Question Number1

You have a coin that, when flipped, has the probability 0.72 of coming up heads. You flip the coin 10 times. What is the probability that it will come up heads an even number of times?

Caculation : i did this using bionomial distribution

prob_head_even_times
=(10C0(1/2)^0(1-1/2)^10-0)+(10C2(1/2)^2(1-1/2)^10-2)+(10C4(1/2)^4(1-1/2)^10-4)+(10C6(1/2)^6(1-1/2)^10-6)+
(10C8(1/2)^8(1-1/2)^10-8)+(10C10(1/2)^10(1-1/2)^10-10)

prob_head_even_times = (0.0009765625)+ (0.04492187499999999)+(0.2500000000000002)+(0.45507812500000044)+(0.49902343750000044)+(0.5000000000000004)

prob_head_even_times = 0.5000000000000004

Question 1

```
from scipy.stats import binom
import matplotlib.pyplot as plt
import numpy as np

def prob_head_even_times(number_of_flips , optional=None):

    prob_head_even_time = 0
    for r in range(number_of_flips+1):
        if r%2==0:
            dist = binom.pmf(r, number_of_flips, 0.5)
            prob_head_even_time = prob_head_even_time+ dist
            print( r, " = ", prob_head_even_time)

    return prob_head_even_time

prob_head_even_times(10)
```

using this formula

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

```
binom.pmf(k) = choose(n, k) * p**k * (1-p)**(n-k)
```

[ ]:
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np

import seaborn as sns
sns.set(color_codes=True)
sns.set_style("white")
```

[ ]:
```
np.random.uniform(low=0.0, high=1.0)  #always return the value between zero and one
```

[ ]:
```
# generate a 'flip'
def flip(num = 10):
    flips = []

    for i in range(num):
        num = np.random.uniform(low=0.0, high=1.0)
        if num > 0.75:
            flips.append('H')             # should be doing yield here if you know 'generators'
        else:
            flips.append('T')
    return flips
```

[ ]:
```
flip()
```

[ ]:
```
flips = flip(10)
print(flips)
```

[ ]:
```
values, counts = np.unique(flips, return_counts=True)
```

values, counts

Repdoducible Randomness

[ ]:
```
np.random.seed(0)  #you can any values fix here
```

[ ]:
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np

import seaborn as sns
sns.set(color_codes=True)
sns.set_style("white")     # See more styling options here: https://seaborn.pydata.org/tutorial/aesthetics.html


# np.random.seed()   # random numbers and seed


# generate a 'flip'
def flip(num = 1):
    flips = []

    for i in range(num):
        num = np.random.uniform(low=0.0, high=1.0)
        if num > 0.75:
            flips.append('H')             # should be doing yield here if you know 'generators'
        else:
            flips.append('T')
    return flips


# Flip
flips = flip(10)
values, counts = np.unique(flips, return_counts=True)
```

[115]:
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np

import seaborn as sns
sns.set(color_codes=True)
sns.set_style("white")
```

[116]:
```
np.random.uniform(low=0.0, high=1.0)  #always return the value between zero and one
```

[116]:
```
0.25430398645595265
```

[117]:
```
# generate a 'flip'
def flip(num = 10):
    flips = []

    for i in range(num):
        num = np.random.uniform(low=0.0, high=1.0)
        if num > 0.75:
            flips.append('H')             # should be doing yield here if you know 'generators'
        else:
            flips.append('T')
    return flips
```

[118]:
```
flip()
```

[118]:
```
['T', 'T', 'T', 'T', 'T', 'T', 'T', 'H', 'T', 'H']
```

[119]:
```
flips = flip(10)
print(flips)
```

```
['H', 'T', 'T', 'T', 'H', 'T', 'H', 'T', 'T', 'T']
```

[120]:
```
values, counts = np.unique(flips, return_counts=True)
```

values, counts

Code

Repdoducible Randomness

[139]:
```
np.random.seed(0)  #you can any values fix here
```

[140]:
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

import numpy as np

import seaborn as sns
sns.set(color_codes=True)
sns.set_style("white")     # See more styling options here: https://seaborn.pydata.org/tutorial/aesthetics.html


# np.random.seed()   # random numbers and seed


# generate a 'flip'
def flip(num = 1):
    flips = []

    for i in range(num):
        num = np.random.uniform(low=0.0, high=1.0)
        if num > 0.75:
            flips.append('H')             # should be doing yield here if you know 'generators'
        else:
            flips.append('T')
    return flips


# Flip
flips = flip(10)
values, counts = np.unique(flips, return_counts=True)
```

[123]:
```
print (values)#/stats
print(flip())
```

```
['H' 'T']
['H']
```

['']

[124]:
```
print(flips)
print(values, counts)
```

```
['T', 'T', 'T', 'T', 'T', 'T', 'T', 'H', 'H', 'T']
['H' 'T'] [2 8]
```

Probability of Flips

[125]:
```
from collections import Counter, defaultdict

def get_freqs(flips):

    keys = Counter(flips).keys()
    vals = Counter(flips).values()

    # print(keys)
    # print(vals)

    # return dict(zip(keys, vals))     # bug: what if there are no 'H' or no 'T'

    return defaultdict(int, dict(zip(keys, vals)))
```

[126]:
```
freqs = get_freqs(flips)
print(freqs)
```

```
defaultdict(<class 'int'>, {'T': 8, 'H': 2})
```

[127]:
```

prob_h = freqs['H'] / len(flips)
print(prob_h)
```

```
0.2
```

Experiment: Prob calculated based on 1 flip upto N flips

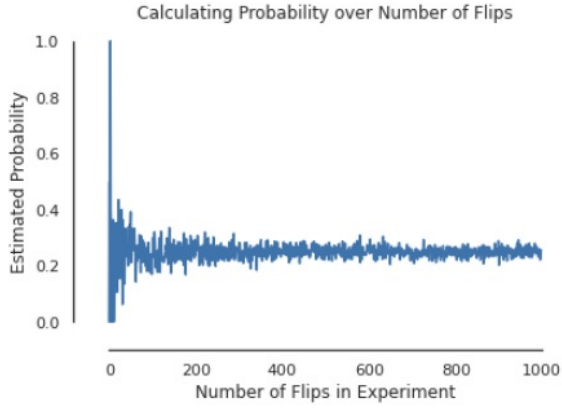
[128]:
```
maximum_flips = 1000

probs = []
for num_flips in range(1, maximum_flips):
    flips = flip(num_flips)
    freqs = get_freqs(flips)
    prob_h = freqs['H'] / len(flips)

    probs.append(prob_h)
```

[129]:
```
#print(probs)
```

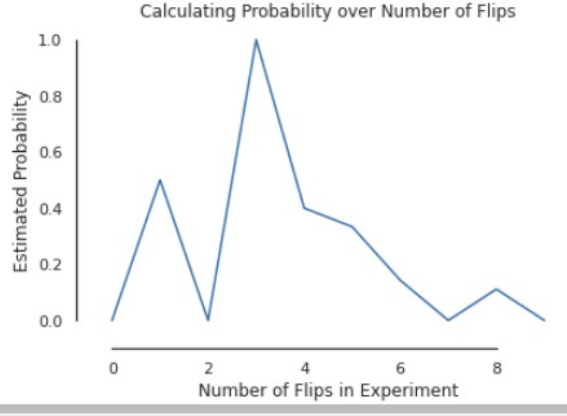
[130]:
```
print(freqs)
```

```
defaultdict(<class 'int'>, {'H': 256, 'T': 743})
```

[131]:
```
plt.plot(probs)
plt.ylabel('Estimated Probability')
plt.xlabel('Number of Flips in Experiment');
plt.title("Calculating Probability over Number of Flips")
sns.despine(offset=10, trim=True);  # move axes away
plt.show()
```

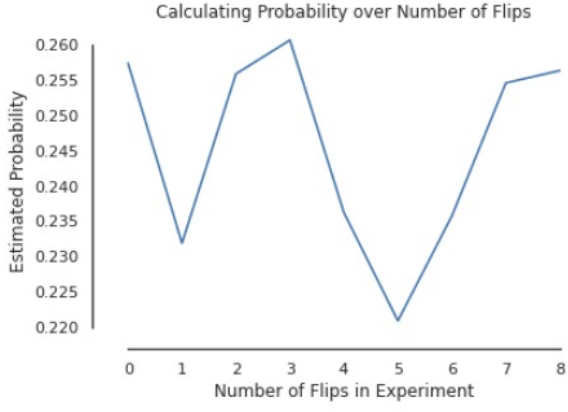


[132]:
```
plt.plot(probs[:10])
plt.ylabel('Estimated Probability')
plt.xlabel('Number of Flips in Experiment');
plt.title("Calculating Probability over Number of Flips")
sns.despine(offset=10, trim=True);  # move axes away
plt.show()
```



[133]:

```
plt.plot(probs[maximum_flips-10:])
plt.ylabel('Estimated Probability')
plt.xlabel('Number of Flips in Experiment');
plt.title("Calculating Probability over Number of Flips")
sns.despine(offset=10, trim=True);  # move axes away
plt.show()
```



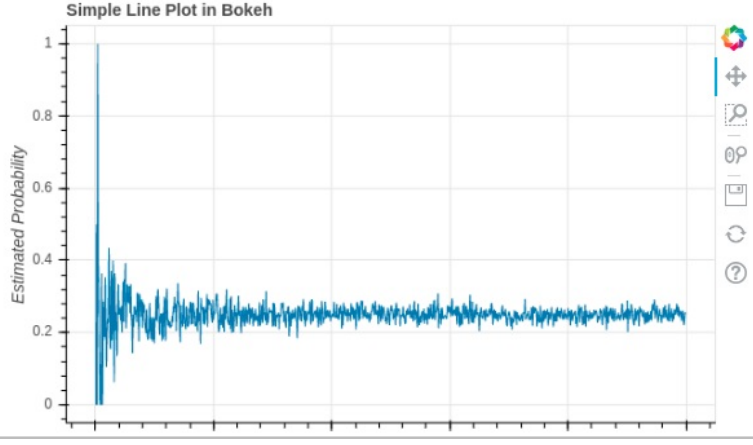
```
Requirement already satisfied: pyparsing>=2.0.2 in /home/iffishells/anaconda3/lib/python3.8/site-packages (from packaging>=16.8->bokeh) (2.4.7)
```

[135]:
```
from bokeh.io import show, output_notebook
from bokeh.plotting import figure

output_notebook()
```

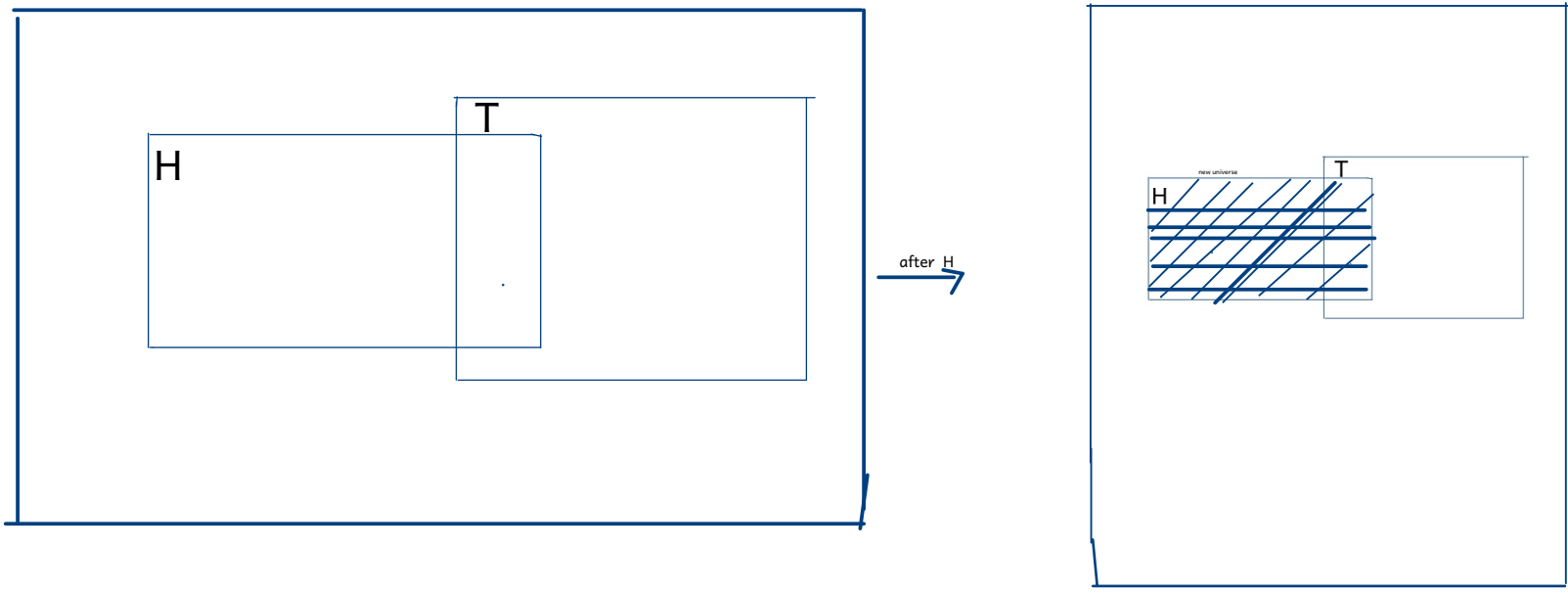
BokehJS 2.1.1 successfully loaded.

[136]:
```
p = figure(title="Simple Line Plot in Bokeh",
           x_axis_label='Number of Flips in Experiment',
           y_axis_label='Estimated Probability',
           plot_width=580, plot_height=380)
```

[137]:
```
# Add a line renderer with legend and line thickness
x = range(1, maximum_flips)
p.line(x=x, y=probs)

# Show the results
show(p)
```



Before Universe



Question 3

1 - You flip a fair coin two times. You know that one of them was heads. What is the probability that the other one was tails.

if i flip a coin two times then there possible outcome is {TH,HT,TT,HH}. bUT we know head is one of them then there remaining possible is {TH,HT,HH} then the probaility of other tail is 2/3

$$P(T|H) = \frac{P(T \cap H)}{P(H)} \quad \text{//P(H) = 1 because is head is accour its new universe}$$

$$P(T|H) = \frac{2/3}{1}$$

Question 3(part 2)

Does your answer change if we change the statement to: You flip a fair coin two times. You know that the second flip was heads. What is the probability that the first one was tails?

Yes.

if i flip a coin two times then there possible outcome is {TH,HT,TT,HH}. bUT we know Seconed one is head is one of them then there remaining possible is {TH,HH} then the probaility of other tail is 1/2