Drop Rate Calculator for Old School RuneScape

Installation and Dependencies

To download, open a terminal (or command prompt) and navigate to the folder where you want to download the project. For example:

```
cd /path/to/your/desired/folder
```

Use Git to clone the repository. Make sure Git is installed first:

```
git clone https://github.com/M-Dunnet/OSRS_DropRateCalc.git
```

For ease of use, it is recommended to run these scripts in an Integrated Development Environment (IDE) such as **Visual Studio Code (VS Code)**. After downloading, you can open the project folder in VS Code and run the scripts directly from the integrated terminal.

These scripts require **Python 3**, as well as the **NumPy** and **SciPy** packages. After installing Python 3, the dependencies can be installed via `pip`:

```
pip install numpy
pip install scipy
```

How it works

For Individual Drops

Use these functions if you are only interested in getting one of the possiable drops.

`drop_chance(drop_rate, kills)` Calculates the chance of getting at least one drop after a certain number of rolls.

Example usage for a 1/50 droprate after 75 rolls:

```
drop_chance(1/50, 75)
>>> Chance of drop after 75 rolls: 78.0236%
```

`kills_for_confidence(drop_rate, confidence)` Calculates the number of rolls needed to achieve a certain confidence level of getting at least one drop.

Example usage for having a 90% chance of getting a 1/50 drop:

```
kills_for_confidence(1/50, 0.9)
>>> Rolls needed for 90.00% chance of drop: 114
```

`drop_rate_interval(drop_rate, interval)` Calculates the range of kills in which a given percentage of players are expected to receive at least one drop.

Example usage for the range of number of rolls in which 50% of players will get at least one drop with a 1/50 drop rate.

```
drop_rate_interval(1/50, interval=0.5)
>>> 50.00% of players will get their first drop between 15 and 69 rolls.
```

For Multiple Drops

Use these Class methods if you are interested in getting multiple drops from a single source. A main Class, `MultiDropSimulator`, is used to generate a Monte Carlo simulation for drop chances. Class methods are then used for specific calculations.

`MultiDropSimulator` `MultiDropSimulator` requires a Python dictionary of drop names and their respective drop rates, for example:

```
drops = {
    "Dragon 2h sword": 1/358,
    "Dragon pickaxe": 1/358,
    "Skull of vet'ion": 1/618
}
simulations = MultiDropSimulator(drops)
```

After initating the `MultiDropSimulator` class as above, use the following class methods

`simulator.combined_expected_kills()` Calculates the expected number of kills to collect all drops.

Example usage for the expected number of rolls to collect all drops:

```
simulations.combined_expected_kills()
>>> Average rolls for all drops: 844
```

`simulator.combined_kills_for_confidence(confidence)` Calculates the chance of getting at all drops after a certain number of rolls.

Example usage for a 90% chance of all drops:

```
simulations.combined_kills_for_confidence(0.9)
>>> Rolls needed for 90.00% chance of all drops: 1590
```

`simulator.individual_drop_rate_intervals(interval)` Calculates the range of kills in which a given percentage of players are expected to receive at least one drop, for all drops individually.

Example usage for the range of number of rolls in which 50% of players will get at least one drop.

```
simulations.individual_drop_rate_intervals(0.5)
>>> Individual item rolls needed for a 50.00% chance of drop:
        Dragon 2h sword: 103 to 496 rolls
```

```
Dragon pickaxe: 103 to 496 rolls
Skull of vet'ion: 178 to 857 rolls
```

`simulator.combined_drop_interval(interval)` Calculates the range of kills in which a given percentage of players are expected to receive all drops.

Example usage for the range of number of rolls in which 50% of players will get all drops:

```
simulations.simulator.combined_drop_interval(0.5)
>>> 50.00% of players will get all drop between 435 and 1091 rolls.
```