Generating Hero Ability Values

- In most role-playing games, heroes have abilities such as strength, dexterity, intelligence, charism, etc.
- Initial abilities are often measured in ranges like **3 – 18**
- At the beginning of the game, players ***roll dice*** to determine the initial values for each ability
- The higher the value, the more likely the player will succeed while adventuring

	ABILITY SCORE	ABILITY MODIFIER	TEMP SCORE	TEMP MODIFIER
STR				
DEX				
CON				
INT				
WIS				
CHA				

Generating Hero Ability Values

- Two ways of rolling for initial abilities between **3 and 18**

1. Roll one **20**-sided die (**1d20**), but *reroll* if face value is 1, 2, 19, or 20
2. Roll three **6**-sided dice (**3d6**), summing the value of all three dice

- Using the **1d20** method is faster than **3d6**, especially when having to roll for six separate abilities

	ABILITY SCORE	ABILITY MODIFIER	TEMP SCORE	TEMP MODIFIER
STR	17			
DEX	11			
CON	15			
INT	5			
WIS	7			
CHA	3			

Generating Hero Ability Values

- Two ways of rolling for initial abilities between **3 and 18**

1. Roll one **20**-sided die (**1d20**), but *reroll* if face value is 1, 2, 19, or 20
2. Roll three **6**-sided dice (**3d6**), summing the value of all three dice

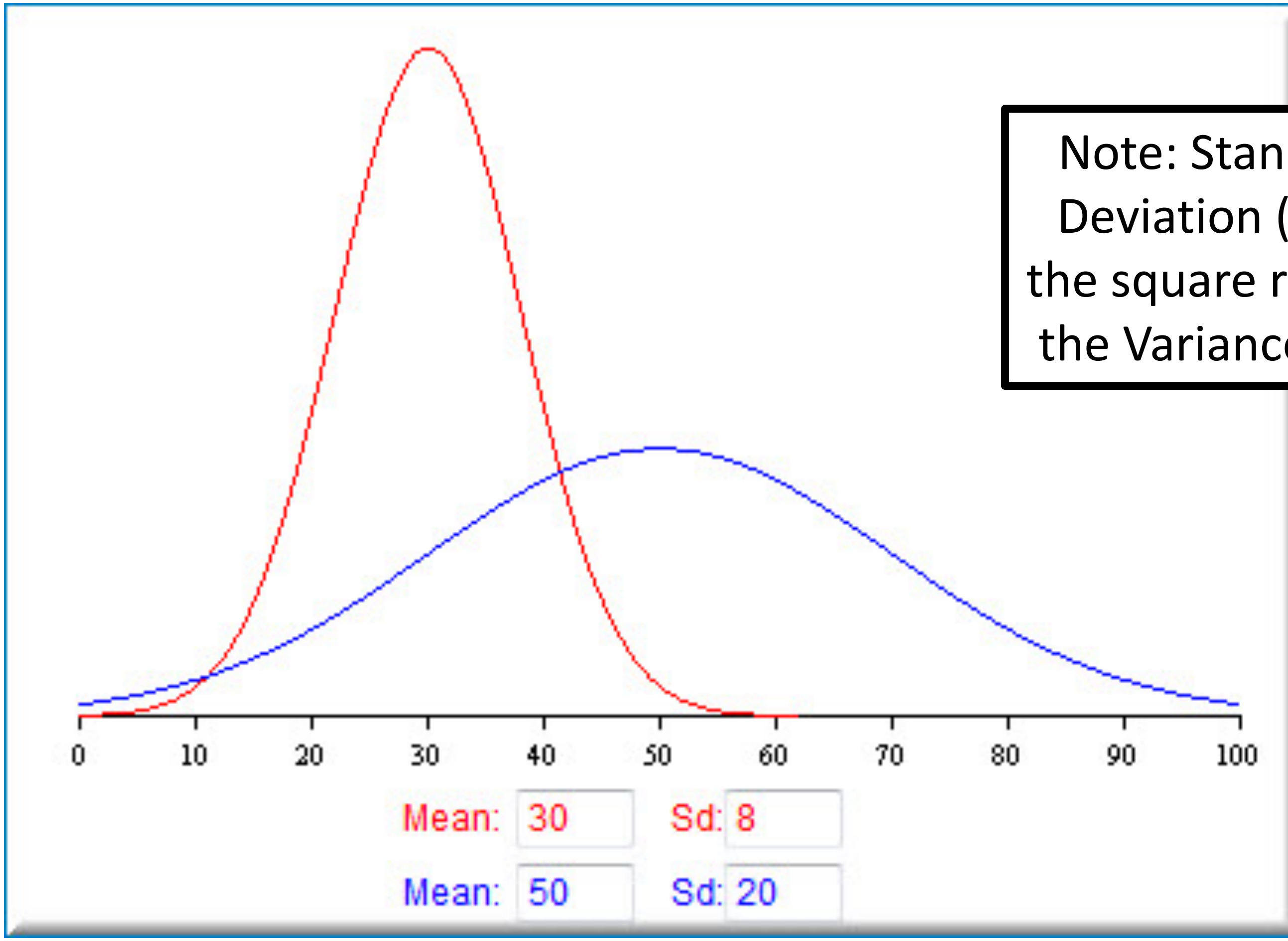
- **Which method would you choose? Why?**

	ABILITY SCORE	ABILITY MODIFIER	TEMP SCORE	TEMP MODIFIER
STR	17			
DEX	11			
CON	15			
INT	5			
WIS	7			
CHA	3			

Mean vs. Variance

- Imagine two different classes take the same test
  - 1st period students score between 50 and 100 with μ = 75
  - 2nd period students score between 70 and 80 with μ = 75
- **Variance** $(\boldsymbol{\sigma}^2)$ is the average "distance" between each number in a set and the mean $(\boldsymbol{\mu})$ of that set
  - 1st period students have a greater **variance** in scores than 2nd period
  - Variance is a measure of **central tendency** – on average how close around the mean do all the numbers fall?
  - For every data point, we sum the ***square*** of the difference between the number and the mean. Then we divide that sum by the total number of data points

Mean vs. Standard Deviation



Note: Standard Deviation (σ) is the square root of the Variance (σ^2)

Mean, Variance, Standard Deviation

Mean

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$x = \{2, 9, 11, 5, 6\} \therefore n = 5$$

$$\sum_{i=1}^{n} x_i = (2 + 9 + 11 + 5 + 6) = 33$$

$$\mu = \frac{33}{5} = 6.6$$

Mean, Variance, Standard Deviation

Variance

Mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \quad \text{Where} \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$x = \{2, 9, 11, 5, 6\} \therefore n = 5, \mu = 6.6$$

$$\sum_{i=1}^{n}(x_i - \mu)^2 = (2 - 6.6)^{\boxed{2}} + (9 - 6.6)^{\boxed{2}} + (11 - 6.6)^{\boxed{2}} + \cdots = 49.2$$

$$\sigma^2 = \frac{49.2}{5} = 9.84$$

Mean, Variance, Standard Deviation

Variance

Mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \quad \text{Where} \quad \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

Standard Deviation

By taking square root of variance, the **standard deviation** has the same <u>units</u> as the *source* data

$$x = \{2, 9, 11, 5, 6\}$$

$$\sigma^2 = 9.84$$

$$\sigma = \sqrt{9.84} \cong 3.13$$

Determine Which Roll Method is Best

- Write a program to generate **1,000,000** hero ability scores, comparing the *mean* and *standard deviation* of the 1d20 versus the 3d6 dice roll methods
- **Which dice roll method would you want to use to generate your hero's abilities – and why?**

Edit hero_abilities.ipynb – **Cells 1…4**

Import packages used in this notebook

```
[1] # Cell 1
    import numpy as np
```
①

Set n (the number of rolls) to be 1,000,000 ②

```
[2] # Cell 2
    n = 1_000_000
```
③

Simulate n number of 3d8 rolls and store each roll in array a ④

```
[3] # Cell 3
    a = np.random.randint(1, 7, n)
    a = a + np.random.randint(1, 7, n)
    a = a + np.random.randint(1, 7, n)
```
⑤

Display the first ten 3d8 rolls ⑥

```
[4] # Cell 4
    print(a[:10])
```
⑦

```
[12  9  9  5 10  7  9 15 10 10]
```
⑧

Edit hero_abilities.ipynb – **Cells 5…8**

Simulate n number of 1d20 rolls and store each roll in array b ①

```
[5] # Cell 5
    b = np.random.randint(3, 19, n)
```
②

Display the first ten 1d20 rolls ③

```
[6] # Cell 6
    print(b[:10])
```
④

```
[16 11 17 16  3  6  5  5  9  8]
```

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2$$

$$\sigma = \sqrt{\sigma^2}$$

Print the mean (μ) and standard deviation (σ) across all 3d8 rolls ⑤

```
[7] # Cell 7
    print(np.mean(a), np.std(a))
```
⑥

```
10.498994 2.9605399982497112
```
⑦

Print the mean (μ) and standard deviation (σ) across all 1d20 rolls ⑧

```
[8] # Cell 8
    print(np.mean(b), np.std(b))
```
⑨

```
10.503175 4.6116914380056 93
```
⑩

Which roll type will most likely give you the highest **average** score across all abilities?

Common Statistics

- Beyond mean, variance, and standard deviation, other **common statistics** can inform us about the "**shape**" (or distribution) of the source data
- Students learn early on about **median** and **mode**
  - The **median** is the "middle" value if the source data is first **sorted** by increasing magnitude
  - The **mode** is the value (or values) that occur most frequently in the source data
- How would you write Python functions *from scratch* to calculate the **mean**, **median**, and **mode** of an array of numbers?

Run common_statistics.ipynb – **Cells 1…3**

Note: You should not edit this file!

Import packages used in this notebook

```
[1] # Cell 1
    import collections
    import numpy as np
```
①

Set the numpy random number `seed` **to 2016** ②

```
[2] # Cell 2
    np.random.seed(2016)
```
③

Create and display an array a of 30 integers where each element is in the range [0,100] ④

```
[3] # Cell 3
    a = np.random.randint(1, 101, 30)
    a
```
⑤

```
array([63, 11, 84, 14, 40, 35, 12, 36, 44, 45, 53, 76, 35, 87, 98, 10, 98,
       19, 30, 61, 99, 73, 28, 31, 57, 38, 37, 43, 38, 25])
```
⑥

Run common_statistics.ipynb – **Cells 1…3**

Define a function `mean(s)` that returns the average of the values in the array s ①
then print the value returned by that function when passed array a ②

```
[4]  # Cell 4
     def mean(s):                        ③
         return np.sum(s) / len(s)       ④

     print(f"{mean(a) = }")              ⑤

     mean(a) = 47.333333333333336        ⑥
```

Define a function `median(s)` that returns the median value of the array s
then print the value returned by that function when passed array a
NOTE: The array s may have an <u>even</u> number of items!

```
[5]  # Cell 5
     def median(s):
         s.sort()
         i = len(s) // 2
         if len(s) % 2 == 1: # s has an odd number of elements
             return s[i]
         else: # s has an even number of elements
             return (s[i - 1] + s[i]) / 2

     print(f"{median(a) = }")

     median(a) = 39.0
```

Run common_statistics.ipynb – **Cells 4…5**

Define a function `mean(s)` that returns the average of the values in the array s
then print the value returned by that function when passed array a

```
[4]  # Cell 4
     def mean(s):
         return np.sum(s) / len(s)

     print(f"{mean(a) = }")
```

```
mean(a) = 47.333333333333336
```

Define a function `median(s)` that returns the median value of the array s ①
then print the value returned by that function when passed array a ②
NOTE: The array s may have an <u>even</u> number of items! ③

```
[5]  # Cell 5
     def median(s):                                              ④
         s.sort()                                                ⑤
         i = len(s) // 2                                         ⑥
         if len(s) % 2 == 1: # s has an odd number of elements   ⑦
             return s[i]
         else: # s has an even number of elements                ⑧
             return (s[i - 1] + s[i]) / 2

     print(f"{median(a) = }")                                    ⑨
```

```
median(a) = 39.0                                                 ⑩
```

Run common_statistics.ipynb – Cell 6

Tally the total occurrences of each number in array a ①

```
[6] # Cell 6
    collections.Counter(a)
```
②

```
Counter({10: 1,
         11: 1,
         12: 1,
         14: 1,
         19: 1,
         25: 1,
         28: 1,
         30: 1,
         31: 1,
         35: 2,
         36: 1,
         37: 1,
         38: 2,
         40: 1,
         43: 1,
         44: 1,
         45: 1,
         53: 1,
         57: 1,
         61: 1,
         63: 1,
         73: 1,
         76: 1,
         84: 1,
         87: 1,
         98: 2,
         99: 1})
```
③

Run common_statistics.ipynb – Cell 6

Define a function `mode(s)` that returns the mode value of the array s ①

then print the value returned by that function when passed array a ②

NOTE: The array s may be <u>multi-modal</u>! ③

```
[7] # Cell 7
    def mode(s):
        counts = collections.Counter(s)
        max_count = max(counts.values())
        if max_count == 1:
            return None
        else:
            return [k for k, v in counts.items() if v == max_count]

    print(f"{mode(a) = }")

    mode(a) = [35, 38, 98]
```

https://realpython.com/list-comprehension-python

Variance of Uniform Distributions

- Your scientist needs a program that can:
  - Generate 15 sets of **random sizes** between **10,000** and **200,000** items
  - Within each set, every item is a random integer chosen within a range between a random **lower limit** and a random **upper limit**
  - The **lower limit** for each set is a random number between **0 and 10,000**
  - The **upper limit** is that set's lower limit **plus** another random number between 0 and 100,000
  - Calculate the mean (μ) and variance (σ^2) for each set's population

$$\boxed{\sigma^2 = \frac{1}{n}\sum_{i=i}^{n}(x_i - \mu)^2 \;\; where \;\; \mu = \frac{1}{n}\sum_{i=1}^{n} x_i}$$

Variance of Uniform Distributions

- The research goal is to determine if a magic number hides within **all** *uniform* random number distributions
  - Calculate and display this "constant" for each set:

$$\textit{Magic Number} = \frac{(upperLimit - lowerLimit)^2}{variance}$$

  - Is this number the same for ALL uniform distributions?
- Can we use this value to test if dice are loaded?

Run uniform_variance.ipynb – **Cells 1…2**

Import packages used in this notebook

```
[1] # Cell 1
    import numpy as np                                    ①
```

Define a function `run_trial(trial_num)` that: ②

1. Creates a random array ③
2. Computes the magic number $\frac{(upper\ limit - lower\ limit)^2}{\sigma^2}$ ④
3. Prints the various statistics for this trial ⑤

```
[2] # Cell 2
    def run_trial(trial_num):                             ⑥
        lower_limit = np.random.randint(10_001)
        upper_limit = lower_limit + np.random.randint(100_001)   ⑦
        size = np.random.randint(10_000, 200_001)
        a = np.random.randint(lower_limit, upper_limit, size)    ⑧
        mean, var = np.mean(a), np.var(a)
        magic = (upper_limit - lower_limit) ** 2 / var           ⑨
        print(f"{trial_num:>8}", end="")
        print(f"{lower_limit:>9,}", end="")
        print(f"{upper_limit:>9,}", end="")                      ⑩
        print(f"{size:>9,}", end="")
        print(f"{mean:>14.3f}", end="")
        print(f"{var:>16.3f}", end="")
        print(f"{magic:>10.3f}")
```

Note: You should not edit this file!

Run uniform_variance.ipynb – **Cell 3**

Print the table headers then run 15 trials of this experiment

```python
# Cell 3
print(f"{'Trial #':>8}", end="")
print(f"{'Lower':>9}", end="")
print(f"{'Upper':>9}", end="")
print(f"{'Size':>9}", end="")
print(f"{'Mean':>14}", end="")
print(f"{'Variance':>16}", end="")
print(f"{'Magic':>10}")

for trial_num in range(1, 16):
    run_trial(trial_num)
```

```
 Trial #    Lower    Upper     Size          Mean        Variance     Magic
       1    3,621   77,012  101,397     40331.583   448358673.446    12.013
       2    1,030   38,670  104,978     19837.612   118274763.322    11.979
       3      910  100,746  161,436     50863.410   832864656.334    11.967
       4    2,740   36,896   44,032     19849.948    97738548.624    11.936
       5    4,947   11,408   87,748      8182.547     3464328.541    12.050
       6    3,931   79,606  114,077     41779.457   479300380.117    11.948
       7    5,298   73,859  116,363     39480.076   391681820.063    12.001
       8    4,955   55,824   35,566     30417.527   217509563.899    11.897
       9    7,415    9,901   81,025      8656.040      514782.757    12.005
      10    2,628   75,904   50,343     39192.092   445470722.786    12.053
      11    6,545   51,823   78,789     29198.462   170540400.849    12.021
      12    4,178   38,195   50,325     21126.543    96317614.595    12.014
      13    4,428   63,085  159,597     33771.320   285995086.252    12.030
      14    2,747   33,008   61,260     17844.770    76358842.198    11.992
      15    6,998   60,707  158,343     33854.692   240563603.967    11.991
```

Variance of Uniform Distributions

Trial #	Lower	Upper	Size	Mean	Variance	Magic
1	2,186	97,609	100,308	50061.375	763204878.817	11.931
2	2,456	41,355	83,467	21981.261	125285980.368	12.077
3	832	18,461	65,817	9648.839	25938232.503	11.982
4	4,233	42,165	31,918	23231.598	119992088.765	11.991
5	8,879	91,012	160,019	49962.086	563505796.451	11.971
6	1,765	87,215	140,124	44436.213	606677745.464	12.036
7	1,549	43,086	23,841	22178.161	143154389.004	12.052
8	8,587	105,157	130,589	56981.826	777105956.238	12.001
9	7,127	89,418	37,812	47946.706	568515233.060	11.911
10	1,265	11,018	102,292	6142.628	7955048.841	11.957
11	6,830	74,990	132,704	40882.409	386369369.576	12.024
12	9,786	27,604	148,185	18702.335	26342315.791	12.052
13	963	10,211	14,035	5572.470	7251379.077	11.794
14	5,717	9,443	23,348	7581.759	1146793.735	12.106
15	2,533	29,988	135,261	16234.108	62987583.045	11.967

- Every set had a different lower and upper limit, size, mean, and variance… yet the magic number was ~ 12 for all of them!
- Why would Mother Nature choose 12 for this magic number? What is so special about 12? Why not pick a nice even 10?
- Boundless natural curiosity is what makes a good scientist…

Variance of Uniform Distributions

$$\boxed{\sigma^2 = \frac{1}{n}\sum_{i=i}^{n}(x_i - \mu)^2 \;\; where \;\; \mu = \frac{1}{n}\sum_{i=1}^{n} x_i}$$

The *expected* value ($\mathbb{E}$) of a random variable $\boldsymbol{X}$ is its <u>mean</u> value (μ)

$$\mathbb{E}(X) = \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Variance (σ^2) is the <u>mean</u> difference *squared* between every $\boldsymbol{X}$ and its $\mathbb{E}(X)$

$$\sigma^2 = \mathbb{E}\left[\left(X - \mathbb{E}(X)\right)^2\right]$$

The *expected* value ($\mathbb{E}$) returns a **constant** value → $\mathbb{E}(X) = \mu$

The *expected* value ($\mathbb{E}$) of a **constant** value returns that same value → $\mathbb{E}(\mu) = \mu$

$$\mathbb{E}\left(\mathbb{E}(X)\right) = \mathbb{E}(X)$$

$$\mathbb{E}\left(\mathbb{E}\left(\mathbb{E}(X)\right)\right) = \mathbb{E}(X)$$

$\mathbb{E}(X)$ is **idempotent**

Variance of Uniform Distributions

$$\boxed{\sigma^2 = \frac{1}{n}\sum_{i=i}^{n}(x_i - \mu)^2 \ \ where \ \ \mu = \frac{1}{n}\sum_{i=1}^{n} x_i}$$

$$\mu = \mathbb{E}(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma^2 = \mathbb{E}\left[\left(X - \mathbb{E}(X)\right)^2\right]$$

$$\mathbb{E}(\mu) = \mu$$

$$\mathbb{E}\big(\mathbb{E}(X)\big) = \mathbb{E}(X)$$

Faster because only <u>one</u> subtraction is required!

$$\sigma^2 = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) - \mu^2$$

$$\sigma^2 = \mathbb{E}\left[\left(X - \mathbb{E}(X)\right)^2\right]$$ FOIL

$$\sigma^2 = \boxed{\mathbb{E}}[\boxed{X^2} - \boxed{2X\mathbb{E}(X)} + \boxed{\mathbb{E}(X)^2}]$$

Note: $\mathbb{E}(x)$ is a distributive linear operator

$$\sigma^2 = \mathbb{E}(X^2) - \mathbb{E}\big(2X\mathbb{E}(X)\big) + \mathbb{E}(\mathbb{E}(X)^2)$$

$$\sigma^2 = \mathbb{E}(X^2) - 2\mathbb{E}(X)\mathbb{E}(X) + \mathbb{E}(X)^2$$

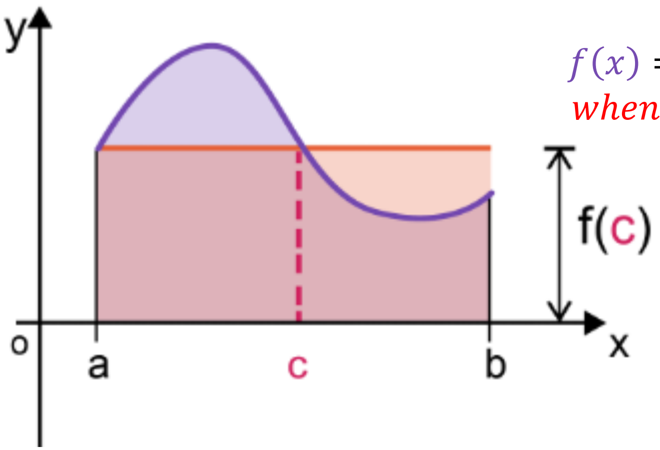
$$\sigma^2 = \mathbb{E}(X^2) - 2\mathbb{E}(X)^2 + \mathbb{E}(X)^2$$

$$\sigma^2 = \mathbb{E}(X^2) - \mathbb{E}(X)^2$$

$$\sigma^2 = \mathbb{E}(X^2) - \mu^2$$

Variance of Uniform Distributions

$f(c)$ = the average value of the function



$f(x) = f(c)$
when $x = c$

Random Variable (Uniform Distribution)

Discrete: $\mathbb{E}(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$

Continuous: $\mathbb{E}(X) = \frac{1}{(b-a)}\int_a^b \boxed{x}\, dx$

Mean Value Theorem (*Integrals*)

$$Area_{red} = Area_{curve}$$

$$Area_{red} = f(c)\times(b-a)$$

$$Area_{curve} = \int_a^b f(x)\, dx$$

$$f(c)\times(b-a) = \int_a^b f(x)\, dx$$

$$f(c) = \frac{1}{(b-a)}\int_a^b \boxed{f(x)}\, dx$$

$$f(c) = \mu = \mathbb{E}(X)$$

Variance of Uniform Distributions

Moment Generating Functions

$$\mathbb{E}(\boxed{X}) = \frac{1}{(b-a)}\int_a^b \boxed{x}\, dx$$

$$\mathbb{E}(\boxed{X^2}) = \frac{1}{(b-a)}\int_a^b \boxed{x^2}\, dx$$

$$\sigma^2 = \mathbb{E}(X^2) - \mu^2$$

$$\mu = \frac{1}{b-a}\int_a^b x\, dx = \frac{1}{b-a}\left(\frac{x^2}{2}\bigg|_a^b\right) = \frac{b+a}{2}$$

Hero Ability Results

```
Mean ability (1d20): 10.49
Mean ability (3d6):  10.50
```

$$\frac{(18+3)}{2} = 10.5$$

$$\mathbb{E}(X^2) = \frac{1}{b-a}\int_a^b x^2\, dx = \frac{1}{b-a}\left(\frac{x^3}{3}\bigg|_a^b\right) = \frac{b^2 + ab + a^2}{3}$$

Variance of Uniform Distributions

Moment Generating Functions

$$\boxed{12 = \frac{(\boldsymbol{upperLimit} - \boldsymbol{lowerLimit})^{\mathbf{2}}}{variance}}$$

$$\mathbb{E}(X) = \frac{1}{(b-a)}\int_a^b x\,dx$$

$$\mathbb{E}(X^2) = \frac{1}{(b-a)}\int_a^b x^2\,dx$$

$$\sigma^2 = \mathbb{E}(X^2) - \mu^2$$

$$\mu = \frac{1}{b-a}\int_a^b x\,dx = \frac{1}{b-a}\left(\frac{x^2}{2}\Big|_a^b\right) = \frac{b+a}{2}$$

$$\mathbb{E}(X^2) = \frac{1}{b-a}\int_a^b x^2\,dx = \frac{1}{b-a}\left(\frac{x^3}{3}\Big|_a^b\right) = \frac{b^2+ab+a^2}{3}$$

This is the **second central moment** of a uniform distribution

$$\sigma^2 = \mathbb{E}(X^2) - \mu^2 = \frac{b^2+ab+a^2}{3} - \left(\frac{b+a}{2}\right)^2 = \boxed{\frac{(b-a)^2}{12}}$$

Variance

Session **05** – Now You Know…

- The **mean** of a set is often <u>not</u> enough of a *meaningful* statistic to describe the **shape** of the distribution – **variance** is a measure of **central tendency**
- All random variable distributions have a 1st and 2nd **moment** which describes the long-term *behavior* of those random numbers
- A perfect **Normal distribution** ensures that **68.26%** of all values fall within **one** (1) standard deviation from the **mean**
  - 99.73% of all values in a perfect normal distribution are within **three (3)** standard deviations from the mean
  - The normal distribution is known as the "**bell curve**"