```
 1 # stream_plasma.py - implementation of
   oldskool plasma effect
 2 # see http://www.bidouille.org/prog/
   plasma
 3 #
 4 # 2017_0122 PePo adopted for default
   neopixel MicroPython library
 5 #           and LED matrix 8*8
 6 #
 7 # Sources: Youtube https://www.youtube.
   com/watch?v=QcyuYvyvOEI&index=14&list=
   PLuuAy8GJr5z1WoOJAFh1adr_yjCMJQ2Yl
 8 # Tony Dicola source: https://gist.
   github.com/tdicola/
   6fe1fbc173dcd49de3a95be5fd9594f6
 9
10 import machine
11 import math
12 import neopixel
13 import time
14
15 # LED matrix: 8 * 8 pixels
16 PIXEL_WIDTH = 8
17 PIXEL_HEIGHT = 8
18 MAX_BRIGHT = 50.0  # 100.0
19
20 # create a neopixel array
21 # NodeMU: neopixel connected to pin
   GPIO13 (D7)
22 np = neopixel.NeoPixel(machine.Pin(13),
   PIXEL_WIDTH * PIXEL_HEIGHT)
23
24 # Clear all the pixels and turn them off
   .
25 np.fill((0, 0, 0))
26 np.write()
27
28 while True:
```

```
29      np.fill((0, 0, 0))
30      current = time.ticks_ms() / 1000.0
31      for x in range(PIXEL_WIDTH):
32          for y in range(PIXEL_HEIGHT):
33              v = 0.0
34              v += math.sin(x + current)
35              v += math.sin(1.0 * (x *
    math.sin(current / 0.5) + y * math.cos(
    current / 0.25)) + current)
36              cx = x + 0.5 * math.sin(
    current / 5.0)
37              cy = y + 0.5 * math.cos(
    current / 3.0)
38              v += math.sin(math.sqrt((
    math.pow(cx, 2.0) + math.pow(cy, 2.0)) +
     1.0) + current)
39              v = (v + 3.0) / 6.0
40              # 2017_0122 added: color r,g
    ,b must be always > 0
41              r = math.sin(v * math.pi)
42              r = (r + 1.0) / 2.0 # scale
    to 0..1
43              g = math.sin(v * math.pi + 2
    .0 * math.pi / 3.0)
44              g = (g + 1.0) / 2.0 # scale
    to 0..1
45              b = math.sin(v * math.pi + 4
    .0 * math.pi / 3.0)
46              b = (b + 1.0) / 2.0 # scale
    to 0..1
47              np[y * PIXEL_WIDTH + x] = (
    int(MAX_BRIGHT * r),
48
    int(MAX_BRIGHT * g),
49
    int(MAX_BRIGHT * b))
50              # np[y * PIXEL_WIDTH + x
    ] = (int(MAX_BRIGHT * math.fabs(r)),
```

```
51
   #                                int(
   MAX_BRIGHT * math.fabs(g)),
52
   #                                int(
   MAX_BRIGHT * math.fabs(b)))
53      np.write()
54
```