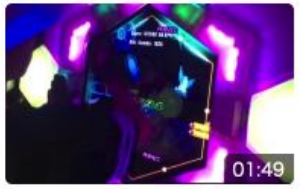


【sessdsa19】第08组：24SPEED

收藏于： 5-6

【sessdsa19】第21组：决战数算之巅

收藏于： 5-6



【sessdsa19】第16组：彩虹屁

收藏于： 5-6

【sessdsa19】第32组：True Music

收藏于： 5-6

【sessdsa19】第29组：泡泡堂

收藏于： 5-6

【sessdsa19】第34组：Doodle Jumper

收藏于： 5-6

【sessdsa19】第26组：Fly Bitch & Snake

收藏于： 5-6

【sessdsa19】第23组：五棍棋

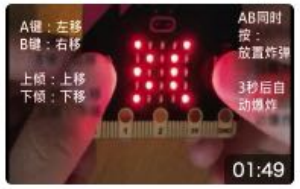
收藏于： 5-6

【sessdsa19】第36组：节奏大师

收藏于： 5-6

【sessdsa19】第22组：最强大脑

收藏于： 5-6



【sessdsa19】第20组：炸弹超人

收藏于： 5-6

【sessdsa19】第30组：寝室管家

收藏于： 5-6

【sessdsa19】第37组：Man Down

收藏于： 5-6

【sessdsa19】第38组：迷失丛林

收藏于： 5-6

【sessdsa19】第33组：盆栽拯救者+各款游戏

收藏于： 5-6

【sessdsa19】第25组：快闪之星

收藏于： 5-6

https://space.bilibili.com/275008758/favlist?fid=452936958