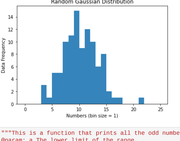
In [2]:
```
#A01733702 Nezih Nieto Gutiérrez
#This cell was reserved for imports only and it must be run before any other cell
from contextlib import nullcontext
import random
from tkinter.font import names
import numpy as np
import random
from matplotlib import pyplot as plt
import pandas as pd
```

In [106]:
```
"""This function generates a random list of a normal distribution, and shows the subsequent plotted hi
@param: size The size of the array
@param: standDev The standard deviation of the normal distribution
@param: mean The expected value of the normal distribution
@param: binsize The size of the bins showed in the histogram
"""
def randomGaussianNumbers(size,standDev,mean,binsize):
    nums = []
    for i in range(0,size):
        nums.append(random.normalvariate(mean,standDev))
    bins = np.arange(-100, 100, binsize)
    plt.xlim([min(nums)-5, max(nums)+5])
    plt.hist(nums, bins=bins, alpha=0.9)
    plt.title('Random Gaussian Distribution')
    plt.xlabel('Numbers (bin size = '+str(binsize)+')')
    plt.ylabel('Data Frequency [Units]')
    plt.show()

randomGaussianNumbers(100,3,10,1)
```



In [46]:
```
"""This is a function that prints all the odd numbers in a range, with two different sorts of loop.
@param: a The lower limit of the range.
@param: b The upper limit of the range.
"""
def oddHorizontalNumbers(a,b):
    if(a%2==0):
        nums = range(a+1,b,2)
    else:
        nums = range(a,b,2)

    for i in nums:
        print(str(i), end="\t")

    print()
    j=0
    while j < len(nums):
        print(str(nums[j]), end="\t")
        j+j+1

oddHorizontalNumbers(0,30)
```

```
1	3	5	7	9	11	13	15	17	19	21	23	25	
27	29	
1	3	5	7	9	11	13	15	17	19	21	23	25	
27	29
```

In [105]:
```
"""This is a function that performs a basic data analysis of the
@param: URL The address from where the Data Set must be downloaded.
"""
def irisDataSet(URL):
    col_names = ['Sepal_Length','Sepal_Width', 'Petal_Length','Petal_Width','Class']
    DataSet =  pd.DataFrame(pd.read_csv(URL, names = col_names))
    print('Total Registers:', DataSet.shape[0])
    freq = DataSet.groupby(['Class']).count()
    print('Total number of classes ',str(len(freq)))
    slStats=[DataSet['Sepal_Length'].mean(), DataSet['Sepal_Length'].median(), DataSet['Sepal_Leng
    plStats=[DataSet['Petal_Length'].mean(), DataSet['Petal_Length'].median(),DataSet['Petal_Length
    stats = {'SL': slStats, 'PL': plStats, 'Statistic Measures': ['Mean', 'Median', 'Standard Devia
    Stats = pd.DataFrame(stats)
    Stats
    fig = plt.figure(figsize = (15,15), )
    ax = fig.gca()
    DataSet.hist(bins=20, ax=ax)
    print(DataSet.describe())
    return DataSet

DataSet=irisDataSet('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data')
DataSet
```

```
Total Registers: 150
Total number of classes 3
```

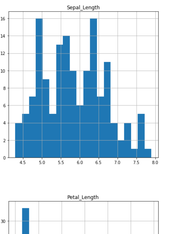
```
/tmp/ipykernel_705/361000008.py:14: UserWarning: To output multiple subplots, the figure containing t
he passed axes is being cleared.
  DataSet.hist(bins=20, ax=ax)
```

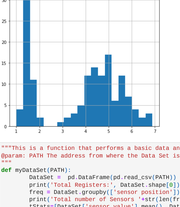
```
       Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

Out[105]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns





In [103]:
```
"""This is a function that performs a basic data analysis of a EEG of a control subject.
@param: PATH The address from where the Data Set is extracted.
"""
def myDataSet(PATH):
    DataSet =  pd.DataFrame(pd.read_csv(PATH))
    print('Total Registers:', DataSet.shape[0])
    freq = DataSet.groupby(['sensor position']).count()
    print('Total number of Sensors ',str(len(freq)))#Nos da el número de Electrodos Utilizados en
    tStats=[DataSet['sensor value'].mean(), DataSet['sensor value'].median(), DataSet['sensor valu
    stats = {'sensor value': tStats, 'Statistic Measures': ['Mean', 'Median', 'Standard Deviation'
    Stats = pd.DataFrame(stats)
    print(Stats)
    print()
    fig = plt.figure(figsize = (15,20), )
    ax = fig.gca()
    DataSet.hist(bins=10, ax=ax)
    print(DataSet.describe())
    return DataSet

DataSet = myDataSet('Data1.csv')#EEG Data Set downloaded from Kaggle (https://www.kaggle.com/zhixx018/
```

```
Total Registers: 16384
Total number of Sensors 64
   sensor value  Statistic Measures
0   1.989997                  Mean
1   1.221000                Median
2   7.505248  Standard Deviation
```

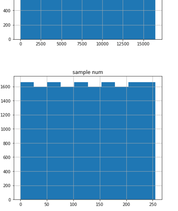
```
/tmp/ipykernel_705/593676396.py:13: UserWarning: To output multiple subplots, the figure containing t
he passed axes is being cleared.
  DataSet.hist(bins=10, ax=ax)
```

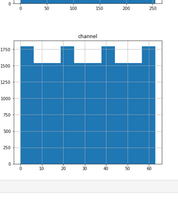
```
       Unnamed: 0  trial number    sample num  sensor value        channel  \
count   16384.000000   16384.0  16384.000000  16384.000000  16384.000000   
mean     8228.000000       0.0    127.500000      1.989997     31.500000   
std      4748.269011       0.0     73.902526      7.505248     18.473517   
min         0.000000       0.0      0.000000    -39.825000      0.000000   
25%      4116.500000       0.0     63.750000     -2.228000     15.750000   
50%      8228.000000       0.0    127.500000      1.221000     31.500000   
75%     12339.500000       0.0    191.250000      5.402000     47.250000   
max     16451.000000       0.0    255.000000     51.900000     63.000000   

                time  
count   16384.000000  
mean        0.498047  
std         0.288682  
min         0.000000  
25%         0.249023  
50%         0.498047  
75%         0.747070  
max         0.996094  
```

Out[103]:

	Unnamed: 0	trial number	sensor position	sample num	sensor value	subject identifier	matching condition	channel	name	time
0	5	0	FP1	0	-8.921	a	S1 obj	0	co2a0000364	0.000000
1	6	0	FP1	1	-8.433	a	S1 obj	0	co2a0000364	0.003906
2	7	0	FP1	2	-2.574	a	S1 obj	0	co2a0000364	0.007812
3	8	0	FP1	3	5.239	a	S1 obj	0	co2a0000364	0.011719
4	9	0	FP1	4	11.587	a	S1 obj	0	co2a0000364	0.015625
...	...	...	...	...	...	...	...	...	...	...
16379	16447	0	Y	251	6.571	a	S1 obj	63	co2a0000364	0.980469
16380	16448	0	Y	252	12.431	a	S1 obj	63	co2a0000364	0.984375
16381	16449	0	Y	253	15.849	a	S1 obj	63	co2a0000364	0.988281
16382	16450	0	Y	254	16.337	a	S1 obj	63	co2a0000364	0.992188
16383	16451	0	Y	255	14.872	a	S1 obj	63	co2a0000364	0.996094

16384 rows × 10 columns





In [ ]:
```

```