Homework #1 – Tools and Fundamentals

Name: Tejas Harishchandra Acharya

Date: 02-08-2025

ID: 9933-6984-15

In [1]:
```python
# Imports for HW_1
import random
import matplotlib.pyplot as plt
```

In [2]:
```python
# Constants for Q_1
P_HEAD = 0.7
```

In [3]:
```python
def is_head():
    return random.uniform(0, 1) < P_HEAD
```

In [4]:
```python
def get_trials(num_trials):
    return [is_head() for i in range(num_trials)]
```

In [5]:
```python
def get_longest_run_heads(trials):
    max_run = 0
    curr_run = 1 if trials[0] else 0

    for i in range(1, len(trials)):
        if trials[i]:
            if trials[i] == trials[i - 1]:
                curr_run += 1
            else:
                curr_run = 1
        else:
            max_run = max(max_run, curr_run)
            curr_run = 0
    return max_run
```

1 (a)

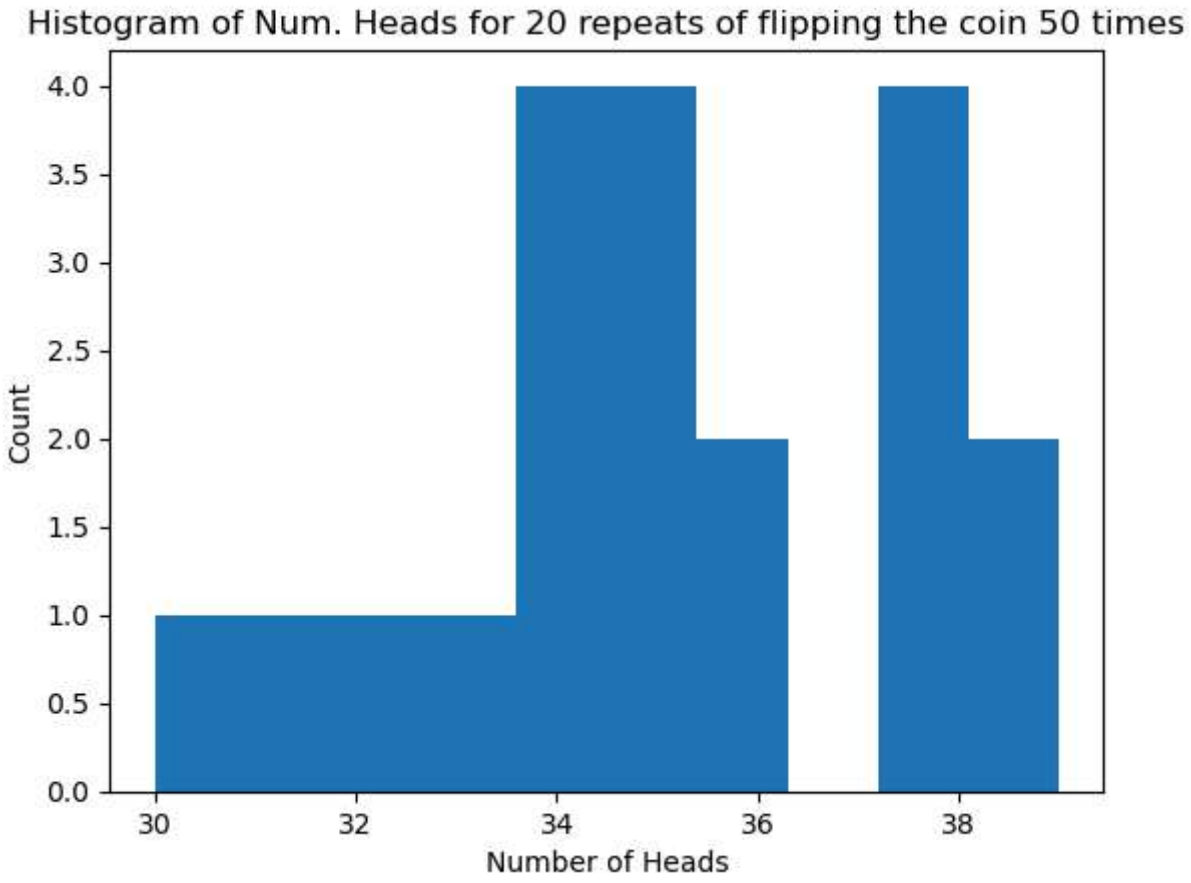
In [6]:
```
num_trials = 50
trials = get_trials(num_trials)
num_heads = sum(trials)
print(f"For 50 trials, the number of Heads = {num_heads}")
```

```
For 50 trials, the number of Heads = 35
```

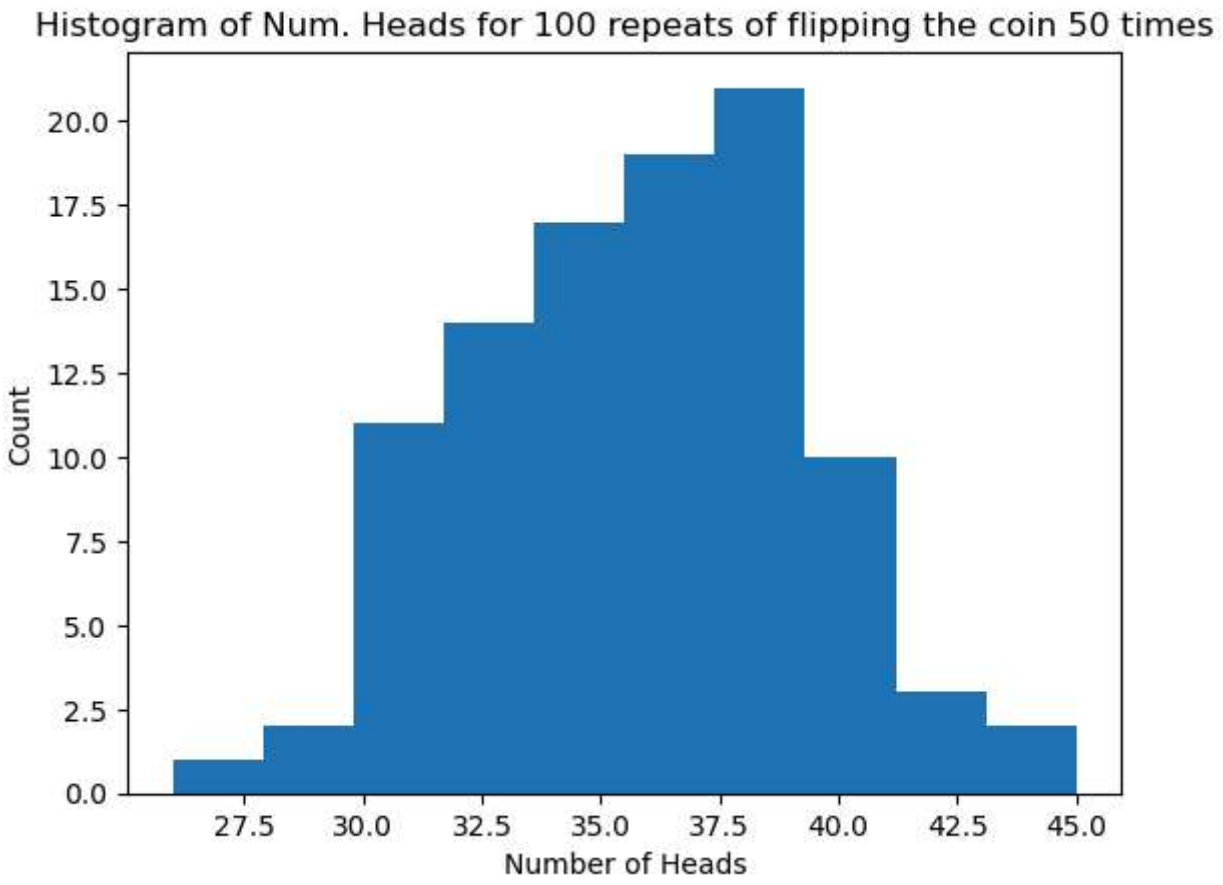
In [7]:
```
longest_run_heads = get_longest_run_heads(trials)
print(f"For 50 trials, the longest run of heads = {longest_run_heads}")
```

```
For 50 trials, the longest run of heads = 6
```

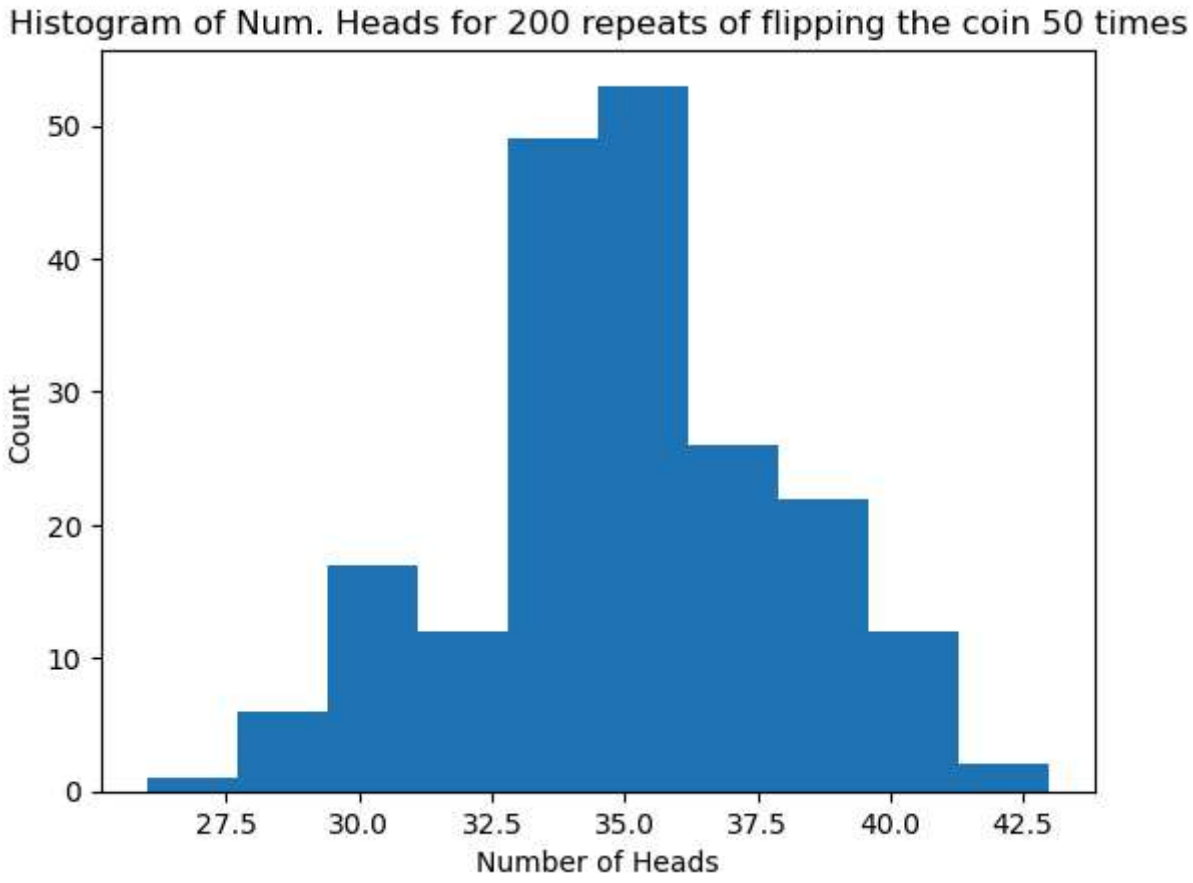
In [8]:
```
num_repeats = 20

num_heads_list = [sum(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Number of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Num. Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```



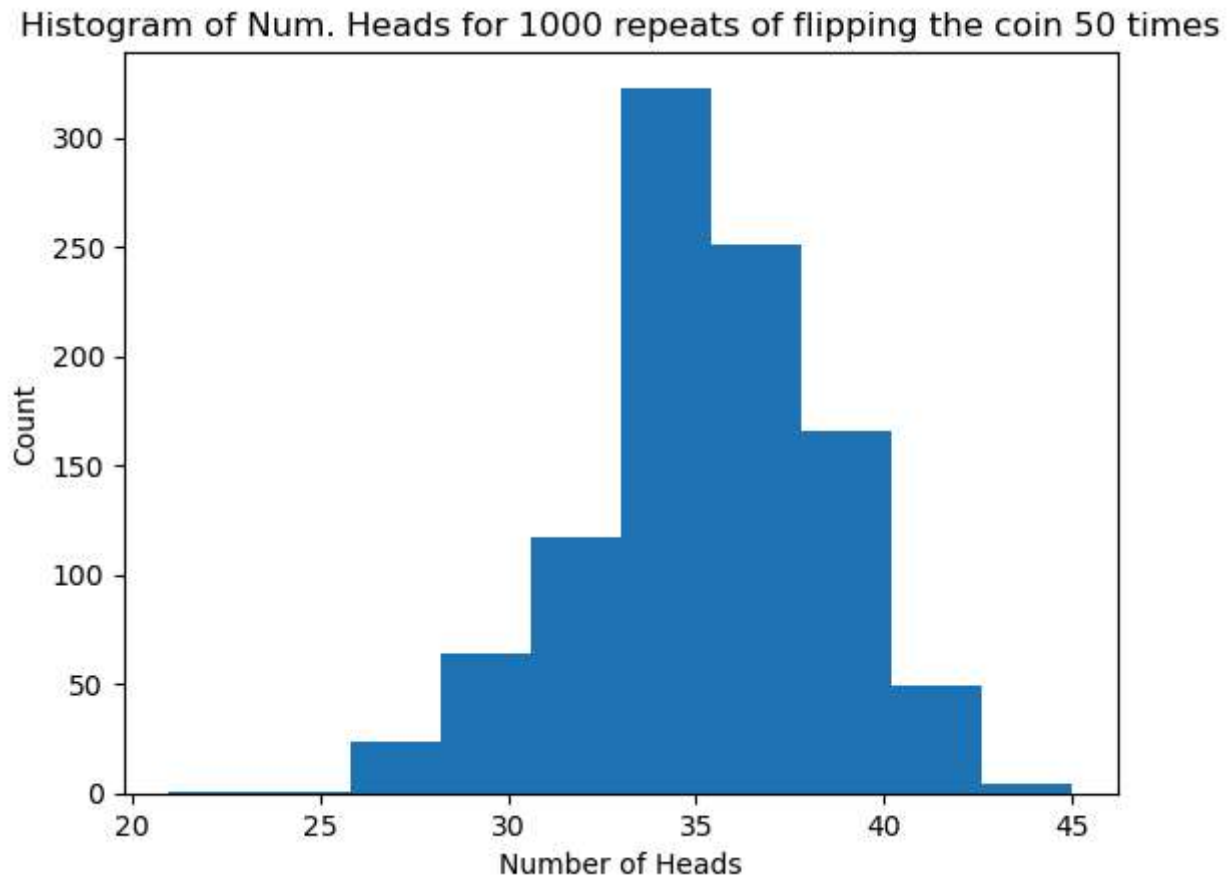
In [9]:
```
num_repeats = 100

num_heads_list = [sum(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Number of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Num. Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```



In [10]:
```
num_repeats = 200

num_heads_list = [sum(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Number of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Num. Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```



In [11]:
```
num_repeats = 1000

num_heads_list = [sum(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Number of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Num. Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```



In [12]:
```
print(f"X-Axis limit: {min(num_heads_list)} to {max(num_heads_list)}")
```

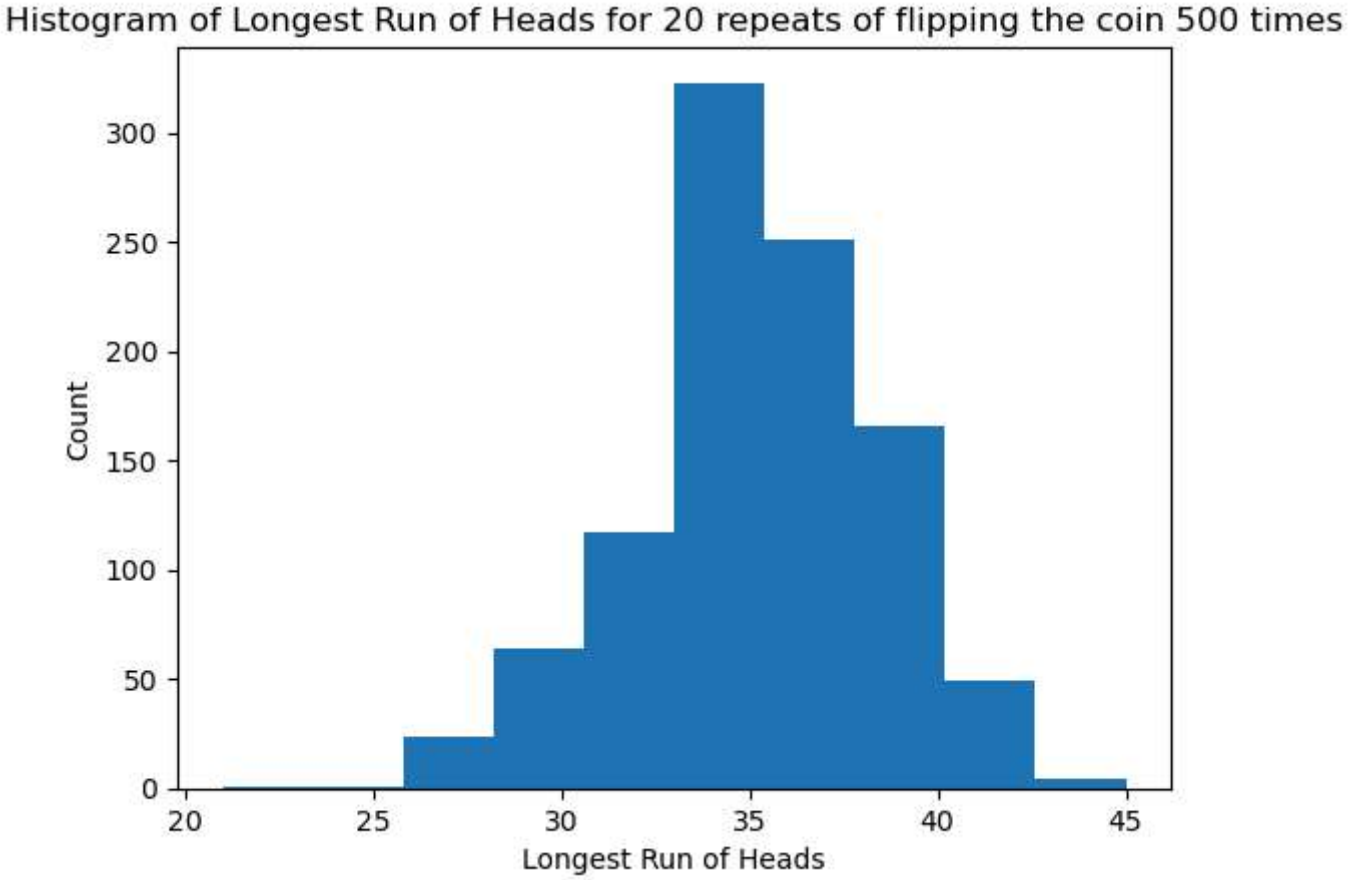
```
X-Axis limit: 21 to 45
```

1 (b)

In [13]:
```
num_trials = 500
```

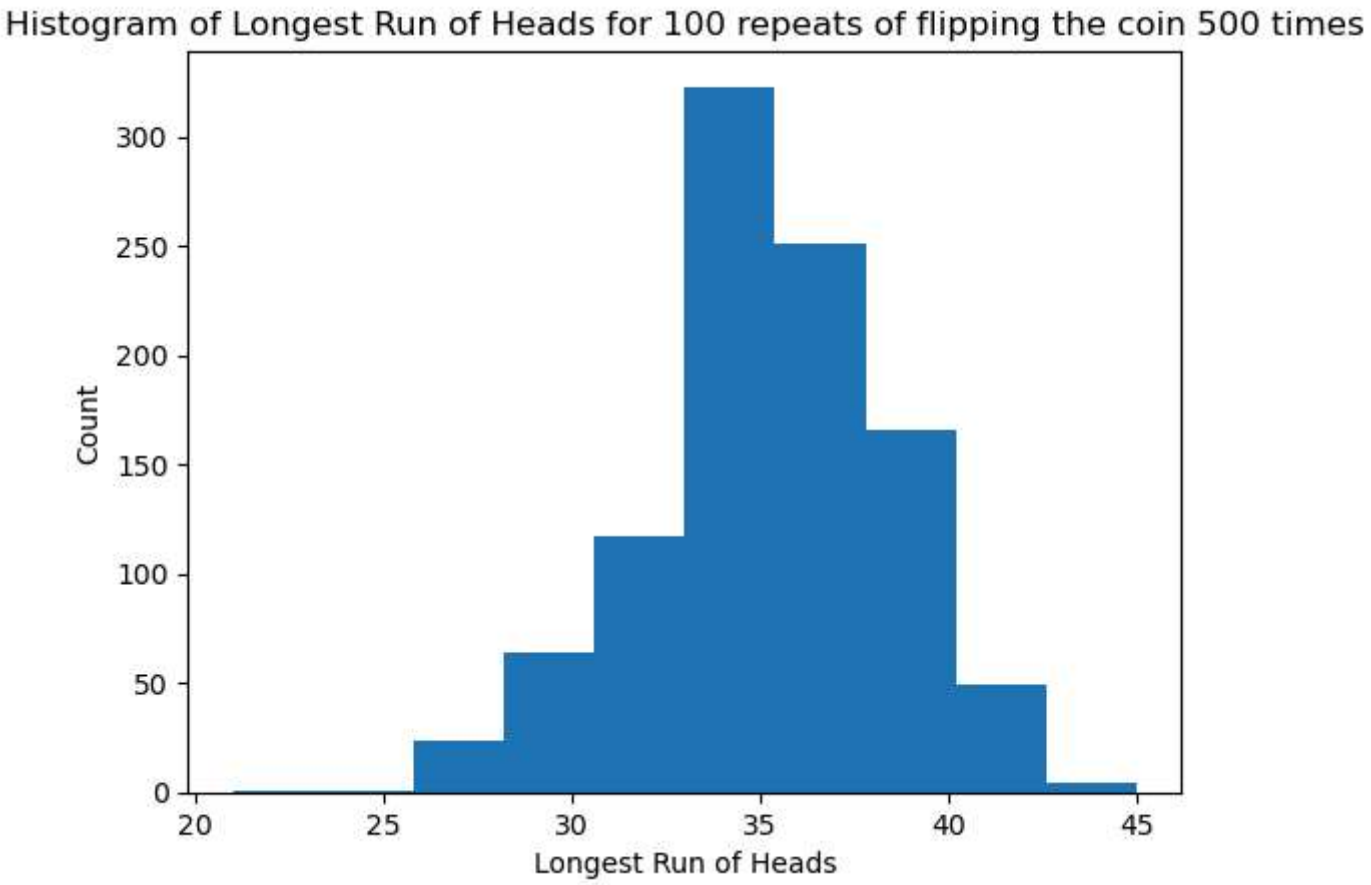
In [14]:
```
num_repeats = 20

longest_heads_run_list = [get_longest_run_heads(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
```

```
plt.xlabel("Longest Run of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Longest Run of Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```

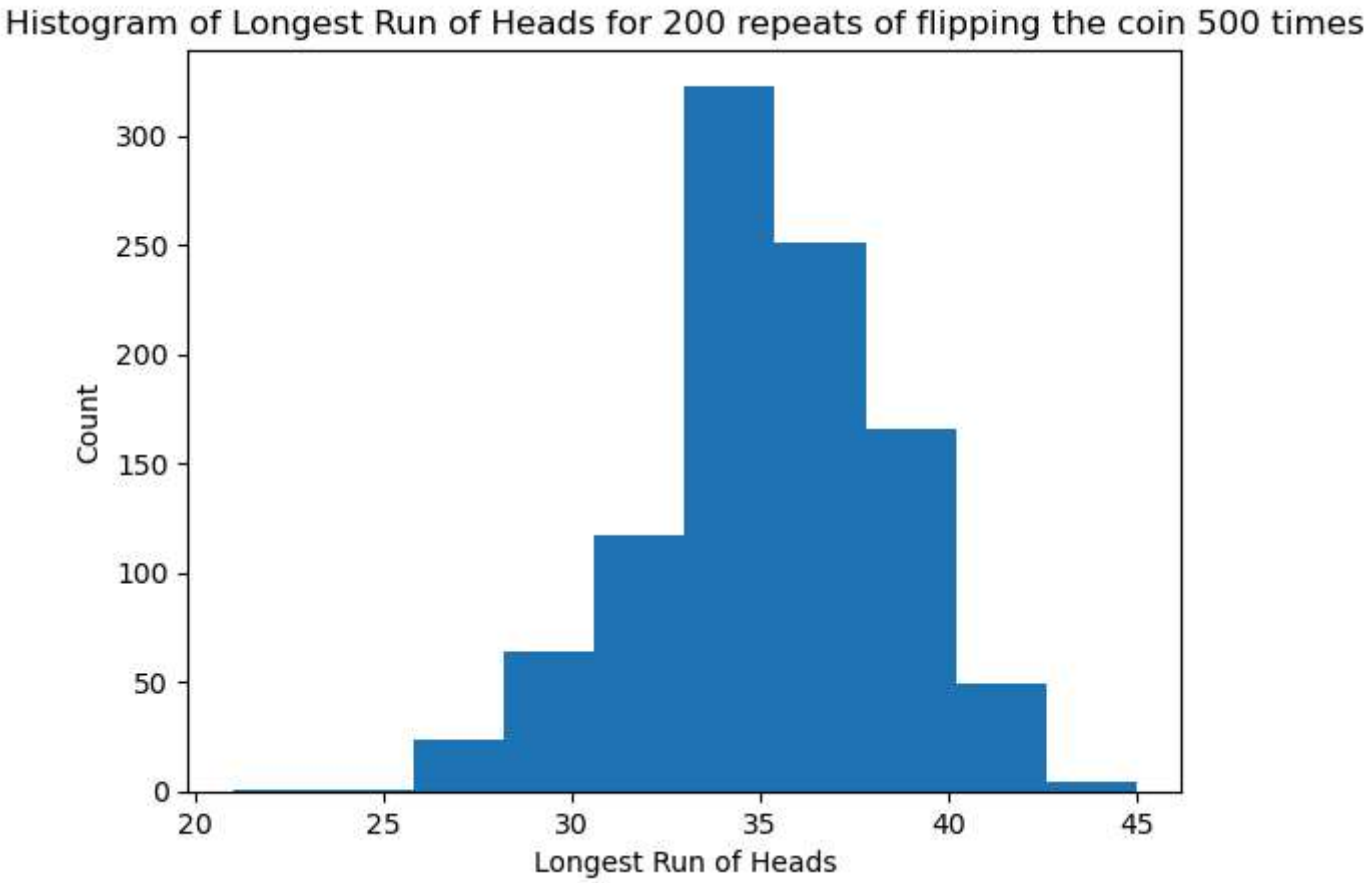


In [15]:
```
num_repeats = 100

longest_heads_run_list = [get_longest_run_heads(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Longest Run of Heads")
plt.ylabel("Count")
```

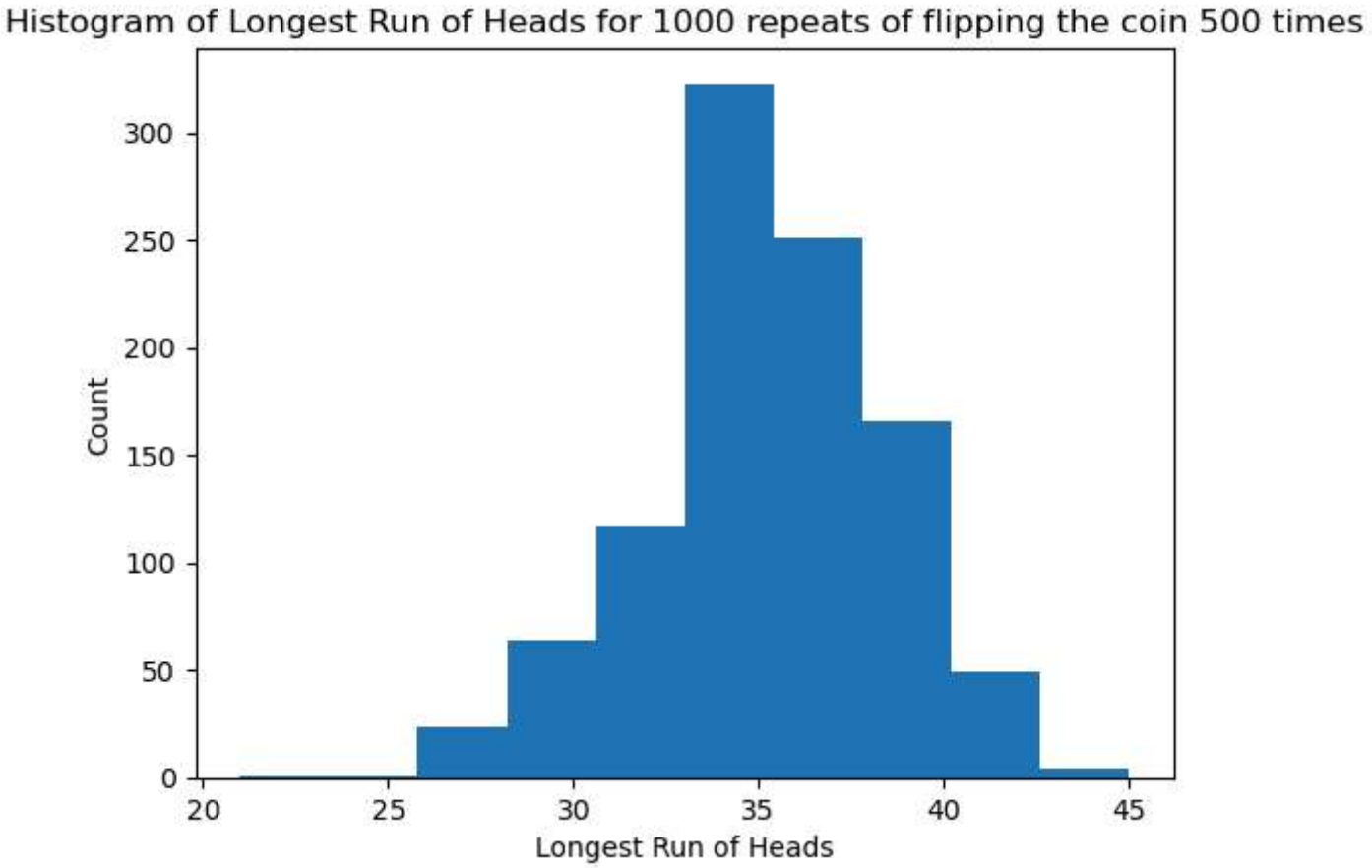
```
plt.title(f"Histogram of Longest Run of Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```



In [16]:
```
num_repeats = 200

longest_heads_run_list = [get_longest_run_heads(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Longest Run of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Longest Run of Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```



In [17]:
```
num_repeats = 1000

longest_heads_run_list = [get_longest_run_heads(get_trials(num_trials)) for i in range(num_repeats)]

plt.figure()
plt.hist(num_heads_list)
plt.xlabel("Longest Run of Heads")
plt.ylabel("Count")
plt.title(f"Histogram of Longest Run of Heads for {num_repeats} repeats of flipping the coin {num_trials} times")
plt.show()
```

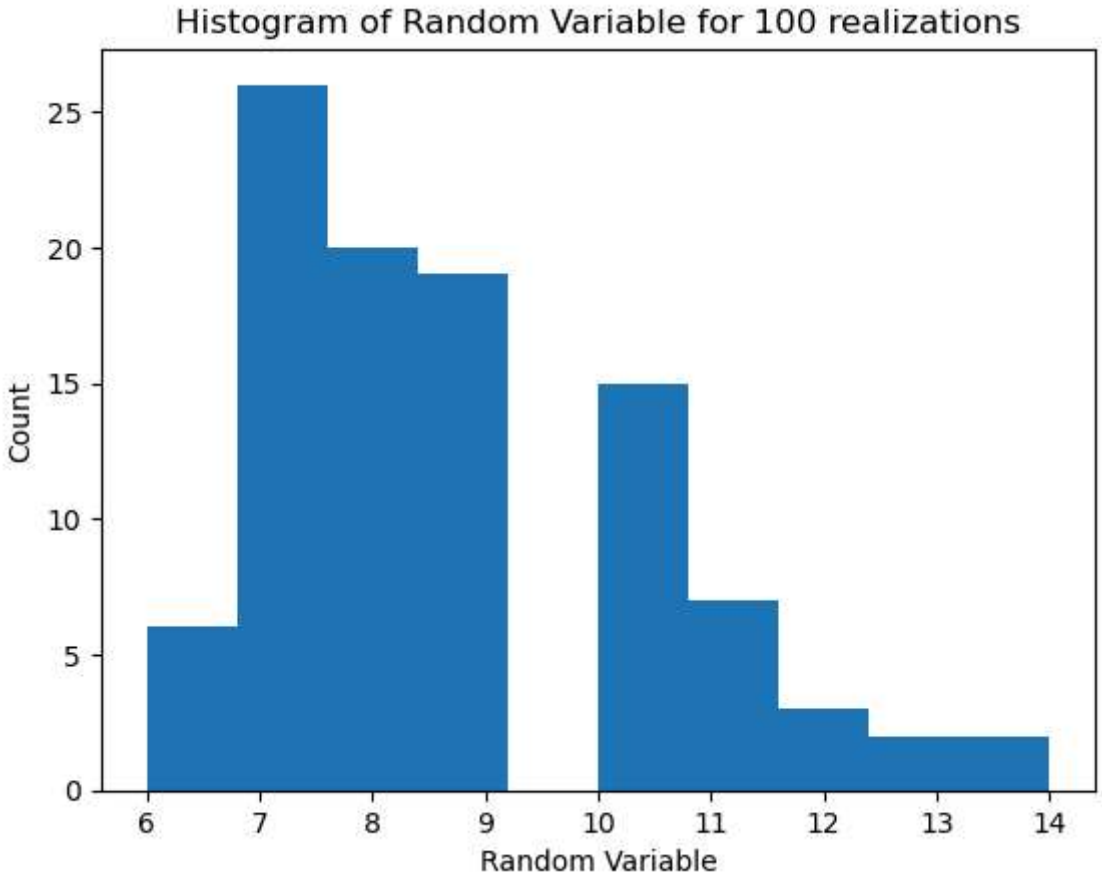


2

In [18]:
```
# Constants for Q_2
SUM = 4
```

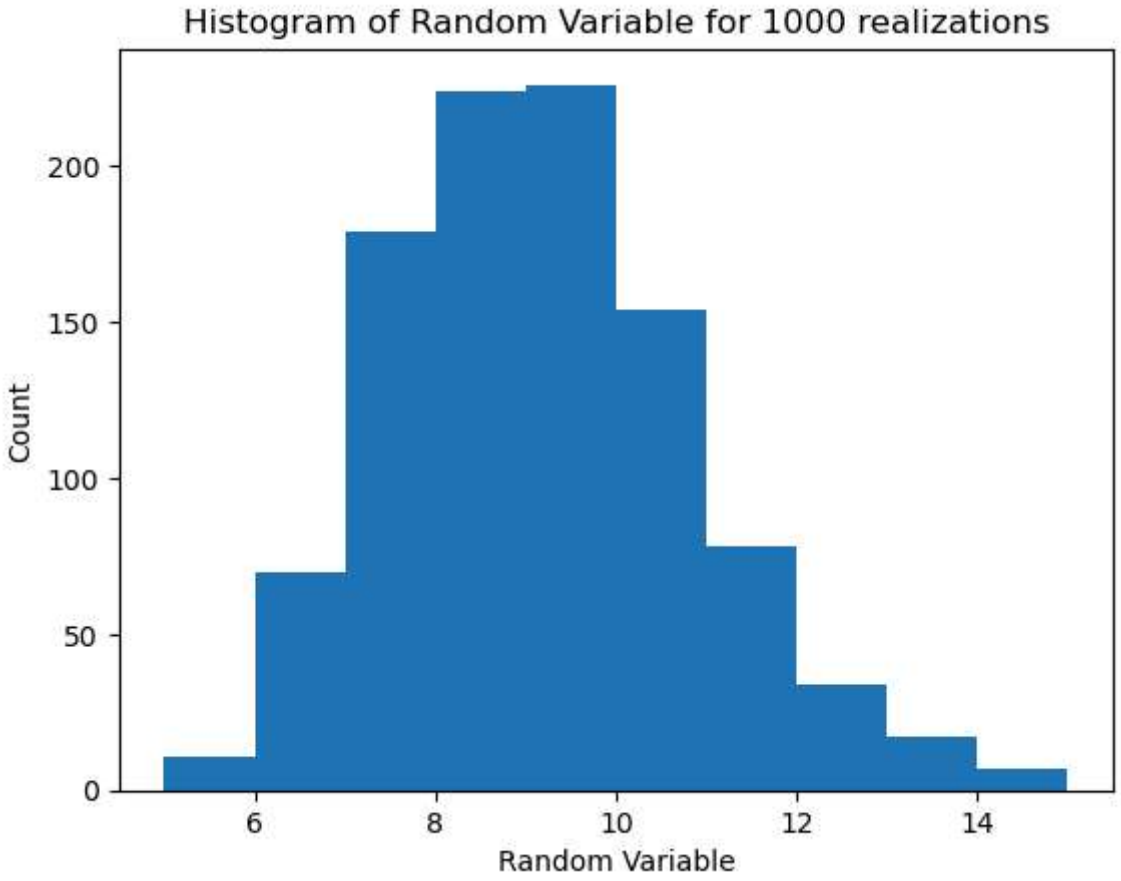
In [19]:
```
def get_random_variable():
    n = 0
    running_sum = 0
    while running_sum <= SUM:
        n += 1
        running_sum += random.uniform(0, 1)
    return n
```

In [20]:
```
realizations = 100

random_var_list = [get_random_variable() for i in range(realizations)]

plt.figure()
plt.hist(random_var_list)
plt.xlabel("Random Variable")
plt.ylabel("Count")
plt.title(f"Histogram of Random Variable for {realizations} realizations")
plt.show()
```



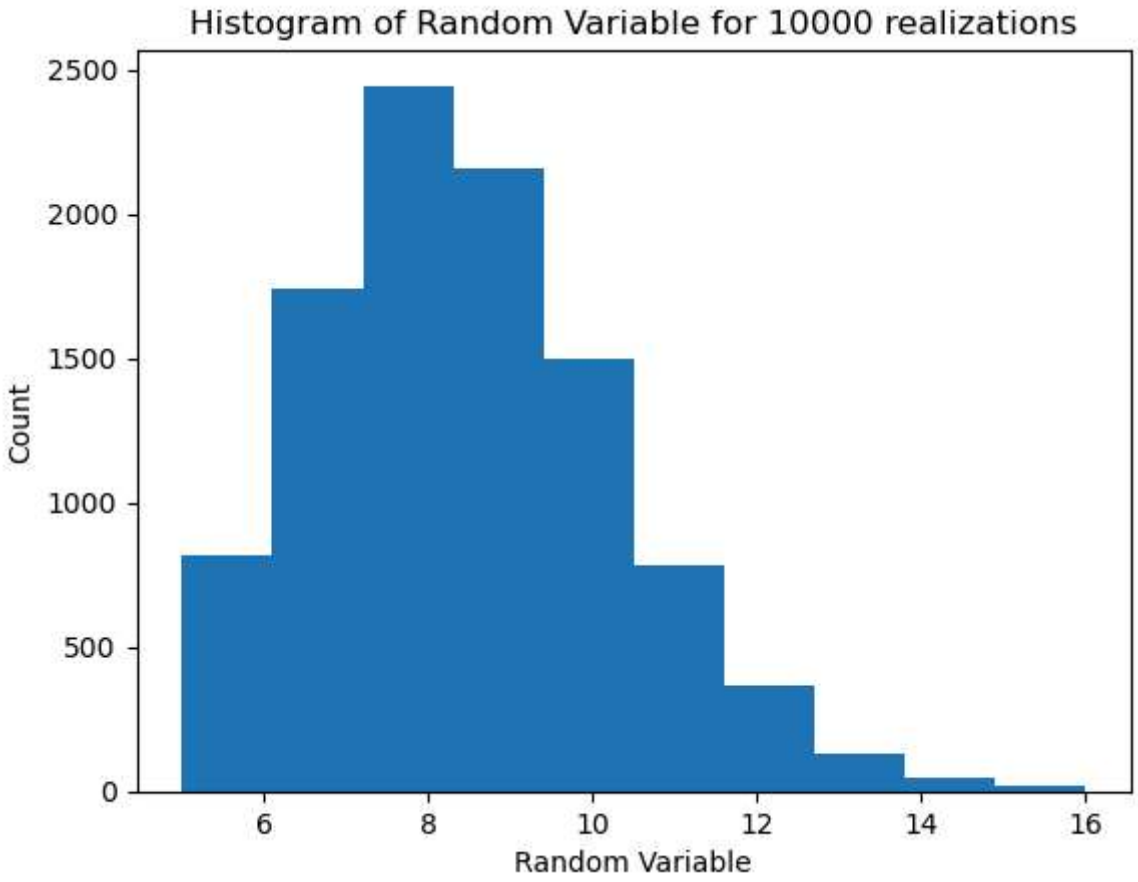
In [21]:
```
realizations = 1000

random_var_list = [get_random_variable() for i in range(realizations)]
```

```
plt.figure()
plt.hist(random_var_list)
plt.xlabel("Random Variable")
plt.ylabel("Count")
plt.title(f"Histogram of Random Variable for {realizations} realizations")
plt.show()
```



In [22]:
```
realizations = 10000

random_var_list = [get_random_variable() for i in range(realizations)]

plt.figure()
plt.hist(random_var_list)
```

```
plt.xlabel("Random Variable")
plt.ylabel("Count")
plt.title(f"Histogram of Random Variable for {realizations} realizations")
plt.show()
```



In [23]:
```
print("E[N] = 8")
```

```
E[N] = 8
```

3

In [ ]:
```
# Imports for Q_3
from func import f
import sys
```

In [ ]:
```
# Constants for Q-3
CONVERGENCE_CRITERION = 10E-10
```

In [ ]:
```
def get_secant_root(a, b):
    xn_1 = b
    xn_2 = a
    N = 0

    if abs(xn_1 - xn_2) < CONVERGENCE_CRITERION:
        xn = (xn_1 + xn_2) / 2
        return N, xn_2, xn_1, xn
    else:
        xn = xn_1 - f(xn_1) * (xn_1 - xn_2) / (f(xn_1) - f(xn_2))
        N = 1
        xn_2 = xn_1
        xn_1 = xn
        while abs(xn_1 - xn_2) >= CONVERGENCE_CRITERION:
            xn = xn_1 - f(xn_1) * (xn_1 - xn_2) / (f(xn_1) - f(xn_2))
            N += 1
            xn_2 = xn_1
            xn_1 = xn
        return N, xn_2, xn_1, xn
```

In [ ]:
```
a = sys.argv[1]
b = sys.argv[2]

try:
    a = float(a)
    b = float(b)
except:s
    sys.stderr.write("Range error")
    sys.exit(1)

if a >= b:
    sys.stderr.write("Range error")
    sys.exit(1)

if (f(a) * f(b)) >= 0:
    sys.stderr.write("Range error")
    sys.exit(1)
```

```
N, xn_2, xn_1, xn = get_secant_root(a, b)

print(N)
print(xn_2)
print(xn_1)
print(xn)
```