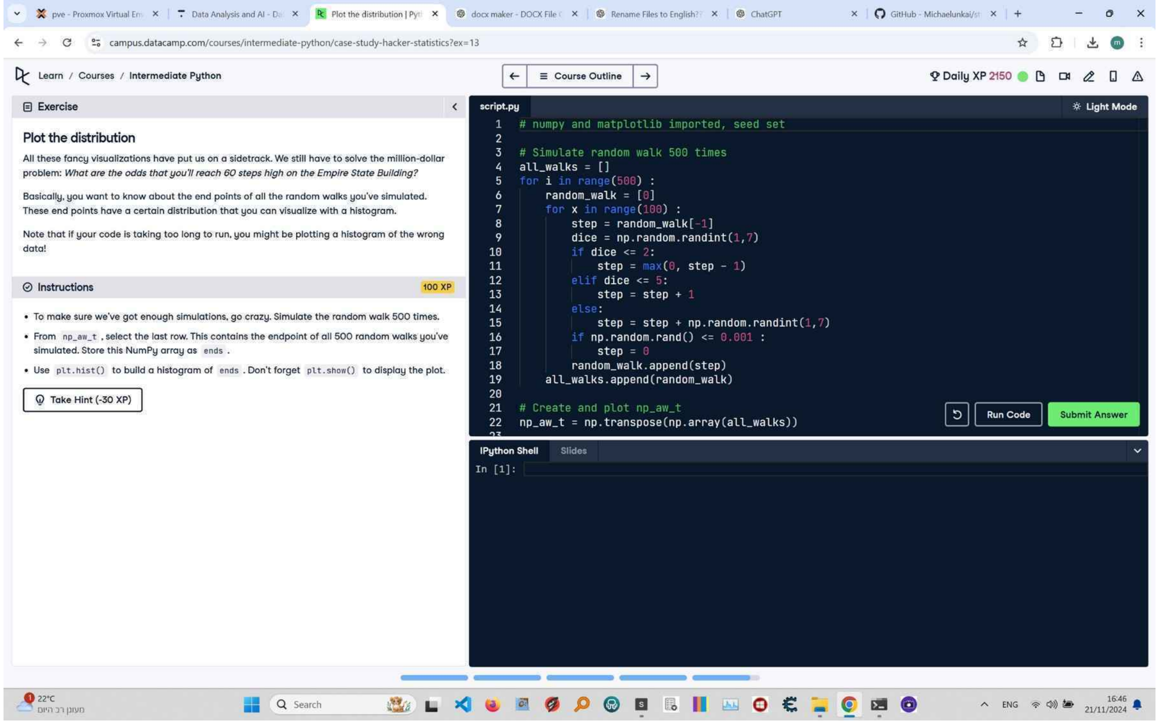
Plot the Distribution - Corrected Solution



Plot the distribution

All these fancy visualizations have put us on a sidetrack. We still have to solve the million-dollar problem: *What are the odds that you'll reach 60 steps high on the Empire State Building?*

Basically, you want to know about the end points of all the random walks you've simulated. These end points have a certain distribution that you can visualize with a histogram.

Note that if your code is taking too long to run, you might be plotting a histogram of the wrong data!

Instructions 100 XP

- To make sure we've got enough simulations, go crazy. Simulate the random walk 500 times.
- From `np_aw_t` , select the last row. This contains the endpoint of all 500 random walks you've simulated. Store this NumPy array as `ends` .
- Use `plt.hist()` to build a histogram of `ends` . Don't forget `plt.show()` to display the plot.

Take Hint (-30 XP)

```
# numpy and matplotlib imported, seed set

# Simulate random walk 500 times
all_walks = []
for i in range(500) :
    random_walk = [0]
    for x in range(100) :
        step = random_walk[-1]
        dice = np.random.randint(1,7)
        if dice <= 2:
            step = max(0, step - 1)
        elif dice <= 5:
            step = step + 1
        else:
            step = step + np.random.randint(1,7)
        if np.random.rand() <= 0.001 :
            step = 0
        random_walk.append(step)
    all_walks.append(random_walk)

# Create and plot np_aw_t
np_aw_t = np.transpose(np.array(all_walks))
```

```
In [1]:
```

Below is the corrected solution for the 'Plot the Distribution' exercise from the Python course. The image includes the instructions, code, and task details.

Corrected Solution:

```
# numpy and matplotlib imported, seed set
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(123)

# Simulate random walk 500 times
all_walks = []
for i in range(500):  # Loop runs 500 times
    random_walk = [0]
    for x in range(100):  # Loop runs 100 times for each walk
        step = random_walk[-1]
        dice = np.random.randint(1, 7)
        if dice <= 2:
            step = max(0, step - 1)
        elif dice <= 5:
            step = step + 1
        else:
            step = step + np.random.randint(1, 7)
        if np.random.rand() <= 0.001:  # Introduce clumsiness
            step = 0
        random_walk.append(step)
    all_walks.append(random_walk)

# Create and plot np_aw_t
np_aw_t = np.transpose(np.array(all_walks))

# Select last row from np_aw_t: ends
ends = np_aw_t[-1]

# Plot histogram of ends and display the plot
plt.hist(ends)
plt.show()
```

Explanation:

1. Import numpy as np and matplotlib.pyplot as plt, and set the random seed using np.random.seed(123).

2. Simulate 500 random walks using an outer loop. Each walk consists of 100 steps simulated by an inner loop.

3. For each step, determine the next step based on the dice roll, ensuring steps don't go below 0.

4. Introduce a 'clumsiness' factor by resetting the step to 0 if np.random.rand() is less than or equal to 0.001.

5. Append the final step of each walk to all_walks.

6. Convert all_walks to a NumPy array, transpose it, and store the result in np_aw_t.

7. Extract the last row from np_aw_t to analyze the end points of the walks and store it in ends.

8. Plot a histogram of the values in ends using plt.hist(ends) and display it with plt.show().