Solutions 1

Jumping Rivers

Lists

The following code will import the package associated with this course as `jr` and assign a variable which contains a series of random numbers.

```
import jrpyml as jr
x1 = jr.get_numeric_list()
```

Using this list:

1. How many elements are in `x1`?

   ```
   print(len(x1))
   ```

   ```
   ## 55748
   ```

2. What is the 55^{th} element of `x1`? Remember, indexing starts at 0

   ```
   print(x1[54])
   ```

   ```
   ## -12.85
   ```

3. What is the final value of `x1`?

   ```
   print(x1[-1])
   ```

   ```
   ## -30.42
   ```

4. What is the 50^{th} smallest values in `x1`?

   ```
   x1.sort()
   print(x1[49])
   ```

   ```
   ## -39.87
   ```

5. Get a fresh copy of the random numbers with `x1 = jr.get_numeric_list()`. The `sum()` function can be used on a list of numbers to calculate the total. What is the sum of the first 5 values?

   ```
   x1 = jr.get_numeric_list()
   print(sum(x1[:5]))
   ```

   ```
   ## 168.59
   ```

6. What is the average value of the list? Note that there is no `mean` function for lists. Mean is total sum divided by number of values.

   ```
   print(sum(x1)/len(x1))
   ```

   ```
   ## 24.736813876730785
   ```

Numpy arrays

For mathematics and statistics **numpy** provides a more convenient data structure, the `array`. For this section import the numpy library and get a new copy of the random list:

```
import numpy as np
x1 = jr.get_numeric_list()
```

1. Convert your list to a **numpy** array.

```
x1_array = np.array(x1)
```

2. Calculate the mean, median and standard deviation of the array

```
x1_mean = np.mean(x1_array)
x1_median = np.median(x1_array)
x1_std = np.std(x1_array)
print(x1_mean)
```

```
## 24.736813876731006
```

```
print(x1_median)
```

```
## 24.81
```

```
print(x1_std)
```

```
# the std() function returns population standard deviation rather
# than sample standard deviation,
# to get sample standard deviation
# x1_sample_std = np.std(x1_array, ddof = 1)
```

```
## 37.58518023172968
```

3. By default the answers to the previous question have a lot of decimal places so look a bit messy, round them to 2 decimal places. Hint: look at the `round()` function from **numpy**.

```
print(round(x1_mean, 2))
```

```
## 24.74
```

```
print(round(x1_std, 2))
```

```
## 37.59
```