

```
display.scroll("Hello, microbit_python!")
```

#Scroll to display ' Hello, microbit_python!' in micro:bit dot matrix, It can be replaced by any character.

```
display.show(Image.HAPPY)
```

#A smiley face showing on a micro:bit dot matrix,BBC Microbit_Python also has a number of built-in images that can be displayed on the screen (copy urls <http://www.qingchuangzhiyi.com/doc/image.htm>,open in the browser, and pull down to find properties.) is a list of built-in images

```
display.get_pixel(x, y)
```

Gets the brightness of the pixel (x, y), which can be 0 (off) to 9 (pixel at maximum brightness).

```
display.set_pixel(x, y, val)
```

Set the brightness of the pixel (x, y) to val (between 0 and 9)

```
display.clear()    #Clear display
```

```
display.scroll(string, delay=400)    #Scroll a string on the display, with  
400 milliseconds between each character.
```

```
image = Image('90009:09090:00900:09090:90009:'),    #Custom microbit  
display image, numerical brightness. Zero means extinguished. The  
brightest level of 9, the colon ":" indicates the end of an LED line.
```

```
image = Image(width, height)    #Create an empty image and give the size
```

```
image = Image(width, height, buffer)    #Initializes an image with the  
specified width and height. The buffer is an array of lengths and widths
```

`Image.width ()` # the width of the return image (usually 5)

`Image.height ()` # the height of the return image (usually 5)

`Image.set_pixel (x, y, value)` # sets the pixel to the specified location, with the value pixel between 0 and 9

`image.get_pixel(x, y)` #Gets the pixel at the specified location, with a value between 0 and 9

`image.shift_left(n)` #Returns a new image by moving the image to the left n times

`image.shift_right(n)` #Returns a new image by moving the image to the right n times

`image.shift_up(n)` #Returns a new image by moving the image to up n times

`image.shift_down(n)` #Returns a new image by moving the image to down n times

`repr(image)` # a string representation of the getting image

`accelerometer.get_x()` #Measure the force of gravity on the X-axis, in g

`accelerometer.get_y()` #Measure the force of gravity on the Y-axis, in g

`accelerometer.get_z()` #Measure the force of gravity on the Z-axis, in g

`accelerometer.get_values()` #Gets the gravity values for the X, Y, and Z axes (listed in this order)

`accelerometer.current_gesture()` #Get the current gesture value, BBC

Microbit_Python can recognize the following gestures: up, down, left,

right, face down, face down, free fall, 3 g, 6 g, 8 g, shake. The gestures in the program are up, down, left, right, face up, face down, freefall, 3g, 6g, 8g, shake.

`accelerometer.is_gesture(name)` #Returns true or false to indicate whether the current activity is a specified gesture.

`accelerometer.was_gesture(name)` #Returns true or false to indicate whether the last activity is a specified gesture.

`accelerometer.get_gestures()` #Returns a tuple of gesture history, the last activity being the last.

`compass.calibrate()` #Alignment compass

`compass.heading()` #Returns a number representing the degree offset of "north".

`compass.get_field_strength()` #Returns a numerical indication of the strength of the magnetic field

`compass.is_calibrated()` #Returns True if the electronic compass is successfully calibrated, or False if it is not.

`compass.clear_calibration()` #Uncalibrate and initialize to uncalibrated state.

`microbit.compass.get_x()` #The magnetic force on the X-axis is read as a positive integer or a negative integer, depending on the direction of the magnetic force

`microbit.compass.get_y()` #The magnetic force on the Y-axis is read as a positive integer or a negative integer, depending on the direction of the magnetic force

`microbit.compass.get_z()` #The magnetic force on the Z-axis is read as a positive integer or a negative integer, depending on the direction of the magnetic force

`button_a.is_pressed()` #Indicates that the button is being pressed and returns true or false

`button_a.was_pressed()` #Indicates whether the button was pressed, returns true or false, and then starts or finally calls the function.

`button_a.get_presses()` #This function returns the number of times A was pressed before. After this function is called, the count is cleared and the count is resumed.

`while True:` #An infinite loop

`For num in range(1,10):` # if the value of num is between 1 and 10, loop

`The if...` If... The event is correct or has occurred

`# do one thing execute this command`

`Elif...` If... The event is correct or has occurred

`# do another thing execute this command`

`The else:` # otherwise

`# do yet another thing execute this command`

`Ping.write_digital (value)` # writes a numeric value to the pin, which can be 0,1, False, True

`Ping.read_digital ()` # reads the numeric value of the pin and returns either 0 or 1

Write an analog value to the pin, which can range from 1 to 1023

`Ping.read_analog ()` # reads the analog value of the pin and returns a value between 1 and 1023

`Set_analog_period (int)` # sets the cycle of the PWM output in milliseconds

`Pin. Set_analog_period_microseconds (int)` # sets the output pins of the PWM cycle to microseconds

`Ping.is_touched ()` # returns a Boolean value if the pin is touched

`Music. Play (music.NYAN)` # play a melody

Built-in melodies complete list:

- `DADADADUM` - The fifth symphony in Beethoven -C minor begins.
- `ENTERTAINER` -Scott Joplin's Ragtime classic "The Entertainer" opening sequence.
- `PRELUDE` - Bach's 48 preludes in C major and fugue.
- `ODE` - Theme of Beethoven's symphony no. 9 in D minor, ode to joy
- `NYAN` - Yan Cat theme。 (<http://www.nyan.cat/>) Composer unknown. Fair in educational intent (as they say in New York).
- `RINGTONE` - It sounds like a ringtone. Used to indicate incoming messages.
- `FUNK` - A rough bass line for spies and criminal plotters.
- `BLUES` - The boogie-woogie 12 blues group continues
- `BIRTHDAY` - "Happy birthday to you..." For copyright information, <http://www.bbc.co.uk/news/world-us-canada-34332853>
- `WEDDING` -The wedding chorus from Wagner's Iohengrin.

- **FUNERAL** - "Funeral march" is the third movement of Chopin's piano sonata no. 2 in b flat minor (op. 35).
- **PUNCHLINE** - A funny clip of a joke has been made.
- **PYTHON** - John Philip Sousa's march also known as liberty bell, Monty Python's theme song (the Python programming language was later named)
- **BADDY** - The entrance of the silent film era is a bad man.
- **CHASE** - The silent film era chase scene.
- **BA_DING** - A short signal that something is happening.
- **WAWAWAWAA** - A very sad trombone
- **JUMP_UP** - Used in a game to indicate upward movement.
- **JUMP_DOWN** - For in-game use, indicate move down.
- **POWER_UP** - A campaign to unlock an achievement.
- **POWER_DOWN** - A sad advertisement for the loss of an achievement.

`import random`

`display.show(str(random.randint(1, 6)))`

#Gets a random number between 1 and 6 and converts it to a character that is displayed on the screen

`emakefunnames = ["Mary", "Yolanda", "Damien", "Alia", "Kushal", "Mei Xiu", "Zoltan"]`

`display.scroll(random.choice(emakefunnames))`

Gets a random name in the list emakefunnames and displays it on the microbit screen

`sleep(ms)` #Sleep (delay) time for a given number in milliseconds.

`running_time()` #Returns the last micro:bit startup time, in

microseconds

`reset()` #reset micro:bit