Report

```
[41] import numpy as np
     import matplotlib.pyplot as plt
     import random
```

Importing required libraries

```
temp=[[] for x in range(80)]
sd = []
for i in range(80):
    mu = i/2
    sd_ = random.uniform(0.2,0.4)
    sd.append(sd_)
    temp[i] = np.random.normal(mu,sd_,pow(10,4))
```

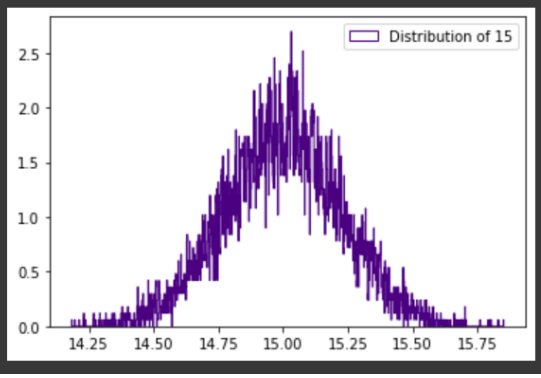
Generating normal distributions of error for temperatures in the range of 0-40 with 0.5 interval

Stores all measured temperature values (10^4 values for each marking)

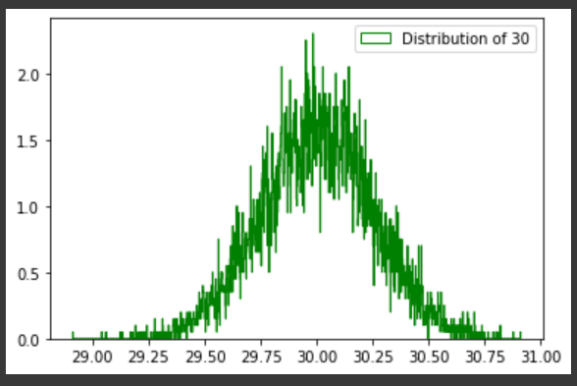
sd = [] #stores standard deviation of all temperature's error distributions

Illustration of some graphs for different temperatures:

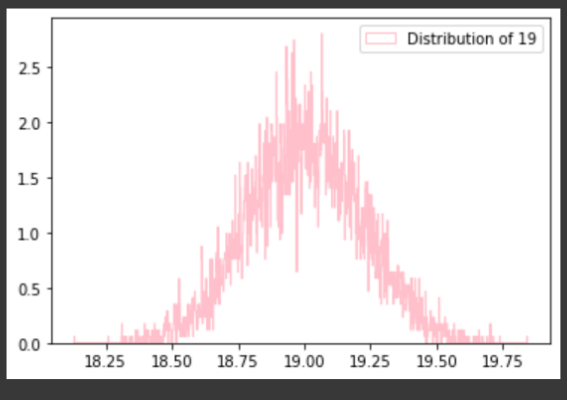
```
plt.hist(temp[30],bins=1000,density=True,label="Distribution of 15",histtype="step",color="indigo")
plt.legend()
plt.show()
```



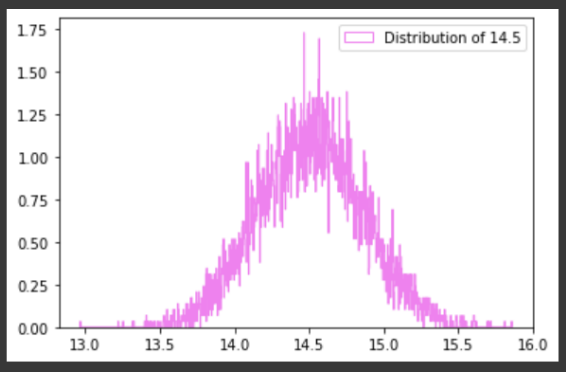
```
[30] plt.hist(temp[60],bins=1000,density=True,label="Distribution of 30",histtype="step",color="green")
     plt.legend()
     plt.show()
```



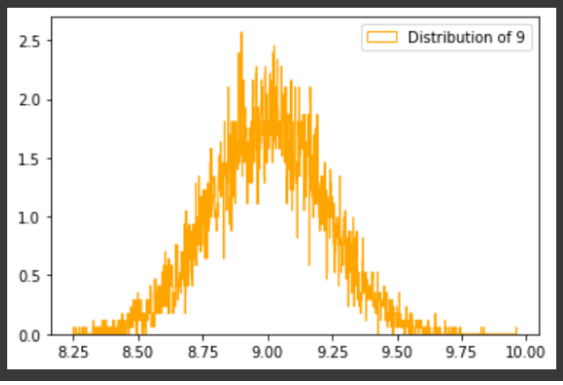

```
[31] plt.hist(temp[38],bins=1000,density=True,label="Distribution of 19",histtype="step",color="pink")
     plt.legend()
     plt.show()
```




```
[32] plt.hist(temp[29],bins=1000,density=True,label="Distribution of 14.5",histtype="step",color="violet")
     plt.legend()
     plt.show()
```



```
[36] plt.hist(temp[18],bins=1000,density=True,label="Distribution of 9",histtype="step",color="orange")
     plt.legend()
     plt.show()
```




```
[67] prob_error = []
     for i in range(80):
         count = 0
         for j in range(pow(10,4)):
             if ((temp[i][j] > (i/2+0.3)) or (temp[i][j] < (i/2-0.3))):
                 count = count + 1
         prob_error.append(count/pow(10,4))
```

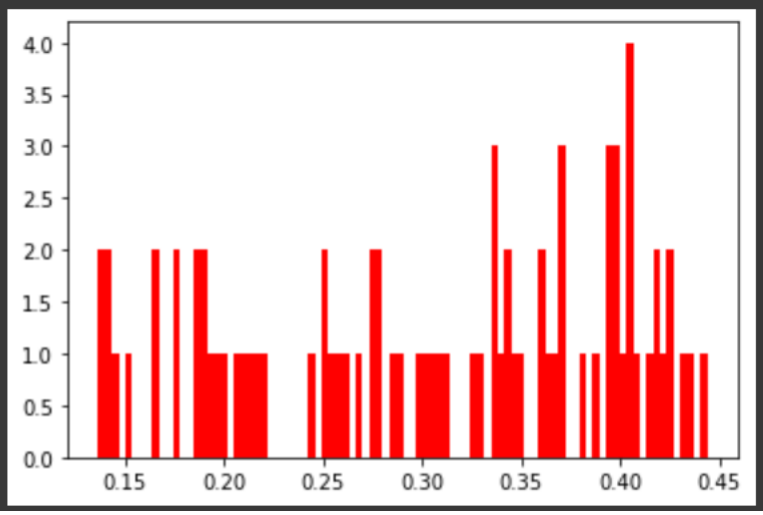
#probability of error prob_error

#print(prob_error)

print (prob_error [30], prob_error [60], prob_error [38], prob_error [29], prob_error [18], max(prob_error), min (prob_error), sd [prob_error.index (min(prob_error))])

Calculating probability of value being outside accepted range (+-0.3)

```
[68] plt.hist(prob_error,bins=90,color="red")
     plt.show()
```



Plotting the probability of errors being greater than accepted range.

```
prob=[0 for x in range(80)]
cost=[0 for x in range(80)]
for i in range(80):
    if i>=30 and i<=50:
        prob[i] = random.uniform(14,20)
    elif i<5 or i>70:
        prob[i] = random.uniform(1,4)
    else:
        prob[i] = random.uniform(4,14)

sum = 0
for i in range(80):
    sum = sum + prob[i]
for i in range(80):
    prob[i] = prob[i]/sum

for i in range(80):
    if i>=30 and i<=50:
        cost[i] = random.uniform(14,20)
    elif i<5 or i>70:
        cost[i] = random.uniform(1,4)
    else:
        cost[i] = random.uniform(4,14)
```

prob=[0 for x in range(80)] # Stores the probability of a true temperature occurring

cost=[0 for x in range(80)] # Stores the cost of error for each temperature

for i in range(80):

  if i>=30 and i<=50:

    prob[i] = random.uniform(14,20) # More probability is given to temperatures in the middle of the range, decreases gradually

  elif i<5 or i>70:

    prob[i] = random.uniform(1,4)

  else:

    prob[i] = random.uniform(4,14)

sum = 0

for i in range(80):

  sum = sum + prob[i]

for i in range(80): #Normalizing the probabilities

  prob[i] = prob[i]/sum

for i in range(80):

```
    if i>=30 and i<=50:
        cost[i] = random.uniform(14,20)     # More probability is given to temperatures in the middle of t
he range, decreases gradually
    elif i<5 or i>70:
        cost[i] = random.uniform(1,4)
    else:
        cost[i] = random.uniform(4,14)
# print(prob)
# print(cost)
```

```
[70] prob_e = 0
     for i in range(71):
         prob_e = prob_e + prob_error[i]*prob[i]
     print("The probability of an error happening is: ")
     print(prob_e)
     exp_cost = 0
     for i in range(71):
         exp_cost = exp_cost + prob_error[i]*prob[i]*cost[i]
     print("The expected cost is: ")
     print(exp_cost)

     The probability of an error happening is:
     0.3052054015608932
     The expected cost is:
     3.8662163354618406
```

Calculating the required probabilities and cost