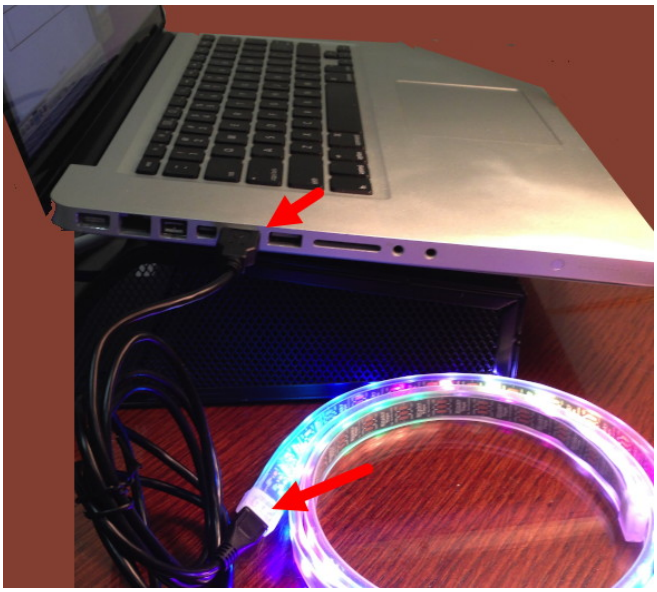
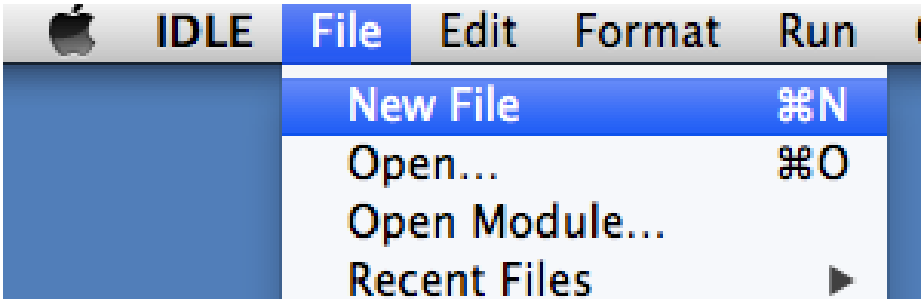
Python Programming with Blinky Tape Lessons 19-32 v9

Content created by Dr. Mark Miller, Chris Miller, and Ed Loeswick – Learningtech.org

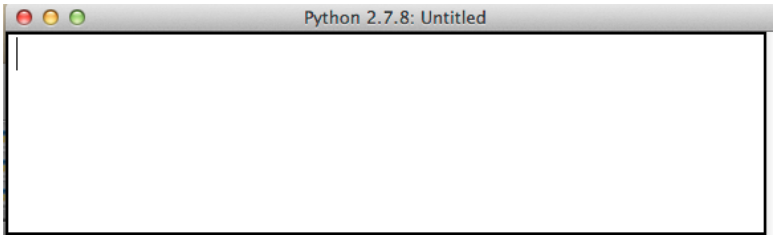


Make New File for Blinky Tape

Click:

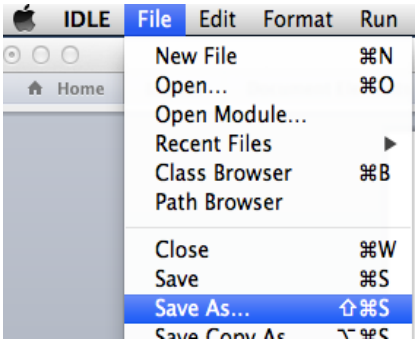


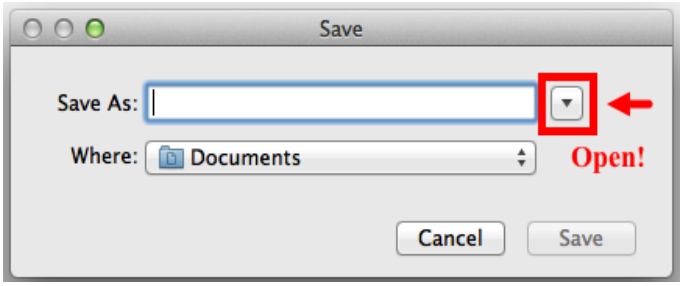
See:



Save As ... Documents ...

See:

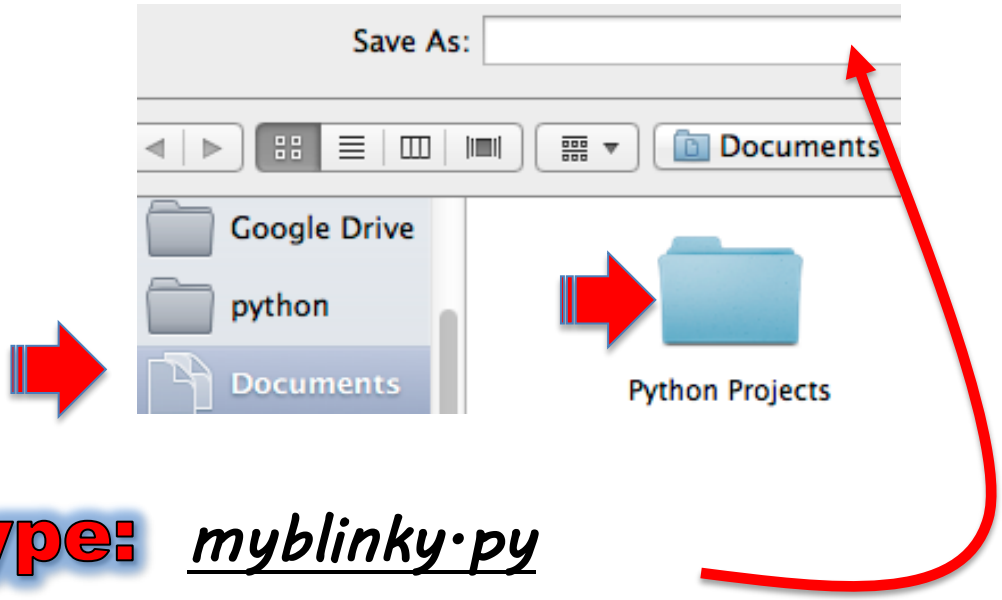




Python Projects/myblinky.py

Click: Documents

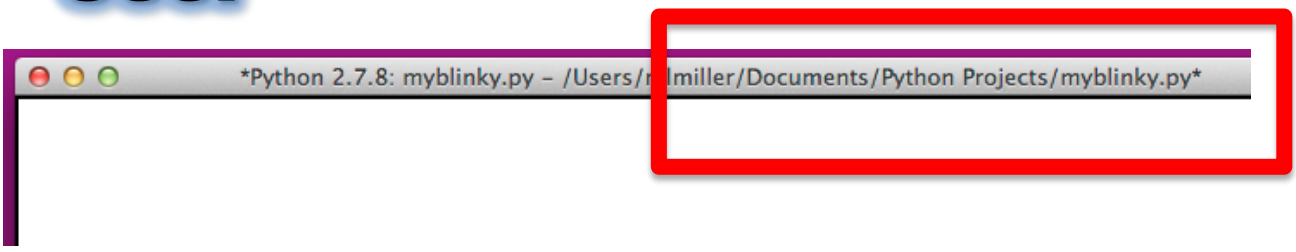
Double-Click: Python Projects



Type: myblinky.py

Click: Save

See:



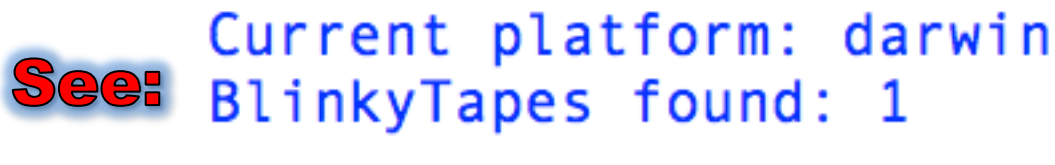
20a. Plug in Blinky Tape

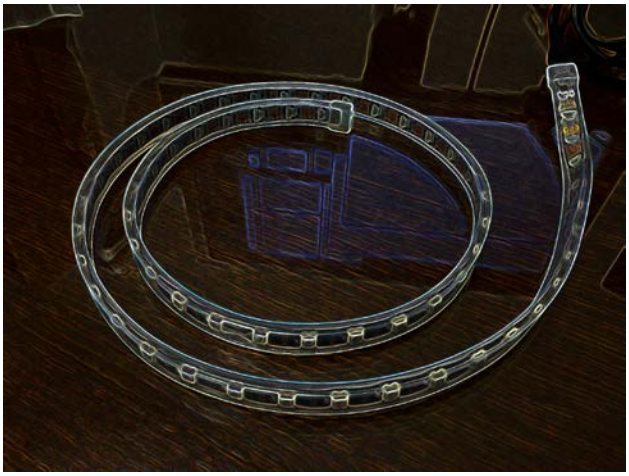
20b. Import Blinky Tape Essentials

Type :

```
from blinkytape import *
from random import *
from time import *
```

20c. Run Blinky Essentials



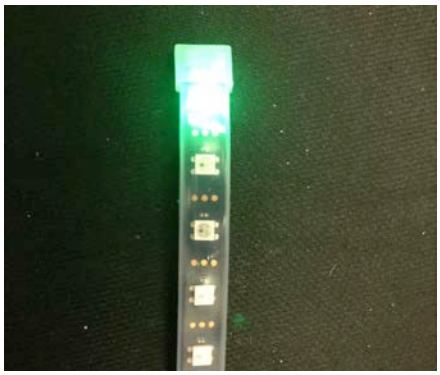


20d. Test Blinky Commands

Type – Shell

```
>>> colors
>>> led(0, 'red')
>>> clear()
>>> led(59, 'green')
```

See:



21a. Teach BlinkyTape a New Word

Type:

```
def onoff():
    for bulb in range(60):
        led(bulb, 'white')
        sleep(2)
        led(bulb, 'black')
```

Click: Save ⌘S **Click:** Run Module F5

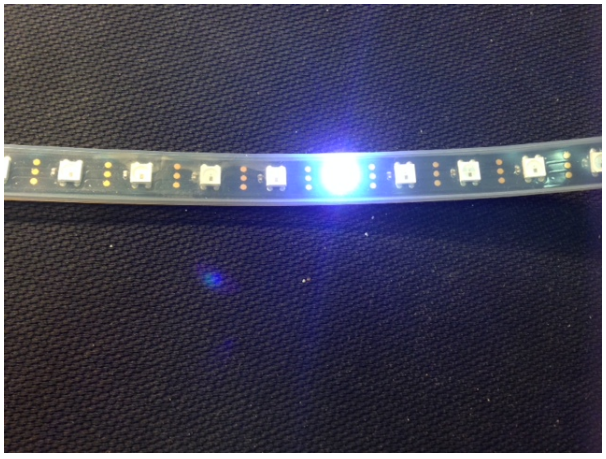
See:

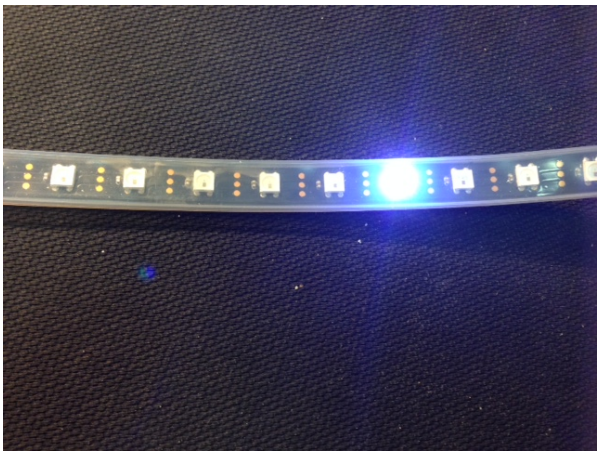
```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> onoff()
```

See:





Q: What about sleep(1)? sleep(30)?

21b. Add Random Colors

Type:

```
def onoffrc():
    for bulb in range(60):
        led(bulb, choice(colors))
        sleep(1)
        led(bulb, 'black')
```

Click: Save ⌘S **Click:** Run Module F5

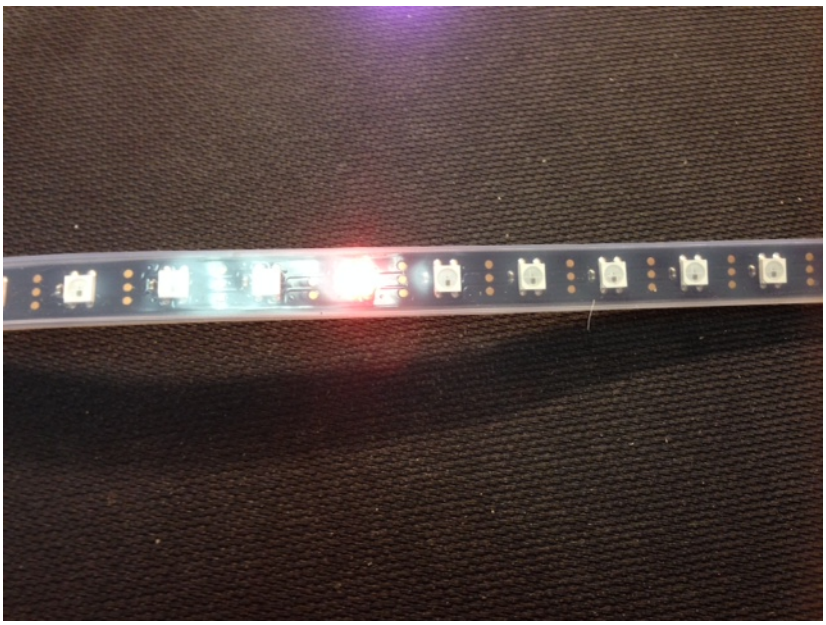
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> onoffrc()
```

See: ***(your colors may vary)***



22a. displayColor

Upper and lower case matters !

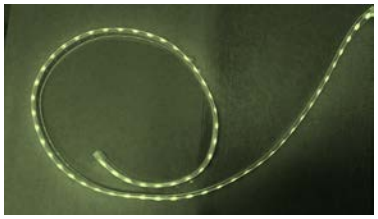
1st Input is How Much *RED* (0 – 255)

Shell: >>> displayColor(255,0,0)

See:

Shell: >>> displayColor(0,255,0)

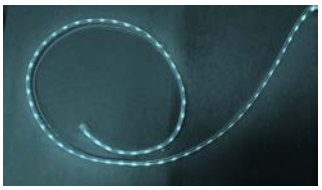
See:



2nd Input is How Much *Green* (0 – 255)

Shell: >>> displayColor(0,0,255)

See:



2nd Input is How Much *Blue* (0 – 255)

23a. Color Mixing - White

Type:

```
def allwhite():
    displayColor(255,255,255)
```

Click: Save ⌘S **Click:** Run Module F5

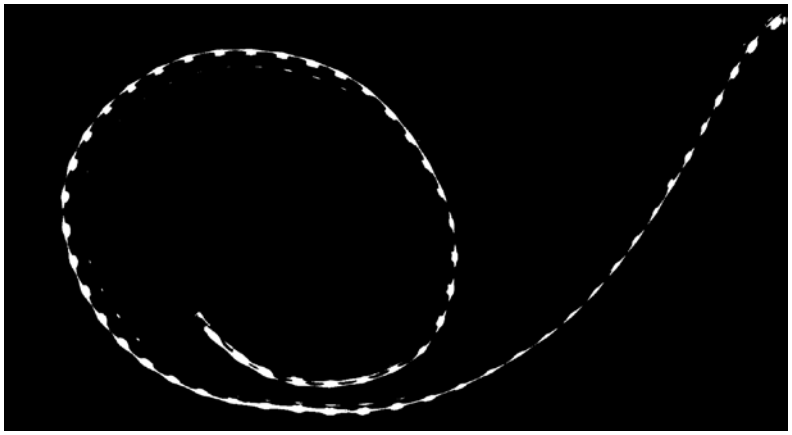
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> allwhite()
```

See:



23b. Color Mixing - Black

Type:

```
def allblack():
    displayColor(0,0,0)
```

Click: Save ⌘S **Click:** Run Module F5

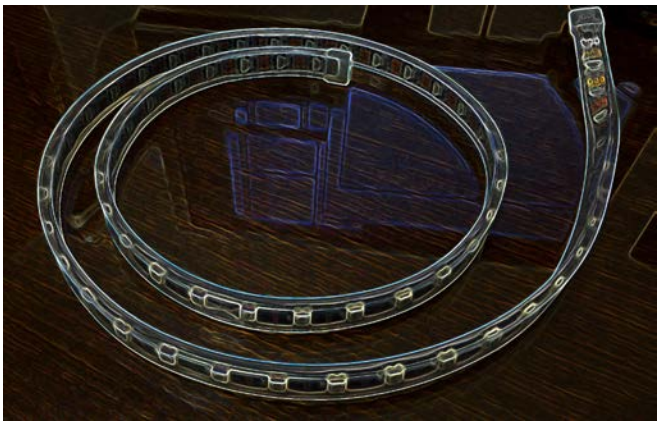
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> allblack()
```

See:



24a. Color Mixing - Purple

Type:

```
def allpurple():
    displayColor(255,0,255)
```

Click: Save ⌘S **Click:** Run Module F5

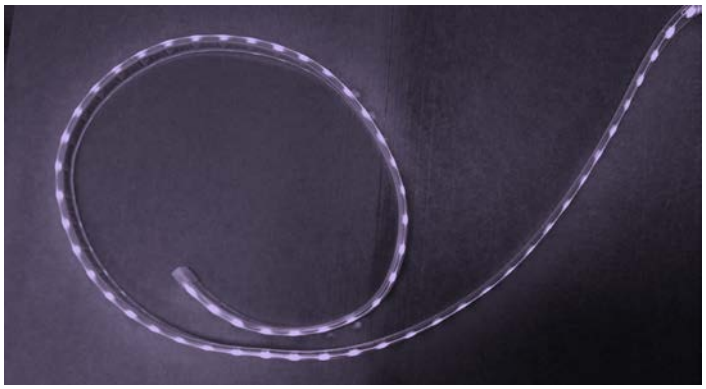
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> allpurple()
```

See:



24b. Color Mixing - Yellow

Type :

```
def allyellow():
    displayColor(255,255,0)
```

Click: Save ⌘S **Click:** Run Module F5

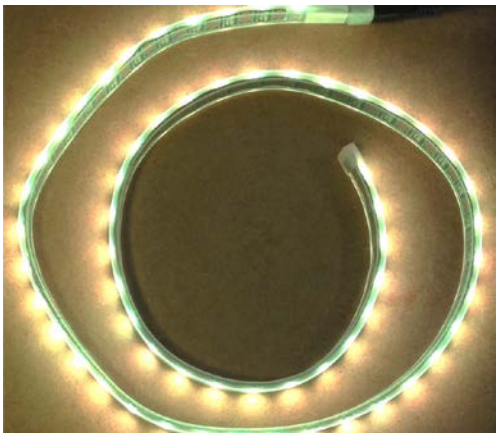
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> allyellow()
```

See:



24c. Color Mixing – Orange

Type:

You Decide! Experiment With Different Inputs!

```
def allorange()
    displayColor( ???, ???, ??? )
```

Save/Run/See ?

25a. Color Mixing – More Colors!

Try to mix Teal, Taupe, Magenta or your own favorite 2-3 colors. If you get stuck, you may search the Internet for RGB color codes, *after* you have exhausted trial and error!

Type:

```
def allteal():
    displayColor(???,???,???)
```

Click: Save ⌘S **Click:** Run Module F5

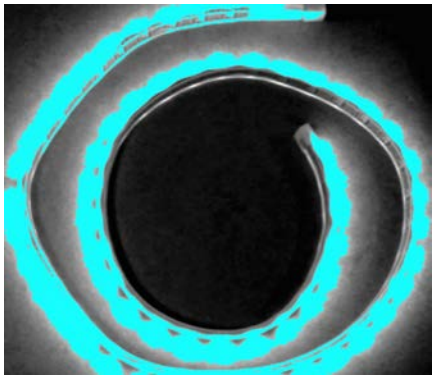
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> allteal()
```

See:



25b. setPixel # immediately

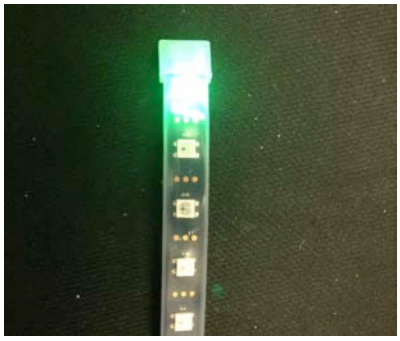
Shell:

```
>>> clear()
```

True: "Update Pixel NOW"

```
>>> setPixel(59,0,255,0,True)
```

See:



25c. setPixel # on sendUpdate()

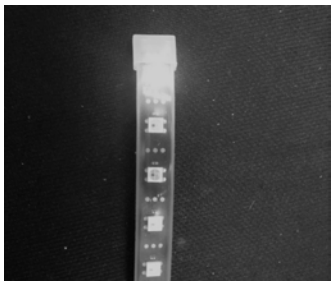
Shell:

```
>>> clear()
```

False: "Wait for sendUpdate()"

```
>>> setPixel(59,0,255,0,False)
```

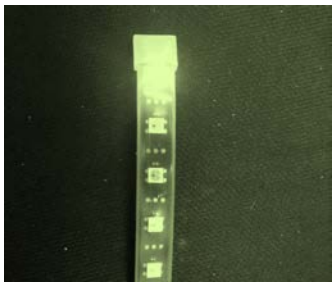
See:



Shell:

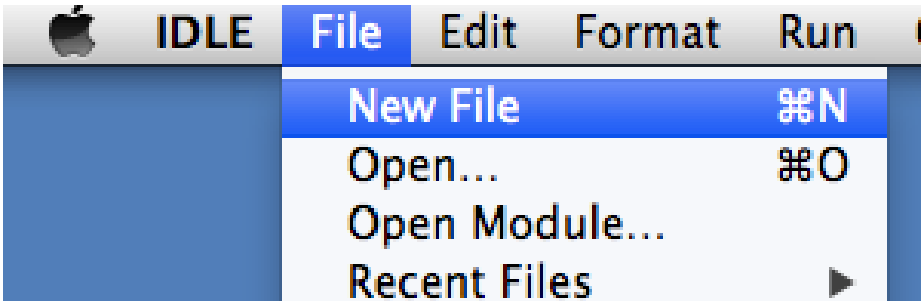
```
>>> setPixel(58,0,255,0,False)
>>> setPixel(57,0,255,0,False)
>>> setPixel(56,0,255,0,False)
>>> sendUpdate()
```

See:

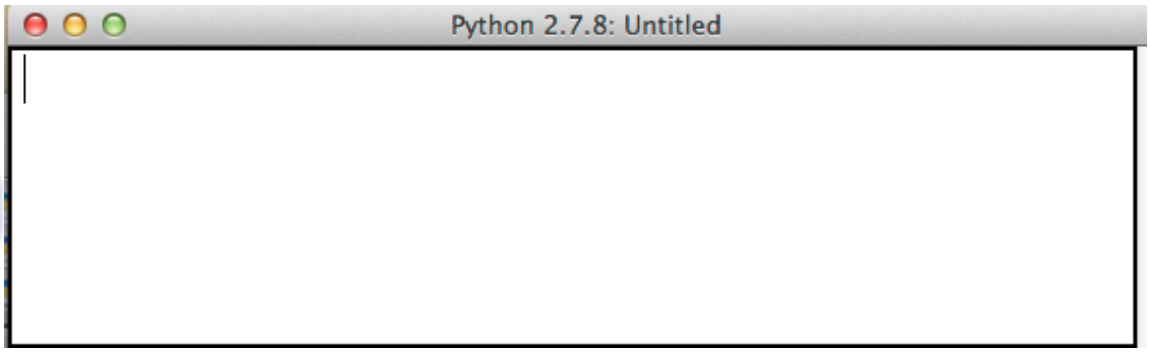


Make New File for Programs #26-28

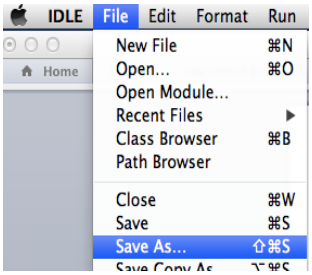
Click:



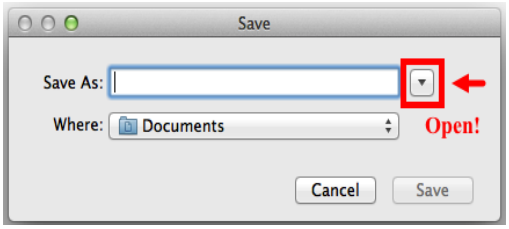
See:



Save As ... Documents ...



See:

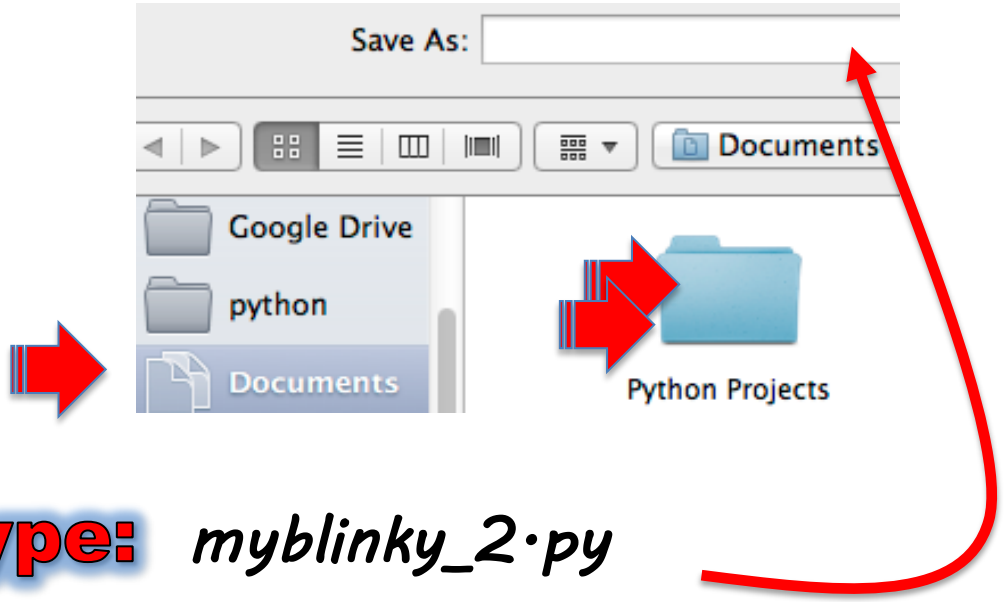


Python Projects/myblinky_2.py

Use “myblinky_2.py” as a new file name.

Click: *Documents*

Double-Click: *Python Projects*



Type: *myblinky_2·py*

Click: Save

See:

26a. All Blues, Up Tempo

Type:

```
from blinkytape import *
from random import *
from time import *
def allblues():
    for p in range (60):
        setPixel(p,0,0,255,True)
```

Click: Save ⌘S **Click:** Run Module F5

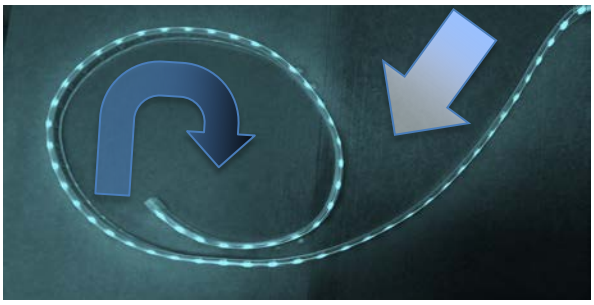
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> allblues()
```

See:



26b. Slow Blues

Type:

```
def slowblues():
    for p in range (60):
        setPixel(p,0,0,255,True)
        sleep(2.5)
```

Save/Run/Shell: `>>> slowblues()`

See: ***(same idea but slower)***

26c. Suddenly Blue

Type:

```
def suddenlyblue():
    for p in range (60):
        setPixel(p,0,0,255,False)
    sendUpdate()
```

Click: Save ⌘S **Click:** Run Module F5

See:

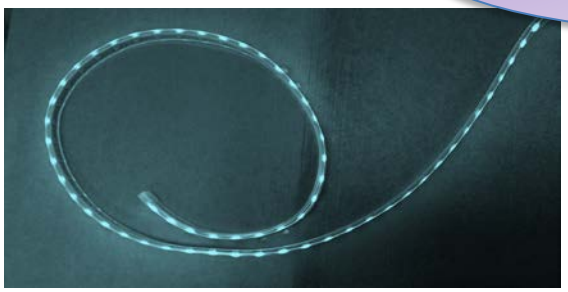
```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> suddenlyblue()
```

Is this faster or slower than allblues()?

See:



27a. Up and Back

Type:

```
def upandback():
    for p in range (60):
        setPixel(p,0,0,255,True)
    for q in range (60):
        setPixel(59-q,255,128,0,True)
```

Do you see how to count backwards?

Click: Save ⌘S **Click:** Run Module F5

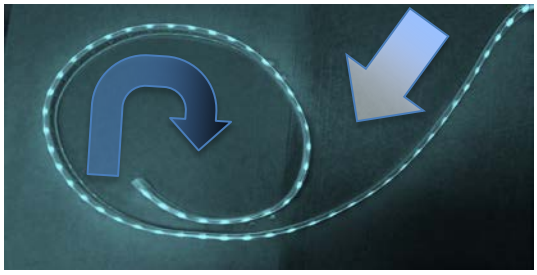
See:

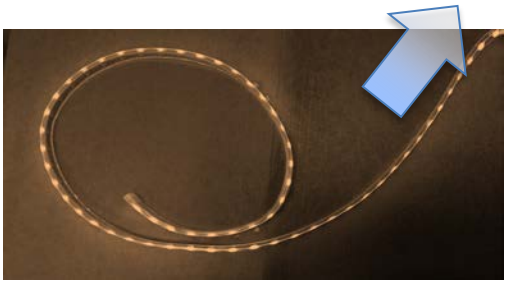
```
Current platform: darwin
BlinkyTapes found: 1
```

Shell:

```
>>> upandback()
```

See:





27b. Up and Back, N Times

Edit – myblinky_2.py:

```
## Comment out the previous version first
def upandback(n):
    for p in range (60):
        setPixel(p,0,0,255,True)
    for q in range (60):
        setPixel(59-q,255,128,0,True)
    if n > 0:upandback(n-1)
```

Click: Save ⌘S

Click: Run Module F5

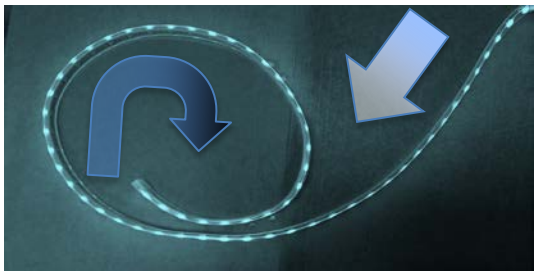
See:

```
Current platform: darwin
BlinkyTapes found: 1
```

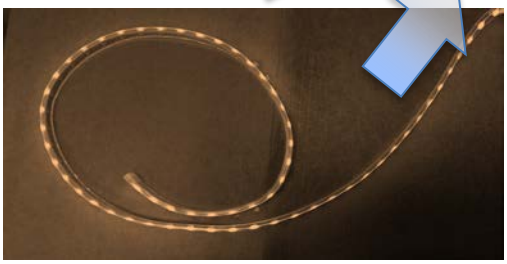
Shell:

```
>>> upandback(10)
```

See:



Back and Forth, 10 Times



28a. Flash Gordon

Type:

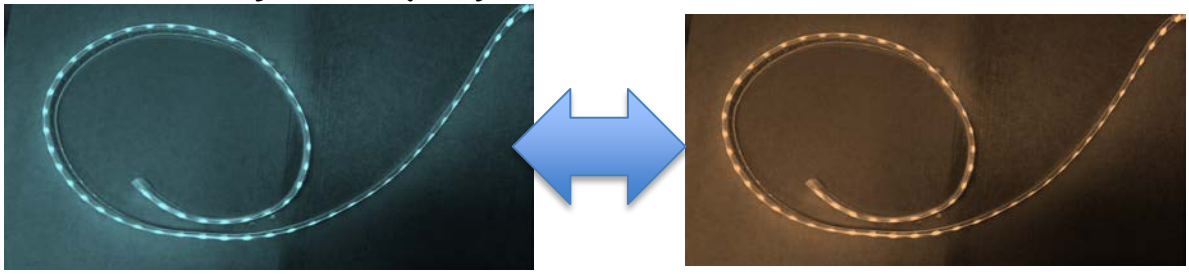
How is this different from Up and Back?

```
from blinkytape import *
from random import *
from time import *
def flashgordon(n):
    for p in range (60):
        setPixel(p,0,0,255,False)
    sendUpdate()
    sleep(0.5)
    for q in range (60):
        setPixel(59-q,255,128,0,False)
    sendUpdate()
    sleep(0.5)
    if n > 0: flashgordon(n-1)
```

Shell:

```
>>> flashgordon(10)
```

See:



28b. While True Coolness

Type:

```
def coolness():
    clear()
    sleep(1)
    while True:
        for p in range (60):
            setPixel(p,0,255,0,False)
        sendUpdate()
        sleep(0.1)
        for q in range (60):
            setPixel(59-q,255,128,0,False)
        sendUpdate()
        sleep(0.1)
```

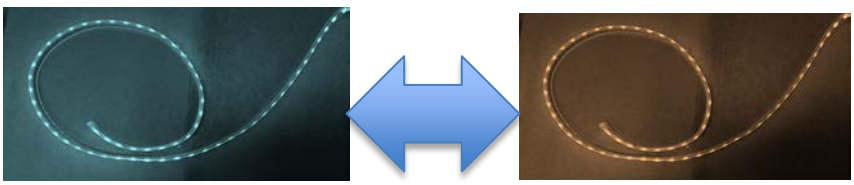
Click: Save ⌘S **Click:** Run Module F5

See:

```
Current platform: darwin
BlinkyTapes found: 1
```

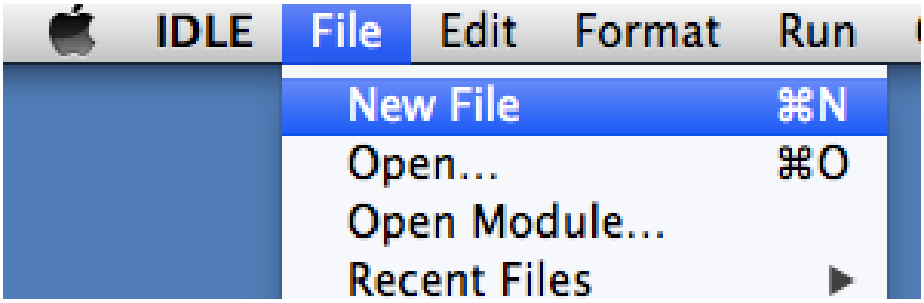
Shell: `>>> coolness()` ***(Control-C to stop)***

See:

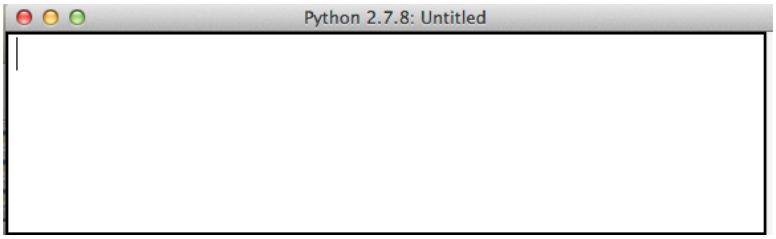


Make New File for Programs #29-32

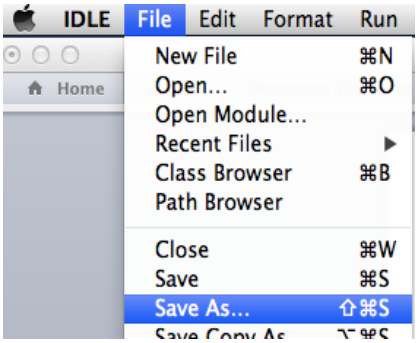
Click:



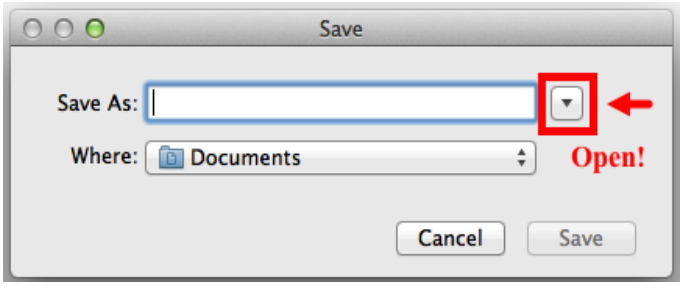
See:



Save As ... Documents ...



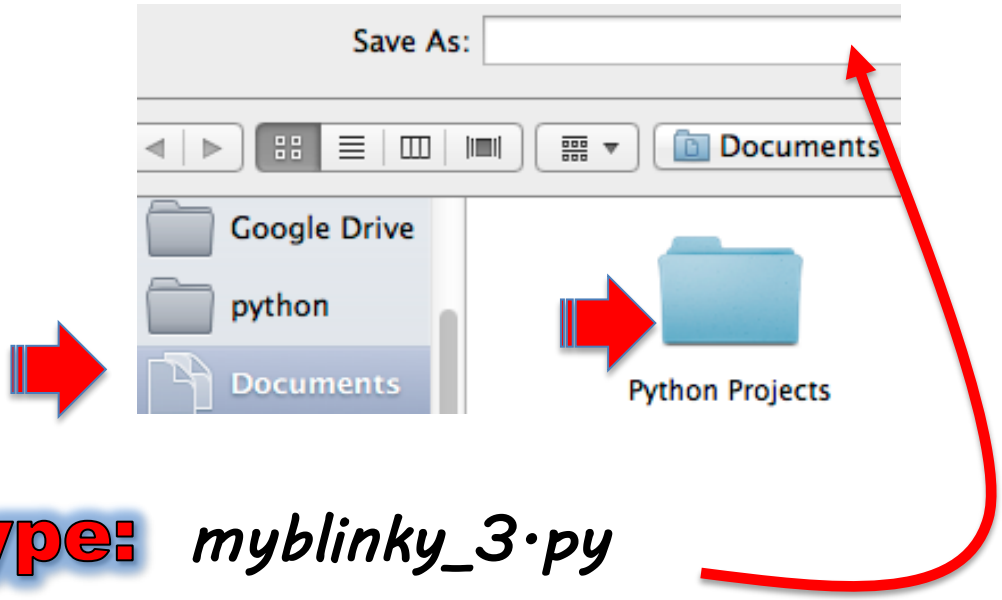
See:



Python Projects/myblinky_3.py

Click: Documents

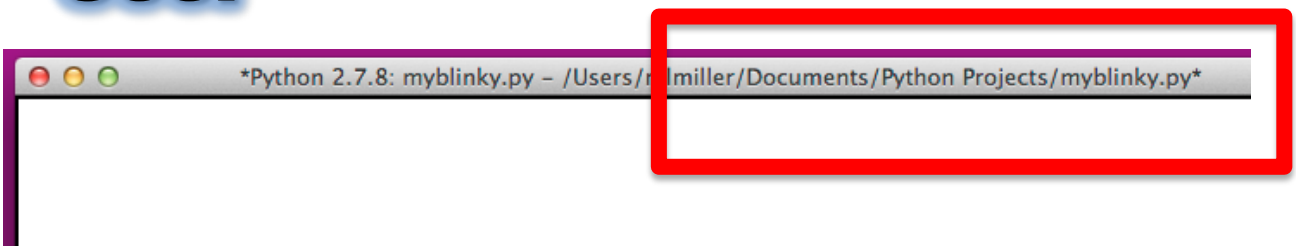
Double-Click: Python Projects



Type: myblinky_3·py

Click: Save

See:



29a. RGB Randomness Out of Loop

Shell:

```
>>> rval = randint(0,255)
>>> rval
>>> gval = randint(0,255)
>>> gval
>>> bval = randint(0,255)
>>> bval
```

Numbers should vary -- randomly!

See: 52

177

38

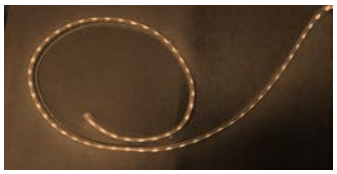
Type:

```
from blinkytape import *
from random import *
from time import *
def outerrand():
    rval = randint(0,255)
    gval = randint(0,255)
    bval = randint(0,255)
    for pxl in range (60):
        setPixel(pxl,rval,gval,bval,True)
```

All LEDs should be the same (random) color. Your colors may vary.

29b. RGB Randomness Inside Loop

Shell: `>>> outerrand()`



Type:

```
def innerrand():
    for pxl in range (60):
        rval = randint(0,255)
        gval = randint(0,255)
        bval = randint(0,255)
        setPixel(pxl,rval,gval,bval,True)
```

Each individual LED should be its own random color

Click: Save ⌘S **Click:** Run Module F5

See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell: `>>> innerrand()`

See:

30a. Individual Versus Group

Type:

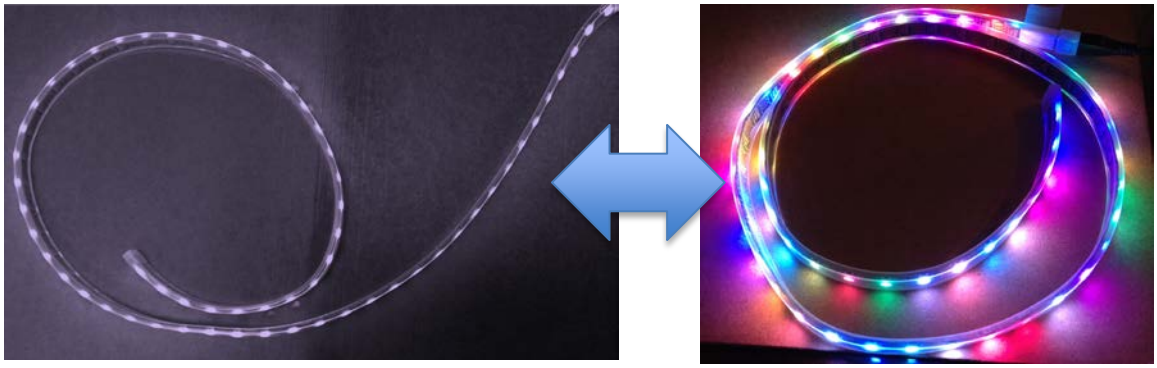
```
def alternate():
    while True:
        clear()
        outerrand()
        sleep(0.5)
        innerrand()
        sleep(0.5)
```

Click: Save ⌘S **Click:** Run Module F5

See:

```
Current platform: darwin
BlinkyTapes found: 1
```

Shell: `>>> alternate()` ***(Control-C to stop)***



30b. Cylon

Type:

```
def cylon():
  # pulsating red "eye" goes back & forth
  print "Now running: cylon()"
  blue = 10
  red = 255
  #start by setting entire strip blue
  displayColor(0,0,blue)
  counter = 0 #eye position starts @ pixel 0
  while (counter < 59): # move eye up tape
        #set counter location to blue
        setPixel(counter,0,0,blue,True)
        #set pixel above it to red
        setPixel(counter+1,red,0,0,True)
        counter += 1 #increase counter by 1
  while (counter > 0): # move eye down tape
         #set counter location to blue
         setPixel(counter,0,0,blue,True)
         #set pixel below it to red
         setPixel(counter-1,red,0,0,True)
         counter -= 1 #decrease counter by 1
```

See:

Shell:

```
>>> cylon()
```

31. Open Project

Open File – sunnyvaledemo.py:

Try running these demos! However, please do not make changes to this file! Try out some of the examples individually. Read the code! All good programmers learn by reading other people's code. (If you want to be a great writer, you first need to read a lot.)

Notice that familiar functions such as **setPixel** have an extra word in the front such as **tape.setPixel**. This just tells Python which add-on modules you are using, such as turtles or blinkytapes. Sometimes a functions like clear() that have one meaning when it is tape.clear() and another meeting when it is turtle.clear().

Find a demo you like, **make a copy of it into your own myblinky.py file**, and then experiment with ways to improve it. Remember, do not change the original Sunnyvale file. This is your chance to get really creative! You might want to keep in mind our Holiday Amusement Park theme.

32. Counting in Binary

Type (in a new file: mybinary.py):

```
from blinkytape import *

def is_odd(num):
    return ((num % 2) == 1)

def binary(num):
    clear()
    print "Decimal:", num
    print "Binary:", binarydigits(num,0,"")

def binarydigits (num, nextled,bits):
    if is_odd(num):
        bits = "1" + bits
        led(nextled, "white")
    else:
        bits = "0" + bits
        led(nextled, "black")
    if ((nextled < 59) and (num >= 0)):
        return(binarydigits(num/2, nextled+1,bits))
    else:
        return(bits)
```

What does + mean here?

Why not just use *for i in range (n)* here?

```
def countup(n):
    i = 0
    answer = ""
    while ( (i <= n) and answer != "q"      ):
        binary(i)
        i = i+1
        answer = raw_input("Hit q to quit, RETURN
                           to continue: ")
    print ("Done!")
```

What does % mean here?

Click: Save ⌘S **Click:** Run Module F5

Shell:

```
>>> binary(5)
```

See:

Decimal: 5

Binary: 0 … 00000000000000000000000000000101

Shell (more examples to try)

```
>>> binary(0)
>>> binary(1)
>>> binary(2)
>>> binary(3)
>>> binary(4)
>>> binary(6)
>>> binary(7)
>>> binary(8)
>>> binary(15)
>>> binary(16)
>>> print (2**60) - 1
>>> binary(2**60 - 1)
>>> countup(7)
>>> countup(64)
>>> countup(1152921504606846975) #((2**60 - 1)
```