

Simulate Multiple Walks

The screenshot shows a web browser with multiple tabs. The active tab is 'campus.datacamp.com/courses/intermediate-python/case-study-hacker-statistics?ex=10'. The page displays an exercise titled 'Simulate multiple walks' with instructions and a code editor. The instructions state: 'A single random walk is one thing, but that doesn't tell you if you have a good chance at winning the bet. To get an idea about how big your chances are of reaching 60 steps, you can repeatedly simulate the random walk and collect the results. That's exactly what you'll do in this exercise. The sample code already sets you off in the right direction. Another for loop is wrapped around the code you already wrote. It's up to you to add some bits and pieces to make sure all of the results are recorded correctly. Note: Don't change anything about the initialization of all_walks that is given. Setting any number inside the list will cause the exercise to crash!'. The code editor shows a Python script with the following content:

```
1 # NumPy is imported; seed is set
2
3 # Initialize all_walks (don't change this line)
4 all_walks = []
5
6 # Simulate random walk five times
7 for i in range(5):
8
9     # Code from before
10    random_walk = [0]
11    for x in range(100):
12        step = random_walk[-1]
13        dice = np.random.randint(1,7)
14
15        if dice <= 2:
16            step = max(0, step - 1)
17        elif dice <= 5:
18            step = step + 1
19        else:
20            step = step + np.random.randint(1,7)
21        random_walk.append(step)
22
23    # Append random_walk to all_walks
```

The IPython shell shows the prompt 'In [1]:'.

Below is the exercise on 'Simulate Multiple Walks' from the Python course. The image includes the instructions, code, and task details.

Solution:

```
# NumPy is imported; seed is set
import numpy as np
np.random.seed(123)
```

```
# Initialize all_walks (don't change this line)
all_walks = []
```

```
# Simulate random walk five times
for i in range(5): # Loop runs 5 times
    # Code from before
    random_walk = [0]
    for x in range(100): # Loop runs 100 times for each walk
        # Set step: last element in random_walk
        step = random_walk[-1]
```

```
    # Roll the dice
```

```

dice = np.random.randint(1, 7)

# Determine next step using max to prevent step from going below 0
if dice <= 2:
    step = max(0, step - 1) # Move down but ensure step doesn't go
below 0
elif dice <= 5:
    step = step + 1 # Move up
else:
    step = step + np.random.randint(1, 7) # Move up by a random value

# Append next_step to random_walk
random_walk.append(step)

# Append random_walk to all_walks
all_walks.append(random_walk)

# Print all_walks
print(all_walks)

```

Explanation:

1. Import numpy as np and set the random seed using np.random.seed(123) to ensure reproducibility.
2. Initialize all_walks as an empty list to store the results of multiple random walks.
3. Use a for loop to repeat the random walk simulation 5 times:
 - For each walk, initialize random_walk as a list containing the first step, 0.
 - Use a nested for loop that runs 100 times to simulate the steps of each random walk:
 - Get the current step as the last element of the random_walk list using random_walk[-1].
 - Roll the dice using np.random.randint(1, 7) to generate a random integer between 1 and 6.
 - Use max() to ensure that step doesn't go below 0 when dice <= 2. This prevents negative steps:
 - If dice is 1 or 2, use max(0, step - 1) to decrease step by 1 but not below 0.

- If dice is 3, 4, or 5, increase step by 1.
 - If dice is 6, roll the dice again and add the new result to step.
 - Append the updated step to the random_walk list.
 - Append the completed random_walk to the all_walks list.
4. Print all_walks to see the results of the 5 random walks.