Simulation Project EXTRA CREDIT

- Name: Aakansh Murali
- Class period: 4th Period
- Problem number: 4
- Problem text: On a certain day, the blood bank needs 4 donors with Type O blood. How many donors, on average, would they have to see to get exactly four donors with Type O blood, assuming that 45% of the population has Type O blood?

Partial Trial Setup

- Example: 1 person walks into the ice cream parlor to order either vanilla, chocolate or strawberry ice cream

In [2]:
```
# Write code to run a partial trial
# Refer to your previous simulations as a resource!
import numpy

donor = numpy.random.randint(1,101)

if donor <= 45:
    print("Type O")
else:
    print("Other")
```

```
Other
```

Partial Trial Function

- Can you give the above procedure a descriptive name so you can easily call it multiple times?

In [3]:
```
# Define a function that runs a partial trial and returns its result appropriat

def single_trial_donor():
    donor = numpy.random.randint(1,101)

    if donor <= 45:
        return("Type O")
    else:
        return("Other")
```

Full Trial Setup

- Example: 5 people walk into the ice cream parlor together: do 2 or more order chocolate?

In [4]:
```
# Write code to run a FULL trial
# Refer to your previous simulations as a resource!

import numpy

type_o_donors = 0
other_donors = 0

while type_o_donors < 4:
    result = single_trial_donor()
    if result == "Type O":
        type_o_donors += 1
    else:
        other_donors += 1

total_donors = type_o_donors + other_donors

print(total_donors)
```

```
7
```

Full Trial Function

- Can you give the above procedure a descriptive name so you can easily call it multiple times?

In [5]:
```
# Define a function that runs a FULL trial and returns its result appropriately

def full_trial_donors():
    type_o_donors = 0
    other_donors = 0

    while type_o_donors < 4:
        result = single_trial_donor()
        if result == "Type O":
            type_o_donors += 1
        else:
            other_donors += 1

    total_donors = type_o_donors + other_donors

    return(total_donors)
```

Full Simulation

- Make sure to look at your previous simulations for the number 10,000

In [6]:
```
# Run your full trial a large number of times
# Record your data in an array

donor_results = []

for i in range(10000):
    results = full_trial_donors()
    donor_results.append(results)

print(donor_results)
```

```
3, 8, 8, 7, 6, 12, 8, 9, 11, 7, 8, 9, 7, 7, 7, 13, 12, 7, 4, 12, 11, 5, 8,
7, 11, 9, 12, 12, 7, 7, 4, 11, 4, 7, 8, 10, 10, 5, 6, 9, 9, 5, 5, 10, 7, 9,
8, 6, 8, 7, 9, 10, 10, 9, 5, 10, 15, 9, 5, 6, 11, 5, 12, 12, 15, 13, 6, 7,
12, 12, 15, 7, 9, 6, 9, 8, 6, 8, 9, 8, 6, 8, 8, 9, 15, 7, 5, 10, 17, 12, 8,
9, 15, 10, 5, 10, 9, 18, 7, 6, 9, 7, 6, 9, 6, 13, 5, 6, 9, 12, 9, 9, 9, 7,
14, 7, 5, 8, 14, 16, 12, 16, 6, 8, 6, 6, 6, 7, 17, 12, 14, 6, 10, 4, 9, 10,
12, 11, 5, 9, 4, 12, 9, 5, 11, 7, 14, 16, 5, 8, 7, 10, 9, 7, 8, 8, 9, 5, 9,
7, 6, 11, 14, 7, 4, 4, 7, 10, 10, 10, 14, 8, 11, 9, 5, 15, 10, 6, 14, 8, 7,
11, 6, 11, 10, 8, 9, 5, 9, 16, 14, 8, 12, 11, 15, 14, 9, 18, 6, 8, 4, 14, 1
4, 13, 14, 10, 6, 9, 11, 17, 16, 6, 6, 5, 8, 6, 9, 7, 8, 5, 7, 7, 7, 10, 7,
9, 10, 9, 8, 8, 16, 8, 6, 6, 18, 9, 12, 15, 9, 11, 11, 9, 11, 14, 7, 7, 5,
5, 7, 7, 5, 8, 7, 8, 6, 12, 10, 8, 7, 20, 6, 5, 8, 12, 8, 6, 17, 10, 8, 10,
7, 7, 8, 8, 11, 6, 11, 8, 4, 17, 7, 8, 11, 14, 5, 12, 19, 12, 7, 6, 6, 8,
9, 7, 7, 6, 6, 12, 7, 4, 10, 9, 7, 10, 12, 5, 8, 12, 11, 11, 10, 9, 11, 10,
5, 5, 7, 6, 9, 6, 9, 9, 8, 8, 4, 11, 7, 8, 19, 9, 9, 15, 7, 9, 8, 11, 18,
5, 7, 11, 4, 7, 9, 4, 13, 9, 7, 6, 11, 8, 7, 23, 8, 16, 6, 14, 6, 6, 8, 9,
11, 8, 9, 11, 8, 9, 5, 9, 9, 8, 11, 16, 12, 9, 11, 5, 5, 15, 7, 10, 6, 6,
8, 8, 8, 12, 4, 6, 13, 4, 9, 6, 6, 12, 6, 7, 9, 6, 10, 6, 4, 11, 18, 9, 9,
6, 9, 9, 9, 7, 11, 11, 10, 7, 12, 6, 6, 9, 11, 12, 18, 12, 13, 7, 7, 12, 7,
```

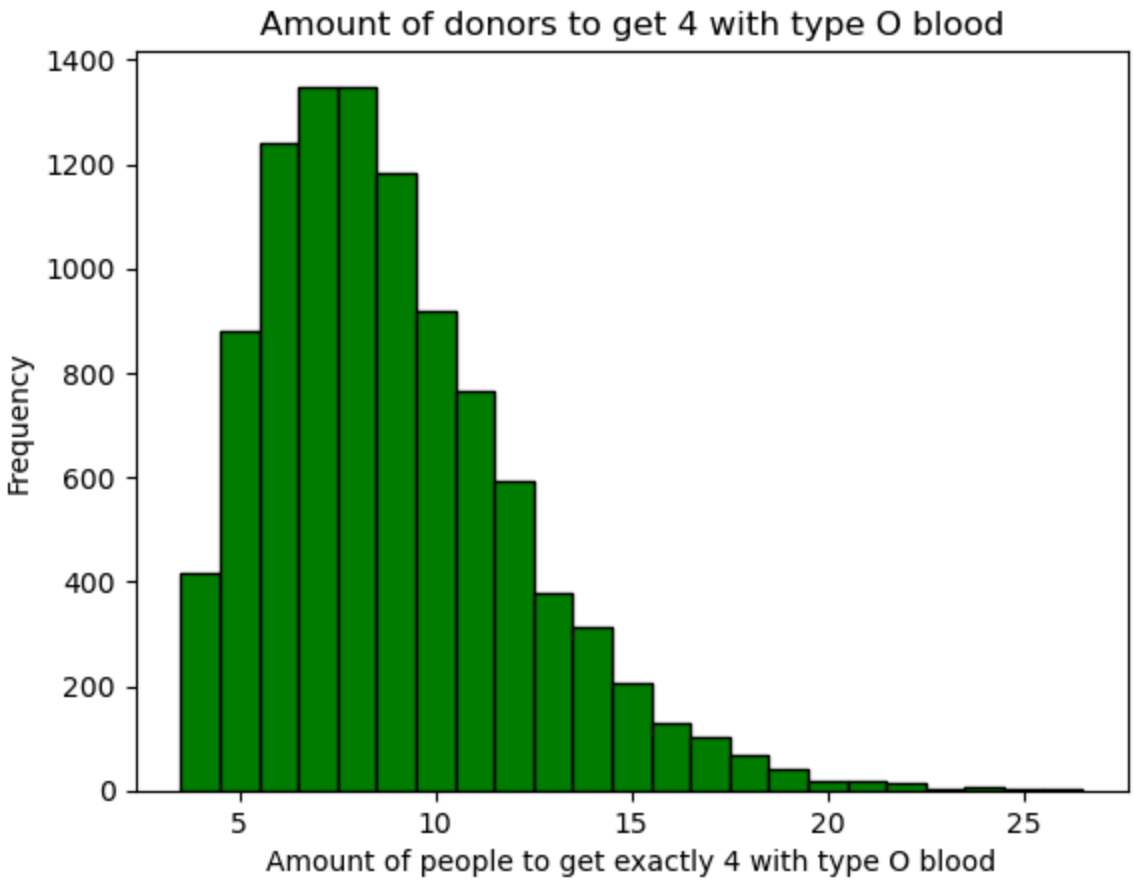
Calculate the Results

In [7]:
```
# Using the results array, grab appropriate data and perform the necessary calc

sum_of_results = 0

for trial in donor_results:
    sum_of_results += trial

average = sum_of_results/10000

print(average)
```

```
8.8765
```

Display the Results

In [12]:
```
# Import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

# Create bins array
bins = [4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]

# Create title and lables
plt.title("Amount of donors to get 4 with type O blood")
plt.xlabel("Amount of people to get exactly 4 with type O blood")
plt.ylabel("Frequency")

# plt.hist(array, dividers/bins, alignment, graph color, border color)
plt.hist(donor_results, bins, align="left", color="green", edgecolor="black")

# Show
plt.show()
```



Answer

*Provide an answer and reasoning here

According to my simulation, on average it takes 8.8

Additional Question

- State one additional question of interest (of your choice) using the data from your simulation

What is the probability that the first 4 donors will have type O blood

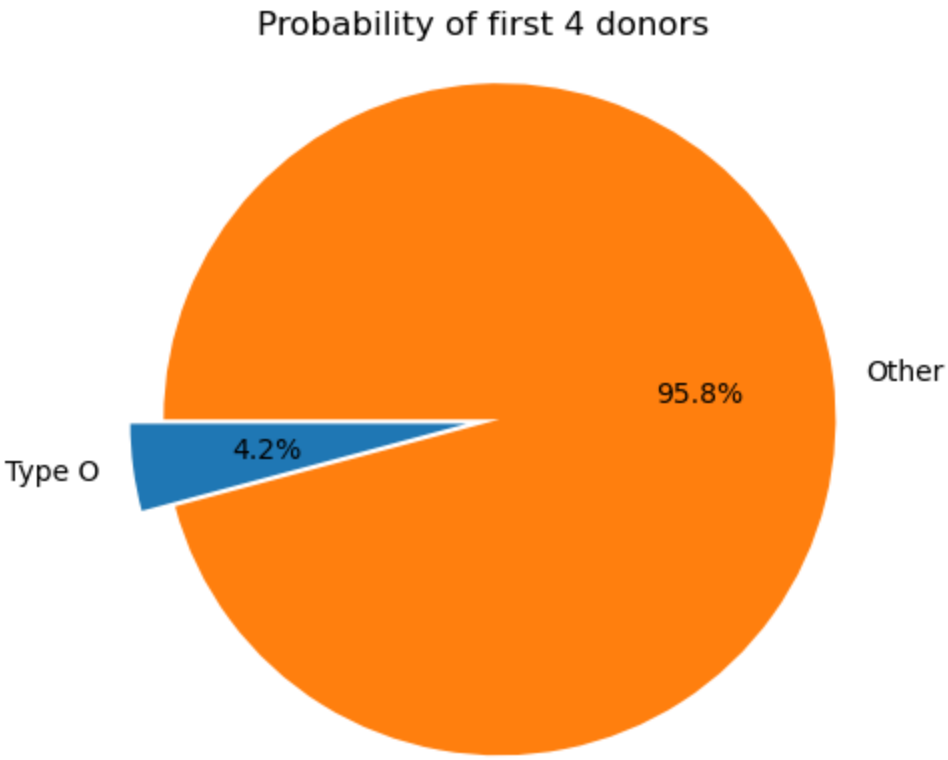
Calculate the Results

In [15]:
```
# Using the results array, grab appropriate data and perform the necessary calc

sum_of_type_O = 0
sum_of_others = 0

for trial in donor_results:
    if trial == 4:
        sum_of_type_O += 1
    else:
        sum_of_others += 1

probability_type_O = sum_of_type_O/100
probability_other = sum_of_others/100

print(probability_type_O)
```

```
4.17
```

Display the Results

In [17]:
```
# Graphically display results related directly to your question
import matplotlib.pyplot as plot
%matplotlib inline
# Magic to allow the graph to display directly in this notebook

# Create an array of labels
labels = ["Type O", "Other"]

# Create an array of your results
results = [probability_type_O, probability_other]

# Explode option
    # 'Slices' appear distanced from the center
        # Larger numbers = further explosion
    # Explode array should be same size as labels
explode = (0, 0.1)

# Use matplotlib module subplots() to get data for various charts
    # Returns a tuple in the form (figure,axes)
fig1,ax1 = plot.subplots()

# Use axes to create a pie chart
    # ax1.pie(data array, explode array, labels array, starting angle)
ax1.pie(results, explode, labels, autopct='%1.1f%%', startangle=180)

# Equal aspect ratio ensures that pie is drawn as a circle.
ax1.axis('equal')
plot.title("Probability of first 4 donors")

plot.show()
```



According to my simulation there is a 4.2% chance that the first 4 donors will have type O blood

In [ ]: