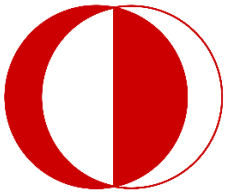


Principles of Measurement & Instrumentation I

Laboratory

PHYS417

Preliminary Work

BATUHAN DERELİ

1851690

Experiment 3– Data Acquisition

Experiment Date: 23.10.2019

Submission Date: 23.10.2019

Question 1: In Python, what do these functions do: *split() – encode() – decode() – len() – min() – max() – sorted() – sum() – type() – strip() – count() – clear() – range() – map() – str() – int() – repr() – index() – find() – enumerate() – next() – open() – round()*

Answer:

****

split() *is the splitting separating strings I will give the example*

```
txt = "WE ARE WRITING LAB REPORT"

x = txt.split()

print(x)
```

RESULT IS THE ['WE', 'ARE', 'WRITING', 'LAB', 'REPORT'] SO EVERY SINGLE WORD OR IN COMPUTER LANGUAGE STRING SEPARATE .

****

encode()

UTF-8 encode the string , **UTF-8** is a variable width character encoding capable of encoding all 1,112,064 valid code points in Unicode using one to four **8**-bit bytes. The encoding is defined by the Unicode Standard, and was originally designed by Ken Thompson and Rob Pike.

```
txt = "My name is Ståle"

x = txt.encode()

print(x)
```

##########

```
C:\Users\My Name>python demo_string_encode.py
b'My name is St\xc3\xe5le'
```

****

decode() is the Python string method **decode()** decodes the string using the codec registered for *encoding*. It defaults to the default string encoding.

```
Str = "this is string example....wow!
```

****

```
!!";
Str = Str.encode('base64','strict');

print "Encoded String: " + Str
print "Decoded String: " + Str.decode('base64','strict')
```

```
result ==
Encoded String: dGhpcyBpcyBzdHJpbmcgZXhhbXBsZS4uLi53b3chISE=
Decoded String: this is string example....wow!!!
```

len() The len() function returns the number of items of an object.

```
testList = []
print(testList, 'length is', len(testList))

testList = [1, 2, 3]
print(testList, 'length is', len(testList))
```

```
answer :

[] length is 0
[1, 2, 3] length is 3
[] length is 0
[1, 2, 3] length is 3
```

****

min()

Return the lowest number

```
x = min(5, 10)
```

```
C:\Users\My Name>python demo_min.py
5
```

max()

Return the largest number:

```
x = min(5, 10)
```

```
C:\Users\My Name>python demo_max.py
10
**********
```

sorted()

Sorting any sequence

```
x = [2, 8, 1, 4, 6, 3, 7]
print "Sorted List returned :",
print sorted(x)
```

```
Sorted List returned : [1, 2, 3, 4, 6, 7, 8]
```

****

sum()

Add all items in a tuple, and return the result:

```
a = (1, 2, 3, 4, 5)
x = sum(a)
C:\Users\My Name>python demo_sum.py
15
```

****

type()

method returns class type of the argument(object) passed as parameter. type() function is mostly used for debugging purposes.

```
print(type([]) is list)

print(type([]) is not list)

print(type(()) is tuple)

print(type({}) is dict)

print(type({}) is not list)
```

```
True

False

True

True

True
```

strip()

cters stripped. When the combination of characters in the chars argument mismatches the character of the string in the left, it stops removing the leading characters.

```
string = ' xoxo love xoxo   '

# Leading whitepsace are removed
print(string.strip())

print(string.strip(' xoxoe'))

# Argument doesn't contain space
# No characters are removed.
print(string.strip('sti'))
```

```
string = 'android is awesome'
print(string.strip('an'))
```

```
xoxo love xoxo
lov
 xoxo love xoxo
droid is awesome
```

---

count()

Return the number of times the value "cherry" appears int the `fruits` list:

```
fruits = ['apple', 'banana', 'cherry']

x = fruits.count("cherry")
```

```
C:\Users\My Name>python demo_list_count.py
1
```

---

clear() The clear() method removes all items from the dictionary.

```
fruits = ['apple', 'banana', 'cherry', 'orange']

fruits.clear()
```

```
C:\Users\My Name>python demo_list_clear.py
[]
```

---

range()

Create a sequence of numbers from 0 to 5, and print each item in the sequence:

```
x = range(6)
for n in x:
  print(n)
```

```
C:\Users\My Name>python demo_range.py
0
1
2
3
4
5
```

map()

function returns a list of the results after applying the given function to each item of a given iterable (list, tuple etc.)

```
def addition(n):
    return n + n

# We double all numbers using map()
numbers = (1, 2, 3, 4)
result = map(addition, numbers)
print(list(result))
```

```
{2, 4, 6, 8}
```

str()

Convert the number 3.5 into an string:

```
x = str(3.5)
```

```
C:\Users\My Name>python demo_str.py
3.5
```

int()

function in Python and Python3 converts a number in given base to decimal.

```
num =  13
String =  '187'
# Stores the result value of
# binary "187" and num addition
result_1 =  int(String) +  num
print("int('187') + 13 = ", result_1, "\n")
```

```
nt('187') + 13 =  200
```

repr()

```
s = 'Hello, Geeks.'
print repr(s)
print repr(2.0/11.0)
```

```
'Hello, Geeks.'
```

```
0.18181818181818182
```

index()
index is an inbuilt function in Python, which searches for given element from start of the list and returns the lowest index where the element appears.

```
list1 = [1, 2, 3, 4, 1, 1, 1, 4, 5]

# Will print the index of '4' in list1
print(list1.index(4))
```

```
2
```

find()

The find() method returns the lowest index of the substring if it is found in given string. If its is not found then it returns -1.

```
word = 'geeks for geeks'

# Substring is searched in 'eks for geeks'
print(word.find('ge', 2))

# Substring is searched in 'eks for geeks'
print(word.find('geeks ', 2))

# Substring is searched in 's for g'
print(word.find('g', 4, 10))

# Substring is searched in 's for g'
print(word.find('for ', 4, 11))
```

```
10

-1

-1
```

```
6
```

enumerate()
A lot of times when dealing with iterators, we also get a need to keep a count of iterations. Python eases the programmers' task by providing a built-in function enumerate() for this task.
Enumerate() method adds a counter to an iterable and returns it in a form of enumerate object. This enumerate object can then be used directly in for loops or be converted into a list of tuples using list() method.

```
l1 = ["eat","sleep","repeat"]
s1 = "geek"

# creating enumerate objects
obj1 = enumerate(l1)
obj2 = enumerate(s1)

print "Return type:",type(obj1)
print list(enumerate(l1))

# changing start index to 2 from 0
print list(enumerate(s1,2))
```

```
Return type: < type 'enumerate' >
```

```
[(0, 'eat'), (1, 'sleep'), (2, 'repeat')]
[(2, 'g'), (3, 'e'), (4, 'e'), (5, 'k')]
```

****

next()

The next() returns the next item from the iterator.

```
mylist = iter(["apple", "banana", "cherry"])
x = next(mylist)
print(x)
x = next(mylist)
print(x)
x = next(mylist)
print(x)
```

```
C:\Users\My Name>python demo_next.py
apple
banana
cherry
```

---

open()

Open a file and print the content:

```
f = open("demofile.txt", "r")
print(f.read())
```

```
C:\Users\My Name>python demo_open.py
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

---

round()

Round a number to only two decimals:

```
x = round(5.76543, 2)
print(x)
```

```
C:\Users\My Name>python demo_round.py
5.77
```

Question 2: In the python pyserial package; ➢ Explain the purpose of the following functions: write() – open() – close() – read() – readline() – flushInput() – flushOutput() – inWaiting()

Write() ==➔ Write the bytes *data* to the port. This should be of type `bytes` (or compatible such as `bytearray` or `memoryview`). Unicode strings must be encoded (e.g. `'hello'.encode('utf-8'`).You want to send to Arduino turn on light for this define for example turn on light command but if you try to send this command directly from usb port communication will not fit each others because of that we have to convert to ascii form after send to Arduino command we have to encode () python code you can see what is that at question 1 so as a result external any microcontroller connected with usb it will understand what did we write.

open()

we are using for the this program for the serial communication with my computer to my Arduino or whatever we are using if we want to make this communication we should use ways in computer language these are port , if we want to star communication we should call open function and open port .

close()

As open function if we call the close function our port will close .

read()

We opened and closed our ports now if we choose read function after opened our port ,data will star to reading . So it helps to us read data on the port .

readline()

If you know that your input is always properly terminated with EOL characters, better way is to use **** The **End of Line** ("**EOL**") **character** (0x0D0A, \r\n) is actually two ASCII **characters** and is a combination of the CR and LF **characters**. It moves the cursor both down to the next line and to the beginning of that line.

flushInput()

Flush of file like objects. In this case, wait until all data is written. The function flush(stdin) is used to flush or clear the output buffer of the stream.

flushOutput()

fflush**()** is typically used for output stream only. Its purpose is to clear (or flush) the output buffer and move the buffered data to console (in case of stdout) or disk (in case of file output stream).

inWaiting()

Get the number of bytes in the input buffer

****

NOTE : FROM WEB SITE TAKEN SOME INFORMATIONS BECAUSE THE LIBRARY DICTIONARY IS THE USEFUL FOR WHO DOES NOT KNOW PYSERIAL LIBRARY *https://pyserial.readthedocs.io/en/latest/pyserial_api.html*

Question 3: And define the following parameters: port – baudrate – bytesize – parity

PORT ➔ on the computer we are using some parts for the data transfer ınput and output and these all operations will happen on the special data transfer ,this area on the electronic devices called PORT .

Baudrate ➔ this is very basic because in the serial communication it provide to understand to us data speed rate in change in second .

Bytesize ➔ it is single small amount for the data .

Parity ➔ when you connect to your computer and any hardware if they communicate each other that's parity even or they are not pair parity none .

Question 4: In the python pyvisa package; ➢ Explain the purpose of the following functions: *get_instruments_list() – instrument() – ask() – write() – trigger() – read_raw() - read() – clear() – query_values()*

get_instruments_list()

When I connect the my computer with osiloscope ,after call the get_instruments_list I saw that which device connected after python code show physical address ,so basicly it is show the our devices on the port .

instrument()

The class SerialInstrument defines the following additional properties. Note that all properties can also be given as keyword arguments when calling the class constructor or instrument()

ask()

Instead of separate write and read operations, you can do both with one ask() call.

– write()

After read data from data line we have to write this data and after show on the program so we have to write to this write function make calculate what we wish on the line ,this logic after give to us data from port device .

trigger()

Sends a software trigger to the device.

read_raw()

returns a string sent from the instrument to the computer. In contrast to read(), no termination characters are checked or stripped. You get the pristine message.

read()

reading strings from the device on the computer

clear()

resets the device.

query_values() we are asking did I connect to my device or not so on the port if device there is answer will come or not connect answer will no .

NOTE = PYVISA DATA DOCUMENT REFERENCES ARE USED .
https://buildmedia.readthedocs.org/media/pdf/pyvisa/1.5-docs/pyvisa.pdf

Question 5: And define what these messages mean: '*IDN?' – 'CURV?' – '*RST'

***IDN?**

I send the message "*IDN?" to the device, which is the standard GPIB message for "what are you?" or – in some cases – "what's on your display at the moment?". Basic mean we want to learn our device position ,is there or not ? sometimes our data devices might far away with just one click or code we can learn where it is , is it connected ,or are we taking data from our device ? or signal ?

'CURV?

If your oscilloscope (open in the variable `inst`) has been configured to transfer data in **ASCII** when the `CURV?` command is issued, you can just query the values like this:

```
>>> values = inst.query_ascii_values('CURV?')
```

`values` will be `list` containing the values from the device.

*****taken this information from** https://pyvisa.readthedocs.io/en/latest/introduction/rvalues.html

***RST**

we send the initialisation and reset message to the instrument. I took this centence from pyvisa reference so rst mean reseting before starting data taken .

**rst about information relative the reference pyvisa address
https://pyvisa.readthedocs.io/en/1.8/example.html?highlight=rst

Question 6 In the python struct package, what is the result of using unpack() function

Struct package is the conversation python values between C structs ,because python is the different language so in the programing bytes are different equalivent so this make conversation . unfack() function is the very usefull because imagine that you have a data and all data like this a,b,c,d,e….so you want to order them make clear the understable list so unfack function helps to make them

A
B
C
D
….. order like this ….

Question 7 In the python NumPy package, explain the purpose of the following functions: array() – arange() – linspace() – append() – indices() – shape() – reshape() – split() – delete()

array()

when we write the array it contains elements so array big part of the strings and elements for example we want to create matric we should create array after print this array we will see the as we wish .it can be think basic base of the library .

arange()

we want to start to create number list and count them , lets start from 1 and goes to 10 but we want to go to these steps 3 by 3 steps so if we call this arrange function we will get array [1,4,7] so basicly positional steps are shown at the computer screen .

linspace()

it is easy for example I want to draw the grap for temperature and I want to show my temperature values 0 degrees to 200 degrees but how …. So I need star stop and also between this interval how many points I need so … after call this function you can create nice number sequence .

append()

we are copying elements and after again write with order the axis array if loking like([1,2,3]……,[7,8,9]

this function will convert the shown like this array ([[1,2,3]

[4,5,6]

[7,8,9]])

indices()

Return an array representing the indices of a grid. This function I do not clear information This information taken from https://docs.scipy.org/doc/numpy/reference/generated/numpy.indices.html

shape()

Tuple of array dimensions.

The shape property is usually used to get the current shape of an array, but may also be used to reshape the array in-place by assigning a tuple of array dimensions to it. As with **numpy.reshape**, one of the new shape dimensions can be -1, in which case its value is inferred from the size of the array and the remaining dimensions. Reshaping an array in-place will fail if a copy is required.

This information taken from https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html

reshape()

Gives a new shape to an array without changing its data.

split()

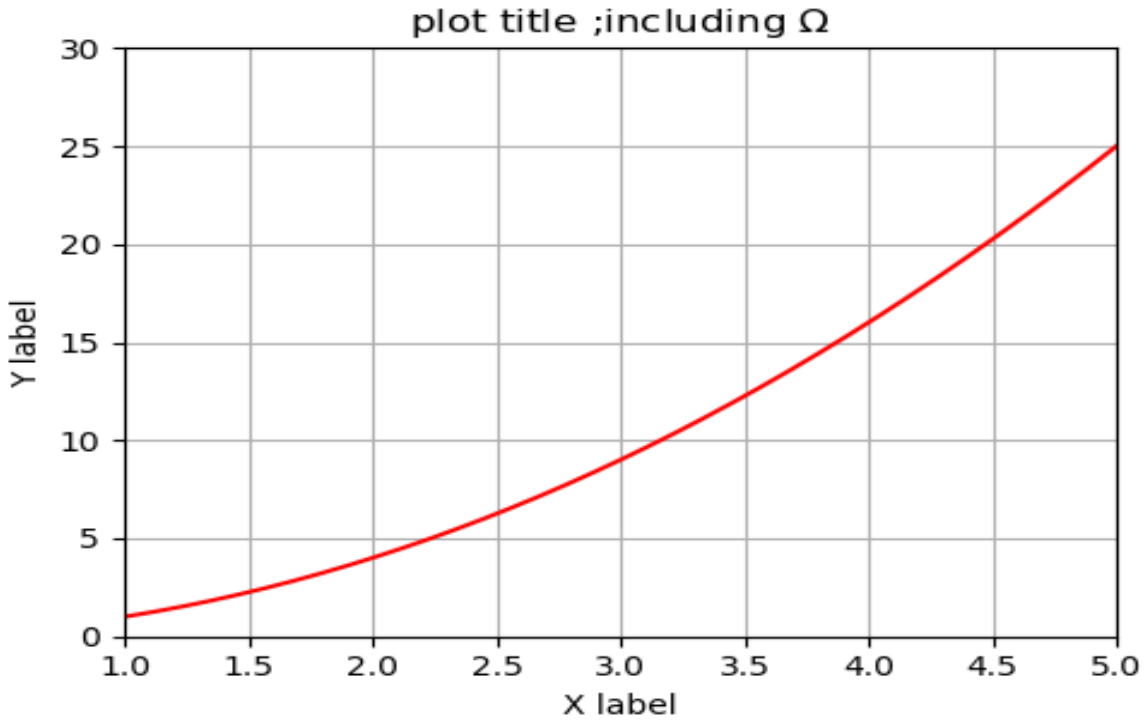
it is separating array multi arrays .

delete()

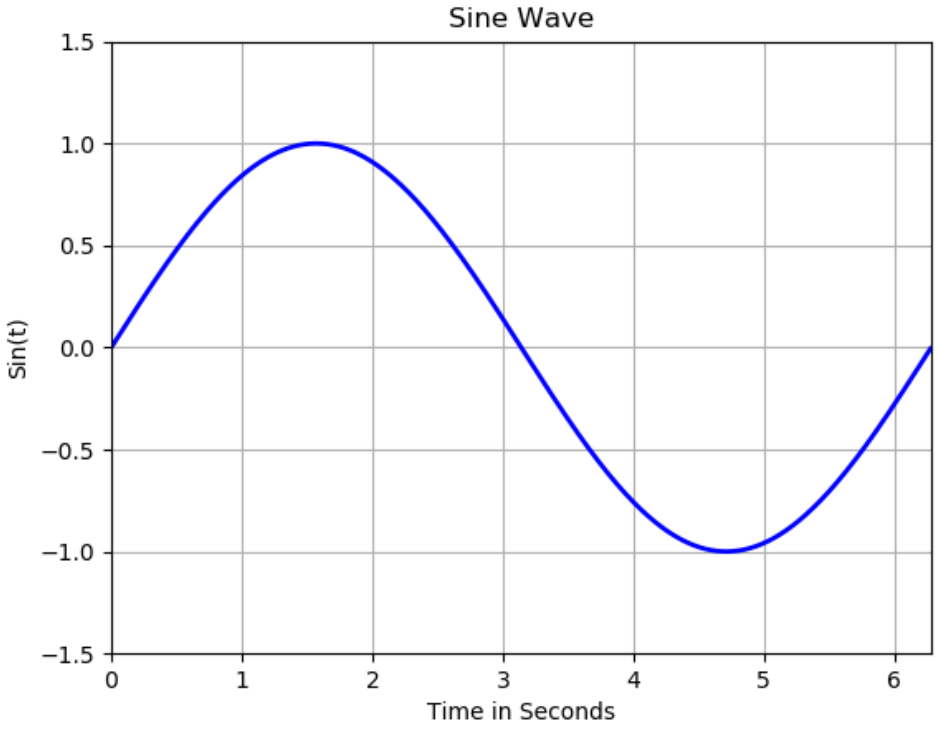
Using the NumPy function `np.delete()`, you can delete any row and column from the NumPy array `ndarray`.

This information is taken from https://note.nkmk.me/en/python-numpy-delete/

EXTRA PART FOR BONUS POINT



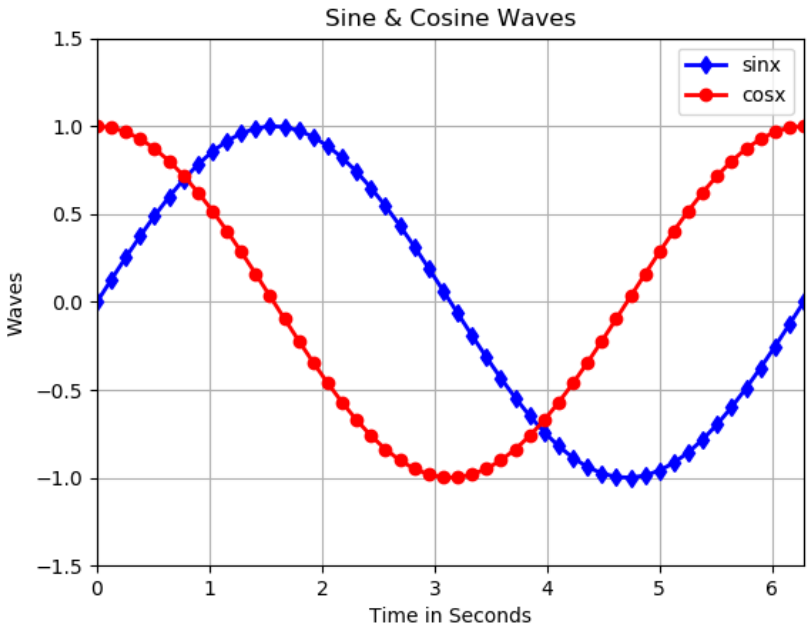

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
f, ax=plt.subplots(1,1, figsize=(5,4))
x= np.linspace(0,10,1000)
y=np.power(x,2)
ax.plot(x,y,color='r')
ax.set_xlim((1,5))
ax.set_ylim((0,30))
ax.set_xlabel('X label')
ax.set_ylabel('Y label')
ax.set_title('plot title ;including $\Omega$')
plt.tight_layout()
plt.savefig('linegraph.pdf')
x, y = np.meshgrid(x, y)
plt.grid()
plt.show()
```




```
import numpy as np  # numpy lib we called
import matplotlib.pyplot as plt # for plot we need this library
x=[] #we define x
y=[]  # we define y


for i in np.arange(0,2*np.pi,2*np.pi/1000):  #try to draw sin graph
    x.append(i) # i x values
    y.append(np.sin(i)) #when we put x values we will get y values


plt.plot(x,y,'b-' ,linewidth=2) # line describe
plt.grid(True) #grid for background
plt.axis([0,2*np.pi,-1.5,1.5]) #plot axis
plt.title('Sine Wave') #title of graph
plt.xlabel('Time in Seconds') #x axis name
plt.ylabel('Sin(t)')  #y axis name
plt.show() #shown graph not enough plot if you dont write this
```




```
import numpy as np  # numpy lib we called
import matplotlib.pyplot as plt # for plot we need this library
x=np.linspace(0,2*np.pi,50) #this is my array
y=np.sin(x) #depens on x  , my y array
z=np.cos(x) # my z array for depend x value
plt.plot(x,y,'b-d' ,linewidth=2,label='sinx') # line describe blue colar means b
d shape describe
plt.plot(x,z,'r-o' ,linewidth=2,label='cosx') # line describe red means r also
shape o describe
plt.grid(True) #grid for background
plt.axis([0,2*np.pi,-1.5,1.5]) #plot axis range
plt.title('Sine & Cosine Waves') #title of graph
plt.xlabel('Time in Seconds') #x axis name
plt.ylabel('Waves')  #y axis name
plt.legend() #show the legent
plt.show() #shown graph not enough plot if you dont write this
```

USB PORT LEARN CODE

```
import visa

rm=visa.ResourceManager(r'C:\Windows\System32\visa32.dll')
rm.list_resources()
print(rm.list_resources()[0])

myinst=rm.open_resource(rm.list_resource()[0])

print(myinst.query('*IDN?'))

mydevice=rm.get_instrument(rm.list_resources()[0])
mydevice.write("RST")


#    USB0::0x0699::0x0368::C027917::INSTR
```

PYTHON SERIAL PORT READING AND ARDUINO ANALOG DATA SEND TO PORT VALUE

```
#UPLOUD THE CODE  PYTHON READ IN IDE

import serial


ser = serial.Serial('COM4', baudrate = 9600, timeout=1)


while 1:


   arduinoData = ser.readline().decode('ascii')

   print(arduinoData)
```

```
 //ARDUINO  CODE UPLOUD TO ARDUDIO
int analogPin = 0;
int data = 0;
char userInput;

void setup(){

  Serial.begin(9600);         //  setup serial
}

void loop(){


     data = analogRead(analogPin);    // read the input pin
     Serial.println(data);
     delay(1);

} // Void Loop
```

OSILOSCOPE DATA TAKEN FROM CHANNEL 2 OR CHANNEL 1 JUST ONE CHANNEL

```
import visa
import numpy as np
import pylab

from struct import unpack

rm = visa.ResourceManager()
print(rm.list_resources())
inst=rm.open_resource('USB0::0x0699::0x0368::C027917::INSTR')
print(inst.query("*IDN?"))


inst.write('DATA:SOU CH2')
inst.write('DATA:WIDTH 1')
inst.write('DATA:ENC RPB')

ymult = float(inst.ask('WFMPRE:YMULT?'))
yzero = float(inst.ask('WFMPRE:YZERO?'))
yoff = float(inst.ask('WFMPRE:YOFF?'))
xincr = float(inst.ask('WFMPRE:XINCR?'))


inst.write('CURVE?')
data =inst.read_raw()
headerlen= 2 + int(data[1])
header=data[:headerlen]
ADC_wave= data[headerlen:-1]

ADC_wave =np.array(unpack('%sB' % len(ADC_wave),ADC_wave))
Volts =(ADC_wave - yoff) * ymult +yzero
Time= np.arrange(0,xincr * len(Volts),xincr)

pylab.plot(Time,Volts)
```

Arduino LED ATTACHED CODE

```
const int ledPin = 13; // the pin that the LED is attached to
int incomingByte;      // a variable to read incoming serial data into

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // see if there's incoming serial data:
  if (Serial.available() > 0) {
    // read the oldest byte in the serial buffer:
    incomingByte = Serial.read();
    // if it's a capital H (ASCII 72), turn on the LED:
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    // if it's an L (ASCII 76) turn off the LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

PYTHON FOR THE LED CONTROL FROM THE PORT COMUNICATION SERIAL

```
import serial
 import time

 ser = serial.Serial('COM4', 9600)  # open serial port With same bound rate arduino
 time.sleep(2)                  # wait 10 seconds
ser.name()
'COM4'

ser.write(b'H')
# LED turns on

 ser.write(b'L')
# LED turns off

ser.write(b'H')
# LED turns on

 ser.write(b'L')
# LED turns off

 ser.close()
 exit()
```

References

“Numpy.delete(): Delete Rows and Columns.” numpy.delete(): Delete rows and columns. Accessed October 20, 2019. https://note.nkmk.me/en/python-numpy-delete/.

“Numpy.ndarray.shape¶.” numpy.ndarray.shape - NumPy v1.17 Manual. Accessed October 20, 2019. https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html.

“Numpy.indices¶.” numpy.indices - NumPy v1.17 Manual. Accessed October 20, 2019. https://docs.scipy.org/doc/numpy/reference/generated/numpy.indices.html.

“A More Complex Example¶.” A more complex example - PyVISA 1.8 documentation. Accessed October 20, 2019. https://pyvisa.readthedocs.io/en/1.8/example.html?highlight=rst.

“Reading and Writing Values¶.” Reading and Writing values - PyVISA 1.11.0.dev0 documentation. Accessed October 20, 2019. https://pyvisa.readthedocs.io/en/latest/introduction/rvalues.html.

“Control Your Instruments with Python¶.” PyVISA. Accessed October 20, 2019. https://pyvisa.readthedocs.io/en/latest/.

“PySerial API¶.” pySerial API - pySerial 3.4 documentation. Accessed October 20, 2019. https://pyserial.readthedocs.io/en/latest/pyserial_api.html.