

Visualize the Walk

The screenshot shows a web browser window with the URL `campus.datacamp.com/courses/intermediate-python/case-study-hacker-statistics?ex=8`. The page is titled 'Visualize the walk' and is part of a course on 'Intermediate Python'. The exercise instructions are as follows:

Visualize the walk
Let's visualize this random walk! Remember how you could use `matplotlib` to build a line plot?

```
import matplotlib.pyplot as plt
plt.plot(x, y)
plt.show()
```

The first list you pass is mapped onto the `x` axis and the second list is mapped onto the `y` axis.

If you pass only one argument, Python will know what to do and will use the index of the list to map onto the `x` axis, and the values in the list onto the `y` axis.

Instructions (100 XP)

Add some lines of code after the `for` loop:

- Import `matplotlib.pyplot` as `plt`.
- Use `plt.plot()` to plot `random_walk`.
- Finish off with `plt.show()` to actually display the plot.

[Take Hint \(-30 XP\)](#)

The code editor shows the following solution:

```
1 # NumPy is imported, seed is set
2
3 # Initialization
4 random_walk = [0]
5
6 for x in range(100):
7     step = random_walk[-1]
8     dice = np.random.randint(1,7)
9
10    if dice <= 2:
11        step = max(0, step - 1)
12    elif dice <= 5:
13        step = step + 1
14    else:
15        step = step + np.random.randint(1,7)
16
17    random_walk.append(step)
18
19 # Import matplotlib.pyplot as plt
20
21 # Plot random_walk
22
```

The IPython Shell shows the prompt `In [1]:`.

Below is the exercise on 'Visualize the Walk' 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 random_walk
random_walk = [0]
```

```
# Complete the for loop
for x in range(100): # Loop runs 100 times
    # 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)

# Import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Plot random_walk
plt.plot(random_walk)

# Show the plot
plt.show()

```

Explanation:

1. Import numpy as np and set the random seed using np.random.seed(123) to ensure reproducibility.

2. Initialize random_walk as a list containing the first step, 0.

3. Use a for loop that runs 100 times to simulate the steps of the 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.

4. Append the updated step to the `random_walk` list.
5. Import `matplotlib.pyplot` as `plt` to visualize the random walk.
6. Use `plt.plot(random_walk)` to plot the random walk on the y-axis, with the indices of `random_walk` automatically plotted on the x-axis.
7. Call `plt.show()` to display the plot.