BPC Problem Set 1

Secret Code: 20

9/5/2020

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

Problem A1

In [2]:
```python
a = [1, np.pi, 0.37, np.power(np.e,6)]
print(a)
```

```
[1, 3.141592653589793, 0.37, 403.428793492735]
```

Problem A2

In [3]:
```python
type(a)
```

Out[3]:
```
list
```

Problem A3

In [4]:
```python
b = [x+2 for x in a]
print(b)
```

```
[3, 5.141592653589793, 2.37, 405.428793492735]
```

Problem A4

In [5]:
```python
c = [x/2 for x in b]
print(c)
```

```
[1.5, 2.5707963267948966, 1.185, 202.7143967463675]
```

Problem A5

In [6]:
```python
_1_to_37_step_1 = np.arange(1, 38, 1)
print(_1_to_37_step_1)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37]
```

Problem A6

In [7]:
```python
d = np.arange(0, 6.1, 0.1)
print(d)
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6. ]
```

Problem A7

In [8]:
```python
len(d)
```

Out[8]:
```
61
```

Problem A8

In [9]:
```python
myprimes = []
count = 2
while len(myprimes) < 20:
    flag = True
    for i in range(count):
        if ((i+1) != 1) and ((i+1) != count) and (count / (i+1) == count // (
            flag = False

    if flag :
        myprimes.append(count)
        count += 1
    else :
        count += 1

print(myprimes)
```

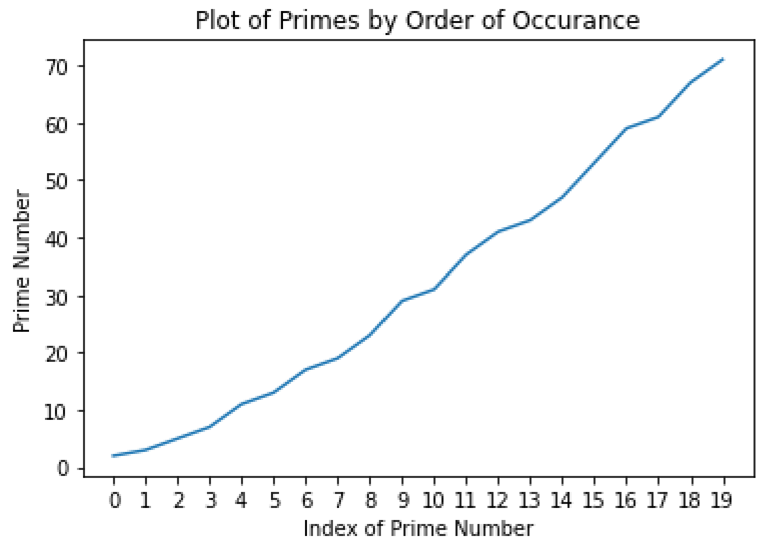
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 7
1]
```

Problem A9

Plot of Primes By Order They Occur

In [10]:

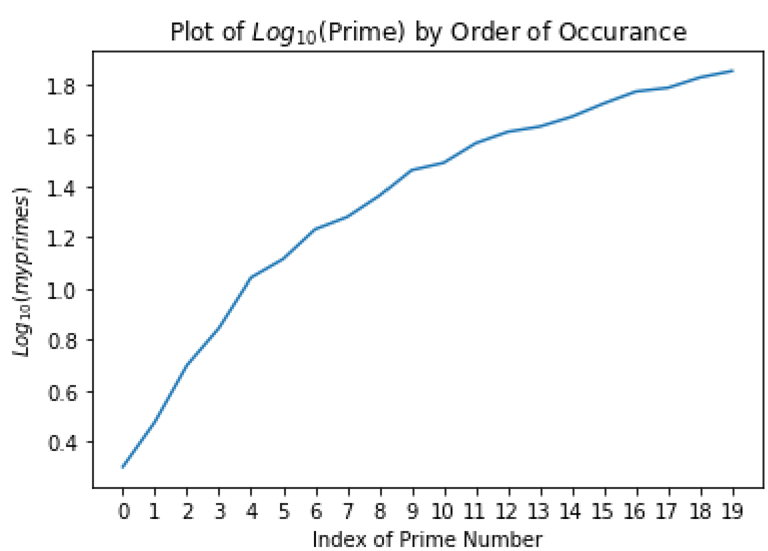
```python
plt.plot(myprimes)
plt.xticks(np.arange(0, 20, 1))
plt.ylabel("Prime Number")
plt.xlabel("Index of Prime Number")
plt.title("Plot of Primes by Order of Occurance")
print("", end="")
```



Plot of Log base 10 Primes By Order They Occur

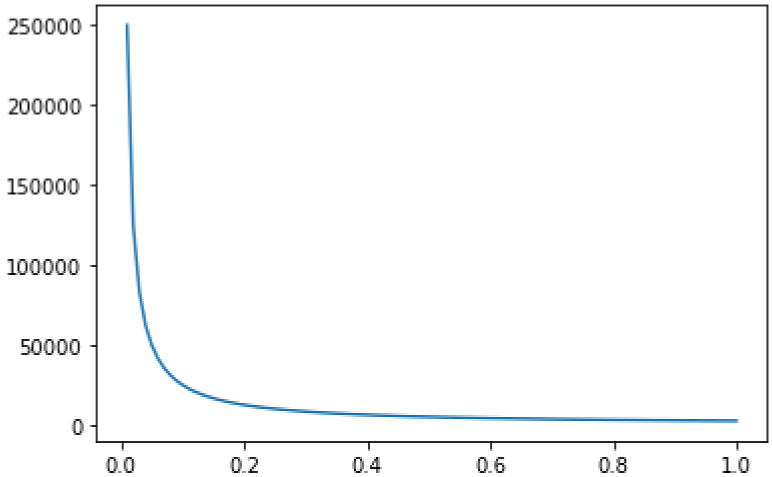
In [11]:

```python
logprimes = [np.log10(x) for x in myprimes]
plt.plot(logprimes)
plt.xticks(np.arange(0, 20, 1))
plt.ylabel("$Log_{10}(myprimes)$")
plt.xlabel("Index of Prime Number")
plt.title("Plot of $Log_{10}$(Prime) by Order of Occurance")
print("", end="")
```



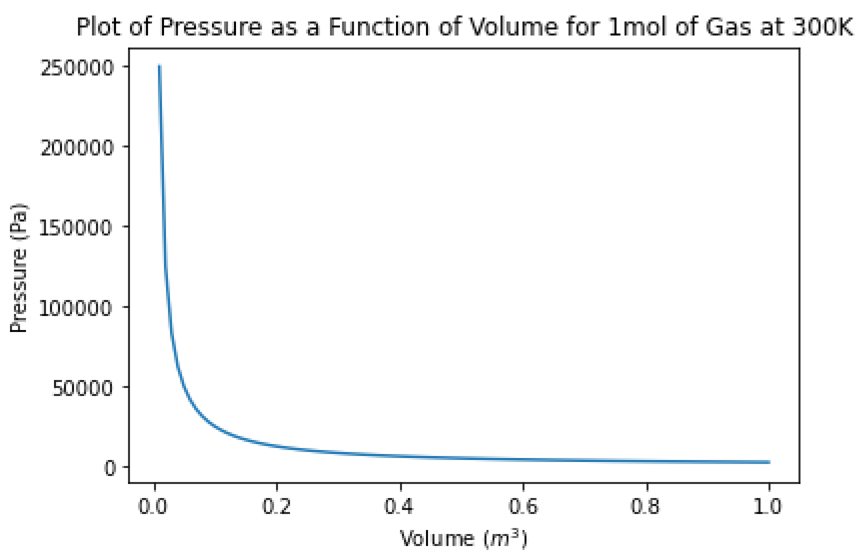
Problem A10

In [12]:
```
T = 300 #Temperature in K
n = 1 #number of moles of gas
R = 8.3145 #ideal gas constant in m3 Pa K-1 mol-1
V = np.arange(0.01, 1.01, 0.01) #Volume range in m3
P = [(n*R*T)/x for x in V] #Pressure in Pascals
plt.plot(V, P)
print("", end="")
```



Problem A11

In [13]:
```
plt.plot(V, P)
plt.title("Plot of Pressure as a Function of Volume for 1mol of Gas at 300K")
plt.ylabel("Pressure (Pa)")
plt.xlabel("Volume ($m^3$)")
print("", end="")
```



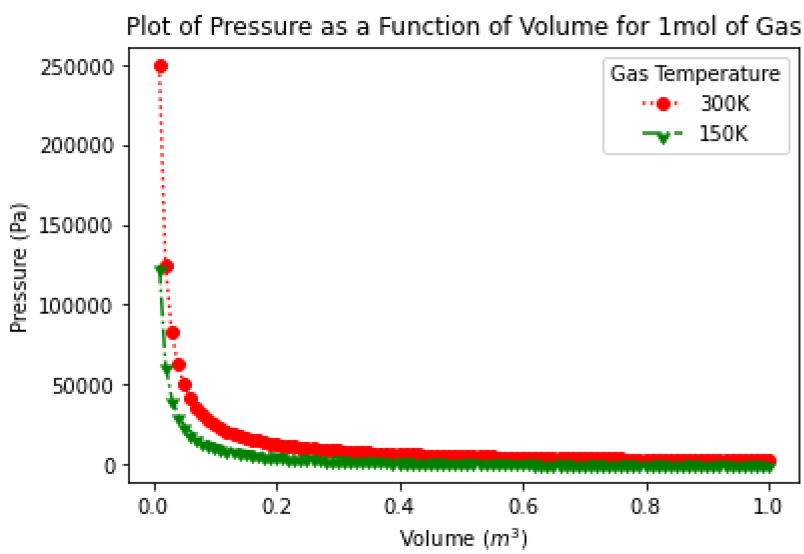
Problem A12

In [14]:
```
T = 300 #Temperature in K
n = 1 #number of moles of gas
R = 8.3145 #ideal gas constant in m3 Pa K-1 mol-1
V = np.arange(0.01, 1.01, 0.01) #Volume range in m3
P = [(n*R*T)/x for x in V] #Pressure in Pascals

T2 = 150 #Temperature for second plot in K
P2 = [(n*R*T2)/x for x in V] #Pressure range in Pascals at second Temperature

plot1, = plt.plot(V, P, color = "red", linestyle = ':', marker="o")
plot2, = plt.plot(V, P2, color = "green", linestyle='-.', marker=11)

plt.title("Plot of Pressure as a Function of Volume for 1mol of Gas")
plt.ylabel("Pressure (Pa)")
plt.xlabel("Volume ($m^3$)")
plt.legend([plot1, plot2], ["300K", "150K"], title="Gas Temperature")
print("", end="")
```



Problem B2

Flip Ten Coins Once

In [15]:
```
np.random.seed(18218)

def Flip_Ten():
    ten_flips = np.random.randint(0,2,10)
    Nh = np.sum(ten_flips)
    return Nh


print("10 Coins Flipped Once Returns - Nh: {}\n".format(Flip_Ten()))
```

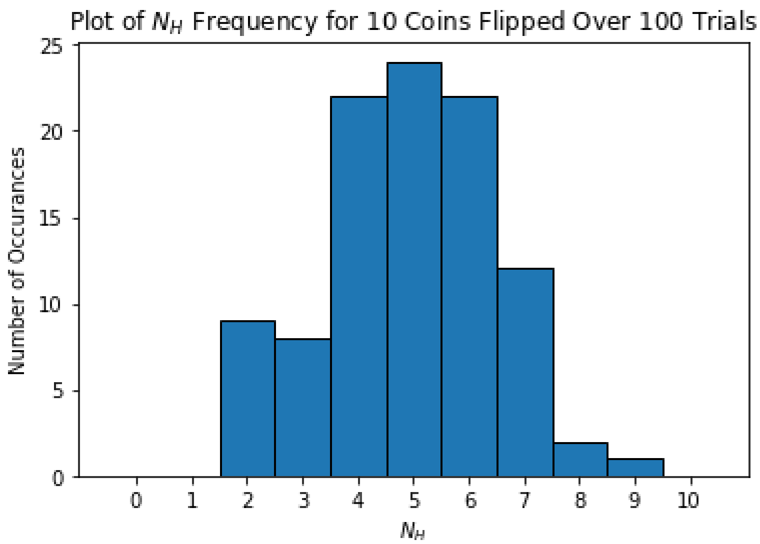
```
10 Coins Flipped Once Returns - Nh: 4
```

Flip Ten Coins 100 Times

In [16]:

```
np.random.seed(18218)
count_array = []
for x in range(100):
    count_array.append(Flip_Ten())

bins = np.arange(0, 12, 1)
plt.hist(count_array, bins = bins, align="left", ec="black")
plt.xticks(bins[:-1])
plt.ylabel("Number of Occurances")
plt.xlabel("$N_H$")
plt.title("Plot of $N_H$ Frequency for 10 Coins Flipped Over 100 Trials")
print("", end="")
```



Flip 1,000 Coins 10,000 Times

In [17]:

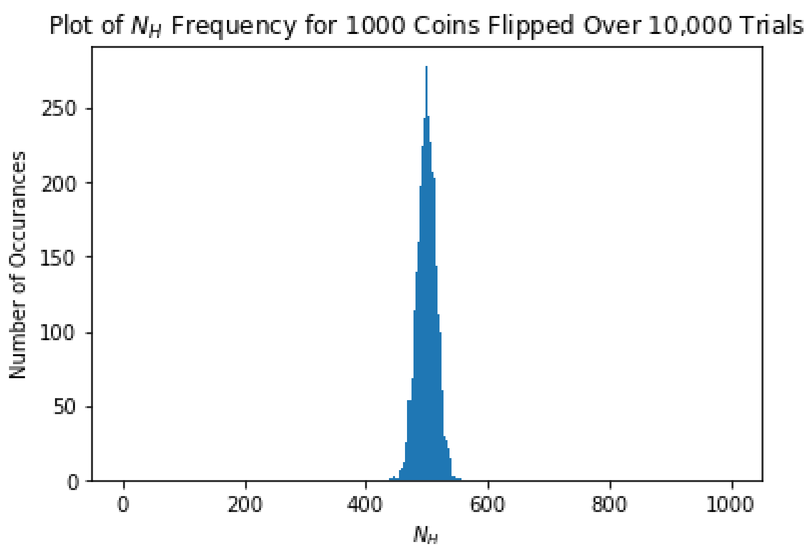
```
np.random.seed(18218)

def Flip_Thousand():
    thousand_flips = np.random.randint(0,2,1000)
    Nh = np.sum(thousand_flips)
    return Nh

thousand_count_array = []
for x in range(10000):
    thousand_count_array.append(Flip_Thousand())

new_bins = np.arange(0, 1002, 1)

plt.hist(thousand_count_array, bins = new_bins, align="left")
plt.ylabel("Number of Occurances")
plt.xlabel("$N_H$")
plt.title("Plot of $N_H$ Frequency for 1000 Coins Flipped Over 10,000 Trials"
print("", end="")
```



Compare Standard Deviations

According to textbook, standard deviations for binomial distribution = sqrt(Ntot * Pa * Pb) as we see below, the standard deviation increases with the sqrt of Ntot. The predicted value of std. dev for the 10-flip trial is sqrt(10*0.5*0.5) = 1.5811 and for the 1000-flip trials sqrt(1000*0.5*0.5) = 15.811. The approximation values below improve with more trials as the sampling approaches a perfect binomial distribution!

In [18]:

```
std_dev_10 = np.std(count_array)
std_dev_1000 = np.std(thousand_count_array)
print("Standard Deviation for 10-Flip Trials: {}".format(std_dev_10))
print("Standard Deviation for 1000-Flip Trials: {}".format(std_dev_1000))
```

```
Standard Deviation for 10-Flip Trials: 1.5433405327405874
Standard Deviation for 1000-Flip Trials: 15.872882712034384
```

Compare Sharpness of Distributions

As expected, the 1000 flip trial is much sharper than the 10 flip trials as it is less likely for outliers to make a significant impact on the spread of the data over so many more trials.

In [19]:
```
sharpness_10 = std_dev_10 / 10
sharpness_1000 = std_dev_1000 / 1000

print("Sharpness of 10-Flip Trials: {}".format(sharpness_10))
print("Sharpness of 1000-Flip Trials: {}".format(sharpness_1000))
```

```
Sharpness of 10-Flip Trials: 0.15433405327405875
Sharpness of 1000-Flip Trials: 0.015872882712034384
```

Problem B3

To fit the binomial distributions over the sample data we need to multiply its values by Ntot in order to convert the probability values for each point in x converted to Nh values as P(i;N) * Ntot = Nh,i(expected) for each point along the distribution.

Plot Binomial Distribution For 10-Flip Trials

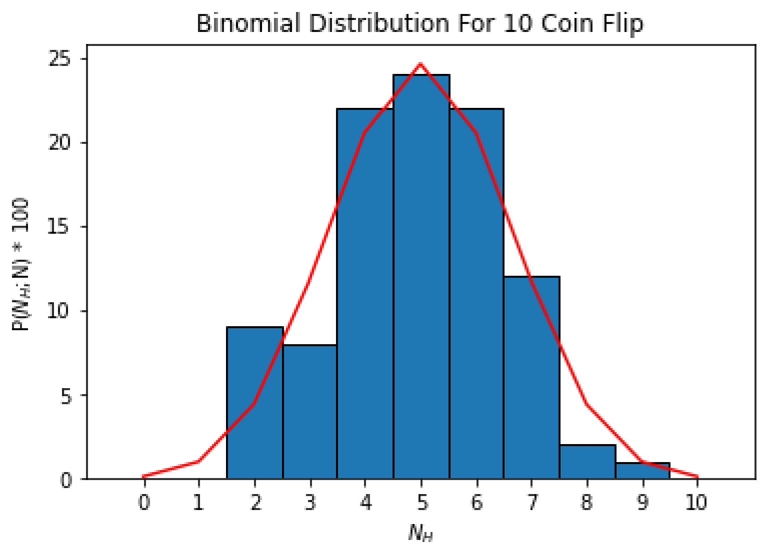
In [20]:

```
# P(Nh ; N) = (N! / (a!*(n-a)!)) * (Pa**a) * (1-Pa**N-a)

def binomial(Nh, Ntot):
    Pa = 0.5
    out = (np.math.factorial(Ntot) / (np.math.factorial(Nh)*np.math.factorial
    return out

range = np.arange(0,11, 1)
output = [binomial(x, 10) for x in range]

plt.hist(count_array, bins = bins, align="left", ec="black")
plt.plot([x * 100 for x in output], 'red') #use scale factor of 100 for 100 t
plt.xticks(range)
plt.xlabel("$N_H$")
plt.ylabel("P($N_H$;N) * 100")
plt.title("Binomial Distribution For 10 Coin Flip")
print("", end="")
```

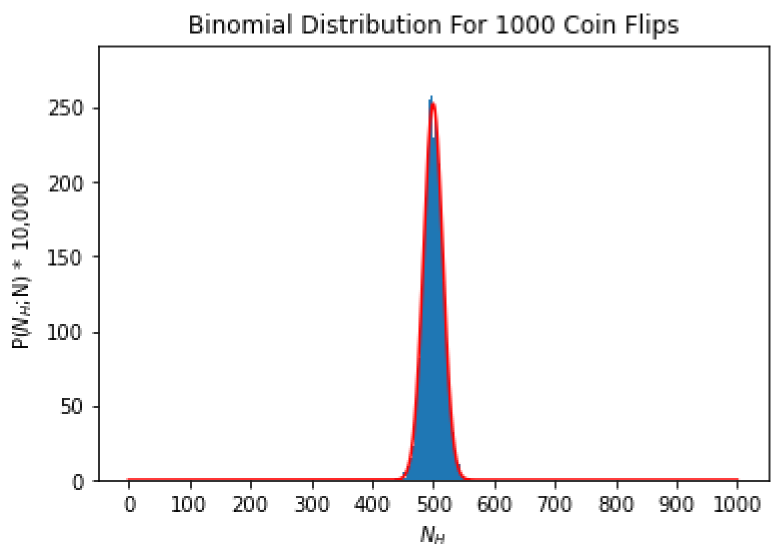


Plot Binomial Distribution For 1000-Flip Trials

In [21]:

```
range_2 = np.arange(0,1001, 1)
output_2 = [binomial(x, 1000) for x in range_2]

plt.hist(thousand_count_array, bins = new_bins, align="left")
plt.plot([x * 10000 for x in output_2], 'red')  #use scale factor of 10000 fo
plt.xticks(np.arange(0, 1100, 100))
plt.xlabel("$N_H$")
plt.ylabel("P($N_H$;N) * 10,000")
plt.title("Binomial Distribution For 1000 Coin Flips")
print("", end="")
```



Problem B4

In [22]:

```
num_k = np.arange(1, 201, 1)

def plane_binomial(k):
    P_f_int_k = (2000/300000000) * (200/300000000)
    probability = (np.math.factorial(200) / (np.math.factorial(k) * np.math.f
    return probability

all_probs = [plane_binomial(k) for k in num_k]

total_probability = sum(all_probs)
print("The probability of enountering someone you know on the plane is: {}".f
```

```
The probability of enountering someone you know on the plane is: 8.88888888
4958025e-10
```