Sophie's Dice - Documentation

Dice Notation

Sophie's Dice supports standard dice notation and a few other things, here's the full list:

- Basics
- Roll Conditions
  - Keep
  - Keep Lowest
  - Keep Middle
  - Drop Highest
  - Drop Lowest
  - Drop Conditionally
  - Cap/Clamp
  - Value Replacement
  - No Repeats/Unique
  - Explosion
  - Reroll
  - Count
  - Text
  - Redefine
  - Paint
  - Scale
- Combinations
- Multiple Results
- Sub-Expressions
- Math Functions
  - Rounding Functions
  - Game-Specific Functions
  - Check/Compare Functions

- Preset Variable and Constant Values
- User Values
- Roll Condition Parameters

Basics

To roll a dice, enter the dice name. For example:

```
d6
```

Will roll a die with 6 sides. You can do this for any number and if there isn't a premade dice with that many sides, the program will make one and roll that. (if a fair dice can't be made, it will roll a shape whose face values are set randomly)

To roll multiple dice, put a number in front of the dice name:

```
4d8
```

This will roll four eight-sided dice.

You can roll multiple pools of dice in a single expression, and include simple maths operations too, for example:

```
d6 + 4d8 - 3
```

Which will roll both the expressions above at the same time, add them together, and subtract three.

> **Warning:**
> Be careful when rolling large amounts of dice, Sophie's Dice will attempt to roll as many as you ask it to, but some computers/devices will have trouble if there

are a great many dice - and the program may crash. For this reason, it is not advised to roll much more than a hundred or so dice at once.

Roll Conditions

The basics above will suit almost all of your dice needs, but there are more interesting ways of rolling dice! By adding roll conditions to a dice pool (a dice pool is any group of dice in an expression, eg; "4d6 - 2d12" has two pools, a d6 pool and a d12 pool) you can change how that dice pool is treated in the expression.

Keep

This condition will disregard all dice except those which roll the highest values. By default it will keep all but one dice, but you can specify a different amount. For example:

```
2d20K
```

This will roll two twenty-sided dice, and keep the highest of the two values. (this is equivalent to an advantage roll in DnD)

```
5d10K3
```

This will roll five ten-sided dice, and keep only the highest three values.

Keep Lowest

This condition will disregard all dice except those which roll the lowest values. By default it will only disregard one dice, but you can specify a different amount. For example:

```
2d20KL
```

This will roll two twenty-sided dice, and keep the lowest of the two values. (this is equivalent to a disadvantage roll in DnD)

```
5d10KL3
```

This will roll five ten-sided dice, and keep only the lowest three values.

Keep Middle

This condition will disregard all dice except those which roll the middle values. By default it will only disregard one dice, but you can specify a different amount. For example:

```
3d20KM
```

This will roll three twenty-sided dice, and keep the middle of the three values, disregarding the other two.

```
5d10KM2
```

This will roll five ten-sided dice, and keep only the middle two values. (Because there is no clear 'middle two' of five, it is rounded down - disregarding the lowest value and the two highest values).

If you wanted to keep the *upper* middle of such a roll:

```
5d10KMm2
```

Drop Highest

This condition will disregard the dice which roll the highest values. By default it will only disregard one dice, but you can specify a different amount. For example:

```
2d20H
```

This will roll two twenty-sided dice, and disregard the highest of the two values. (this is equivalent to a disadvantage roll in DnD)

```
5d10H3
```

This will roll five ten-sided dice, and disregard the highest three values.

Drop Lowest

This condition will disregard the dice which roll the lowest values. By default it will only disregard one dice, but you can specify a different amount. For example:

```
2d20L
```

This will roll two twenty-sided dice, and disregard the lowest of the two values. (this is equivalent to a advantage roll in DnD)

```
5d10L3
```

This will roll five ten-sided dice, and disregard the lowest three values.

Drop conditionally

With this you can disregard dice that roll over a value, below a value, or by specific values.

4d6D{<3}

Roll 4d6, drop any that roll a value less than 3.

4d6D{>4}

Roll 4d6, drop any that roll a value greater than 4.

4d6D{<3,>4}

Roll 4d6, drop any that roll a value less than 3, and also drop dice with values greater than 4.

4d6D{4}

Roll 4d6, drop any that roll a value of 4.

4d20D{<5,9,11,>15}

Roll 4d20, drop dice below 5, dice above 15, and dice that roll 9 or 11.

Cap/Clamp

Clamp values that roll above/below a specified value

4d6C<2

Roll 4d6, treating any value less than 2, as 2.

```
4d6C>5
```

Roll 4d6, treating any value greater than 5, as 5.

```
4d6C<2>5
```

Roll 4d6, treating any value less than 2, as 2, and any value greater than 5, as 5.

Value Replacement

Use this condition to replace dice face values with others

```
4d6V{4=6,5=6}
```

Roll 4d6, and count face values of 4 or 5 as if they were 6.

```
d20V{20=9001}
```

Roll a d20, and count the 20 face as extra powerful.

You can also replace all values less than (<) or greater than (>) a specific value

```
d20V{<5=0,>15=20}
```

Roll a d20, and count the faces below 5 as 0, and above 15 as 20.

It is also possible to replace number values with string values (without spaces) and vice-versa:

```
d6V{6=Success}
```

Roll a d6, and count the 6 face as a "Success" string value.

You can also replace a single face with a range of possible values, which are sampled as the die rolls:

```
d6V{6=Win|Lose,5=10..20}
```

Roll a d6, and count the 6 face as either a "Win" or "Lose" string value, and the 5 face as a random number from 10 to 20.

You can even replace a face value with the value of another dice roll by providing a sub-expression:

```
d20V{20=[10d6]}
```

Roll a d20, and if it settles on the 20 face, roll 10d6 (and don't count the 20).

No Repeats/Unique

This will roll the pool so that if any two dice in the pool have rolled the same value, one of them will re-roll.

```
4d6U
```

It's worth keeping in mind, some dice pools can't have unique results, for example, 7d6 will always have at least one dice that matches another, so 7d6u will re-roll forever (which is fun, so don't worry about it too much).

You can specify values which should not be rerolled by this condition:

```
4d6U{5}
```

Roll 4d6 and re-roll repeat values, but don't re-roll values of 5.

Explosion

With this roll condition, dice that roll certain values will spawn new dice.

4d6!

Roll 4d6, and explode dice that land on the maximum value (6).

4d6!2

Limited explosion: Roll 4d6 exploding as above, but each die can explode no more than 2 times.

4d6!!

Inflation: Roll 4d6 exploding, but exploded dice increase in how many sides they have (the increase is determined by the "Ladder Type" dice setting).

4d6!!!

Reductive explosion: Whatever value each dice rolls (except 1) a dice with that many sides will be exploded from it.

4d6!{4,5}

Roll 4d6, and explode on values 4 or 5 (and no others).

4d6!{<4,>5}

Roll 4d6, and explode on values less than 4 or greater than 5.

```
4proficiency!{Success}
```

Roll 4 proficiency dice, and explode on "Success" string values. (this requires you have dice labelled "proficiency" in your dice bag, with a string value face of "Success").

```
d20!{20=[4d8]}
```

Roll a d20, and explode on the 20 face into the sub-expression "4d8".

```
4d6!{(6,6,6),(1,2,3)}
```

Pattern explosion: Roll 4d6, and if three dice match the patterns 6-6-6 or 1-2-3, explode.

```
4d6!.L2
```

Explosion without disregarding: Roll 4d6 exploding on the maximum value, with a condition to drop the lowest two dice. Because the explosion condition is followed by a '.', exploded dice will not be disregarded by other roll conditions.

Reroll

This condition allows you to specify certain values that dice in a pool *cannot* settle on, if they do then the dice will be re-rolled.

```
4d6R{1,2}
```

Roll 4d6, and re-roll any dice that settle on values 1 or 2.

```
4d6R{<3,>5}
```

Roll 4d6, and re-roll any dice that settle on values less than 3 or greater than 5.

```
4challengeR{Failure}
```

Roll 4 challenge dice, and re-roll any dice that settle on string values of "Failure".

You can also specify a maximum number of rerolls per die:

```
4d6R{1,2}3
```

Roll 4d6, and re-roll any dice that settle on values 1 or 2, but do not allow any die to be re-rolled more than three times.

If you use the Reroll condition without any reroll values (things inside the curly brackets) number values of '1' will be rerolled.

Count

Count the number of dice in the pool that meet specified conditions.

```
4d6#
```

Roll 4d6 and count how many dice roll the maximum value (6).

```
4d6#{>4}
```

Roll 4d6 and count how many dice roll a value greater than 4.

```
4d6#{5}
```

Roll 4d6 and count how many dice roll a value of 5.

4d6#{<4,5}

Roll 4d6 and count how many dice roll a value less than 4, or equal to 5.

4d6#{odd}

Roll 4d6 and count how many dice roll an odd-numbered value.

Text

Convert a dice pool's numeric values to text (string) values.

2d20T

Roll 2d20 and treat the sum as a string value.

2d20Ti

Roll 2d20 and treat the individual values as string values.

2d20Tw

Roll 2d20 and convert the sum to English words (eg, "Twenty One").

2d20Twi

Roll 2d20 and convert the individual values to English words (eg, "Nine, Fifteen").

Redefine

Rolls an existing die as a different specified number-die or any other custom/generatable die.

Assuming you have a custom die named "myDie" in your dice bags:

myDie_20 + myDie_4

Roll a d20 and a d4, each with the appearance of "myDie"

4myDie_dF

Roll four FATE/Fudge dice with the appearance of "myDie"

myDie_myOtherDie

Roll a die like "myOtherDie" with the appearance of "myDie"

Paint

Changes the coloring of dice in a pool. You can set up to four colors using words, letters, or HTML values:

d20P{Green}

Roll a green d20.

d20P{Green, Yellow}

Roll a green and yellow d20.

d20P{G,Y,R}

Roll a green and yellow d20 with red face labels.

```
d20P{#0F0,#FF0,#F00,#FF00FF}
```

Roll a green and yellow d20 with red face labels, highlighting the top face in magenta.

```
d20P
```

Roll a d20 with random colors.

Scale

Scales the dice in a pool by a specified amount.

```
2d20S{0.5}
```

Roll 2d20 that are half-sized.

```
2d20S{2}
```

Roll 2d20 that are double-sized.

That's all of the supported roll conditions for now!

Combinations

The roll conditions can generally be combined, for example:

```
10d6H3L2D{4}!{2}
```

This will roll 10d6, drop the highest 3, drop the lowest 2, drop any dice that landed on face 4, and explode any dice that rolled a 2.

Multiple Results

Roll expressions can output multiple results simultaneously, this can be done by breaking up an expression with commas:

```
4d6, 3d8
```

This will roll 4d6 and 3d8, outputting the results seperately.

You can also seperate results by naming them like so:

```
attack=2d20 damage=4d12
```

This will roll 2d20 and 4d12, outputting the named results seperately.

Named results can also be used as part of subsequent results:

```
a=4d6, b=5d10, a+b
```

This will roll 4d6 and 5d10, outputting the named results seperately, and finally outputting their sum.

Named results can have some roll conditions applied:

```
a=4d6, a#
```

This will roll 4d6, outputting the named result, and then outputting the count of dice which rolled the maximum value.

The roll conditions which may be applied to named results are those which don't

change rolled dice or add new ones. You may Keep, Drop, Cap/Clamp, Value Replace, and Count named results - treating all dice which form the result as a single dice pool.

Sub-Expressions

Roll expressions can include within them, instructions to roll yet other roll expressions if certain conditions are met.

These conditions are explosion (where the result of the sub-expression will be rolled and *added* to the face result:

```
d6!{6=[4d8]}
```

And value replacement (where the result of the sub-expression will be rolled and *replace* the face result:

```
d6V{6=[4d8]}
```

sub-expressions are indicated by square brackets ("[" and "]"), and can contain any valid roll expression (including further sub-expressions).

Math Functions

You can include several functions in your dice expressions:

```
round(d20/2)
```

Roll a d20, divide by two, and round the result to the nearest whole number.

Many tabletop games have specific rounding rules, check the Rounding Functions section for a comprehensive list of rounding options that are supported.

min(2d20, 4d10)

Roll 2d20 and 4d10, returning the sum of whichever dice pool rolled the smallest number. This function can compare an arbitrary number of parameters.

max(2d20, 4d10)

Roll 2d20 and 4d10, returning the sum of whichever dice pool rolled the largest number. This function can compare an arbitrary number of parameters.

clamp(d20+d4, 5, 15)

Roll a d20 and a d4, and clamp the result so it is never less than 5, or greater than 15.

rnd(1,20)

Return a random number from 1 to 20 without rolling any dice.

There are other mathematical functions, less common for use with dice but available to you if you want them: sin(), cos(), tan(), asin(), acos(), atan(), atan2(,), tanh(), exp(), sqrt(), ln(), abs(), pow(,), lerp(,,), mod(,), sign().

Rounding Functions

There are lots of different ways to round numbers in Sophies Dice:

round(d20/2)

Roll a d20, divide by two, and round the result to the nearest whole number.

rounddown(d20/2)

Roll a d20, divide by two, and round the result down. This function can also be called "floor()" or "rdown()".

roundup(d20/2)

Roll a d20, divide by two, and round the result up. This function can also be called "ceiling()", "ceil()", or "rup()".

roundeven(d20/2)

Roll a d20, divide by two, and round the result to the nearest even number. This function can also be called "roundtoeven()" or "reven()".

roundodd(d20/2)

Roll a d20, divide by two, and round the result to the nearest odd number. This function can also be called "roundtoodd()" or "rodd()".

roundtozero(d20/2)

Roll a d20, divide by two, and round the result to the adjacent whole number closest to zero. This function can also be called "trunc()" or "truncate()".
This is equivalent to rounding down when above zero, and rounding up when below zero.

```
roundfromzero(d20/2)
```

Roll a d20, divide by two, and round the result to the adjacent whole number furthest to zero.
This is equivalent to rounding up when above zero, and rounding down when below zero.

When rounding with any of these functions, there will be no change if you are trying to round a number that is already a whole number. This applies even in the case of rounding to odd or even.

Game-Specific Functions

There are a handful of functions for use with specific tabletop game systems

Convert a 5e Ability Score to an Ability Modifier:

```
dnda2m(4d6L)
```

Roll four d6s, drop the lowest, and convert resulting ability score to an ability modifier.

Convert a 5e Character Level to a Proficiency Modifier:

```
dndl2p(d20)
```

Roll a d20, and convert resulting character level to a proficiency modifier.

Convert a Lancer License Level to GRIT score:

```
lanll2g(d13-1)
```

Roll a d13, subtract 1, and convert resulting licence level to a GRIT score.

Check/Compare Functions

Check/Compare functions compare two numbers and add a string value to the results determined by the comparison.

```
check(d20+2, 10)
```

Roll a d20+2, and if the result is 10 or higher output "Pass", otherwise output "Fail".

```
compare(d20+2, 10)
```

Roll a d20+2, and if the result is; higher then output "Greater", smaller then output "Less", or equal then output "Equal".

Both check() and compare() are essentially the same function, but with different default string outputs. These defaults can be overriden, for example:

```
check(d20+2, 10, "win", "lose", "tie")
```

Roll a d20+2, and if the result is; higher then output "win", smaller then output "lose", or equal then output "tie".

Variable and Constant Values

There are severable variables you can include in your expressions that will change based on the results of the roll:

```
d20R{1} * rerollCount
```

Roll a d20, reroll it whenever it lands on a 1, and multiply the result by the number of times it has been re-rolled.

d20! * diceCount

Roll a d20, spawning new d20s whenever they roll max value, and multiply the result by the total number of dice rolled by the expression.

4d6 * pinCount

Roll 4d6, and multiply the result by the total number of dice that have been pinned.

4d6 * pinSum

Roll 4d6, and multiply the result by the total values of all dice that have been pinned.

4d6 * disregardCount

Roll 4d6, and multiply the result by the total number of dice that have been disregarded.

4d6 * disregardSum

Roll 4d6, and multiply the result by the total values of all dice that have been disregarded.

4d6!2 * explodedCount

Roll 4d6, exploding two new d6s whenever they roll max value, and multiply the result by the number of dice that exploded.

4d6!2 * explosionCount

Roll 4d6, exploding two new d6s whenever they roll max value, and multiply the result by the number of dice that have been spawned by explosions.

4d6K0 + diceSum

Roll 4d6, disregard them all, but count the sum of them anyway.

Preset variables only refer to dice that are part of the result/sub-expression where they are specified, to refer to all dice regardless of this scope, use an "all" version of the variable name. (eg, "allPinSum", "allExplodedCount").

There are also constant values which do not change: PI, Phi, DegToRad, RadToDeg.

User Values

You can set up your own values to use in roll expressions too. By creating named variables and assigning them values.

(These examples assume you have an active User Value named "Strength".)

d20 + Strength

Roll a d20, and add the value of 'Strength' to it.

Strengthd6

Roll a number of d6 equal to the Strength value.

6dStrength

Roll 6 dice that have a number of sides equal to the Strength value.

To create/edit user values, go to the 'User Values' section of the menu, and create a value sheet. Sheets contain a list of values which you name yourself, and can then

include in roll expressions.

Value sheets have been designed to be roughly equivelant to character sheets, so they can be disabled for characters you are not using, and coloured in order to distinguish them in the results display.

Roll Condition Parameters

Most Roll Conditions can take one or more parameters - values that alter how they behave. These follow the condition directly or are placed between curly brackets "{}".

- If you have a roll condition "H2" (Keep Highest 2), the '2' is the parameter.
- If you have a roll condition "!{4,Success}" (Explode on 4 and 'Success'), the parameters are '4' and 'Success'.

These parameters can also be specified using named expression results, User Values, or Preset Variables:

```
x=2d4, 8d6Kx
```

Roll a result called "x" (2d4), and use it as a parameter for a "Keep" condition to determine how many dice to keep from a pool.

```
d20!{>Strength}
```

Assuming you have a User Value named "Strength", roll a d20 and explode when it rolls greater than the Strength Value.

```
10d6D{<=pinCount}
```

Roll 10 d6 and disregard values that are less than or equal to the number of pinned dice.

There are some preset parameter terms that can be compared to (only within curly brackets "{}")

- "odd" is a number odd?
- "even" is a number even?
- "max" is a number the largest one the die can roll?
- "min" is a number the smallest one the die can roll?
- "multofX" is a number a multiple of X (where 'X' is an integer from 3 to 10)
- "integer" is a value an integer?
- "string" is a value a text string?
- "float" is a value a floating-point number?

Sometimes parameters can be specified as part of a condition.

- "<X" is "Less Than parameter X"
- "<=X" is "Less Than or Equal to parameter X"
- "X" is "Matching (Equal to) parameter X"
- ">=X" is "Greater than or Equal to parameter X"
- ">X" is "Greater than parameter X"
- "X..Y" is "In the range of values between parameter X and parameter Y (inclusive)"
- "(X,Y,Z)" is "Match a pattern of the three parameters X, Y, and Z"

Not all conditional parameters will work with all Roll Conditions, it will largely depend on context but generally if it makes sense to do, it should work.

Documentation index

Sophie's Dice