

How Low Can You Go?

The screenshot shows a web browser with multiple tabs. The active tab is 'campus.datacamp.com/courses/intermediate-python/case-study-hacker-statistics?ex=7'. The page displays an exercise titled 'How low can you go?' with instructions and a code editor. The code editor shows a solution for the exercise, which involves simulating a random walk and ensuring it doesn't go below zero. The IPython shell is also visible at the bottom.

Exercise: How low can you go?

Things are shaping up nicely! You already have code that calculates your location in the Empire State Building after 100 dice throws. However, there's something we haven't thought about - you can't go below 0!

A typical way to solve problems like this is by using `max()`. If you pass `max()` two arguments, the biggest one gets returned. For example, to make sure that a variable `x` never goes below 10 when you decrease it, you can use:

```
x = max(10, x - 1)
```

Instructions (100 XP)

- Use `max()` in a similar way to make sure that `step` doesn't go below zero if `dice <= 2`.
- Hit **Submit Answer** and check the contents of `random_walk`.

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

Code Editor (script.py):

```
1 # NumPy is imported, seed is set
2
3 # Initialize random_walk
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        # Replace below: use max to make sure step can't go below 0
12        step = step - 1
13    elif dice <= 5:
14        step = step + 1
15    else:
16        step = step + np.random.randint(1,7)
17
18    random_walk.append(step)
19
20 print(random_walk)
```

IPython Shell:

```
In [1]:
```

Below is the exercise on 'How Low Can You Go?' 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)

# Print out random_walk
print(random_walk)

```

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. Print the final random_walk list after completing the loop.