

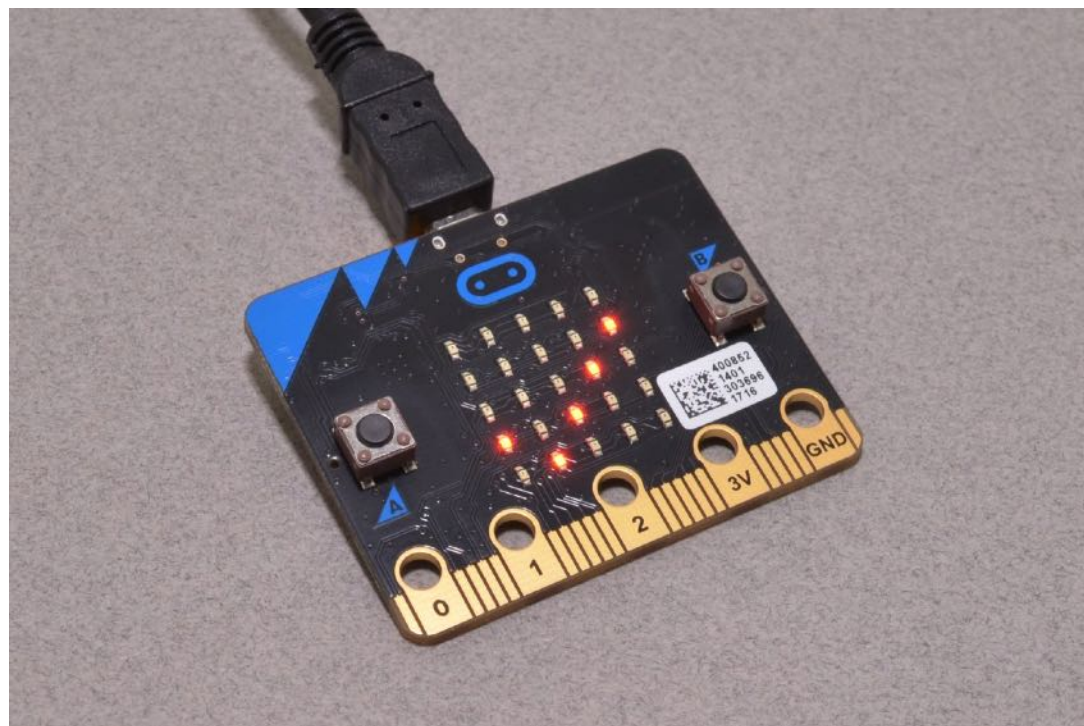
# Python程序设计06-microbit编程

北京大学 陈斌

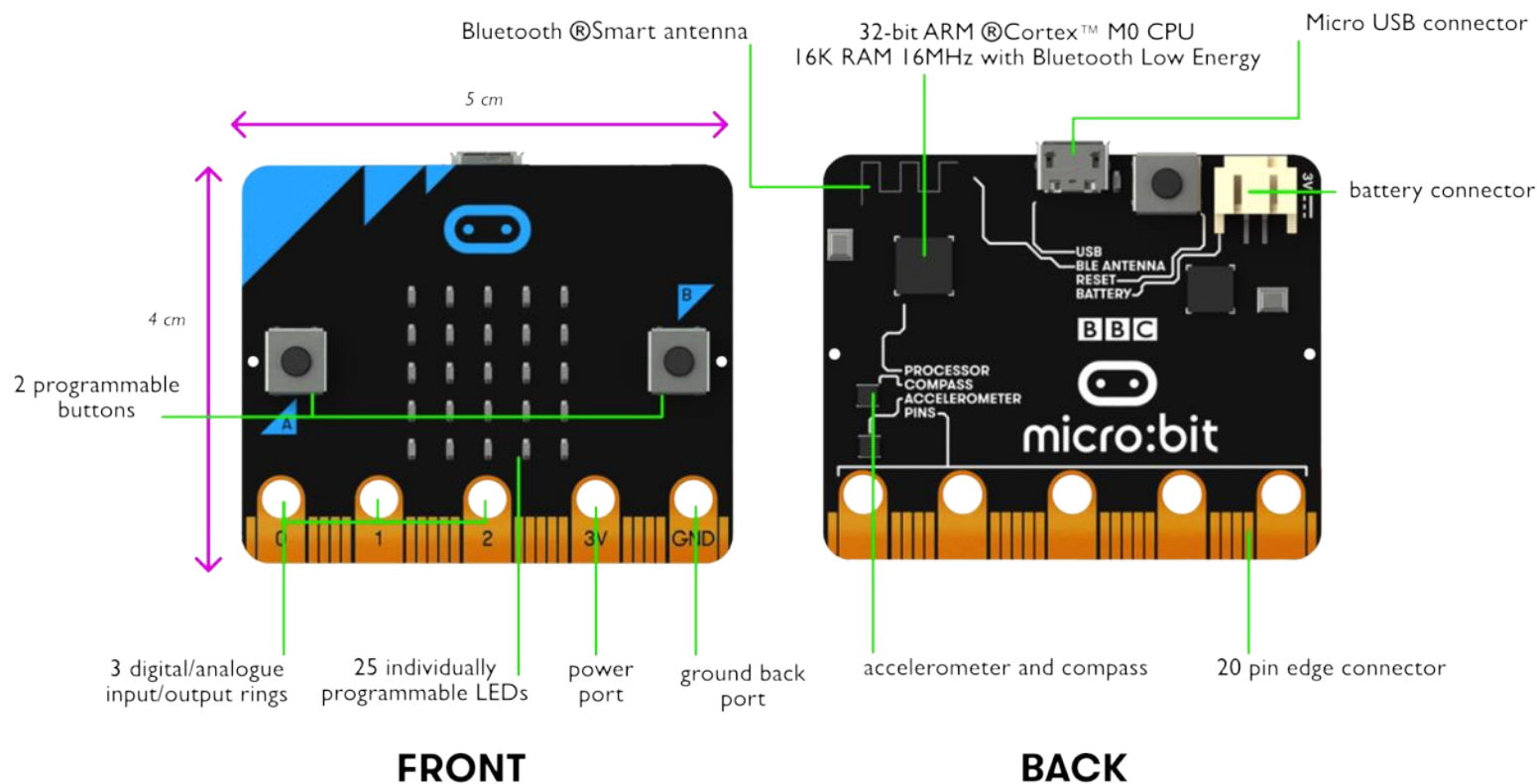
2019.04.08

# 目录

- microbit单片机介绍
- LED点阵控制
- 按钮控制



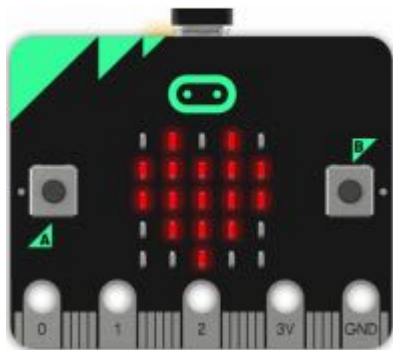
# microbit from BBC介绍



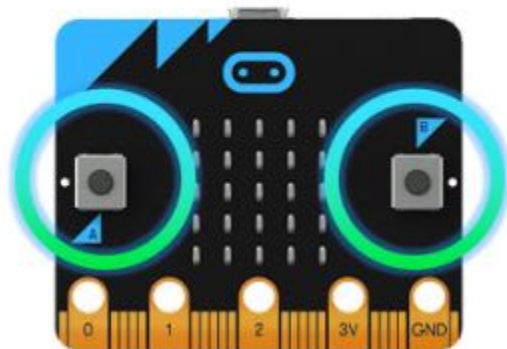
- 25个独立编程的LED
- 2个可编程的按钮
- 1个reset按钮
- microUSB接口
- 3V电源接口
- 光线传感器、温度传感器
- 加速计、电子罗盘
- 无线通信：射频以及蓝牙

# microbit概貌

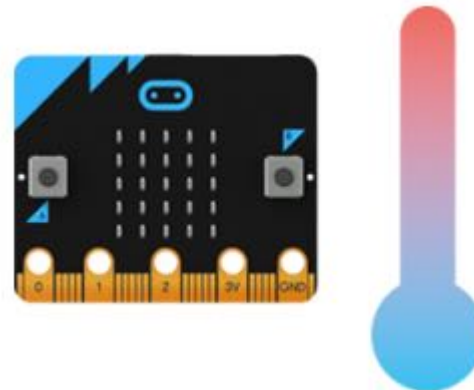
LED



按钮传感器



温度传感器



射频



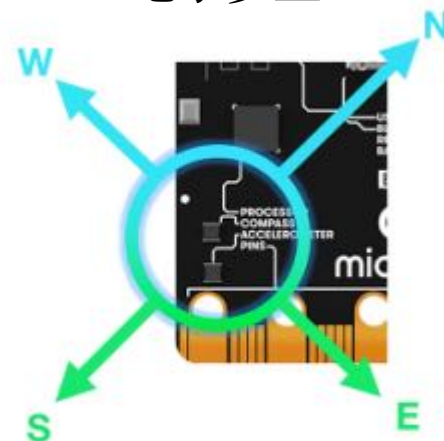
光线传感器



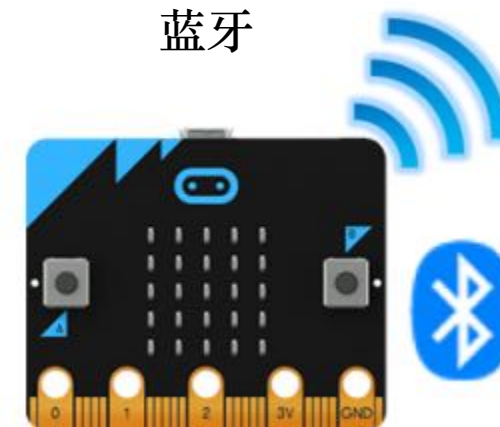
加速度计



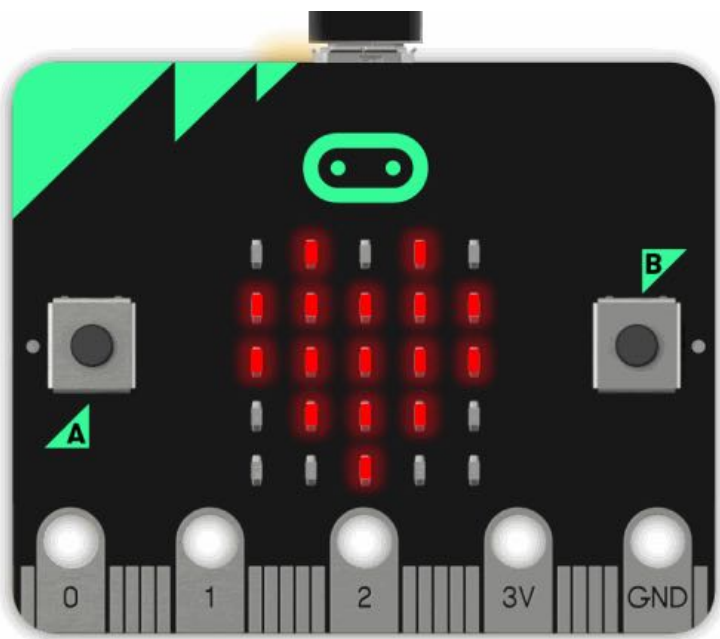
电子罗盘



蓝牙

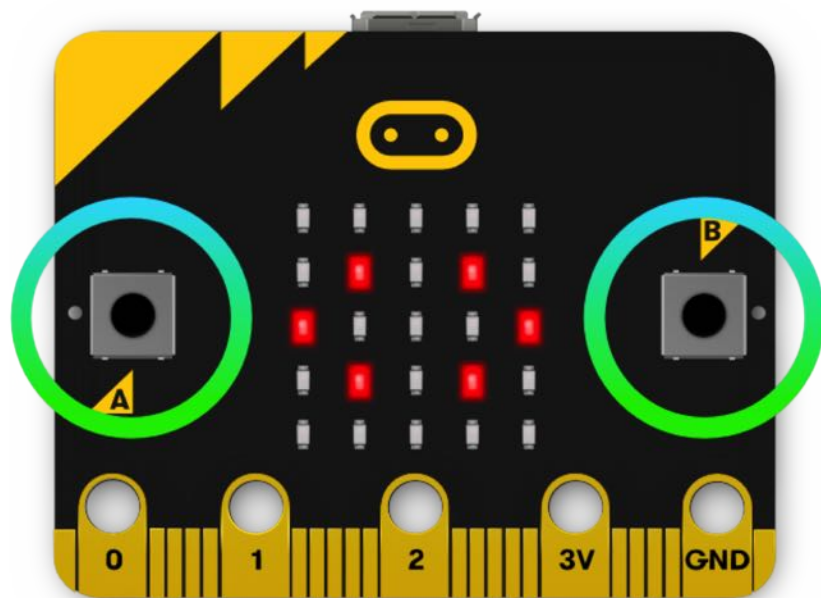


# microbit特征： LED



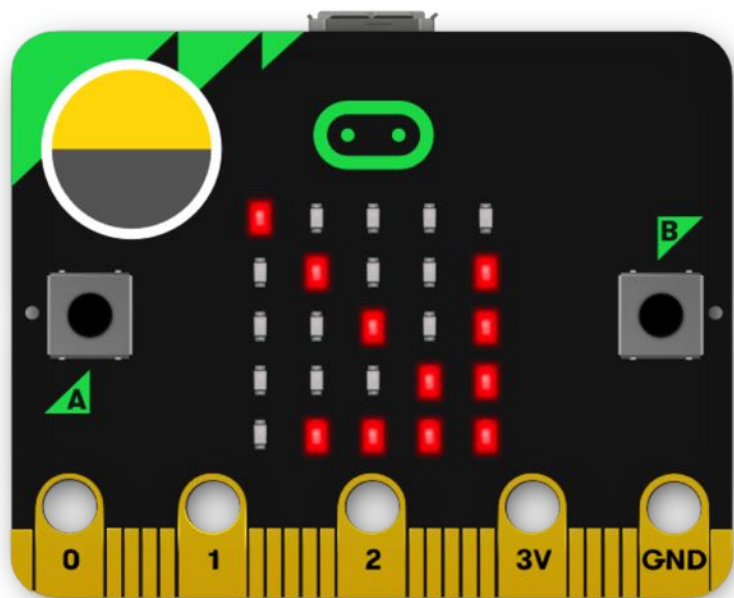
- micro:bit有25颗可独立编程的LED灯
- 可以用来显示文本，数字以及图像

# 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

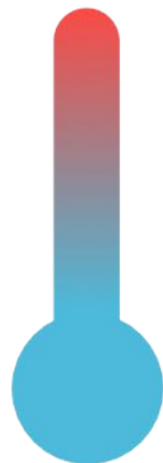
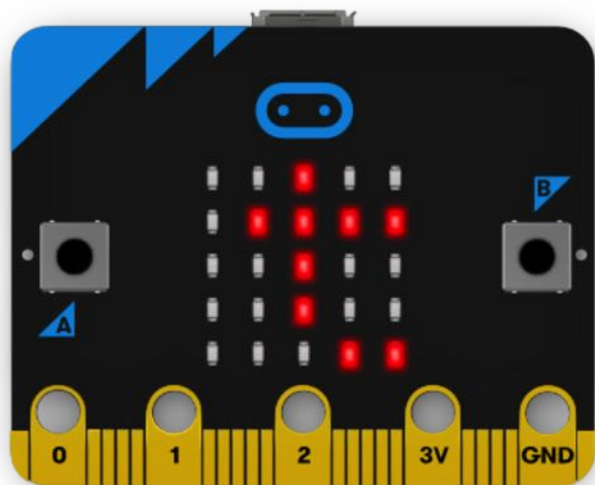
# microbit特征： 光线传感器



- 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



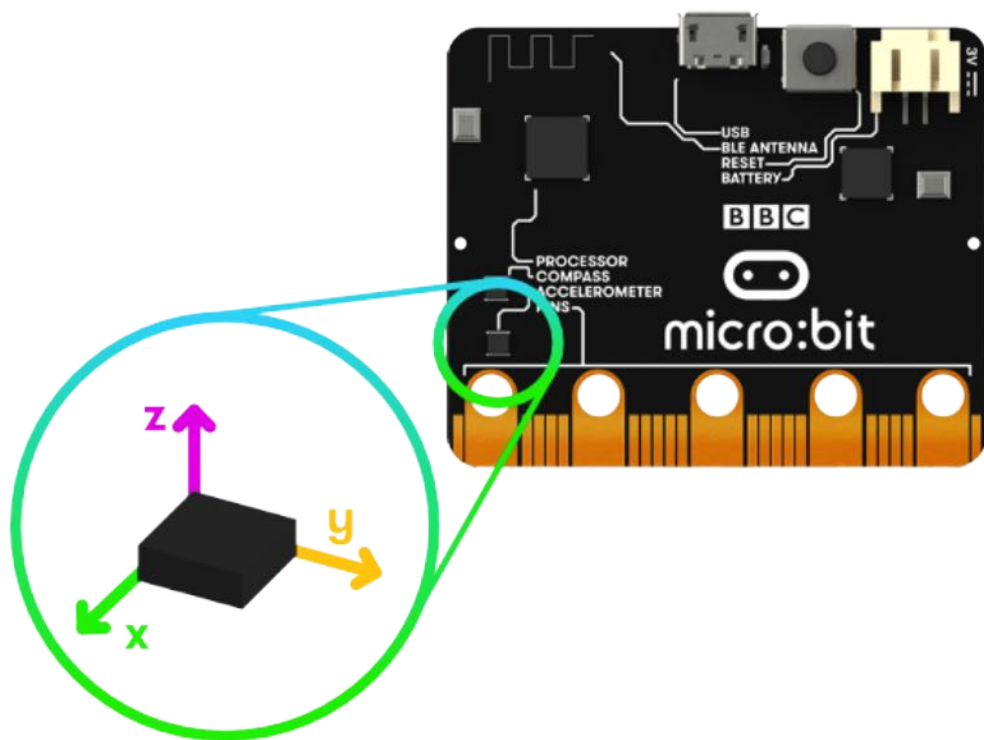
# microbit特征： 温度传感器



- 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

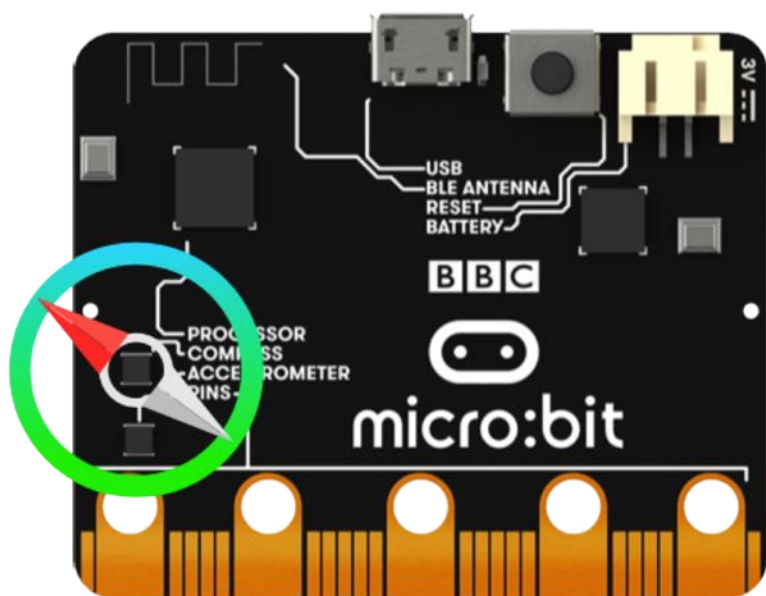


# microbit特征： 加速度传感器



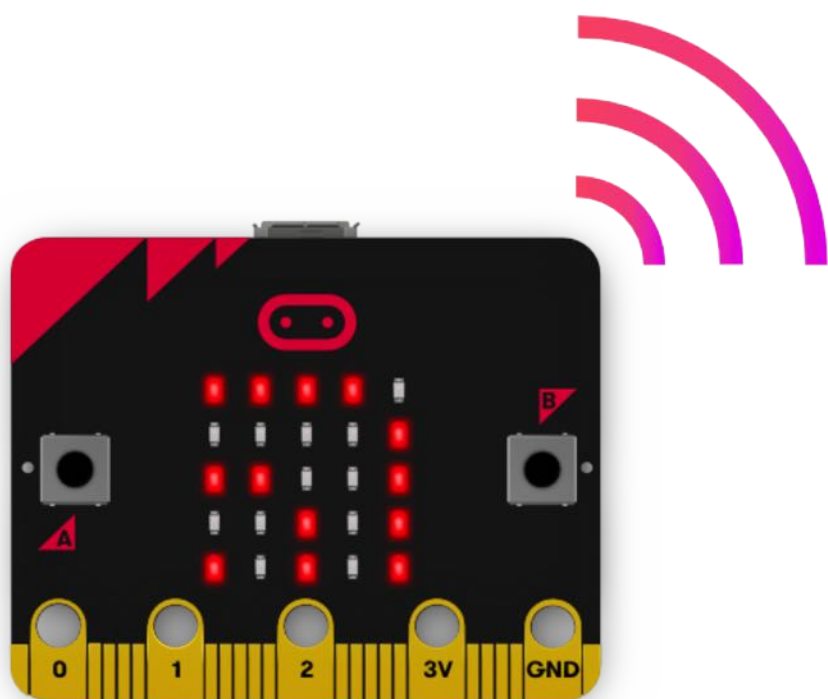
- 加速度传感器可以测量micro:bit的加速度
- 可以检测micro:bit的移动
- 也可以检测其他的动作
- 例如：摇动，倾斜以及自由落体。

# microbit特征：指南针



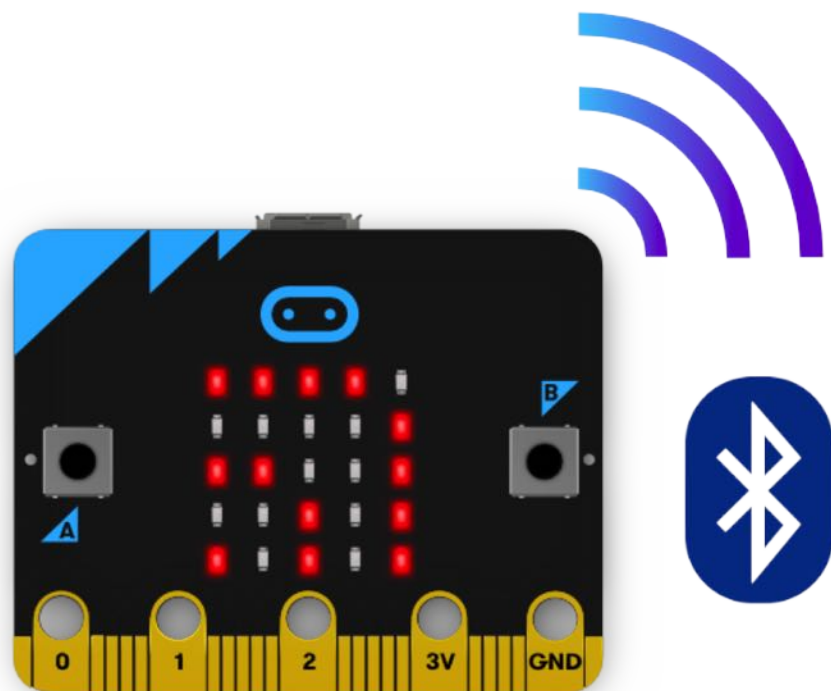
- 指南针用于检测地球磁场，可以探测到micro:bit面对的方向
- 在使用之前，需要校准指南针。
- “校准”是为了确保指南针的结果正确
- 在JavaScript积木块编辑器中，使用“指南针校准”积木块
- 在Python中用 `compass.calibrate()` 校准指南针

# microbit特征：无线电



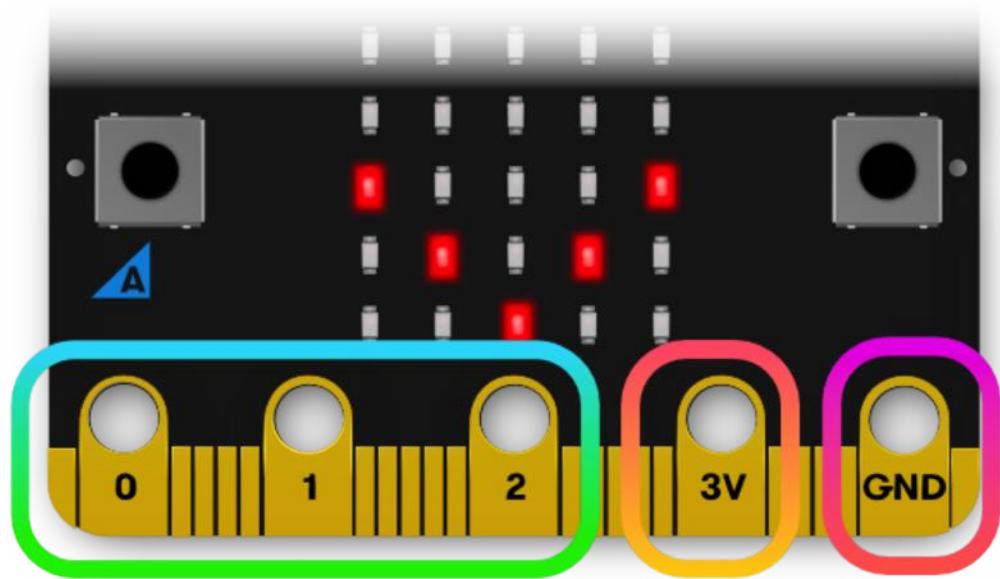
- 可以在2块甚至多块micro:bit板子之间进行无线通讯
- 用无线电发送信息到其他的micro:bit板子上
- 无线电可以有100个频道，调节发射功率，以免互相干扰
- 用于创建多人游戏以及更多有趣的发明！

# 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件编程
- 或者是连接外部传感器控制代码

# 俄罗斯方块

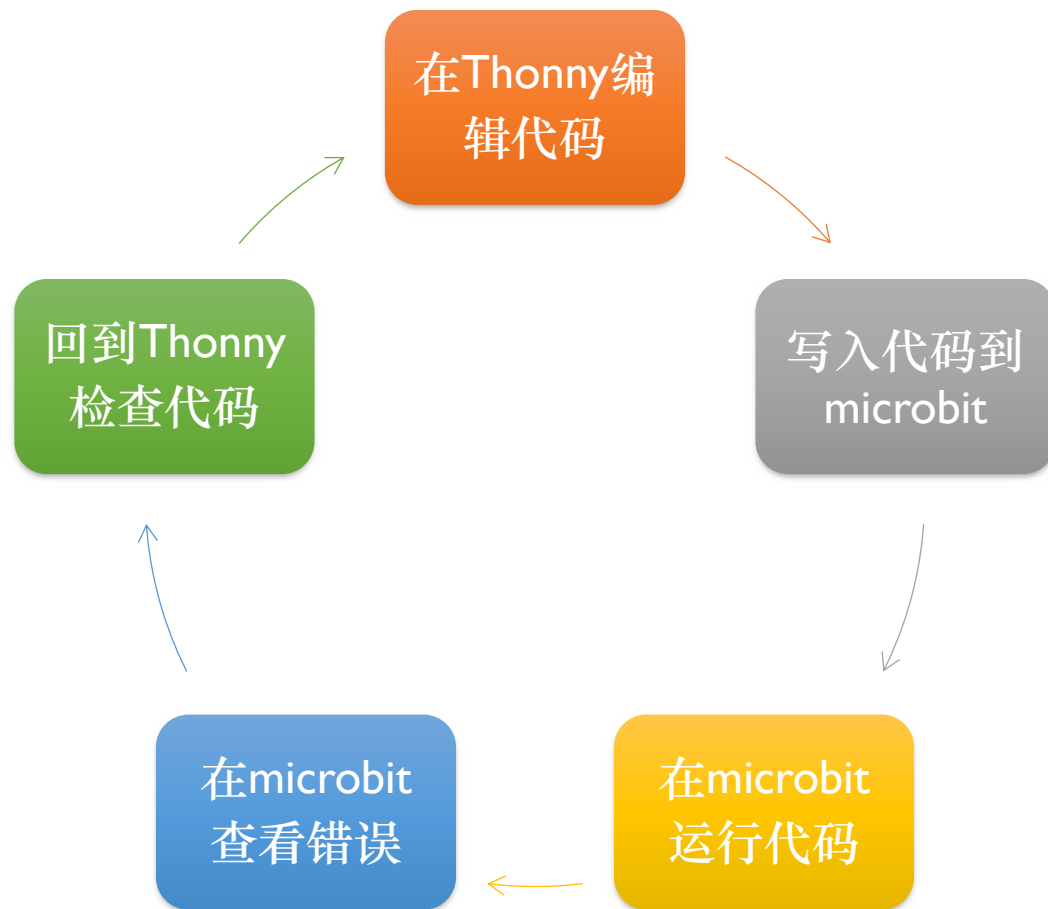
俄罗斯方块  
改编的  
microbit方块

26组

陈丹丘&马涵聪&冉瑾瑜

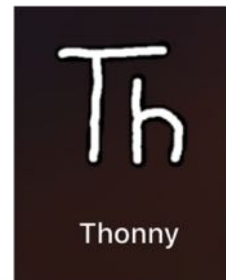
# microbit-micropython编程的过程

- 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

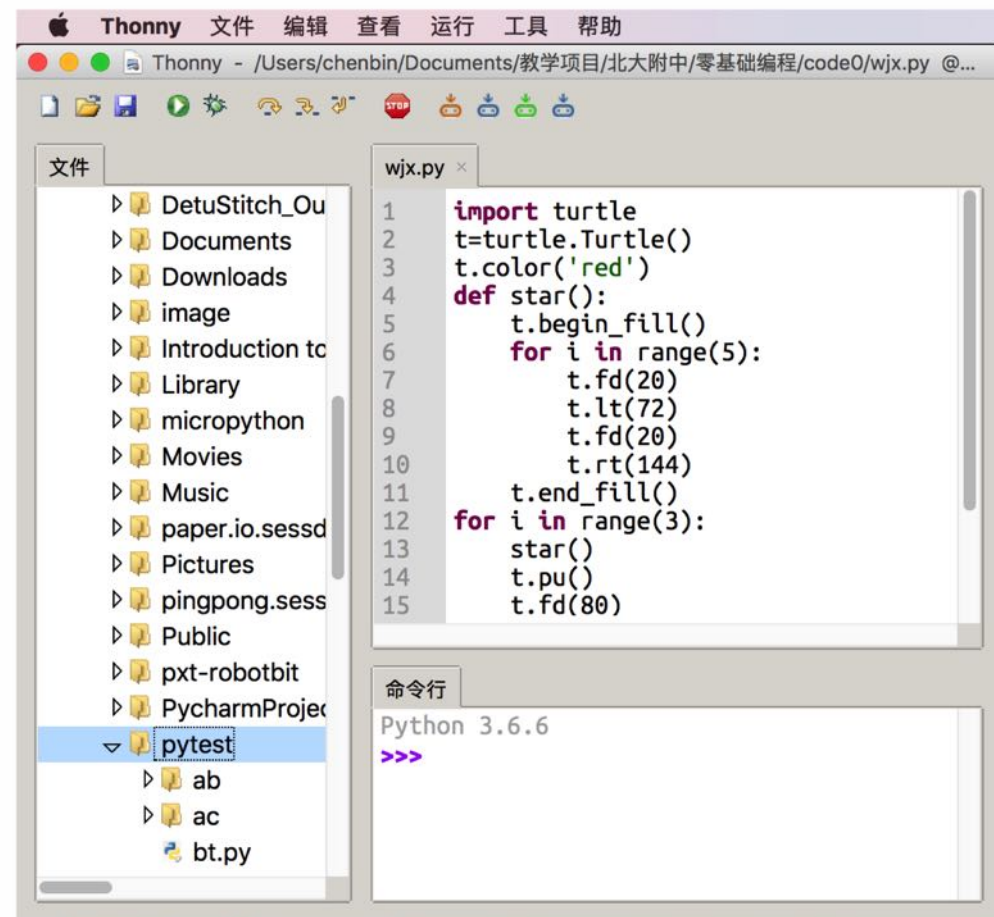




# 集成开发环境：Thonny



- 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



- 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 I：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

```
1 from microbit import *
2 from random import randrange
3 import radio
4
5 display.show(Image.HAPPY)
6
7 f = [] # 一个列表，从99999到00000的图像
8 for i in range(9, -1, -1):
9     imgstr = (str(i)*5+":"+")*5
10    f.append(Image(imgstr))
11
12 radio.on()
13
14 while True:
15     if button_a.was_pressed():
16         radio.send('flash') # 按键发送呼叫信息
17     r = radio.receive()
18     if r == 'flash':
19         sleep(randrange(1000)+50)
20         display.show(f, delay=100, wait=False)
21         if randrange(10) <= 0: #响应的机会0-9，数字越大，机会越大
22             sleep(500)
23             radio.send('flash')
```





# 实例2：萤火虫

```
1 from microbit import *
2 from random import *
3 import radio
4
5 display.show(Image.HAPPY)
6 f = [Image().invert()*(i/9) for i in range(9, -1, -1)]
7
8 radio.on()
9
10 while True:
11     if button_a.was_pressed():
12         radio.send('flash') # 按键发送呼叫信息
13     r = radio.receive()
14     if r == 'flash':
15         sleep(randrange(1000)+50)
16         display.show(f, delay=100, wait=False)
17         if randrange(10) <= 0: #回应的机会0-9, 数字越大, 机会越大
18             sleep(500)
19             radio.send('flash') |
```



# 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



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

# 第一个程序：Hello World!

要点：观察和修正错误

- 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

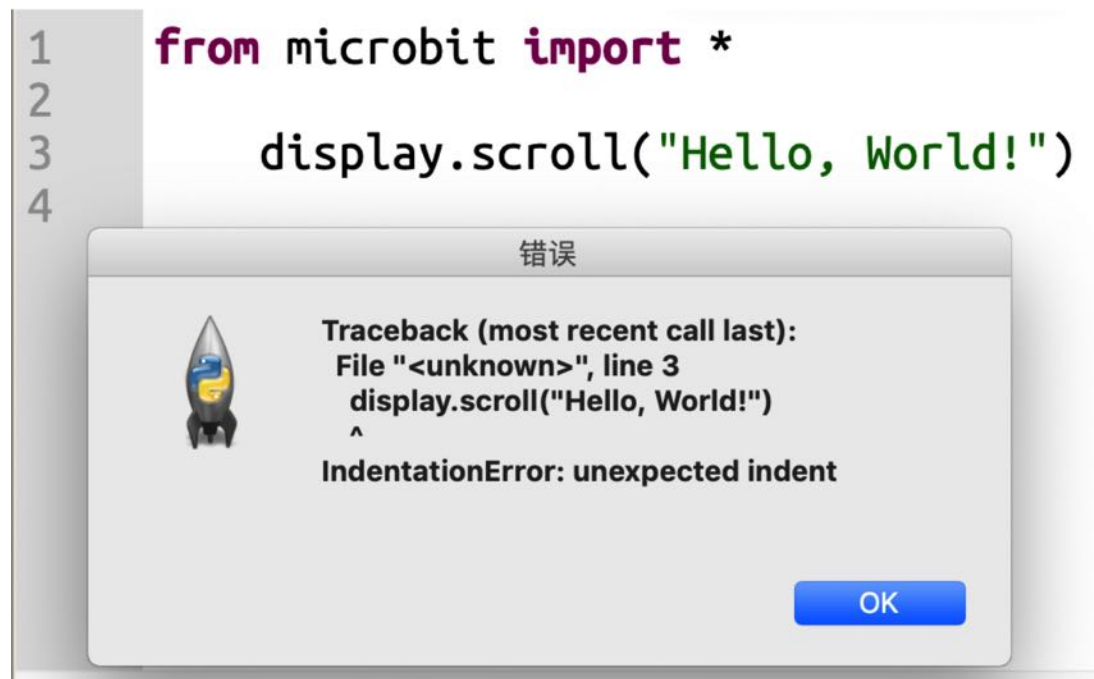


```
1 from microbit import *  
2  
3 Display.scroll("Hello, World!")
```



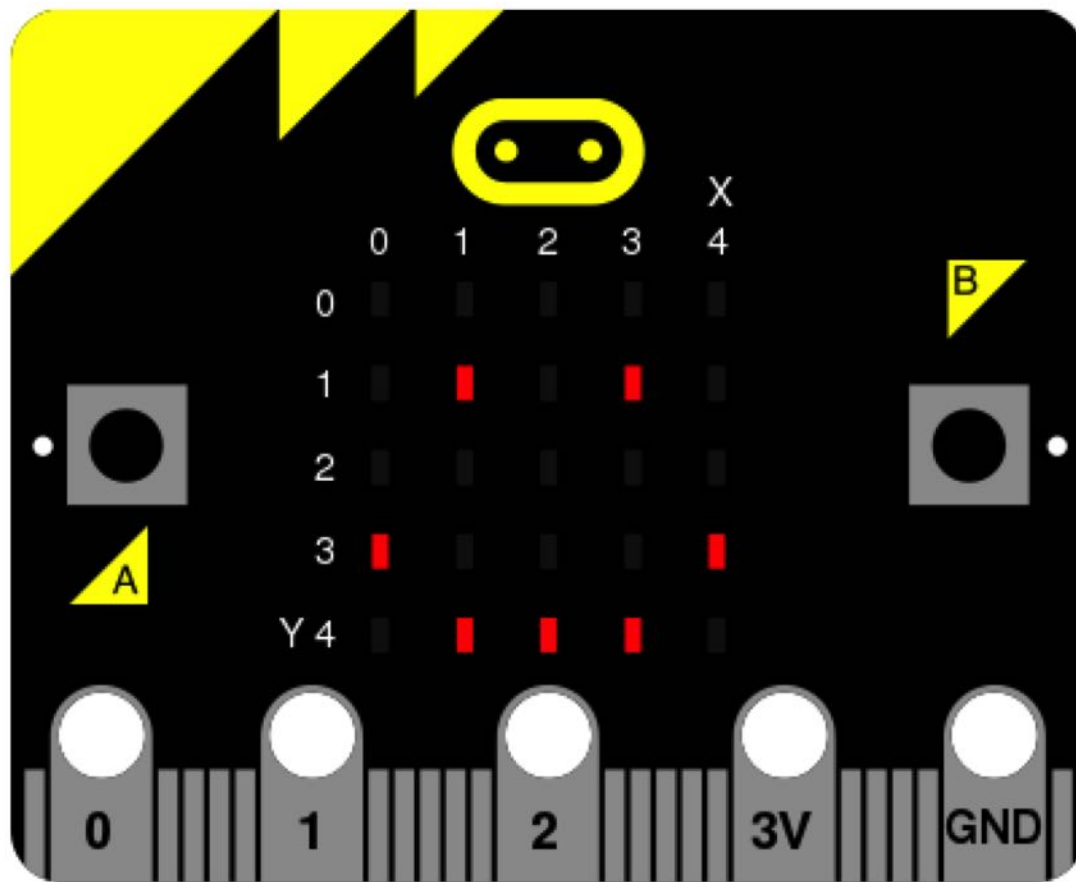
# 第一个程序：Hello World!

- 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

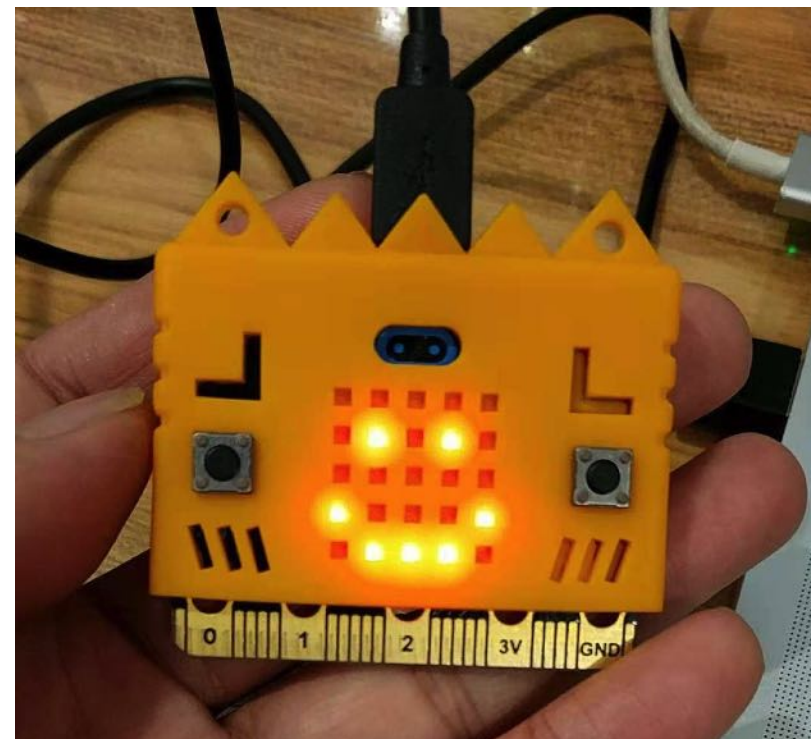


image1.py

```
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"

要点:  
\*\* 对象和赋值;  
\*\* 字符串表示



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

要点:

\*\* 序列和列表的应用;

\*\* 函数的关键字参数;

- 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

要点：

- \*\* 序列和列表的应用；
- \*\* 函数的关键字参数；

```
1  from microbit import *
2
3  boat1 = Image("05050:"
4               "05050:"
5               "05050:"
6               "99999:"
7               "09990")
8  boat2 = Image("00000:"
9               "05050:"
10              "05050:"
11              "05050:"
12              "99999")
13  boat3 = Image("00000:"
14               "00000:"
15               "05050:"
16               "05050:"
17               "05050")
```

```
18  boat4 = Image("00000:"
19               "00000:"
20               "00000:"
21               "05050:"
22               "05050")
23  boat5 = Image("00000:"
24               "00000:"
25               "00000:"
26               "00000:"
27               "05050")
28  boat6 = Image("00000:"
29               "00000:"
30               "00000:"
31               "00000:"
32               "00000")
33  all_boats = [boat1, boat2, boat3, boat4, boat5, boat6]
34  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( $b=0\sim9$ )
  - `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()

要点：嵌套的函数调用

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

要点：条件循环while

## 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

要点：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()
```

# 列表的一些操作

- 字符串的操作

- 字符串添加字符: `astr = astr + "1"`
- 字符串的长度: `len(astr)`


- 列表的操作

- 定义一个字符串的列表: `alist = ["11111", "01111", "00111"]`
- 判断一个字符串是不是在列表中: `astr in alist`
- 一个字符串在列表中的位置: `alist.index(astr)`
  - 例如: `alist.index("01111")`将返回1

# 【H7】 摩尔斯码

- 利用A, B按钮来输入点、划
- 显示一个笑脸, 开始按钮
- 接收5次按钮, 翻译成数字
- 如果翻译成功, 显示**数字**
- 如果翻译不成功, 显示**叉**  
(Image.NO)
- 返回显示笑脸

A: 点; B: 划



A table showing the Morse code for digits 1 through 0. Each digit is represented by a sequence of dots (A) and dashes (B) in a 5-column grid. The digits are listed on the left, and the corresponding Morse code is shown to their right.

1	●	■	■	■	■
2	●	●	■	■	■
3	●	●	●	■	■
4	●	●	●	●	■
5	●	●	●	●	●
6	■	●	●	●	●
7	■	■	●	●	●
8	■	■	■	●	●
9	■	■	■	■	●
0	■	■	■	■	■