CSCI 1010 Class 1

Profs. Michael Linderman and Phil Chodrow
Department of Computer Science
Middlebury College

```
data = np.array([1.0, 2.0, 3.0, 4.0])
math.sqrt(np.sum(np.power(data - np.mean(data), 2))/(len(data) - 1))
```

L

$$2.5 \qquad 1$$

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} - \begin{bmatrix}2.5\\2.5\\2.5\\2.5\end{bmatrix} \qquad \begin{matrix}-1.5^2\\-0.5^2\\0.5^2\\1.5^2\end{matrix}$$

$$2.25 + 0.25 + 0.25 + 2.25 \qquad 5$$

$$\sqrt{\frac{5}{4-1}}$$

Slide 1 Notes

This code performs the following operations:

1. Creates a 1-D array from a list.
2. Performs a "reduction", computing the mean, to produce the scalar 2.5.
3. Performs an element-wise subtraction to compute the difference from the mean. Note that the scalar argument, the mean, is "broadcasted" to be the same size as the vector.

$$\begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{bmatrix} - \begin{bmatrix} 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \end{bmatrix}$$

4. Performs an element-wise "squaring" via the `**` operator.

$$\begin{bmatrix} -1.5^2 \\ -0.5^2 \\ 0.5^2 \\ 1.5^2 \end{bmatrix}$$

5. Performs a sum reduction of the intermediate vector, producing the scalar 5.

$$2.25 + 0.25 + 0.25 + 2.25$$

6. Performs the division and square root operations over scalar values.

$$\sqrt{\frac{5}{4-1}}$$

```python
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
x = 3 * b + a
```

After above the code executes what is the value of `x`?

a. `13`

b. `np.array([13, 17, 21])`

c. `np.array([15, 21, 27])`

d. `np.array([7, 7, 9])`

Slide 2 Notes

Answer: B

3*b is np.array([12, 15, 18]) and the addition is element-wise so the result is np.array([13, 17, 21])

```python
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
x = np.sum(np.power(b-a, 2))
```

After above the code executes what is the value of `x`?

a. `13`

b. `21`

c. `27`

d. `np.array([27, 27, 27])`

Slide 3 Notes

Answer: C

b-a is np.array([3, 3, 3]) thus the element-wise power operation produces np.array([9, 9, 9]). The resulting sum of that vector is the scalar 27.

```
returns = np.cumprod(np.random.laplace(mean, scale, 240))
```

Which of the following snippets are equivalent to the above NumPy code? Assume there is a `laplace` function that has the mean and scale as arguments and returns a single sample.

a.

```
returns = []
for i in range(240):
  sample = laplace(mean, scale)
  returns.append(sample)
```

b.

```
returns = []
prod = 1.0
for i in range(240):
  sample = laplace(mean, scale)
  prod = prod * sample
  returns.append(prod)
```

c.

```
returns = []
prod = 1.0
for i in range(240):
  sample = laplace(mean, scale)
  returns.append(sample)
  prod = prod * sample
```

d.

```
returns = []
prod = 1.0
for i in range(240):
  sample = laplace(mean, scale)
  returns.append(prod)
  prod = prod * sample
```

Slide 4 Notes

Answer: B

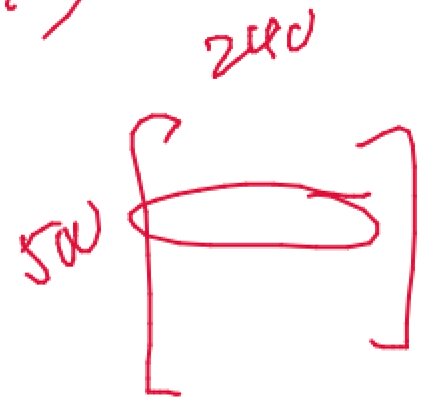
The `cumprod` function computes a *cumulative* product. Answers A and C only record the samples. Answer D has a “off by one”, that is starts with the initial values and doesn’t record the final product.

$$P_{240} = \sum_{m=1}^{240} \$100 \prod_{i=m}^{240} r_i$$

$$100 \sum \prod$$

$$100\left(\prod_{1}^{240} r + \prod_{2}^{240} + \cdots r_{240}\right)$$

$$100\left(\underbrace{r_1 + r_1 \cdot r_2 + \cdots \prod_{1}^{240} r_1}_{\text{cum prod}}\right)$$



Slide 5 Notes

Our first instinct might be to convert the product ($\prod$) and sum ($\sum$) operations into `for` loops, as they are iterative computations over our 240 month time period. As we saw already, we can implement the product operation as a vectorized operation across a 2-D array. Could we do so with the sum as well?

$$\begin{aligned}
P_{240} &= \sum_{m=1}^{240} \$100 \prod_{i=m}^{240} r_i \\
&= \$100 \sum_{m=1}^{240} \prod_{i=m}^{240} r_i \\
&= \$100 (\prod_{i=1}^{240} r_i + \prod_{i=2}^{240} r_i + ... + r_{240}) \\
&= \$100 (r_{240} + (r_{240} \cdot r_{239}) + ... \prod_{i=1}^{240} r_i) \\
&= \$100 (r_1 + (r_1 \cdot r_2) + ... \prod_{i=1}^{240} r_i)
\end{aligned}$$

The sequence $r_1, (r_1 \cdot r_2), ..., \prod_{i=1}^{240} r_i$ is the cumulative product! That is the right hand side of the last expression is the sum of the cumulative product of the monthly returns!