M5-07: Z Tests in Python

Part of the "Polling, Confidence Intervals, and the Normal Distribution" Learning Badge

Video Walkthrough: https://discovery.cs.illinois.edu/m5-07/

Running Z Tests in Python

As you grow as a Data Scientist, you will discover new methods and techniques you may have not known about before. For example, you can write a function to calculate a Z test yourself:

Python:

```python
# Step 1+2: Find test statistics
# x, a list of surveyed data results
# value, mean value under the null hypothesis (H_0)
def my_ztest(x, value):

  # Step 3: Find test statistics
  EV = sum(x) / len(x)
  SD = stat.stdev(x)
  SE = SD / ( len(x)**0.5 )
  z = (val - EV) / SE
  print('z value: ' + str(z))

  # Step 4: Find the p-value
  p = norm.cdf(z)

  # Step 5: Return the conclusion
  return 1-p
```

However, we'll find that Python is an amazing language because so many useful functions are already available. When looking for a useful function, the best way to find a useful search result is to **(1):** search with the programming language name first, **(2):** then search for the concept you're looking for:

Search Query:	`python z test`

The search results will change from time to time, but you'll almost certainly find technical documentation for a library or an example code to use. Here's the documentation I found:

statsmodels.stats.weightstats.ztest(x1, x2=None, value=0, alternative='two-sided', usevar='pooled', ddof=1.0)

Test for mean based on the normal distribution, one or two samples

In the case of two samples, the samples are assumed to be independent.

Source: https://www.statsmodels.org/stable/generated/statsmodels.stats.weightstats.ztest.html

M5-07: Z Tests in Python

Part of the "Polling, Confidence Intervals, and the Normal Distribution" Learning Badge

Video Walkthrough: https://discovery.cs.illinois.edu/m5-07/

Using Python Libraries

For Python libraries that are widely available, you can simply import them without needing to do anything else! For the ztest, we saw that the full name for the Python function to do a ztest was:

```
statsmodels.stats.weightstats.ztest(...)
```

This function name is the **fully qualified function name** -- referencing both the library where it is found and the name of the function.

- Everything that appears **after the last dot** (ex: `ztest(...)`) is the function itself.
- Everything that appears **before the last dot** refers to where Python can find the function.

To import a specific function in Python, we use the syntax:

General Import Format:	`from <location> import <function name>`

Specific to the **`ztest`** function from the statsmodel library:

```
from statsmodels.stats.weightstats import ztest
```

Once the **`ztest`** function is imported, we can use it as described.

Puzzle #1: Simulate 100 rolls of an unfair die, that is 3x more likely to roll a 6 than any other roll:

Python:	

Puzzle #2: Use ztest to find if our dice rolls were likely to be from a fair die?

Python:	