```
In [67]: import pandas as pd
         import numpy as np
         import seaborn as sns
```

```
In [68]: df=pd.read_csv('Iris.csv')
```

```
In [84]: ##remember this
         datagrp=df.groupby(df['Species'])
         df['Species'].unique()
```

```
Out[84]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
In [85]: setosa=datagrp.get_group('Iris-setosa')
```

```
In [87]: setosa.drop('Species',axis=1)
```

Out[87]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
5	6	5.4	3.9	1.7	0.4
6	7	4.6	3.4	1.4	0.3
7	8	5.0	3.4	1.5	0.2
8	9	4.4	2.9	1.4	0.2
9	10	4.9	3.1	1.5	0.1
10	11	5.4	3.7	1.5	0.2
11	12	4.8	3.4	1.6	0.2
12	13	4.8	3.0	1.4	0.1
13	14	4.3	3.0	1.1	0.1
14	15	5.8	4.0	1.2	0.2
15	16	5.7	4.4	1.5	0.4
16	17	5.4	3.9	1.3	0.4
17	18	5.1	3.5	1.4	0.3
18	19	5.7	3.8	1.7	0.3
19	20	5.1	3.8	1.5	0.3
20	21	5.4	3.4	1.7	0.2
21	22	5.1	3.7	1.5	0.4
22	23	4.6	3.6	1.0	0.2
23	24	5.1	3.3	1.7	0.5
24	25	4.8	3.4	1.9	0.2
25	26	5.0	3.0	1.6	0.2
26	27	5.0	3.4	1.6	0.4
27	28	5.2	3.5	1.5	0.2
28	29	5.2	3.4	1.4	0.2
29	30	4.7	3.2	1.6	0.2
30	31	4.8	3.1	1.6	0.2
31	32	5.4	3.4	1.5	0.4
32	33	5.2	4.1	1.5	0.1
33	34	5.5	4.2	1.4	0.2
34	35	4.9	3.1	1.5	0.1
35	36	5.0	3.2	1.2	0.2

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
36	37	5.5	3.5	1.3	0.2
37	38	4.9	3.1	1.5	0.1
38	39	4.4	3.0	1.3	0.2
39	40	5.1	3.4	1.5	0.2
40	41	5.0	3.5	1.3	0.3
41	42	4.5	2.3	1.3	0.3
42	43	4.4	3.2	1.3	0.2
43	44	5.0	3.5	1.6	0.6
44	45	5.1	3.8	1.9	0.4
45	46	4.8	3.0	1.4	0.3
46	47	5.1	3.8	1.6	0.2
47	48	4.6	3.2	1.4	0.2
48	49	5.3	3.7	1.5	0.2
49	50	5.0	3.3	1.4	0.2

In [88]:
```
import math
```

In [106…
```
def show(series):
    summ=0
    count=0
    mean=0
    stdsum=0
    variance=0
    std=0
    for i in series:
        summ+=i
        count+=1
    mean=summ/count
    print("Mean: ",mean)
    for i in series:
        dist=(i-mean)*(i-mean)
        stdsum+=dist
    variance=stdsum/count
    print("Variance: ",variance)
    std=math.sqrt(variance)
    print("Standard deviation: ",std)
    q1=np.percentile(series,0.25)
    print("Q1: ",q1)
    print("\n")
```

In [107…
```
setosa.apply(show,0)
```

```
Mean:  25.5
Variance:  208.25
Standard deviation:  14.430869689661812
Q1:  1.1225


Mean:  5.005999999999999
Variance:  0.12176399999999993
Standard deviation:  0.348946987377739
Q1:  4.31225


Mean:  3.4180000000000006
Variance:  0.142276
Standard deviation:  0.37719490982779713
Q1:  2.3735


Mean:  1.464
Variance:  0.029504000000000001
Standard deviation:  0.17176728442867115
Q1:  1.01225


Mean:  0.2439999999999999
Variance:  0.011263999999999996
Standard deviation:  0.10613199329137278
Q1:  0.1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [107], in <cell line: 1>()
----> 1 setosa.apply(show,0)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:8839, in DataFrame.apply(s
elf, func, axis, raw, result_type, args, **kwargs)
   8828 from pandas.core.apply import frame_apply
   8830 op = frame_apply(
   8831     self,
   8832     func=func,
   (...)
   8837     kwargs=kwargs,
   8838 )
-> 8839 return op.apply().__finalize__(self, method="apply")

File ~\anaconda3\lib\site-packages\pandas\core\apply.py:727, in FrameApply.apply(s
elf)
    724 elif self.raw:
    725     return self.apply_raw()
--> 727 return self.apply_standard()

File ~\anaconda3\lib\site-packages\pandas\core\apply.py:851, in FrameApply.apply_s
tandard(self)
    850 def apply_standard(self):
--> 851     results, res_index = self.apply_series_generator()
    853     # wrap results
    854     return self.wrap_results(results, res_index)

File ~\anaconda3\lib\site-packages\pandas\core\apply.py:867, in FrameApply.apply_s
eries_generator(self)
    864 with option_context("mode.chained_assignment", None):
    865     for i, v in enumerate(series_gen):
    866         # ignore SettingWithCopy here in case the user mutates
--> 867         results[i] = self.f(v)
    868         if isinstance(results[i], ABCSeries):
    869             # If we have a view on v, we need to make a copy because
    870             #  series_generator will swap out the underlying data
    871             results[i] = results[i].copy(deep=False)

Input In [106], in show(series)
      7 std=0
      8 for i in series:
----> 9     summ+=i
     10     count+=1
     11 mean=summ/count

TypeError: unsupported operand type(s) for +=: 'int' and 'str'
```

```
In [ ]:
```