In [1]:
```
import json
import csv
```

In [ ]:
```

```

In [ ]:
```

```

In [2]:
```
with open('worldcup.json',encoding="utf8") as world_cup_file:
    world_cup_data = json.load(world_cup_file)
print(type(world_cup_data))
```

```
<class 'dict'>
```

In [3]:
```
# Run this cell without changes

# Check that the overall data structure is a dictionary
assert type(world_cup_data) == dict

# Check that the dictionary has 2 keys, 'name' and 'rounds'
assert list(world_cup_data.keys()) == ['name', 'rounds']
```

In [4]:
```

with open('population_csv.csv') as population_file:
    population_data = list(csv.DictReader(population_file))

```

In [5]:
```
# Run this cell without changes

# Check that the overall data structure is a list
assert type(population_data) == list

# Check that the 0th element is a dictionary
# (csv.DictReader interface differs slightly by Python version;
# either a dict or an OrderedDict is fine here)
from collections import OrderedDict
assert type(population_data[0]) == dict or type(population_data[0]) == OrderedDict
```

In [6]:
```
world_cup_data.keys()
```

Out[6]:
```
dict_keys(['name', 'rounds'])
```

In [7]:
```
world_cup_data["name"]
```

Out[7]:
```
'World Cup 2018'
```

In [8]:
```
rounds = world_cup_data["rounds"]

print("type(rounds):", type(rounds))
print("len(rounds):", len(rounds))
print("type(rounds[3])", type(rounds[3]))
print("rounds[3]:")
rounds[3]
```

```
type(rounds): <class 'list'>
len(rounds): 20
type(rounds[3]) <class 'dict'>
rounds[3]:
```

Out[8]:

```
{'name': 'Matchday 4',
 'matches': [{'num': 9,
   'date': '2018-06-17',
   'time': '21:00',
   'team1': {'name': 'Brazil', 'code': 'BRA'},
   'team2': {'name': 'Switzerland', 'code': 'SUI'},
   'score1': 1,
   'score2': 1,
   'score1i': 1,
   'score2i': 0,
   'goals1': [{'name': 'Coutinho', 'minute': 20, 'score1': 1, 'score2': 0}],
   'goals2': [{'name': 'Zuber', 'minute': 50, 'score1': 1, 'score2': 1}],
   'group': 'Group E',
   'stadium': {'key': 'rostov', 'name': 'Rostov Arena'},
   'city': 'Rostov-on-Don',
   'timezone': 'UTC+3'},
  {'num': 10,
   'date': '2018-06-17',
   'time': '16:00',
   'team1': {'name': 'Costa Rica', 'code': 'CRC'},
   'team2': {'name': 'Serbia', 'code': 'SRB'},
   'score1': 0,
   'score2': 1,
   'score1i': 0,
   'score2i': 0,
   'goals1': [],
   'goals2': [{'name': 'Kolarov', 'minute': 56, 'score1': 0, 'score2': 1}],
   'group': 'Group E',
   'stadium': {'key': 'samara', 'name': 'Samara Arena'},
   'city': 'Samara',
   'timezone': 'UTC+4'},
  {'num': 11,
   'date': '2018-06-17',
   'time': '18:00',
   'team1': {'name': 'Germany', 'code': 'GER'},
   'team2': {'name': 'Mexico', 'code': 'MEX'},
   'score1': 0,
   'score2': 1,
   'score1i': 0,
   'score2i': 1,
   'goals1': [],
   'goals2': [{'name': 'Lozano', 'minute': 35, 'score1': 0, 'score2': 1}],
   'group': 'Group F',
   'stadium': {'key': 'luzhniki', 'name': 'Luzhniki Stadium'},
   'city': 'Moscow',
   'timezone': 'UTC+3'}]}
```

In [9]:
```
matches = []

# "round" is a built-in function in Python so we use "round_" instead
for round_ in rounds:
    # Extract the list of matches for this round
    round_matches = round_['matches']
    # Add them to the overall list of matches
    matches.extend(round_matches)

matches[0]
```

Out[9]:
```
{'num': 1,
 'date': '2018-06-14',
 'time': '18:00',
 'team1': {'name': 'Russia', 'code': 'RUS'},
 'team2': {'name': 'Saudi Arabia', 'code': 'KSA'},
 'score1': 5,
 'score2': 0,
 'score1i': 2,
 'score2i': 0,
 'goals1': [{'name': 'Gazinsky', 'minute': 12, 'score1': 1, 'score2': 0},
  {'name': 'Cheryshev', 'minute': 43, 'score1': 2, 'score2': 0},
  {'name': 'Dzyuba', 'minute': 71, 'score1': 3, 'score2': 0},
  {'name': 'Cheryshev', 'minute': 90, 'offset': 1, 'score1': 4, 'score2': 0},
  {'name': 'Golovin', 'minute': 90, 'offset': 4, 'score1': 5, 'score2': 0}],
 'goals2': [],
 'group': 'Group A',
 'stadium': {'key': 'luzhniki', 'name': 'Luzhniki Stadium'},
 'city': 'Moscow',
 'timezone': 'UTC+3'}
```

In [10]:
```
# Run this cell without changes

# There should be 64 matches. If the length is 20, that means
# you have a list of lists instead of a list of dictionaries
assert len(matches) == 64

# Each match in the list should be a dictionary
assert type(matches[0]) == dict
```

In [11]:
```
# Run this cell without changes
print(matches[0]["team1"])
print(matches[0]["team2"])
```

```
{'name': 'Russia', 'code': 'RUS'}
{'name': 'Saudi Arabia', 'code': 'KSA'}
```

In [12]:
```
# Replace None with appropriate code
teams_set = set()


for match in matches:
    # Add team1 name value to teams_set
    teams_set.add(match["team1"]["name"])
    # Add team2 name value to teams_set
    teams_set.add(match["team2"]["name"])

teams = sorted(list(teams_set))
print(teams)
```

```
['Argentina', 'Australia', 'Belgium', 'Brazil', 'Colombia', 'Costa Rica', 'Croatia', 'Denmark', 'Egypt', 'England', 'France',
'Germany', 'Iceland', 'Iran', 'Japan', 'Mexico', 'Morocco', 'Nigeria', 'Panama', 'Peru', 'Poland', 'Portugal', 'Russia', 'Saudi
Arabia', 'Senegal', 'Serbia', 'South Korea', 'Spain', 'Sweden', 'Switzerland', 'Tunisia', 'Uruguay']
```

In [13]:
```
# Run this cell without changes

# teams should be a list, not a set
assert type(teams) == list

# 32 teams competed in the 2018 World Cup
assert len(teams) == 32

# Each element of teams should be a string
# (the name), not a dictionary
assert type(teams[0]) == str
```

In [14]:
```
combined_data = {}
for team in teams:
    combined_data[team] = {'wins': 0}

print(combined_data)
```

```
{'Argentina': {'wins': 0}, 'Australia': {'wins': 0}, 'Belgium': {'wins': 0}, 'Brazil': {'wins': 0}, 'Colombia': {'wins': 0}, 'C
osta Rica': {'wins': 0}, 'Croatia': {'wins': 0}, 'Denmark': {'wins': 0}, 'Egypt': {'wins': 0}, 'England': {'wins': 0}, 'Franc
e': {'wins': 0}, 'Germany': {'wins': 0}, 'Iceland': {'wins': 0}, 'Iran': {'wins': 0}, 'Japan': {'wins': 0}, 'Mexico': {'wins':
0}, 'Morocco': {'wins': 0}, 'Nigeria': {'wins': 0}, 'Panama': {'wins': 0}, 'Peru': {'wins': 0}, 'Poland': {'wins': 0}, 'Portuga
l': {'wins': 0}, 'Russia': {'wins': 0}, 'Saudi Arabia': {'wins': 0}, 'Senegal': {'wins': 0}, 'Serbia': {'wins': 0}, 'South Kore
a': {'wins': 0}, 'Spain': {'wins': 0}, 'Sweden': {'wins': 0}, 'Switzerland': {'wins': 0}, 'Tunisia': {'wins': 0}, 'Uruguay':
{'wins': 0}}
```

In [15]:
```
# Run this cell without changes

# combined_data should be a dictionary
assert type(combined_data) == dict

# the keys should be strings
assert type(list(combined_data.keys())[0]) == str

# the values should be dictionaries
assert combined_data["Japan"] == {"wins": 0}
```

In [16]:
```
def find_winner(match):
    """
    Given a dictionary containing information about a match,
    return the name of the winner (or None in the case of a tie)
    """
    if match['score1'] > match['score2']:
        return match['team1']['name']
    elif match['score2'] > match['score1']:
        return match['team2']['name']
    else:
        return None
```

In [17]:
```
# Run this cell without changes
assert find_winner(matches[0]) == "Russia"
assert find_winner(matches[1]) == "Uruguay"
assert find_winner(matches[2]) == None
```

In [18]:
```
# Replace None with appropriate code

for match in matches:
    # Get the name of the winner
    winner = None
    # Only proceed to the next step if there was
    # a winner
    if winner:
        # Add 1 to the associated count of wins
        None

# Visually inspect the output to ensure the wins are
# different for different countries
combined_data
```

Out[18]:
```
{'Argentina': {'wins': 0},
 'Australia': {'wins': 0},
 'Belgium': {'wins': 0},
 'Brazil': {'wins': 0},
 'Colombia': {'wins': 0},
 'Costa Rica': {'wins': 0},
 'Croatia': {'wins': 0},
 'Denmark': {'wins': 0},
 'Egypt': {'wins': 0},
 'England': {'wins': 0},
 'France': {'wins': 0},
 'Germany': {'wins': 0},
 'Iceland': {'wins': 0},
 'Iran': {'wins': 0},
 'Japan': {'wins': 0},
 'Mexico': {'wins': 0},
 'Morocco': {'wins': 0},
 'Nigeria': {'wins': 0},
 'Panama': {'wins': 0},
 'Peru': {'wins': 0},
 'Poland': {'wins': 0},
 'Portugal': {'wins': 0},
 'Russia': {'wins': 0},
 'Saudi Arabia': {'wins': 0},
 'Senegal': {'wins': 0},
 'Serbia': {'wins': 0},
 'South Korea': {'wins': 0},
 'Spain': {'wins': 0},
 'Sweden': {'wins': 0},
 'Switzerland': {'wins': 0},
 'Tunisia': {'wins': 0},
 'Uruguay': {'wins': 0}}
```

In [19]:
```
for match in matches:
    # Get the name of the winner
    winner = find_winner(match)

    # Only proceed to the next step if there was
    # a winner
    if winner:
        # Add 1 to the associated count of wins
        combined_data[winner]['wins'] +=1

# Visually inspect the output to ensure the wins are
# different for different countries
print(combined_data)
```

```
{'Argentina': {'wins': 1}, 'Australia': {'wins': 0}, 'Belgium': {'wins': 6}, 'Brazil': {'wins': 3}, 'Colombia': {'wins': 2}, 'C
osta Rica': {'wins': 0}, 'Croatia': {'wins': 3}, 'Denmark': {'wins': 1}, 'Egypt': {'wins': 0}, 'England': {'wins': 3}, 'Franc
e': {'wins': 6}, 'Germany': {'wins': 1}, 'Iceland': {'wins': 0}, 'Iran': {'wins': 1}, 'Japan': {'wins': 1}, 'Mexico': {'wins':
2}, 'Morocco': {'wins': 0}, 'Nigeria': {'wins': 1}, 'Panama': {'wins': 0}, 'Peru': {'wins': 1}, 'Poland': {'wins': 1}, 'Portuga
l': {'wins': 1}, 'Russia': {'wins': 2}, 'Saudi Arabia': {'wins': 1}, 'Senegal': {'wins': 1}, 'Serbia': {'wins': 1}, 'South Kore
a': {'wins': 1}, 'Spain': {'wins': 1}, 'Sweden': {'wins': 3}, 'Switzerland': {'wins': 1}, 'Tunisia': {'wins': 1}, 'Uruguay':
{'wins': 4}}
```

In [20]:
```
# Run this cell without changes
import numpy as np

wins = [val["wins"] for val in combined_data.values()]

print("Mean number of wins:", np.mean(wins))
print("Median number of wins:", np.median(wins))
print("Standard deviation of number of wins:", np.std(wins))
```

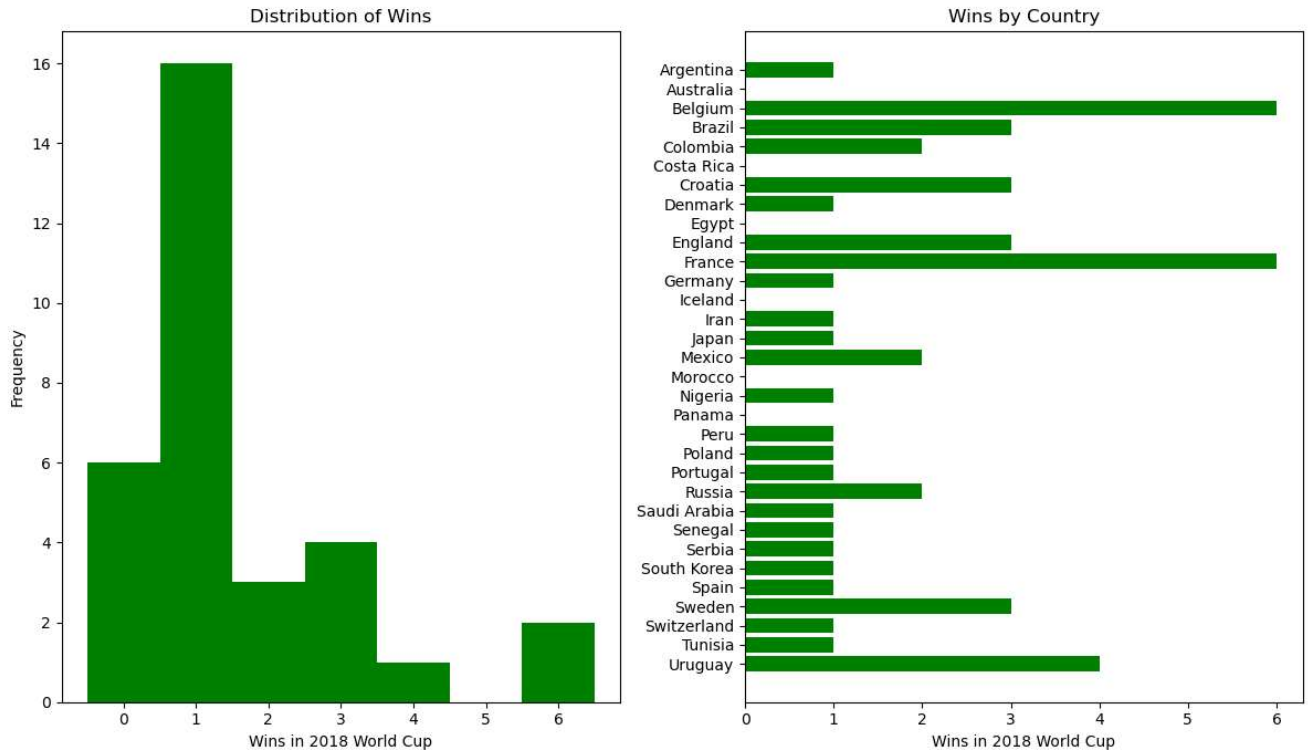
```
Mean number of wins: 1.5625
Median number of wins: 1.0
Standard deviation of number of wins: 1.5194057226429023
```

In [21]:
```
# Run this cell without changes
import matplotlib.pyplot as plt

# Set up figure and axes
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 7))
fig.set_tight_layout(True)

# Histogram of Wins and Frequencies
ax1.hist(x=wins, bins=range(8), align="left", color="green")
ax1.set_xticks(range(7))
ax1.set_xlabel("Wins in 2018 World Cup")
ax1.set_ylabel("Frequency")
ax1.set_title("Distribution of Wins")

# Horizontal Bar Graph of Wins by Country
ax2.barh(teams[::-1], wins[::-1], color="green")
ax2.set_xlabel("Wins in 2018 World Cup")
ax2.set_title("Wins by Country");
```



In [22]:
```
# Run this cell without changes
len(population_data)
```

Out[22]:
```
15409
```

In [23]:
```
# Run this cell without changes
np.random.seed(42)
population_record_samples = np.random.choice(population_data, size=10)
population_record_samples
```

Out[23]:
```
array([{'Country Name': 'Greenland', 'Country Code': 'GRL', 'Year': '1980', 'Value': '50200'},
       {'Country Name': 'High income', 'Country Code': 'HIC', 'Year': '1994', 'Value': '1034461319'},
       {'Country Name': 'Congo, Rep.', 'Country Code': 'COG', 'Year': '1981', 'Value': '1830632'},
       {'Country Name': 'St. Lucia', 'Country Code': 'LCA', 'Year': '2004', 'Value': '161816'},
       {'Country Name': 'China', 'Country Code': 'CHN', 'Year': '2018', 'Value': '1392730000'},
       {'Country Name': 'Poland', 'Country Code': 'POL', 'Year': '2016', 'Value': '37970087'},
       {'Country Name': 'North Macedonia', 'Country Code': 'MKD', 'Year': '1985', 'Value': '1981534'},
       {'Country Name': 'Cyprus', 'Country Code': 'CYP', 'Year': '1971', 'Value': '620859'},
       {'Country Name': 'Equatorial Guinea', 'Country Code': 'GNQ', 'Year': '1971', 'Value': '298846'},
       {'Country Name': 'Euro area', 'Country Code': 'EMU', 'Year': '2013', 'Value': '337302113'}],
      dtype=object)
```

In [24]:
```
# Replace None with appropriate code

population_data_filtered = []
i = 0
for record in population_data:
    # Add record to population_data_filtered if relevant
    if record['Country Name'] in teams and record["Year"] == '2018':
        population_data_filtered.append(record)


len(population_data_filtered) # 27
```

Out[24]: 27

In [26]:
```
# Run this cell without changes
teams[13]
```

Out[26]: 'Iran'

In [27]:
```
# Run this cell without changes
def normalize_location(country_name):
    """
    Given a country name, return the name that the
    country uses when playing in the FIFA World Cup
    """
    name_sub_dict = {
        "Russian Federation": "Russia",
        "Egypt, Arab Rep.": "Egypt",
        "Iran, Islamic Rep.": "Iran",
        "Korea, Rep.": "South Korea",
        "United Kingdom": "England"
    }
    # The .get method returns the corresponding value from
    # the dict if present, otherwise returns country_name
    return name_sub_dict.get(country_name, country_name)

# Example where normalized location is different
print(normalize_location("Russian Federation"))
# Example where normalized location is the same
print(normalize_location("Argentina"))
```

```
Russia
Argentina
```

In [43]:
```
population_data_filtered = []

for record in population_data:
    # Get normalized country name
    normalized_country = normalize_location(record['Country Name'])

    # Add record to population_data_filtered if relevant
    if normalized_country in teams and record['Year'] == '2018':
        # Replace the country name in the record
        record['Country Name'] = normalized_country
        # Append to list
        population_data_filtered.append(record)

len(population_data_filtered)
```

Out[43]: 32

In [48]:
```
print(population_data_filtered[0]["Value"])
```

```
44494502
```

In [51]:
```
# Replace None with appropriate code

    # Convert the population value from str to int


# Look at the last record to make sure the population
# value is an int
x1 = population_data_filtered[-1]
print(x1)
```

```
{'Country Name': 'Uruguay', 'Country Code': 'URY', 'Year': '2018', 'Value': 3449299}
```

In [52]:
```
# Replace None with appropriate code
#for record in population_data_filtered:
    # Convert the population value from str to int
    #record['Value'] = int(record['Value'])

# Look at the last record to make sure the population
# value is an int
population_data_filtered[-1]
```

Out[52]:
```
{'Country Name': 'Uruguay',
 'Country Code': 'URY',
 'Year': '2018',
 'Value': 3449299}
```

In [53]:
```
# Run this cell without changes
assert type(population_data_filtered[-1]["Value"]) == int
```

In [55]:
```
# Run this cell without changes
combined_data
```

Out[55]:
```
{'Argentina': {'wins': 1},
 'Australia': {'wins': 0},
 'Belgium': {'wins': 6},
 'Brazil': {'wins': 3},
 'Colombia': {'wins': 2},
 'Costa Rica': {'wins': 0},
 'Croatia': {'wins': 3},
 'Denmark': {'wins': 1},
 'Egypt': {'wins': 0},
 'England': {'wins': 3},
 'France': {'wins': 6},
 'Germany': {'wins': 1},
 'Iceland': {'wins': 0},
 'Iran': {'wins': 1},
 'Japan': {'wins': 1},
 'Mexico': {'wins': 2},
 'Morocco': {'wins': 0},
 'Nigeria': {'wins': 1},
 'Panama': {'wins': 0},
 'Peru': {'wins': 1},
 'Poland': {'wins': 1},
 'Portugal': {'wins': 1},
 'Russia': {'wins': 2},
 'Saudi Arabia': {'wins': 1},
 'Senegal': {'wins': 1},
 'Serbia': {'wins': 1},
 'South Korea': {'wins': 1},
 'Spain': {'wins': 1},
 'Sweden': {'wins': 3},
 'Switzerland': {'wins': 1},
 'Tunisia': {'wins': 1},
 'Uruguay': {'wins': 4}}
```

In [63]:
```
for record in population_data_filtered:
    # Extract the country name from the record
    country = record["Country Name"]
    # Extract the population value from the record
    population = int(record["Value"])

    # Add population to the appropriate country in combined_data
    for team in teams:
        if normalize_location(country) == team:
            combined_data[team]["population"] = population
combined_data
```

Out[63]:
```
{'Argentina': {'wins': 1, 'population': 44494502},
 'Australia': {'wins': 0, 'population': 24982688},
 'Belgium': {'wins': 6, 'population': 11433256},
 'Brazil': {'wins': 3, 'population': 209469333},
 'Colombia': {'wins': 2, 'population': 49648685},
 'Costa Rica': {'wins': 0, 'population': 4999441},
 'Croatia': {'wins': 3, 'population': 4087843},
 'Denmark': {'wins': 1, 'population': 5793636},
 'Egypt': {'wins': 0, 'population': 98423595},
 'England': {'wins': 3, 'population': 66460344},
 'France': {'wins': 6, 'population': 66977107},
 'Germany': {'wins': 1, 'population': 82905782},
 'Iceland': {'wins': 0, 'population': 352721},
 'Iran': {'wins': 1, 'population': 81800269},
 'Japan': {'wins': 1, 'population': 126529100},
 'Mexico': {'wins': 2, 'population': 126190788},
 'Morocco': {'wins': 0, 'population': 36029138},
 'Nigeria': {'wins': 1, 'population': 195874740},
 'Panama': {'wins': 0, 'population': 4176873},
 'Peru': {'wins': 1, 'population': 31989256},
 'Poland': {'wins': 1, 'population': 37974750},
 'Portugal': {'wins': 1, 'population': 10283822},
 'Russia': {'wins': 2, 'population': 144478050},
 'Saudi Arabia': {'wins': 1, 'population': 33699947},
 'Senegal': {'wins': 1, 'population': 15854360},
 'Serbia': {'wins': 1, 'population': 6982604},
 'South Korea': {'wins': 1, 'population': 51606633},
 'Spain': {'wins': 1, 'population': 46796540},
 'Sweden': {'wins': 3, 'population': 10175214},
 'Switzerland': {'wins': 1, 'population': 8513227},
 'Tunisia': {'wins': 1, 'population': 11565204},
 'Uruguay': {'wins': 4, 'population': 3449299}}
```

In [64]:
```
# Run this cell without changes
assert type(combined_data["Uruguay"]) == dict
assert type(combined_data["Uruguay"]["population"]) == int
```

In [65]:
```
# Run this cell without changes
populations = [val["population"] for val in combined_data.values()]

print("Mean population:", np.mean(populations))
print("Median population:", np.median(populations))
print("Standard deviation of population:", np.std(populations))
```

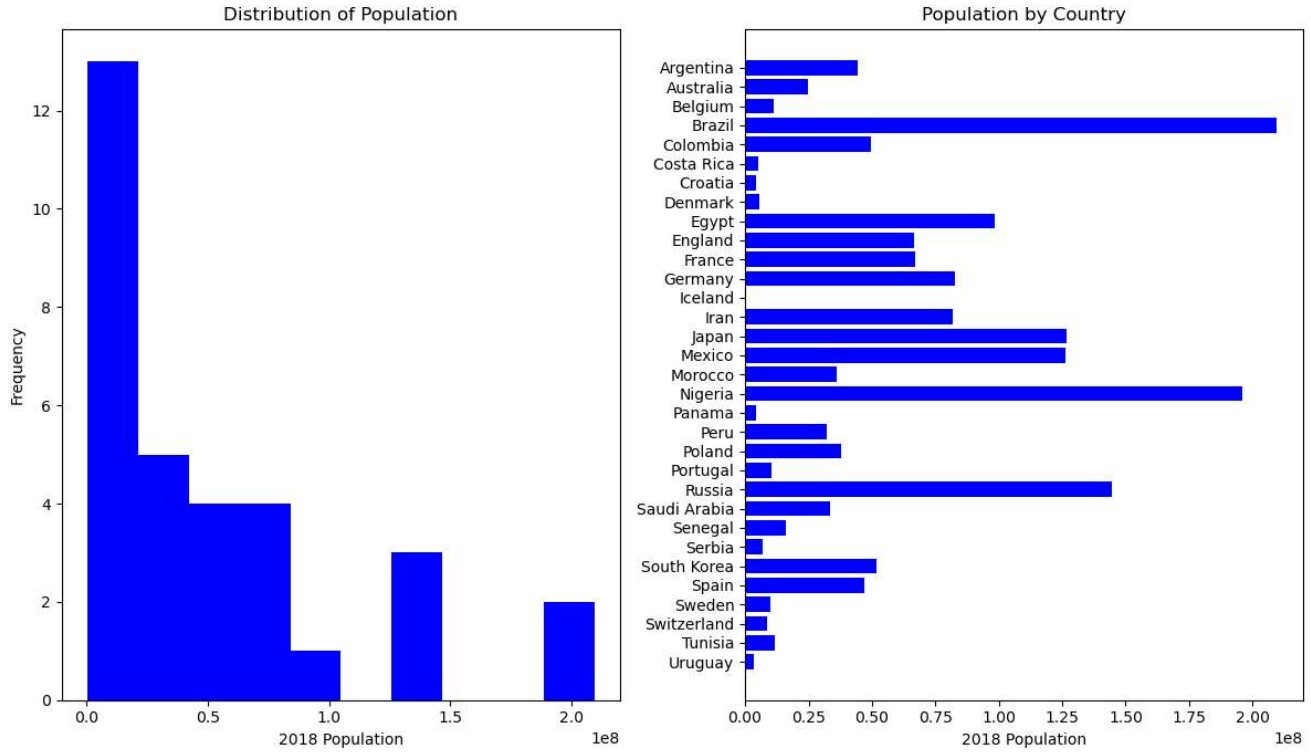
```
Mean population: 51687460.84375
Median population: 34864542.5
Standard deviation of population: 55195121.60871871
```

In [66]:
```
# Run this cell without changes

# Set up figure and axes
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 7))
fig.set_tight_layout(True)

# Histogram of Populations and Frequencies
ax1.hist(x=populations, color="blue")
ax1.set_xlabel("2018 Population")
ax1.set_ylabel("Frequency")
ax1.set_title("Distribution of Population")

# Horizontal Bar Graph of Population by Country
ax2.barh(teams[::-1], populations[::-1], color="blue")
ax2.set_xlabel("2018 Population")
ax2.set_title("Population by Country");
```



In [67]:
```
# Run this cell without changes
np.corrcoef(wins, populations)[0][1]
```

Out[67]:
```
0.07592816849178588
```

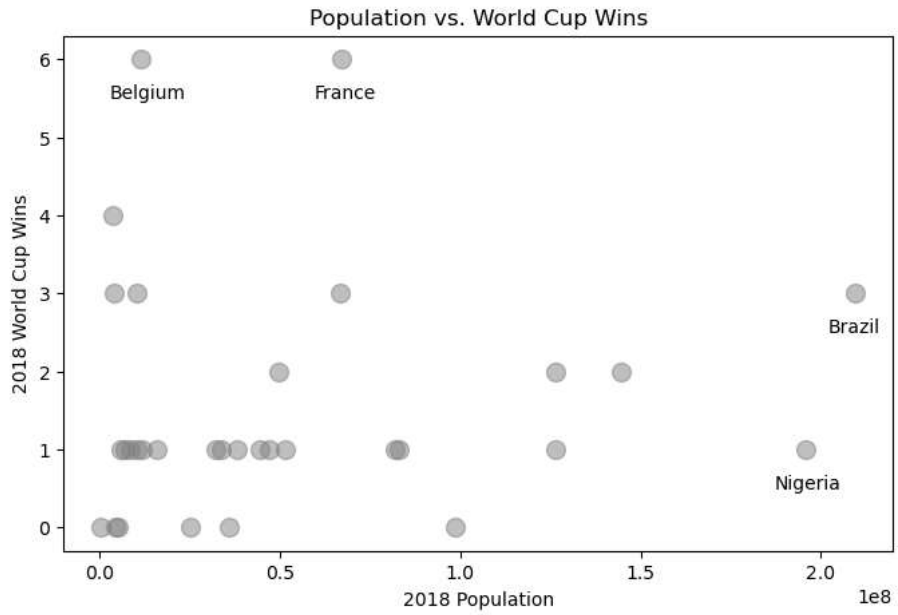
In [ ]:
```
n conclude that there is a corelation between the population of the countries and there wins but not a very strong correlation.
```

In [68]:
```
# Run this cell without changes

# Set up figure
fig, ax = plt.subplots(figsize=(8, 5))

# Basic scatter plot
ax.scatter(
    x=populations,
    y=wins,
    color="gray", alpha=0.5, s=100
)
ax.set_xlabel("2018 Population")
ax.set_ylabel("2018 World Cup Wins")
ax.set_title("Population vs. World Cup Wins")

# Add annotations for specific points of interest
highlighted_points = {
    "Belgium": 2, # Numbers are the index of that
    "Brazil": 3,  # country in populations & wins
    "France": 10,
    "Nigeria": 17
}
for country, index in highlighted_points.items():
    # Get x and y position of data point
    x = populations[index]
    y = wins[index]
    # Move each point slightly down and to the left
    # (numbers were chosen by manually tweaking)
    xtext = x - (1.25e6 * len(country))
    ytext = y - 0.5
    # Annotate with relevant arguments
    ax.annotate(
        text=country,
        xy=(x, y),
        xytext=(xtext, ytext)
    )
```



The weak correlation suggests that there is little to no relationship between the population of a country and their performance in the 2018 FIFA World Cup. In other words, a country's population size does not appear to be a good predictor of how well their national soccer team performs in the World Cup.

There could be a number of reasons for this, including differences in soccer culture and infrastructure, variation in access to resources such as training facilities and coaching, and the specific composition of each team (i.e. the skill level of individual players). Other factors such as luck and chance could also play a role in determining the outcome of individual games and the success of teams overall.