DS1EDP: Homework 06 – Solutions

1. Probability

Question 1:

```
ten_sixes = 2
```

Question 2:

```
five_or_less = 3
```

Question 3:

lottery = 3

Question 4:

list_chances = 2

2. Monkeys Typing Shakespeare

Question 1:

```
datascience_chance = (1/26)**11
```

Question 2:

```
import string

letters = list(string.ascii_lowercase)

def simulate_key_strike():
    return np.random.choice(letters)
```

Question 3:

```
def simulate_several_key_strikes(num_strikes):
     key_strikes_array = make_array()
     for i in np.arange(num_strikes):
          key_strikes_array = np.append(key_strikes_array,
               simulate_key_strike())
     return "".join(key_strikes_array)
```

Question 4:

```
trials = 1000
counter = 0
for i in np.arange(trials):
     x = simulate_several_key_strikes(11)
     if x == "datascience":
          counter += 1
datascience_proportion = counter / trials
```

Question 5:

```
good_approach = False
```

Question 6:

```
1 - ((25 / 26) ** 11)
```

Question 7:

more_effective = True

3. Sampling Basketball Players

Question 1:

full_data = player_data.join("Name", salary_data,

"PlayerName").drop("PlayerName")

Question 2:

values = full_data.column("Points") / (full_data.column("Salary") / 1000)

full_data_with_value_solution = full_data_solution.with_column("Value",

values)

4. Earthquakes

Question 1:

better_sample = 2

Question 2:

representative_sample = earthquakes.sample(500, with_replacement=False)

```
representative_mean = np.mean(representative_sample.column('mag'))
```

Question 3:

```
maximums = make_array()
for i in np.arange(5000):
    m = max(earthquakes.sample(500,
with_replacement=False).column('mag'))
    maximums = np.append(maximums, m)
```

Question 4:

```
strongest_earthquake_magnitude = max(earthquakes.column("mag"))
```

Question 5:

```
determining_max_by_sampling = “No”
```

5. Assessing Gary's Models

Question 1:

```
observated_head_probability = 0.1
expected_head_probability = 0.5
coin_model_probabilities = make_array(0.5, 0.5)
```

Question 2:

```
def test_statistic(expected_p, observed_p):
    return abs(expected_p -  observed_p)
```

Question 3:

```
def coin_simulation_and_statistic(expected_probability, model_probabilities):
    s = sample_proportions(10, model_probabilities)
```

```
    return test_statistic(expected_probability, s.item(0))
```

Question 4:

```
coin_statistics = make_array()

repetitions = 5000

for i in np.arange(repetitions):

    ts = coin_simulation_and_statistic(expected_head_probability,
coin_model_probabilities)

    coin_statistics = np.append(coin_statistics, ts)
```

Question 5:

```
p_value = np.count_nonzero(coin_statistics >= observed_test_statistic)

null_hypothesis_rejected = True
```