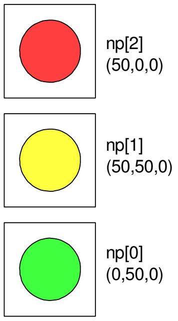
ECE 476/676 - Test #3: Name ____________________

1) NeoPixel: Write a python program to control a stoplight. Assume the Pico is treated like an I2C device where a register (X) controls its operation.

- X[0] = operating mode
  - 0: Normal (green → yellow →red→repeat)
  - 1: Red
- X[1] = green time in ms
- X[2] = yellow time in ms
- X[3] = red time in ms



For example, if you want normal operation with the times being 5000ms (g), 1000ms (y), 6000ms (r), X would be initialized as

```
from machine import Pin, bitstream
X = [0, 5000, 1000, 6000]
```

Write a python program which drives a stoplight made up of three NeoPixels based upon the value of X. You're free to use any method to drive the NeoPixel (bitstream, neopixel library, PIO state machines, etc)

```
from machine import Pin, bitstream
from time import sleep_ms

timing = [300, 900, 700, 500]
np = Pin(12, Pin.OUT)

N = 3

red = bytearray([0,0,0,0,0,0,0,50,0])
yellow = bytearray([0,0,0,50,50,0,0,0,0])
green = bytearray([50,0,0,0,0,0,0,0,0])
X = [0,5000,1000,6000]

while(1):
   if(X[0] == 0):
      bitstream(np, 0, timing, green)
      sleep_ms(X[1])
      bitstream(np, 0, timing, yellow)
      sleep_ms(X[2])
      bitstream(np, 0, timing, red)
      sleep_ms(X[3])
   if(X[0] == 1):
      bitstream(np, 0, timing, red)
```

2) GPS messages look like the following. The altitude shows up under a GPGGA message.

```
'$GPGGA,205246.00,4649.55240,N,09652.11367,W,1,07,1.17,283.7,M,-27.5,M,,*69\r\n'
'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,3.59,1.17,3.39*0C\r\n'
'$GPGSV,2,1,08,08,19,311,09,10,52,288,24,15,28,055,21,18,47,147,25*78\r\n'
'$GPGSV,2,2,08,23,77,015,19,24,39,100,21,27,32,277,1
'$GPRMC,205247.00,A,4649.55258,N,09652.11395,W,0.306,,140724,,,A*62\r\n'
'$GPVTG,,T,,M,0.306,N,0.567,K,A*22\r\n'
'$GPGGA,205247.00,4649.55258,N,09652.11395,W,1,07,1.14,284.1,M,-27.5,M,,*6E\r\n'
'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,2.49,1.14,2.22*04\r\n'
'$GPGSV,2,1,08,08,19,311,08,10,52,288,25,15,28,055,22,18,47,147,26*78\r\n'
'$GPGSV,2,2,08,23,77,015,19,24,39,100,21,27,32,277,1
'$GPRMC,205248.00,A,4649.55297,N,09652.11403,W,0.312,,140724,,,A*63\r\n'
'$GPVTG,,T,,M,0.312,N,0.578,K,A*29\r\n'
'$GPGGA,205248.00,4649.55297,N,09652.11403,W,1,07,1.14,284.5,M,-27.5,M,,*6E\r\n'
'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,2.49,1.14,2.22*04\r\n'
```

Write a Python subroutine which

- Is passed a single GPS message (X) as a text string,
  - *GPS messages start with a $ and end with a* \r\n
- Returns your altitude as a text string
  - The numbers shown in red if it's a GPGGA message
  - Return an empty string if it's not a GPGGA message

```
def GPS_Problem(X):
   if(len(X) < 59):
      reply = ''
   elif(X[0:6] != '$GPGGA')
      reply = ''
   else:
      reply = X[54:59]
   return(reply)
```

Another solution

```
def GPS_Problem(X):
   X = X.strip('\r\n')
   Y = X.split(',')
   return(Y[9])
```

3) BlueTooth: Assume you have received a Bluetooth message with the format of

- *msg = 'aa,bbbbb.bb/r/n'*

where

- *aa* is a number of uncertain length (0 to 99)
- *bbbbb.bb* is a number of uncertain length (0 to 9999.999)
- *a* and *b* are spearated with a comma
- the message is terminated with a carriage-return, line-feed (/r/n)

Write a python subroutine which

- Receives a Bluetooth message like the one above, and
- Stores the number *b*
- At memory location *a*

For example, if *msg = '12,345.6/r/n',* then the number 345.6 would be stored in X[12]

note: This makes the Pico behave like an I2C device where you interract with it by reading and writing to register locations.

```
X = [0]*100
def BlueTooth_Problem(msg):
   global X
   msg = str(msg)
   # write the rest of the code

   n = msg.find(',')
   aa = msg(0:n)
   msg = msg(n+1:)

   n = msg.find('/')
   bb = msg(0:n)

   aa = int(aa)

   X[aa] = float(bb)
```

Another Solution

```
def BlueTooth_Problem(msg):
    global X
    msg = msg.strip('/r/n')
    Y = msg.split(',')
    a = int(Y[0])
    b = float(Y[1])
    X[a] = b
```

4) WiFi: Assume a client is attached to your Pi-Pico set up as a WiFi host. Each time the client clicks on a *Submit* button for a text tag, two messages are sent. For example, if the messages are *N0=Hello+World* and *N1=3.14159*, the WiFi messages are:

Message #1 (text string)

```
x = 'GET /action_page.php?N0=Hello+World&N1=3.14159 HTTP/1.1\r\nHost: 192.168.4.1
\r\nConnection: keep-alive\r\nUpgrade-Insecure-Requests: 1\r\nUser-Agent: Mozilla
/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
131.0.0.0Safari/537.36\r\nAccept:\r\ntext/html, application/xhtml+xml, applicatio
n/xml; q=0.9, image/avif, image/webp,image/apng, */*; q=0.8, application/signed-e
xchange;v=b3; q=0.7\r\nReferer: http://192.168.4.1/\r\nAccept-Encoding: gzip,defl
ate\r\nAccept-Language: en-US,en;q=0.9\r\n'
```

Message #2 (text string)

```
x = 'GET /favicon.ico HTTP/1.1\r\nHost: 192.168.4.1\r\nConnection: keep-alive\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)AppleWebKit/537.36 (KHTML,li
ke Gecko) Chrome/131.0.0.0 Safari/537.36\r\nAccept:image/avif,image/webp,image/ap
ng,image/svg+xml,image/*,*/*;q=0.8\r\nReferer: http://192.168.4.1/action_page.php
?N0=Hello+World&N1=3.14159\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language:e
n-US,en;q=0.9\r\n'
```

Write a python routine which

- Receives a message (x) as a long text string (Message #1 or Message #2)
- If message #1 is received, it returns the number 1 along with the string sent
  `[1, b'N0=Hello+World&N1=3.14159']` *for the above message #1*
- If message #2 is received, it returns the number 2 along with an empty string
  `[2, b'']` *for the above message #2*
- Each message is a text string which can vary in content and length

```
def WiFi_Problem(x):
    n = x.find('/')+1
    x = x[n:]
    msg = x[0:6]
    if(msg == 'action'):
        n = x.find('?') + 1
        x = x[n:]
        n = x.find('HTTP')-1
        x = x[0:n]
        type = 1
    else:
        x = ''
        type = 2
    return(type, x)
```