# NumClass User Manual

## Installation, Usage, and Reference Guide

Manual Version 1.0
For NumClass version 2.0
Author: Marcel M.W. van Dinteren
January 2026


```
Number statistics:
  Input:                 2**5+10
  Number:                42
  Digits:                Count=2, Sum=6, Product=8
  Parity:                Even
  Prime:                 No, nearest primes: ◀41 ▶43
  Digital root of |n|:   6 (sequence: 42 → 6, additive persistence: 1)
  Mult. persistence:     8 (sequence: 42 → 8)
  Zeckendorf:            42 = 34 + 8 (F9 + F6), bits(F9..F2)=10010000
  Prime factorization:   2 × 3 × 7
  Prime factor counts:   Ω(n) = 3, ω(n) = 3
  Euler's totient ϕ(n):  12 ϕ(2)=1, ϕ(3)=2, ϕ(7)=6 ⇒ 1*2*6=12
  Carmichael λ(n):       6 λ(2)=1, λ(3)=2, λ(7)=6 ⇒ lcm(1, 2, 6)=6
  1/n (base 10):         1/n = 0.0238095 repeats, period 6, preperiod 1
  Möbius μ(n):           -1 (squarefree, odd number of prime factors)
  Radical rad(n):        42 (product of distinct primes)

Divisor statistics of |n|:
  Divisor count τ(n):    8
  Divisors:              1, 2, 3, 6, 7, 14, 21, 42
  Classical div. sums:   σ(n)=96, s(n)=σ-n=54, q(n)=s-1=53
  Unitary div. sums:     σ*(n)=96, s*(n)=σ*-n=54, τ*(n)=8, q*(n)=s*-1=53
  Aliquot sequence:      42 → 54 → 66 → 78 → 90 → 144 → 259 → 45 → 33 → 15 → 9 → 4 → 3 → 1 → 0 (steps: 14 – peak 259 @ [7] – 1 ms)
  Quasi-aliquot seq.:    42 → 53 → 0 (steps: 2 – peak 53 @ [2] – 0 ms)

Arithmetic and Divisor-based
  - Abundant number: Proper divisors sum > the given number.
    Details: Sum of proper divisors: 54 > 42.
  - Cube-free number: No prime cube divides n (all prime exponents ≤ 2).
    Details: n = 2 × 3 × 7; max exponent = 1 < 3
  - Highly abundant number: An integer whose sum of divisors is greater than that of any smaller positive integer.
    Details: σ(42) = 96. Previous record: 36 with σ(36) = 91.
  - Practical number: All smaller positive integers can be written as sums of distinct divisors of the number.
    Details: 2 ≤ 1 + σ(2) = 4, 3 ≤ 1 + σ(6) = 13, All conditions satisfied, 42 is practical.
  - Primary pseudoperfect number: Satisfies Σ_{p|n} 1/p + 1/n = 1 (sum over distinct prime divisors).
    Details: primes(n) = 2×3×7; 1/2 + 1/3 + 1/7 + 1/42 = 1 (and Σ n/p + 1 = 42 = n)
  - Semiperfect number: A positive integer equal to the sum of a subset of its proper divisors.
    Details: 42 = 7 + 14 + 21
  - Sphenic number: A positive integer that is the product of exactly three distinct primes.
    Details: 42 = 2 × 3 × 7
  - Squarefree number: No prime factor appears more than once in its factorization.
    Details: n = 2 × 3 × 7; max exponent = 1 < 2
  - Zumkeller number: Divisors can be partitioned into two equal-sum sets.
    Details: Practical & even σ(n) ⇒ Zumkeller; σ(n)=96, target=48; partition found (MITM); witness: [1, 2, 3, 7, 14, 21] (sum 48)

Combinatorial and Geometric
  - Cake number: Max pieces from k planar cuts in 3D: C(k) = (k^3 + 5k + 6)//6.
    Details: 42 is the cake number for k=6 straight cuts.
  - Catalan number: Cₙ = binomial(2n,n)/(n+1), counts Dyck paths.
    Details: 42 is Catalan number C(5).
  - Lah number: Equals L(n,k) for some n≥1,k≥1: partitions of n elements into k nonempty ordered lists.
    Details: 42 = L(7,6).
  - Partition number: Checks if n equals p(k): number of integer partitions of k.
    Details: 42 = p(10).

Digit-based
  - Brazilian number: Representable as a repdigit in some base b with 2 ≤ b ≤ n-2.
    Details: In base 4, n is a repdigit of length 3 with digit 2: 2×R_3(4) where R_3(4)=( 4^3-1 )/(4-1)
  - Harshad number: Base-10 Harshad (Niven) number: divisible by the sum of its decimal digits.
    Details: 42 is divisible by the sum of its digits: 42 ÷ 6 = 7.
  - Moran number: Harshad (base 10) with n/s = prime, where s is the sum of decimal digits of n.
    Details: Harshad with s(n)=6; n/s = 7 is prime.
  - Odious number: Has an odd number of 1's in binary representation for |n|.
    Details: Binary: 101010, number of 1's: 3 (odd), total bits: 6
  - Self number: Cannot be written as m + sum of digits of m for any m.
    Details: 42 cannot be written as m + sum(digits of m) for any m < 42.

Diophantine representations
  - Sum of 3 cubes: Can be written as n = a³ + b³ + c³ for integers a, b, c.
    Details: 42 = (-80538738812075974)³ + 80435758145817515³ + 12602123297335631³
  - Sum of 3 squares: Can be written as n = a² + b² + c² for integers a, b, c.
    Details: found 1: 1²+4²+5²

Fun numbers
  - Fun number: Numbers that are famous or iconic in pop culture, computing, sci-fi, internet humor, or memes.
    Details: The Answer to the Ultimate Question of Life, the Universe, and Everything (Douglas Adams, The Hitchhiker's Guide to the Galaxy).

Mathematical Curiosities
  - Eban number: No letter 'e' in the English spelling.
    Details: 42: forty-two is an eban number (no 'e' in its English spelling)
  - Octal-interpretable number: Number can be interpreted as an octal numeral.
    Details: As octal: 42₈ = 34₁₀ = 100010₂ = 22₁₆
  - Polydivisible number: In base 10: for every k = 1..d, the first k digits form a number divisible by k.
    Details: Let Pₖ be the first k digits. For k=1..2, Pₖ ≡ 0 (mod k). P₁=4 divisible by 1 ; P_2=42 divisible by 2.

Named Sequences
  - 2-term Zeckendorf number: Zeckendorf decomposition uses exactly two Fibonacci terms: n = F_a + F_b with a ≥ b+2.
    Details: 42 = 34 + 8 (F9 + F6)
  - Sparse Zeckendorf representation: Zeckendorf decomposition uses few Fibonacci terms (weight ≤ K).
    Details: weight 2 ≤ 3: 42 = 34 + 8 (F9 + F6)

Polygonal and Figurate Numbers
  - Pronic number: A number of the form n(n+1), the product of two consecutive integers.
    Details: 42 = 6 × 7

Profile: default – Enter an integer, command or profile (h=Help, q=Quit): |
```

*Figure 1 Example of NumClass output for the input 2**5+10, resulting in number 42.*

# Table of Contents

# Introduction

NumClass is a modular and extensible software package written in Python, designed to classify integers into 11 main categories and over 200 distinct number types. The system covers a wide range of mathematical properties, including:

- **Arithmetic and Divisor-based** such as Perfect, Abundant, Deficient, Powerful, Practical, Sociable, Triperfect, and more.
- **Combinatorial and Geometric** such as Bell, cake numbers, Catalan, Motzkin, Stirling, Ramsey, etc.
- **Conjectures and Equation-based** including Egyptian fractions, Erdős–Straus, Goldbach, Legendre, and others.
- **Digit-based** like Palindrome, Automorphic, Happy, Narcissistic, Self, Kaprekar, etc.
- **Diophantine Representations** Numbers expressible as sums of squares or cubes.
- **Dynamical Sequences** such as Collatz, Ducci, Fibonacci mod n and Kaprekar routines.
- **Fun Numbers** those notable in pop culture, internet, computing, or science fiction.
- **Mathematical Curiosities** including, Munchausen, Kaprekar constants, Strobogrammatic, Vampire and Eban numbers.
- **Named Sequences** such as Busy Beaver, Carol, Lucky, Keith, Padovan and Taxicab numbers.
- **Polygonal and Figurate** including Triangular, Pentagonal, Pronic, Cyclic, and Repunit numbers.
- **Primes and Prime-related numbers** including twin, safe, sexy, Sophie Germain, palindromic, emirp, factorial, strong, super primes, and more.
- **Pseudoprimes and Cryptographic numbers** such as Blum, Carmichael, Fermat, and Euler–Jacobi pseudoprimes.

For a complete list of classifiers and detailed explanations, see Appendix A.

NumClass provides comprehensive number and divisor statistics, efficient logic for intersecting properties (e.g., “happy palindromic prime”), and explanations for each classification. It leverages OEIS[1] b-files where speed matters, automatically discovers custom classifiers and settings, and adapts its output to your terminal size.

This manual describes NumClass version 2.0.
The manual itself is versioned independently.
Manual version 1.0 corresponds to the initial public release of NumClass 2.0.
Future manual revisions may document later NumClass releases or clarify existing functionality without changing the software version.

| Manual Version | Date | NumClass Version | Notes |
|---|---|---|---|
| **1.0** | Januari 2026 | 2.0 | Initial manual release |

[1] OEIS Foundation Inc. (2025), The On-Line Encyclopedia of Integer Sequences, Published electronically at https://oeis.org

# Installation

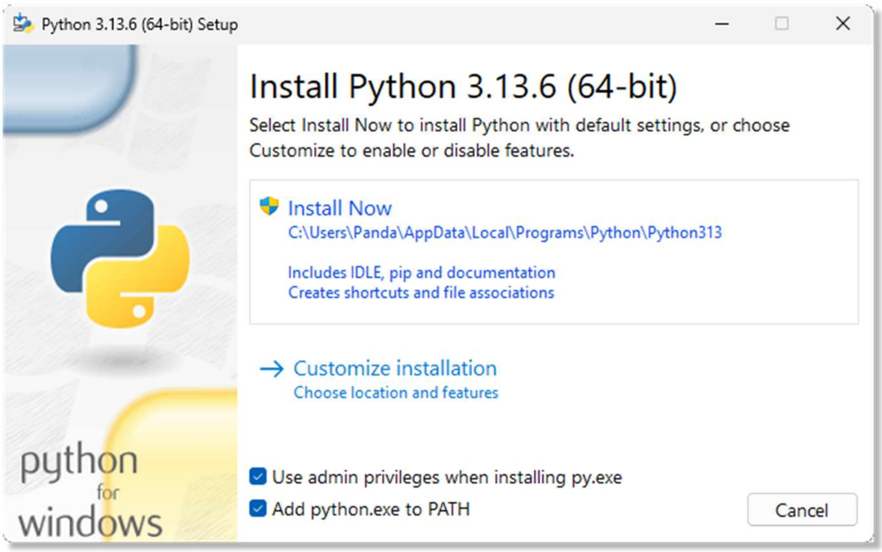
## Prerequisites

- **Operating System:** Windows 10/11, macOS 12+, or any recent 64-bit Linux distribution
- **CPU:** Any 64-bit processor (e.g., Intel i5, Ryzen 3 or better)
- **RAM:** 8 GB or more recommended
- **Disk Space:** 1 GB free (for virtual environment and caches; SSD recommended)
- **Python:** Version 3.11 or higher (64-bit)
- **Terminal:** UTF-8 capable (for proper display of mathematical symbols)

## Install Python

### Windows

1. Download the latest Python release from python.org/downloads/windows.
2. Run the installer and check both boxes at the bottom:
    - "Use admin privileges when installing py.exe"
    - "Add python.exe to PATH"
3. Click "Install Now" to complete the installation.




### Linux

- Most modern Linux distributions include Python 3 by default. To check your version, run:
  `python3 --version`
- If you need to install or upgrade Python 3.11+:
    - **Debian/Ubuntu:**
      ```
      sudo apt update
      sudo apt install python3 python3-pip
      ```
      For the latest Python, you may need to use a PPA or build from source.
    - **Fedora:**
      ```
      sudo dnf install python3 python3-pip
      ```
    - **Arch Linux:**
      ```
      sudo pacman -S python python-pip
      ```

**macOS**

- macOS 12+ may include Python 3, but it is recommended to install the latest version via Homebrew:
  ```
  brew install python
  ```
- Verify installation:
  ```
  python3 --version
  ```

## Install pipx

**pipx** allows you to install and run Python applications in isolated environments.

**Windows**

```
py -m pip install --user pipx
py -m pipx ensurepath
```

Restart your terminal after running ensurepath.

**Linux/macOS**

```
python3 -m pip install --user pipx
python3 -m pipx ensurepath
```

Open a new terminal window, or run source ~/.bashrc, source ~/.zshrc, or the relevant shell configuration file to apply the changes immediately. A full restart is usually not necessary.

## Install NumClass

pipx install "git+https://github.com/c788630/Numclass@main"

**Windows**

```
pipx install "git+https://github.com/c788630/Numclass@main"
```

**Linux / macOS**

```
python3 -m pipx install "git+https://github.com/c788630/Numclass@main"
```

## First-Time Setup

After installation, initialize your workspace:

```
numclass init
```

This creates a NumClass workspace folder in your user Documents folder with classifiers, data, and profile files.

The paths used can be shown / verified with:

```
numclass where
```

# Usage

After installation, you can use NumClass from the command line to classify integers and explore their mathematical properties.

## Basic Command Structure

```
numclass [profile] [integer] [options]
numclass -h | --help
numclass init [overwrite]
numclass list
numclass where
```

`numclass` without an integer specified will start interactive mode.

| | |
|---|---|
| `[profile]` | Starts NumClass interactive mode with the specified profile. |
| `[integer]` | The Integer or mathematical expression to classify. |
| `[<profile> integer]` | Specify a profile to use for this classification followed by the Integer or mathematical expression to classify. The profile is used for this run only. |

If both a profile and an integer are provided, the profile must appear first.

## Options:

| | |
|---|---|
| `-h, --help` | Show help information and exit. |
| `--output OUTPUT` | Write results to a file (also prints to screen unless `--quiet` is used). |
| `--quiet` | Suppress screen output. |
| `--no-details` | Do not show explanations/details. |
| `--debug` | Enable debug mode for detailed timing and discovery information. |

## Commands:

| | |
|---|---|
| `active` | Show the active profile. |
| `init` | Create workspace folders and copy packaged sample files. |
| `init overwrite` | Meant for developers. Requires environment variable `NUMCLASS_DEV=1`. Copies all initial classifiers, datafiles and profiles. All files in Documents/Numclass will be overwritten. |
| `list` | List all available classifiers. |
| `where` | Show the workspace and package paths. |

## Examples

`numclass`

Running `numclass` without arguments starts interactive mode. You will be prompted to enter:

- An integer, e.g. `42`
- A mathematical expression, e.g. `2**100-1`
- One of the following commands:

| | |
|---|---|
| `debug on|off|status` | to switch debug mode on, off or show current debug status. |
| `fast on|off|status` | to switch fast mode on, off or show current fast status. |
| `h` or `help` | to enter Help mode. |
| `hist` | show a history of the entered numbers. |
| `p` | to list available profiles and show the active profile. |
| `q` or `quit` | to quit NumClass. |

- A profile name, like 'default' to switch between profiles.

`numclass 42`

Classifies number 42 using the last active profile.

`numclass 2**100-1`

Converts mathematical expression 2**100-1 to 1267650600228229401496703205375, prints the results and exits.

`numclass aliquot 2856`

Classifies 2856 using the 'aliquot' profile (for this run only).

## Help Mode

Type 'h' at the input prompt to open *help mode*.
This displays an introduction to NumClass, after which you can enter any of the following help subcommands:

**s** - **List effective settings**
Effective settings are auto discovered from Python code and the file: default.toml
Value legend in terminal output: white: default, yellow: set in profile, red: not used in Python code.
(Paginated output, [Enter] to continue, 'q' to quit help mode.)

**c** - **List all classifications by category**
Lists all available classifiers, grouped by category with a compact description of its function. (Paginated output, [Enter] to continue, 'q' to quit help mode.)

**r** - **Show OEIS references**
References to the Online Encyclopedia of Integer Sequences. Ctrl+Click the number to follow the link.
(Paginated output, [Enter] to continue, 'q' to quit help mode.)

**e** - **Show some example numbers you can try**
Some example numbers you can try.
Tip: select and copy the number, quit help mode and paste the value.

**l** - **Show symbols and notation legend**
Explains symbols and shorthands used in NumClass output.

**q** - **Quit help mode and start classifying integers**
Quit help mode and return to the input prompt.

## Working With Profiles

Profiles are a core concept in NumClass. A profile defines how NumClass behaves, including which classifiers are active, which categories are enabled, performance limits, output settings, and display options.
Using profiles allows you to switch between different working modes (for example fast testing, full classification, experimental classifiers, or custom setups) without modifying code or command-line arguments.

**What a Profile Controls**
A profile can control, among others:

- Enabled and disabled classification categories
- Performance limits and time budgets
- Debug and fast modes
- Output behavior (OUTPUT_FILE)
- Display options
- Feature flags

Profiles make NumClass highly configurable while keeping the command line simple.

**Temporary and Persistent Profile Selection**
NumClass distinguishes between temporary profile selection (single-run) and persistent profile switching (interactive mode).

**Command-line Profile Selection (Temporary)**
When a profile name is specified together with a number on the command line, the profile is used only for that single run.
Example:

```
numclass collatz 360
```

Behavior:

- The collatz profile is used for this classification only
- The previously active default profile is not changed
- The next NumClass start will reuse the original default profile

This allows testing different profiles without modifying your persistent configuration.

**Interactive Profile Switching (Persistent)**
When NumClass is started in interactive mode (without providing a number), entering a profile name switches the active profile and makes it the new default.
Example:

```
numclass
```

Then at the prompt: `primes`

Behavior:

- The active profile is changed to primes
- The new profile is stored
- Future NumClass sessions will automatically start using this profile

NumClass persists the most recently selected interactive profile by writing its name to file: numclass\profiles\.current. Each profile is stored as a TOML configuration file.

**Listing Available Profiles**
In interactive mode enter p to show a list of available profile files, stored in the /profiles directory. The currently active profile is marked with an arrow.

The following profiles are initially delivered with NumClass:

| | |
|---|---|
| aliquot | Focus on aliquot theory: σ/τ stats, aliquot/quasi/unitary chains, peaks and timing. |
| all | Enable every category/classifier; useful for demos and maximal coverage. |
| collatz | Explore Collatz: sequence output, stopping time, peaks, optional extended step caps. |
| combinatorial | Combining classical counting sequences: Bell, Stirling, Fubini, Lah, Catalan, Motzkin, and related combinatorial numbers. |
| conjecture | Turn on conjectures: Goldbach/Legendre/Lemoine etc., witness listing; may be slow. |
| core | Classical number theory focus; no recreational/dynamic/curiosity. |
| default | Balanced everyday setup: broad coverage, safe caps, informative details. |
| lean | Fast minimal profile: headline classifiers only, terse output, strict caps. |
| primes | Primes, prime related and Pseudoprimes only. |
| sequences | Dynamical- and Named sequence based only. |
| statsonly | Display the number statistics only. |

**Creating and Editing Profiles**
Profiles are simple text files and can be edited with any text editor.

To create a new profile:

- Copy an existing profile file (best to use all.toml as it contains all settings)
- Rename it
- Modify the settings as needed

Go to your Documents/Numclass/profiles folder, edit a profile file e.g. default.toml change or add any setting and save the file.

**Delete a Profile**
Go to your Documents/Numclass/profiles folder and simply delete the file having the profile name you want to delete.

## Input Formats

NumClass accepts:

- Positive or negative integers (with optional thousand separators: spaces, underscores, periods or commas).
- Binary (0b), octal (0o) and hexadecimal (0x) numbers.
- Mathematical expressions that evaluate to integers.
- Special input translations as defined in Special_inputs.tsv.
  This mechanism allows NumClass to support rich, interactive input beyond plain integers. Two demos are included: Roman numerals (try: *MCMLXVIII = 1968*) and Phonetic Klingon numbers (try: *javSaD wa'vatlh SochmaH loS = 6174)*

## Mathematical Expressions: Operators and Precedence

Mathematical expressions resulting in an integer are allowed.
Example: `2**100-1` (2 to the power of 100 minus 1) will be converted into 1267650600228229401496703205375.

Allowed operations and precedence.

| Prec. | Operators | Example(s) | Meaning |
|---|---|---|---|
| **1** | ( ) | (2 + 3) * 4 | Explicit grouping (highest precedence) |
| **2** | ! | 5! → 120 | Postfix factorial, binds tighter than ** |
| **3** | ** | 2 ** 10 → 1024 | Exponentiation (right associative). Negative exponents not allowed. |
| **4** | +x, -x | -5, +8 | Unary sign operators |
| **5** | *, //, % | 6 * 7, 17 // 3, 17 % 3 | Multiplication, integer division, modulo |
| **6** | +, - | 7 - 4, 2 + 3 | Addition and subtraction |
| **7** | <<, >> | 3 << 4, 100 >> 3 | Bit shifts (shift count ≥ 0) |
| **8** | & | 42 & 15 → 10 | Bitwise AND |
| **9** | ^ | 5 ^ 12 → 9 | Bitwise XOR |
| **10** | \| | 5 \| 12 → 13 | Bitwise OR |
| **11** | : | 12:34 → 1234 | Decimal concatenation (lowest precedence) |
| **12** | ?[n] | ?100 → Random number between 0-100<br>?500-250 → Random number between -250 and +250. | Random number placeholder, replaced before evaluation. Optional n defines the range 0 – n. If n is omitted, the range is 0-999999. |

NumClass can work with Integers having up to 100000 digits by default.
This value can be changed using environment variable PYTHONINTMAXSTRDIGITS.
All known (2025) Mersenne prime expressions are handled even if the number of digits exceeds 100000, but the output is reduced to only show a few elements.
e.g. 2**136279841-1 is a Mersenne prime and has 41024320 digits.

Examples of valid input:

`42` positive integer.
`-7` negative integer.
`123 456 789` 123456789 (using ' ' as thousand separator).
`123.456.789` 123456789 (using . as thousand separator).
`123,456,789` 123456789 (using , as thousand separator).
`1_000_000` 1000000 (using _ as thousand separator).
`0b1010` 10 (using binary prefix 0b).

| | |
|---|---|
| `0o77` | 63 (using octal prefix 0o). |
| `0xFF` | 255 (using hexadecimal prefix 0x). |
| `2**1000-1` | 1267650600228229401496703205375 |

Invalid input:

| | |
|---|---|
| `3.14` | decimal point not allowed as it is not an integer and not a thousand separator. |
| `1,23` | comma not used as a thousand separator (requires groups of 3 digits). |
| `12.34.56` | period not used as a thousand separator (requires groups of 3 digits). |
| `0xG1` | invalid value after hexadecimal prefix (only digits 0-9, letters A-F are allowed). |
| `0o80` | invalid value after octal prefix (only digits 0-7 are allowed). |
| `3+2i` | rejected as a complex number is not an integer by definition. |
| `(2**10)/3` | rejected as non-integer division is not allowed (would create a float 141.333...) |

## Input Redirection

NumClass supports standard input redirection on Windows, Linux, and macOS. This allows numbers to be read from a file or pipe instead of being entered on the command line.
This feature can be used both for very large single integers and for batch processing multiple numbers.

```
numclass --output results.txt < input.txt
```

This method is intended for processing extremely large numbers that exceed practical command-line length limits.
The redirected input file may contain one or multiple integer values or mathematical expressions, separated by line breaks.

Example input.txt:

42
3605678901234567890123456789
2**100-1

Each line is treated as a separate NumClass input. Leading and trailing whitespace is ignored.

**Batch processing behavior**
When multiple input values are provided via redirection:

- Each number is processed sequentially
- Output is generated for each input
- When using file output, results are appended or written per-number depending on the selected output mode

Batch processing is most useful when writing output to files (--output), since interactive screen output is cleared between runs.

**Command-line input precedence**
If both a number/expression is specified on the command line and standard input is redirected, then the command-line number takes precedence and the redirected input is ignored.

## Output Files

NumClass can write classification results either to individual output files (one file per number) or append multiple results to a single file. Output destinations can be specified using the profile setting OUTPUT_FILE or via the command-line option --output.
Both methods use identical path resolution and filename handling rules.

Example (command line):
```
numclass --output C:\NumClass\results
```

**Basic Output Configuration**

Disable file output:

```
OUTPUT_FILE = ""
```

No output file is created. Results are displayed only in the terminal.

**Per-number output (directory mode)**

When OUTPUT_FILE ends with a directory separator (/ or \), NumClass creates one output file per evaluated number.

```
OUTPUT_FILE = "results/"
```

Result: <WORKSPACE>/results/<n>.txt
Example: 42 → <WORKSPACE>/results/42.txt

**Workspace root shortcut**

```
OUTPUT_FILE = "."
```

Writes per-number output files directly into the NumClass Workspace root:
<WORKSPACE>/<n>.txt
**Note:** "." refers to the NumClass Workspace directory, not the current shell directory.

**Single File Output (Append Mode)**
When OUTPUT_FILE specifies a file path (not ending with /), NumClass appends all classification results to the same file.

```
OUTPUT_FILE = "log.txt"
```

Result: <WORKSPACE>/log.txt
Absolute file paths may also be used:

```
OUTPUT_FILE = "/data/numclass.log"
OUTPUT_FILE = "C:\Logs\numclass.txt"
```

**Path Resolution Rules**
NumClass resolves output paths using the following rules.

**Home directory expansion (~)**
Paths starting with ~ are expanded by NumClass to the user's home directory (Linux, macOS and Windows supported):

```
OUTPUT_FILE = "~/numclass/results/"
```

- C:\Users\<username>\numclass\results (Windows)
- /home/user/numclass/results (Linux/MacOS)

**Relative paths**
All relative paths ending with "/" are interpreted relative to the NumClass Workspace directory:

```
OUTPUT_FILE = "results/"
```

Result: <WORKSPACE>/results/<n>.txt

**Absolute paths**
Absolute paths are used unchanged:

```
OUTPUT_FILE = /data/results
OUTPUT_FILE = C:\NumClass\results
```

**Workspace location**
To display the location of the active workspace directory, use:

```
numclass where
```

### Automatic Filename Shortening for Very Large Numbers

When writing per-number output files, NumClass normally uses the evaluated number itself as filename:

```
42.txt
-991.txt
```

However, operating systems impose limits on filename and full path length. Since NumClass supports integers with up to 100000 digits, this limit can be exceeded.
To ensure reliable cross-platform operation, NumClass automatically switches to a shortened, filesystem-safe filename format when the full output path would exceed approximately 250 characters.

Shortened filename format:
```
digits=<D>_<SIGN><FIRST12>...<LAST12>_sha=<HASH>.txt
```

Where:

| Field | Description |
|---|---|
| **D** | Total number of decimal digits (excluding the minus sign) |
| **SIGN** | - for negative numbers, empty for positive numbers |
| **FIRST12** | First 12 digits of the number |
| **LAST12** | Last 12 digits of the number |
| **HASH** | First 12 hexadecimal characters of the SHA-256 hash of the full decimal value |

The hash component is used as a compact, collision-resistant identifier. While cryptographic collisions are theoretically possible, the combined filename structure makes accidental clashes practically negligible.

Large number example:
digits=100000_314159265358…897932384626_sha=73475cb40a56.txt

For small numbers where the full path length remains within limits, the original filename format is preserved, e.g. 42.txt

**Versioned Output Files (No Overwrite)**
If a target filename already exists, a numeric suffix is appended before the file extension:

Example:
42.txt
42_2.txt
42_3.txt

The initial output file uses the base filename without a suffix. Additional versions start at _2.
This behavior also applies to shortened filenames for very large numbers.

Example (large-number versioning):
digits=100000_314159265358...897932384626_sha=73475cb40a56.txt
digits=100000_314159265358...897932384626_sha=73475cb40a56_2.txt

This design is intentional. Re-running NumClass may produce different results (for example due to warmed caches or different effective limits). Automatic versioning prevents loss of earlier output.

# Number and Divisor Statistics

When NumClass evaluates an integer, it first prints a **Number statistics** section, followed by **Divisor statistics** (of $|n|$).
These statistics provide fast, fundamental information about the number before the detailed classification results.

## Number statistics

This section summarizes basic arithmetic, digital, and multiplicative properties of the input number.

**Input**
If the entered value is an expression that evaluates to an integer, NumClass displays the original input expression in addition to the evaluated number.
This provides transparency about how the final number was obtained.

Example:
```
Input:  2**5+10
Number: 42
```

If the input is already a literal integer, the Input line is omitted.

**Number**
The evaluated integer value. This is either the number as entered (which may be negative), or the result of the input expression.

**Digits**
- *Count:* Number of decimal digits of $|n|$
- *Sum:* Sum of the decimal digits
- *Product:* Product of the decimal digits

**Parity**
Indicates whether the number is *Even* or *Odd*.

**Prime**
Indicates whether the number is prime.
If composite, the nearest smaller and larger prime numbers are shown.

**Digital root of |n|**
The repeated sum of decimal digits until a single digit is obtained.
Also shows:
- the reduction sequence
- the *additive persistence* (number of steps)

**Multiplicative persistence**
The number of steps required to reduce $|n|$ to a single digit by repeatedly multiplying its decimal digits. The full reduction sequence is shown.

**Zeckendorf**
The *Zeckendorf decomposition* of the number: a unique representation as a sum of *non-consecutive Fibonacci numbers*.
Displayed as:

- the Fibonacci sum
- the corresponding Fibonacci indices $F_k$
- a compact Fibonacci-base bitstring

The *Fibonacci-base bitstring* represents the Zeckendorf decomposition in Fibonacci base: each bit corresponds to a Fibonacci number $F_k$, ordered from the largest index down to $F_2$.

- a bit value **1** indicates that $F_k$ is present in the decomposition,
- a bit value **0** indicates that it is not,
- no two consecutive bits are 1, reflecting the Zeckendorf rule.

Example:
```
42 = 34 + 8 (F9 + F6), bits(F9..F2)=10010000
```

**Prime factorization**
The canonical factorization of $|n|$ into prime factors.

**Prime factor counts**

- **Ω(n)**: total number of prime factors (with multiplicity)
- **ω(n)**: number of distinct prime factors

**Euler's totient ϕ(n)**
The number of integers $1 \leq k \leq n$ that are coprime to $n$.
The calculation is shown using multiplicativity over the prime factors.

**Carmichael function λ(n)**
The exponent of the multiplicative group modulo $n$.
Computed as the least common multiple of the individual prime-power λ-values.

**1/n (base 10)**
Decimal expansion of the reciprocal of $n$, including:

- repeating or terminating behavior
- length of the repeating period
- length of the preperiod (if any)

**Möbius μ(n)**
The Möbius function value:

- 0 if $n$ is not squarefree
- ±1 if $n$ is squarefree (sign determined by the parity of prime factors)

**Radical rad(n)**
The product of the distinct prime divisors of $n$.

## Divisor statistics of $|n|$

This section summarizes properties derived from the *divisors of* $|\ n\ |$.

**Divisor count τ(n)**
The total number of positive divisors of $n$.

**Divisors**
The complete list of positive divisors, in increasing order.

**Classical divisor sums**
- **σ(n)**: sum of all positive divisors
- **s(n)**: sum of proper divisors ($\sigma(n) - n$)
- **q(n)**: $s(n) - 1$

**Unitary divisor sums**
- **σ*(n)**: sum of unitary divisors
- **s*(n)**: sum of proper unitary divisors
- **τ*(n)**: number of unitary divisors
- **q*(n)**: $s^*(n) - 1$

**Aliquot sequence**
The sequence obtained by repeatedly replacing the number with the sum of its proper divisors. Displayed with:
- full sequence
- number of steps
- peak value and its position
- computation time

**Quasi-aliquot sequence**
A variant aliquot sequence using $q(n)$instead of $s(n)$, shown with the same statistics.

**Notes**
- All divisor statistics are computed for $|n|$.
- Some statistics may be omitted or abbreviated in *fast mode*.
- Expensive computations are guarded by time and size limits to keep NumClass responsive.

# Advanced Usage

## Developer Mode

To enable developer features, set the environment variable:
 set NUMCLASS_DEV=1          (Windows)
 export NUMCLASS_DEV=1     (Linux/macOS)

Currently this enables the `numclass init overwrite` command to overwrite the classifiers, data files and profiles stored in the Documents\Numclass folder with the ones present in the NumClass package folders. Any changes made to the files in the Documents\Numclass folder will be lost!

## Creating a Classifier

NumClass comes with 200+ classifiers, this section is only needed if you want to add your own classifier.

1. Each classifier starts with a decorator:

```
@classifier(
    label="<your classifier label>",
    description="<your classifier description>",
    oeis="<oeis reference>",    # Optional
    category="<existing or new category>",
    limit=<your integer limit>  # Optional
)
```

2. Define a function starting with "is_". This prefix is needed so the function is auto discovered by NumClass:

```
Def is_<your function name>(n: int, ctx: NumCtx | None = None) ->
tuple[bool, str | None]:
```

ctx (context) is optional and can be used if your function uses precomputed factor-based data, When present, ctx is an instance of the following dataclass:

```
@dataclass(frozen=True)
class NumCtx:
    n: int
    fac: dict[int, int]          # May include composite cofactors if factoring
                                    hit limits.
    sigma: int | None           # σ(n); None if incomplete or n in {0}
    tau: int | None             # τ(n); None if incomplete or n in {0}
    unitary_sigma: int | None   # σ*(n); None if incomplete or n in {0}
    unitary_tau: int | None     # τ*(n); None if incomplete or n in {0}
    omega: int | None           # Total prime factors with multiplicity;
                                    None if incomplete or n in {0}
    incomplete: bool = False    # True if fac contains any composite base.
    composite_bases: tuple[int, ...] = ()   # Optional: composite bases,
                                    useful for UI/debug.
 @property
 def is_fully_factored(self) -> bool:
     return not self.incomplete
```

3. Perform your classification logic and return either `True, "your details"` or `False, None`

4. Specifying `False, "your details"`, is also allowed; those details are only shown in debug mode.

**Classifier example**

```
from numclass.utility import build_ctx,      ← if you want to use factor based data

@classifier(
    label="Sphenic number",
    description="A positive integer that is the product of three distinct primes.",
    oeis="A007304",
    category="Arithmetic and Divisor-based",
    limit=999_999_999,
)
def is_sphenic_number(n: int, ctx: NumCtx | None = None) -> tuple[bool, str |
None]:
    """
    Returns (True, details) if n is a product of 3 distinct primes,
    else (False, None).
    """
    if n < 30:  # smallest sphenic number is 2*3*5 = 30
        return False, None

    # Reuse a precomputed context if provided, otherwise build a fresh one.
    ctx = ctx or build_ctx(n)
    f = ctx.fac

    if len(f) == 3 and all(exp == 1 for exp in f.values()):
        primes = list(f.keys())
        return True, f"{n} = {primes[0]} × {primes[1]} × {primes[2]}"

    return False, None
```

**Using profile settings in your classifier**
Import the CFG function:

```
from numclass.runtime import CFG
```

To read a setting from the active profile, e.g. FAST_MODE in section [BEHAVIOUR] use:

```
Fast_mode = CFG("BEHAVIOUR.FAST_MODE", False)
```

It returns the specified setting or the second CFG parameter as a default value if not specified in the active profile.

**Using OEIS b-files**
NumClass can use precomputed integer sequences from the OEIS to accelerate classification and avoid recomputation of known results.
These sequences are stored in so-called b-files, which list each term of a sequence in the form:

```
# A000110 Bell numbers example
0 1
1 1
2 2
3 5
4 15
…
```

To access these efficiently, NumClass provides:

```
from numclass.utility import check_oeis_bfile
```

**Example**
To check whether a number occurs in the Highly abundant numbers (OEIS A002093):
```
found, idx_file, series, idx_set = check_oeis_bfile("b002093.txt", n)
```

This returns a 4-tuple:

| Return name | Type | Meaning |
|---|---|---|
| **found** | bool | True if n appears in the sequence |
| **idx_file** | int \| None | Index from the first column of the b-file (usually the OEIS term number) |
| **series** | list[int] | All sequence values in file order |
| **idx_set** | int \| None | 0-based position of n inside series (useful for retrieving neighbors) |

Example usage:
```
if found:
    print(f"{n} is term {idx_file} in A002093.")
    if idx_set > 0:
        prev = series[idx_set - 1]
        print(f"Previous record: {prev}")
```

You can use series[idx_set – 1] to access the previous value in the sequence if idx_set > 0.

**Resolution order**
When you call check_oeis_bfile(name, n), NumClass automatically looks for the file in:
1. **Your workspace** — <workspace>/data/<name>
2. **The packaged defaults** — numclass.data/<name>

**Notes**
- found → False and empty series means the file was not found or failed to load.

## Data Files Used by NumClass

NumClass uses the following special data files present in folder \data:
Besides OEIS *b-files*, NumClass uses several small, curated data files to recognize special numbers, provide fast lookups, define combined classifications, and support Easter eggs and demonstrations.

All data files share these properties:
- Human-readable
- Safe to edit
- Reloaded on restart
- No Python execution
- Workspace overrides packaged defaults
- Comment lines (starting with #) are allowed and ignored

If a data file contains errors (wrong format, missing fields), NumClass reports a clear, user-friendly error message.

**fun_numbers.tsv**
Purpose: Defines "fun numbers" from culture, science, computing, memes, and popular mathematics. This file allows users to add their own trivia without touching Python code.
Used by: Fun number classifier

Format: number<TAB>description

- Descriptions are free text
- Numbers may be negative

Example:

```
42<TAB>The Answer to Life, the Universe, and Everything.
```

**curiosity_constants.tsv**

Purpose: Defines famous mathematical constants that are single values or small finite sets, not infinite sequences.

Used by: Curiosity constants classifier (data-driven)

Typical entries include: Kaprekar constants, Digit-Reversal Constant, Harshad in all bases, etc.

Format:

label<TAB>n<TAB>description<TAB>details<TAB>oeis

Example:

```
Harshad in all bases<TAB>1,2,4,6<TAB>An integer that is divisible by the
sum of its digits in every base ≥ 2.<TAB>Only 1, 2, 4, and 6 are Harshad in
every base ≥2.<TAB>
```

This design allows NumClass to generate many classifiers automatically from data alone.

**erdos_woods.toml**

Purpose: Provides a complete known list of Erdős–Woods numbers and auxiliary data for fast classification.

Used by: Erdős–Woods number classifier is_erdos_woods_number() in named_sequences.py

Structure:

- [erdos_woods] — known values
- [erdos_woods.min_starts] — minimal starting points for verification

This avoids expensive recomputation and ensures correctness for large inputs.

**hard_factors.txt**

Purpose:

Stores known factorizations of very large or computationally expensive integers, including:

- Fermat numbers
- RSA challenge numbers
- Difficult aliquot sequence values

Used by: Factorization routines and classifiers that depend on divisor structure in utility.py

Format:

Whitespace-separated values (number followed by known factors or expressions).

This file dramatically improves performance for known “hard” cases.

**sum_of_three_cubes.toml**

Purpose: Contains known integer solutions to the equation: $x^3 + y^3 + z^3 = n$

Used by: Sum-of-three-cubes classifier

Structure:

```
[solutions]
33 = [8866128975287528, -8778405442862239, -2736111468807040]
42 = [-80538738812075974, 80435758145817515, 12602123297335631]
```

This allows NumClass to instantly recognize celebrity solutions without recomputation.

**intersections.toml**

Purpose: Defines combined classifications (“intersections”) when a number satisfies multiple atomic properties.

Used by: Intersection engine
Example:

```
[[intersections]]
of = ["Prime number", "Palindrome"]
label = "Palindromic prime"
category = "Primes and Prime-related Numbers"
```

Key points:

- Order does not matter
- More specific intersections take precedence
- Fully user-customizable
- No Python code required

This file allows users to extend NumClass's taxonomy declaratively.

**special_inputs.tsv**

Purpose: Handles non-numeric or symbolic inputs such as Easter eggs, shortcuts, and transformations.
Used by: Input parser (expreval.py)
Format: key<TAB>handler<TAB>index<TAB>description<TAB>digits<TAB>extra

```
2**136279841-1<TAB>mersenne_exact<TAB>52<TAB>Mersenne prime M52 2^136279841-1, discovered in 2024 by L Durant using GIMPS<TAB>41024320<TAB>
0**0<TAB>egg<TAB><TAB>0^0 is undefined — the true neutral element of confusion.<TAB><TAB>
```

This mechanism allows NumClass to support rich, interactive input beyond plain integers.

# Troubleshooting

Start `numclass --debug` to enable debug mode. The following debug information will be shown:

Your terminal available rows and columns:

```
Terminal <columns>x<rows>
```

The number of atomic classifiers that have been discovered:

```
[debug] discovered atomic: 181
```

Classifiers loaded in your workspace folder (ws):

```
[discovery] ws OK arithmetic_divisor.py: 48 label(s)
[discovery] ws OK conjectures.py: 7 label(s)
[discovery] ws OK curiosities.py: 19 label(s)
[discovery] ws OK digit_based.py: 22 label(s)
[discovery] ws OK diophantine.py: 4 label(s)
[discovery] ws OK dynamical_sequences.py: 8 label(s)
[discovery] ws OK fun_number.py: 1 label(s)
[discovery] ws OK named_sequences.py: 19 label(s)
[discovery] ws OK polygonal_figurate.py: 14 label(s)
[discovery] ws OK prime.py: 33 label(s)
[discovery] ws OK pseudoprime_crypto.py: 6 label(s)
```

If your classifier contains an error, it will be listed here. Example:

```
[discovery] ws FAIL digit_based.py: NameError: name 'Decimal' is not defined
```

Classifier files already loaded from the workspace and found duplicate in the package folder will be skipped:

```
[discovery] SKIP 183 duplicate label(s) skipped.
```

The number of intersection rules loaded from file intersections.toml:

```
[discovery] intersections OK: 24 from
C:\Users\<user>\Documents\Numclass\data\intersections.toml
```

The profile used and settings found in the loaded profile (partial example):

```
[debug] active profile: all
[debug] profile file: C:\Users\<user>\Documents\Numclass\profiles\all.toml
[debug] profile keys (raw → runtime value/type):
        _PROFILE.description............................. <description> (str)
        _PROFILE_.name................................... 'all' (str)
        ALIQUOT.MAX_STEPS................................ 250 (int)
        ALIQUOT.STEP_TIME_LIMIT.......................... 0.3 (float)
        ALIQUOT.VALUE_LIMIT.............................. 100000000000 (int)
        BEHAVIOUR.ALLOW_SLOW_CALCULATIONS................ False (bool)
        CATEGORIES.Arithmetic and Divisor-based.......... True (bool)
        CATEGORIES.Combinatorial and Geometric........... True (bool)
```

Now the standard input prompt is displayed, allowing you to enter an integer or an expression.

```
Number Classifier v2.0.0 — Mathematical Classifications & Curiosities

Profile: all — Enter an integer or profile (H=Help, P=List profiles, Q=Quit):
```

After entering, for example, 42, the classification results are displayed along with the time taken to complete the classification.

```
[  0.00 ms] NO   Highly composite number
[  2.61 ms] NO   Superabundant number
[  0.04 ms] NO   Superperfect number
[ 13.20 ms] NO   Untouchable number
[  0.00 ms] NO   Weird number
[  0.17 ms] OK   Egyptian fraction (K=5) — 5/42 = 1/9 + 1/126
[  0.01 ms] OK   Egyptian fraction (K=6) — 6/42 = 1/8 + 1/56
[  0.01 ms] OK   Erdős–Straus (K=4) — 4/42 = 1/11 + 1/232 + 1/53592
[  0.02 ms] OK   Goldbach conjecture — 42 has 4 Goldbach pairs: 5+37, 11+31, 13+29, 19+23
```

(partial example)

If any errors occur during classification, they will also be displayed here, including the corresponding line number.

# Further Resources

GitHub Repository: https://github.com/c788630/Numclass

# Appendix A – Classifications and Categories

This appendix provides reference descriptions for all 11 NumClass categories. Each table lists each available classifier with a short description / mathematical definition, followed by an example, and OEIS reference.

## Arithmetic and Divisor-based Numbers

The following categories contain divisibility- and divisor-function-based classifications.

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Abundant number** | A number is *abundant* if the sum of its proper divisors is greater than the number itself.<br>$s(n) > n$ | **20**: Aliquot sum (sum of proper divisors): s(20) = 1 + 2 + 4 + 5 + 10 = 22 (> 20)<br>OEIS A005101 |
| **Achilles number** | A *powerful number* that is not a perfect power; that is, all prime exponents in its factorization are ≥ 2 and their greatest common divisor equals 1. | **72**: 72 = $2^3 \times 3^2$. Both exponents ≥ 2, but gcd(3, 2) = 1, so 72 cannot be expressed as $m^k$ with m > 1 and k > 1.<br>OEIS A052486 |
| **Almost perfect number** | A number is *almost perfect* if the sum of its divisors σ(n) = 2n−1; only 1 and powers of 2 are known.<br>$\sigma(n) = 2n - 1$ | **8**: Sum of divisors: σ(8) = 1 + 2 + 4 + 8 =15 = 2 × 8−1<br>OEIS A000079 |
| **Amicable number** | Two distinct numbers a and b are *amicable numbers* if each equals the aliquot sum of the other.<br>s(a) = b and s(b) = a. | **220**: Aliquot sum: s(220) = 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 =284. s(284) = 1 + 2 + 4 + 71 + 142 = 220;<br>220 and 284 form an amicable pair since s(220) = 284 and s(284) = 220.<br>OEIS A063990 |
| **Aspiring number** | Number whose aliquot sequence eventually reaches a perfect number, though it is not perfect itself. | **25**: Aliquot sum: s(25) = 1 + 5 = 6, 6 is perfect since the sum of proper divisors s(6) = 1 + 2 + 3 = 6<br>⇒ 25 is an aspiring number.<br>OEIS A063769 |
| **Betrothed number (Quasi amicable number)** | A *betrothed pair* is a pair of positive integers (a, b) such that: s(a)=b+1 and s(b)=a+1. | **48**: Aliquot sum: s(48) = 1 + 2 + 3 + 4 + 6 + 8 + 12 + 16 + 24 = 76. s(75) = 1 + 3 + 5 + 15 + 25 = 49; satisfies s(48) = 75 + 1 and s(75) = 48 + 1. (48,75) is a betrothed pair.<br>OEIS A005276 |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Colossally abundant number** | A number is *colossally abundant* if it makes the ratio of the divisor sum to a slightly higher power of the number as large as possible for some positive ε, $\frac{\sigma(n)}{n^{1+\varepsilon}}$ is maximal for some $\varepsilon > 0$. | **12**: Sum of divisors: σ(12) = 1 + 2 + 3 + 4 + 6 +12 = 28 and for an appropriate small ε, $\frac{\sigma(12)}{12^{1+\varepsilon}}$ exceeds that ratio for all smaller numbers. For a concrete witness, take $\varepsilon$ = 0.2: $F_{12}(0.2) = 28 / 12^{1.2} \approx 1.4195$, while for $k$ = 1,..., 11 the values are: 1.0000, 1.3058, 1.0703, 1.3263, 0.8697, 1.3977, 0.7744, 1.2370, 0.9308, 1.1357, 0.6753, all smaller than 1.4195. OEIS A004490 |
| **Cube-free number** | A *cube-free* number has no prime factor raised to the third power or higher. | **30**: $30 = 2^1 \times 3^1 \times 5^1$ All prime factor exponents are ≤ 2. OEIS A004709 |
| **Cube-full number** | Integer where every prime exponent in its factorization is ≥ 3. | **216**: $216 = 2^3 \times 3^3$ All prime factor exponents are ≥ 3. OEIS A036966 |
| **Deficient number** | Number whose proper divisors sum to less than the number. $\mathrm{s(n)} < 2\mathrm{n}$ | **16**: Aliquot sum: s(16) = 1 + 2 + 4 + 8 = 15 (< 16) OEIS A005100 |
| **Descartes number** | A *Descartes number* is an odd positive integer satisfying σ(n)=2n except that one composite factor is treated as if prime. 198585576189 is the only known Descartes number | **198585576189**: 198585576189 = 19 × 43 × 67 × 1381 (with 1381 composite) OEIS A174292 |
| **Friendly number** | Two positive integers *m* and *n* are *friendly* if they share the same abundancy ratio, defined as $A(x) = \frac{\sigma(x)}{x}$ where σ(x) is the sum of all divisors of x. In other words, *m* and *n* are friendly if $\frac{\sigma(m)}{m} = \frac{\sigma(n)}{n}$. | **6**: Sum of divisors: σ(6) = 1 + 2 + 3 + 6 = 12 ⇒ A(6) = 12/6 = 2<br>28: σ(28) = 1 + 2 + 4 + 7 + 14 + 28 = 56 ⇒ A(28) = 56/28 = 2 ⇒ 6 and 28 are friendly numbers. OEIS A074902 |
| **Giuga number** | A *Giuga number* is a composite integer $n$ such that for every prime divisor $p \mid n$, $p \mid (\frac{n}{p} - 1)$. In other words, when you divide $n$ by each of its prime factors, subtract 1, the result must be divisible by that prime. | **30**: Prime divisors: 2 × 3 × 5 Check each prime divisor $\frac{n}{p} = 15{,}10{,}6$ :<br>• 15 - 1 is divisible by 2<br>• 10 - 1 is divisible by 3<br>• 6 - 1 is divisible by 5<br>All conditions are satisfied ⇒ 30 is a Giuga number. OEIS A007850 |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Hemi-perfect number** (half-perfect number) | A hemiperfect number is a positive integer whose abundancy index is a half-integer: $\frac{\sigma(n)}{n} = \frac{k}{2}$ for some odd integer $k$. Equivalently, $2\sigma(n)/n$ is an odd integer. All known hemiperfect numbers are even; the first few are $2, 24, 4320, 4680, 26208, \ldots$ | **24**: Sum of divisors: $\sigma(24) = 1+2+3+4+5+6+8+12+24 = 60$ Abundancy index: $\frac{\sigma(24)}{24} = \frac{60}{24} = \frac{5}{2}$, which is a half integer where 5 is odd ⇒ 24 is a hemiperfect number. OEIS A159907 |
| **Hexaperfect number** | A *hexaperfect number* (also called a 6-multiperfect number) is an integer whose sum of divisors equals six times the number itself. $\sigma(\mathrm{n}) = 6\mathrm{n}$ | **154345556085770649600**: σ(154345556085770649600) = 926073336514623897600 = 6 × 154345556085770649600 OEIS A046061 |
| **Highly abundant number** | A *highly abundant number* is a positive integer $n$such that the sum of its divisors $\sigma(n)$ is greater than that of any smaller positive integer. In other words, $n$ maximizes the divisor-sum function up to that point: $\sigma(n) > \sigma(k)$ for all $k < n$. | **16**: Sum of divisors: σ(16) = 1 + 2 + 4 + 8 + 16 = 31, among all integers $k < 16$, $\sigma(k) \leq 28$ the largest divisor sum is σ(12) = 28. Since 31 > 28, 16 has the greatest divisor sum among all numbers ≤ 16 ⇒ 16 is highly abundant. OEIS A002093 |
| **Highly composite number** | A *highly composite number* is a positive integer $n$that has more divisors than any smaller positive integer. That is, it maximizes the divisor-counting function $d(n)$(also written $\tau(n)$) up to that point: $d(n) > d(k)$for all $k < n$. | **12**: Divisors: 1, 2, 3, 4, 6, 12 $d(12) = 6$ For all smaller $k < 12$: $d(1) = 1,\ d(2) = 2,\ d(3) = 2,\ d(4) = 3,\ d(6) = 4,\ d(8) = 4,\ d(10) = 4$ None have more than 6 divisors ⇒ 12 is highly composite. OEIS A002182 |
| **k-Hyperperfect number** | A number $n$ is **k-hyperperfect** if for some integer $k \geq 1$: $n = 1 + k(\sigma(n) - n - 1)$. Equivalently: $k = \dfrac{n-1}{\sigma(n) - n - 1}$. In other words: The “divisor excess” $\sigma(n) - n - 1$ must divide $n - 1$. | **301**: Sum of divisors: σ(301) = 1 + 7 + 43 + 301 = 352. σ(n) − n − 1 = 352 − 301 − 1 = 50 (n − 1) / 50 = 300 / 50 = 6 k = 6 is an integer ⇒ 301 is a 6-hyperperfect number. OEIS A034897 |
| **k-Full number** | Positive integer where every prime exponent is at least k (e.g., 4-full). | **7776**: 7776 = $2^5 \times 3^5$, k = 5 OEIS A036967 (k = 4) |
| **Mersenne number** | A *Mersenne number* is a positive integer of the form $n = 2^p - 1$ for some integer $p \geq 1$. Mersenne numbers arise naturally in number theory because powers of 2 play a central role in divisibility, binary representation, and perfect numbers. | **2047**: 2047 = $2^{11} - 1$ 2047 = 23 × 89. 2047 is a Mersenne number, but it is not prime. |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Near-perfect number** | A positive integer $n$ is *near-perfect* if $n$ equals the sum of all its proper divisors except one. | **12**: Sum of proper divisors: s(12) = 1 + 2 + 3 + 4 + 6 = 16. If we omit m = 4, we get 1 + 2 + 3 + 6 = 12 ⇒12 is near-perfect, with redundant divisor 4. OEIS A181595 |
| **Ore (harmonic divisor) number** | A positive integer $n$ is an Ore number if the harmonic mean of its positive divisors is an integer. Let the divisors of $n$ be $d_1, d_2, \ldots, d_k$. Then $n$ is an Ore number if: $H(n) = \frac{k}{\sum_{i=1}^{k} \frac{1}{d_i}}$ is an integer. Using divisor functions, this can be written more compactly as: $H(n) = \frac{n \cdot \tau(n)}{\sigma(n)}$ and $n$ is Ore if and only if $\frac{n \cdot \tau(n)}{\sigma(n)} \in \mathbb{Z}$, where: $\sigma(n)$= sum of divisors $\tau(n)$= number of divisors | **6**: Divisors = 1, 2, 3, 6 $H(6) = \frac{4}{\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{6}} = \frac{4}{2} = 2$ 2 is an integer ⇒ 6 is an Ore number. OEIS A001599 |
| **Pentaperfect number** | A *pentaperfect number* (also called a 5-multiperfect number) is an integer whose sum of divisors equals five times the number itself. σ(n) = 5n. | **14182439040**: Sum of divisors: σ(14182439040) = 70912195200 = 5 × 14182439040 OEIS A046060 |
| **Perfect cube** | A *perfect cube* is a number equal to $k^3$ for some integer k. | **125**: 125 = $5^3$ OEIS A000578 |
| **Perfect number** | A *perfect number* is an integer equal to the sum of its proper divisors. s(n) = n, σ(n) = 2n. | **28**: Aliquot sum: s(28) = 1 + 2 + 4 + 7 + 14 = 28 OEIS A000396 |
| **Perfect power** | A *perfect power* is a number of the form $m^k$ with integers m > 1, k > 1. | **81**: 81 = $3^4$, m = 3, k = 4 OEIS A001597 |
| **Perfect square** | A *perfect square* is a number equal to $k^2$ for some integer k. | **100**: 100 = $10^2$ OEIS A000290 |
| **Perfect totient number** | A *perfect totient number* is an integer that equals the sum of its iterated totients (the count of integers ≤ n that are coprime to n), including the final 1. $n = \varphi(n) + \varphi(\varphi(n)) + \varphi\left(\varphi(\varphi(n))\right) + \cdots + 1$ | **9**: 9 = $3^2$ The formula for Euler's totient is: $\varphi(n) = n \prod_{p\mid n} \left(1 - \frac{1}{p}\right)$ where the product runs over the distinct prime factors $p$ of $n$. $\varphi(9) = 9\left(1 - \frac{1}{3}\right) = 9 \times \frac{2}{3} = 6$, $\varphi(6) = 2, \varphi(2) = 1$, Sum: 6 + 2 + 1 = 9 OEIS A082897 |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Polite number** | A *polite number* is a positive integer that can be expressed as the sum of two or more consecutive positive integers.<br>$n = a + (a+1) + (a+2) + \cdots + (a+k-1)$<br>for some integer $a \geq 1$and $k \geq 2$.<br>All integers except the powers of 2 are polite. | **9**: 9 = 4 + 5 and also:<br>9 = 2 + 3 + 4, ⇒ 9 is a polite number.<br>OEIS A138591 |
| **Powerful number**<br>(Squareful number) | A *powerful number* is a positive integer n in which every prime factor appears with an exponent of at least 2.<br>If $n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$ is the prime factorization of n, then n is powerful if $a_i \geq 2$for every $i$. | **72**: Prime factors: $2^3 \times 3^2$, All exponents are ≥ 2 ⇒ 72 is a powerful number.<br>OEIS A001694 |
| **Practical number** | A practical number is a positive integer n such that every smaller positive integer (from 1 up to n − 1) can be expressed as a sum of distinct divisors of n.<br>In other words, using only the divisors of n (each at most once), you can form *any* smaller number by addition. | **12**: Divisors : 1, 2, 3, 4, 6, 12.<br>All integers 1-11 can be expressed as sums of distinct divisors ⇒ 12 is a practical number.<br>OEIS A005153 |
| **Primary pseudoperfect number** | A positive integer $n > 1$ is called a *primary pseudoperfect number* if $\sum_{p\mid n} \frac{1}{p} + \frac{1}{n} = 1$ where the sum is taken over all distinct prime divisors $p$ of $n$. | **42**: Prime divisors: 2 × 3 × 7:<br>$\frac{1}{2} + \frac{1}{3} + \frac{1}{7} + \frac{1}{42} = \frac{21+14+6+1}{42} = \frac{42}{42} = 1$ ⇒ 42 is a primary pseudoperfect number.<br>OEIS A054377 |
| **Primitive weird number** | A *primitive weird number* is a weird number that has no weird proper divisors — that is, it is *weird* but not a multiple of any smaller weird number. | **70**: Sum of proper divisors: s(70) = 1 + 2 + 5 + 7 + 10 + 14 + 35 = 74 > 70 → abundant.<br>No subset of {1, 2, 5, 7, 10, 14, 35} adds up to 70 → not semi-perfect, So 70 is a weird number, and since there are no smaller weird numbers dividing 70, it is also a primitive weird number.<br>OEIS A006037 |
| **q-aspiring number** | Under the reduced aliquot iteration $q(n) = s(n) - 1$, the sequence reaches a perfect number. | **8**: Nontrivial proper divisors 2,4. q(8) = s(8)-1 = (1 + 2 + 4) - 1 = 6 (equivalently 2 + 4 = 6).<br>q-sequence: 8 → 6 → 5 → 0, the sequence hits perfect 6 at step 1 ⇒ 8 is q-aspiring.<br>OEIS N/A |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **q-sociable number** (Reduced sociable number) | Under the reduced aliquot iteration $q(n) = s(n) - 1$, cycles back to the starting number, with a minimum cycle length ≥ 3. | **1215571544**: Sequence $q(n)$: 1215571544 → 1270824975 → 1467511664 → 1530808335 → 1579407344 → 1638031815 → 1727239544 → 1512587175 → 1215571544 (cycle length = 8) OEIS A309227 |
| **q-socially aspiring number** | Under the reduced aliquot iteration $q(n) = s(n) - 1$, eventually reaches a finite q-cycle of length ≥ 2 that does not contain n itself. | **92**: Sum of proper divisors: q(92) = 2 + 4 + 23 + 46 = 75, ⇒ q(75) = 48, q(48) = 75 So the q-sequence is: 92 → 75 → 48 → 75 → 48 → ⋯ The pair {75,48} is a q-cycle of length 2. 92 is not in the cycle, and it enters it at step 1. ⇒ 92 is a q-socially aspiring number. OEIS N/A |
| **Quadriperfect number** | A *quadriperfect number* (also called a 4-multiperfect number) is an integer whose sum of divisors equals four times the number itself. $\Sigma(\mathrm{n}) = 4\mathrm{n}$ | **30240**: 96 divisors summing to σ(30240) = 120960 = 4 × 30240 → 30240 is a quadriperfect number. OEIS A027687 |
| **Quasi-sociable number** | Under s-iteration, the aliquot sequence enters a sociable cycle (len≥2) not containing n. | **562**: Sum of proper divisors: s(562) = 284, the s-sequence is: 562 → 284 → 220 → 284 → ... This enters a sociable cycle of k = 2 at step 1, and 562 is not in the cycle ⇒ it's quasi-sociable. OEIS A309227 |
| **Semiperfect number** (Pseudoperfect number) | A *semiperfect number* is a positive integer that equals the sum of some (or all) of its proper divisors. | **20**: Proper divisors: 1,2,4,5,10. check subsets sums to 20: 10 + 5 + 4 + 1 = 20 ⇒ 20 is a semiperfect number. OEIS A005835 |
| **Sociable number** | Aliquot sequence forms a cycle that returns to the starting number. | **12496**: Aliquot sequence: s(12496) = 12496 → 14288 → 15472 → 14536 → 14264 → 12496 OEIS A003416 |
| **Socially aspiring number** | Aliquot sequence reaches an amicable or sociable cycle without including n in that cycle. | **562**: Aliquot sequence: s(562) = 284, s(284) = 220, s(220) = 284. The pair 220 ↔ 284 is an amicable (sociable-2) cycle. 562 reaches that 2-cycle at step 1, but is not itself in it ⇒ 562 is a socially aspiring number (it enters a sociable 2-cycle at step 1). OEIS A121508 |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Solitary number** | Shares its abundancy ratio with no other integer; sufficient condition: gcd(n, σ(n)) = 1. | **3**: Sum of divisors:<br>σ(3) = 1 + 3 = 4<br>$\frac{\sigma(3)}{3} = \frac{4}{3}$<br>gcd(3, 4) = 1 which satisfies the condition ⇒ 3 is a solitary number.<br>OEIS N/A |
| **Sphenic number** | A *sphenic number* is a positive integer that is the product of three distinct prime numbers, each appearing exactly once in the factorization. | 30: Prime factors:<br>30 = 2 × 3 × 5.<br>All three factors are prime and distinct ⇒ 30 is a sphenic number.<br>OEIS A007304 |
| **Squarefree number** | A *square-free number* is a positive integer where no prime factor appears with an exponent > 1. | **42**: 42 = $2^1 \times 3^1 \times 7^1$<br>Each prime factor appears only once ⇒ 42 is square-free.<br>OEIS A005117 |
| **Sublime number** | Number with a perfect number of divisors, and their sum is also perfect. Only two sublime numbers are known. | **12**: Sum of divisors:<br>σ(12) = 1 + 2 + 3 + 4 + 6 + 12 = 28, number of divisors = 6. Both 6 and 28 are perfect numbers ⇒ 12 is sublime.<br>OEIS A081357 |
| **Superabundant number** | A number $n$ is *superabundant* if $\frac{\sigma(n)}{n} > \frac{\sigma(m)}{m}$ for every $m < n$. | **12**: Sum of divisors:<br>σ(12) = 1 + 2 + 3 + 4 + 6 + 12 = 28<br>Ratio:<br>$\frac{\sigma(12)}{12} = \frac{28}{12} = 2.333\ldots$<br>All other smaller numbers do not have a higher σ(12n)/12 ratio ⇒ 12 is superabundant.<br>OEIS A004394 |
| **Superperfect number** | A superperfect number is a positive integer n for which the sum of the divisors of the sum of its divisors equals twice the number itself. σ(σ(n)) = 2n. | **16**: Sum of divisors:<br>σ(16) = 1 + 2 + 4 + 8 + 16 = 31<br>σ(31) = 1 + 31 = 32<br>σ(σ(16)) = 32 = 2 × 16 ⇒ 16 is superperfect.<br>OEIS A019279 |
| **Triperfect number** | A *triperfect number* (also called a 3-perfect number) is an integer whose sum of divisors equals three times the number itself. $\Sigma(\mathrm{n}) = 3\mathrm{n}$ | **120**: Sum of divisors:<br>σ(120) = 1 + 2 + 3 + 4 + 5 + 6 + 8 + 10 + 12 + 15 + 20 + 24 + 30 + 40 + 60 + 120 = 360<br>360 / 120 = 3<br>OEIS A005820 |
| **Unitary perfect number** | A *unitary perfect number* is a positive integer $n$ for which the sum of its unitary divisors equals twice the number itself:<br>$\sigma^*(n) = 2n$ | **60**: Prime factors: $2^2 \times 3 \times 5$<br>σ*(60) = $(1 + 2^2)(1 + 3)(1 + 5)$ = 5 × 4 × 6 = 120. 120 / 60 = 2 ⇒ 60 is a unitary perfect number.<br>OEIS A002827 |

| Classifier | Description / Definition | Examples / OEIS reference |
|---|---|---|
| **Untouchable number** | An *untouchable number* is a positive integer that cannot be written as the sum of the proper divisors s(m) for any m > 0.<br>In other words, no integer has this number as its *aliquot sum*. | **5**: For all primes p s(p) = 1, leaving to check: s(1) = 0, s(4) = 3, s(6) = 6, s(8) = 7. After m = 8, s(m) already exceeds 5 for all n so no new n can give 5. (the only known odd one).<br>OEIS A005114 |
| **Weird number** | A *Weird number* is an abundant number that is not semiperfect**,** that is, the sum of its proper divisors exceeds the number itself, but no subset of those divisors adds up exactly to the number. | **70**: Sum of proper divisors: s(70) = 1 + 2 + 5 + 7 + 10 + 14 + 35 = 74 > 70. However, there is no subset of 1,2,5,7,10,14,35 that sums exactly to 70 ⇒ 70 is weird, the smallest known.<br>OEIS A006037 |
| **Zumkeller number** | A *Zumkeller number* is a positive even integer whose divisors can be partitioned into two disjoint subsets having equal sums. | **12**: Sum of divisors:<br>σ(12) = 1 + 2 + 3 + 4 + 6 + 12 = 28, 28 / 2 = 14. We can partition the divisors into: $\{1,3,4,6\}$ and $\{2,12\}$.<br>Since 1 + 3 + 4 + 6 =14 = 2 +12.<br>⇒ 12 is a Zumkeller number.<br>OEIS A083207 |

## Combinatorial and Geometric Numbers

Numbers that arise from discrete counting principles — representing partitions, permutations, arrangements lattice paths or geometric divisions.

Note: Some figurate numbers (like Tetrahedral or Triangular) can also be expressed combinatorially via binomial coefficients. In this manual, they are grouped with Polygonal and Figurate Numbers for their geometric interpretation.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Bell number** | A *Bell number* counts the number of ways a set with n elements can be partitioned into non-empty, disjoint subset.<br>$B(n) = \sum_{k=0}^{n} S(n,k)$<br>where $S(n,k)$is a Stirling number of the second kind (the number of ways to divide $n$ objects into $k$ non-empty groups). | **3**: The set {a, b, c} can be partitioned in 5 ways:<br>1. {{a, b, c}}<br>2. {{a, b}, {c}}<br>3. {{a, c}, {b}}<br>4. {{b, c}, {a}}<br>5. {{a}, {b}, {c}}<br>Hence B(3) = 5.<br>OEIS A000110 |
| **Cake number** (Maximum piece number) | A *cake number* gives the maximum number of pieces into which a solid cake (or any 3-dimensional object) can be divided using *n* straight planar cuts. Each new cut can intersect all previous cuts, thereby maximizing the number of separate regions produced. | **26**: = $C_n$<br>Multiply both sides by 6:<br>$6C_n = n^3 + 5n + 6$<br><br>Rearrange into a cubic in $n$:<br>$n^3 + 5n + (6 - 6C_n) = 0$<br><br>Let $P = 5,\ Q = 6 - 6C_n$ = |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| | Mathematically, the sequence counts regions of space cut by $n$ planes in general position.<br>The $n$-th cake number $C_n$ is given by:<br>$C_n = \frac{n^3 + 5n + 6}{6}$<br>and satisfies the recurrence<br>$C_n = C_{n-1} + \binom{n}{2} + 1$<br>Reverse function according to Cardano's formula for the real root of $x^3 + Px + Q = 0$:<br>$x = \sqrt[3]{-\frac{Q}{2} + \sqrt{\left(\frac{Q}{2}\right)^2 + \left(\frac{P}{3}\right)^3}} + \sqrt[3]{-\frac{Q}{2} - \sqrt{\left(\frac{Q}{2}\right)^2 + \left(\frac{P}{3}\right)^3}}$ | $6 - 6 \times 26 = -150$.<br><br>$n = \sqrt[3]{-\frac{-150}{2} + \sqrt{\left(\frac{-15}{2}\right)^2 + \left(\frac{5}{3}\right)^3}} + \sqrt[3]{-\frac{-15}{2} - \sqrt{\left(\frac{-150}{2}\right)^2 + \left(\frac{5}{3}\right)^3}}$<br>$n = \sqrt[3]{75 + \sqrt{5629.63\ldots}} + \sqrt[3]{75 - \sqrt{5629.63\ldots}} = 5$<br><br>⇒ 26 pieces correspond to 5 straight cuts.<br>OEIS A000125 |
| **Catalan number** | The *Catalan numbers* form a famous sequence in combinatorics that counts many kinds of structured objects.<br>They appear in problems involving balanced parentheses, binary trees, polygon triangulations, and lattice paths. The $n$-th Catalan number is given by the formula:<br>$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\ n!}$ | **3**: $C_3 = \frac{1}{4}\binom{6}{3} = \frac{1}{4} \times 20 = 5.$<br>Interpretation: There are 5 distinct ways to correctly match 3 pairs of parentheses:<br>((())), (()()), (())(), ()(()), ()()()<br>or equivalently, 5 distinct binary trees with 3 internal nodes.<br>OEIS A000108 |
| **Fubini number** (Ordered Bell number) | A *Fubini number* counts the number of weak orders or total preorders on an $n$-element set.<br>Equivalently, it counts the number of ways to partition a set of $n$ labeled elements into non-empty subsets where each block is ordered, and the blocks themselves are ordered.<br>$F(n) = \sum_{k=1}^{n} k!\ S(n,k)$ | **13**: verify from the formula:<br>$F(n) = \sum_{k=0}^{n} k!\ S(n,k)$<br>Take n=3. Using Stirling numbers of the second kind $S(3,k)$:<br>S(3,1) = 1<br>S(3,2) = 3<br>S(3,3) = 1<br>F(3) = 1! × 1 + 2! × 3 + 3! × 1 = 1 + 6 + 6 = 13 ⇒ 13 is the third Fubini number.<br>OEIS A000670 |
| **Lah number** | A *Lah number*, written $L(n,k)$, counts the number of ways to partition $n$ labeled elements into $k$ non-empty linearly ordered subsets (i.e. ordered *lists*, not just sets). Unlike Stirling numbers (which count unordered subsets), Lah numbers take ordering within each subset into account.<br>$L(n,k) = \frac{n!}{k!}\binom{n-1}{k-1}$ | **141120**: Take $n = 8$<br>For $k = 2$:<br>$L(8,2) = \frac{8!}{2!}\binom{7}{1} = \frac{40320}{2} \times 7 = 20160 \times 7 = 141120$<br><br>For $k = 3$:<br>$L(8,3) = \frac{8!}{3!}\binom{7}{2} = \frac{40320}{6} \times 21 = 6720 \times 21 = 141120$<br><br>⇒ $141120 = L(8,2) = L(8,3)$ |

<table>
<tr><th>Classifier</th><th>Definition / Description</th><th>Examples / OEIS</th></tr>
<tr><td></td><td>for integers $n, k \geq 1$.</td><td>OEIS A105278</td></tr>
<tr><td>Motzkin number</td><td>A Motzkin number $M(n)$ counts the number of lattice paths of length $n$ that start at (0, 0), end at (n, 0), and never go below the x-axis, using only the steps Up (1, 1), Flat (1, 0), Down (1, −1). Equivalently, it counts the number of ways to draw non-intersecting chords connecting n labeled points on a circle, or the number of non-crossing matchings in combinatorics.<br>The numbers satisfy the recurrence $M(0) = 1, M(1) = 1, M(n) = M(n-1) + \sum_{k=0}^{n-2} M(k)\, M(n-2-k)\ (n \geq 2)$,<br>and can also be expressed as<br>$M(n) = \sum_{k=0}^{\left\lfloor \frac{n}{2} \right\rfloor} \binom{n}{2k} C_k$,<br>where $C_k$is the k-th Catalan number.</td><td>4: M(0) = 1, M(1) = 1 (M2) = 2, M(3) = 4, M(4) = 9, M(5) = 21, ...<br>Illustration (n = 3):<br>All paths of length 3 from (0, 0) to (3, 0) staying above the x-axis:<br>1. F F F (flat,flat,flat)<br>2. U D F (up,down,flat)<br>3. U F D (up,flat,down)<br>4. F U D (flat,up,down)<br>Hence M(3) = 4 ⇒ 4 is a Motzkin number.<br>OEIS A001006</td></tr>
<tr><td>Partition number</td><td>The partition number $p(n)$counts the number of distinct ways a positive integer $n$ can be written as a sum of positive integers, disregarding order of the terms.<br>Formally, each decomposition<br>$n = a_1 + a_2 + \cdots + a_k (a_i \geq 1)$<br>is a partition of $n$, where the sequence of summands is unordered (so 4 = 3 + 1 = 1 + 3 counts as one partition).</td><td>7:<table>
<tr><td>N</td><td>partitions</td><td>p(n)</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2; 1+1</td><td>2</td></tr>
<tr><td>3</td><td>3; 2+1; 1+1+1</td><td>3</td></tr>
<tr><td>4</td><td>4; 3+1; 2+2; 2+1+1; 1+1+1+1</td><td>5</td></tr>
<tr><td>5</td><td>5; 4+1; 3+2; 3+1+1; 2+2+1; 2+1+1+1; 1+1+1+1+1</td><td>7</td></tr>
</table>⇒ partition number p(5) = 7<br>OEIS A000041</td></tr>
<tr><td>Ramsey number (2-colour, exact)</td><td>A 2-colour Ramsey number $R(r,s)$ is the smallest integer $N$ such that every red/blue colouring of the edges of the complete graph $K_N$ contains either a red $K_r$ or a blue $K_s$. Because the definition is symmetric, $R(r,s) = R(s,r)$.<br>Only a few values are known exactly. Small exact values include $R(3,3) = 6,\ R(3,4) = 9,\ R(3,5) = 14,\ R(4,4) = 18,\ \ldots$<br>For diagonal Ramsey numbers $R(k,k)$, exact values are known only for $k \leq 4$. For $k \geq 5$, a conjectured estimate is used for $(k,k)$.<br>$a(0) = 0, a(1) = 1, a(2) = 2,$<br>$a(k) = \left\lceil \left(\frac{3}{2}\right)^{k-3} k(k-1) \right\rceil\ (k \geq 3)$</td><td>6: For example, R(3,3)=6: in any red/blue coloring of the edges of a complete graph on six vertices, there must be a monochromatic triangle (a 3-clique all in one color), whereas for five vertices it is still possible to avoid one.<br>N=6 ⇒ R(3,3)=6<br>Any 2-coloring of the graph $K_6$ will contain either a red triangle $K_3$or a blue triangle $K_3$. ⇒ 6 = Ramsey number R(3,3)<br>OEIS A059442 (exact values)<br>OEIS A120414 (conjectured diagonal values)</td></tr>
</table>

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Schur number (v1) exact** | The Schur number v1 $S_1(k)$ is the largest integer $n$ such that the entire interval {1,2,...,n} can be colored with k colors with the property that there is no monochromatic solution to the equation x + y = z.<br>Equivalently:<br>$S_1(k)$ is the maximum length of an initial interval that avoids any single-colored triple $(x, y, z)$ with $x + y = z$.<br>Beyond $n = S_1(k)$, any k-coloring of $1 \dots n$ necessarily contains such a triple. These numbers are smaller than the classical Schur numbers (v2), because v1 requires coloring the full interval rather than arbitrary subsets. | 5: For k = 2: $S_1(2) = 5$.<br><br>• There *exists* a 2-colouring of {1,2,3,4,5} avoiding any single-color solution to $x + y = z$.<br>• But for {1,2,3,4,5,6}, *every* 2-colouring gives a monochromatic sum triple (e.g. 1+5=6 or 2+4=6 in one color). ⇒ 5 is maximal, so $5 = S_1(2)$<br>OEIS A030126 |
| **Stirling number (unsigned, 1st kind)** | The *Stirling numbers* of the first kind, written as $c(n, k)$ or $\left[{n \atop k}\right]$, count how many permutations of $n$ distinct elements have exactly $k$ cycles in their cycle decomposition, in other words:<br>$\left[{n \atop k}\right]$= number of ways to arrange $n$ objects into $k$ disjoint cycles.<br>These numbers arise naturally when expressing falling factorials in terms of ordinary powers, or in studying the structure of permutations. | **11**: For n=4:<br>The permutations of 4 elements can be partitioned by number of cycles:<br>• c(4,1)=6 permutations with 1 cycle (the 6 full 4-cycles)<br>• c(4,2)=11 permutations with 2 cycles<br>• c(4,3)=6 permutations with 3 cycles<br>• c(4,4)=1 permutation with 4 cycles (the identity)<br>Thus the 4th row of the unsigned triangle is:<br>4! = 24 = c(4,1) + c(4,2) + c(4,3) + c(4,4) = 6 + 11 + 6 + 1 ⇒ 11 = Stirling number 1$^{st}$ kind c(4,2)<br>OEIS A132393 |
| **Stirling number (2nd kind)** | The *Stirling numbers of the second kind*, written as $S(n, k)$ or $\left\{{n \atop k}\right\}$, count the number of ways to partition a set of $n$ distinct elements into $k$ non-empty, unlabeled subsets (called *blocks*), in other words:<br>$\left\{{n \atop k}\right\}$= number of ways to divide $n$ objects into $k$ groups, ignoring the order of the groups. | **7**: For n=4 and k=2 S(4,2)=7.<br>There are 7 ways to partition the set {1,2,3,4} into 2 non-empty subsets:<br>1. {1,2,3}, {4}<br>2. {1,2,4}, {3}<br>3. {1,3,4}, {2}<br>4. {2,3,4}, {1}<br>5. {1,2}, {3,4}<br>6. {1,3}, {2,4}<br>7. {1,4}, {2,3}<br>⇒ 7 = Stirling number 2$^{nd}$ kind S(4,2)<br>OEIS A008277 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Van der Waerden number (exact)** | A *Van der Waerden* number $W(r,k)$ is the smallest positive integer $N$ such that every r-coloring of the integers $\{1,2,\ldots,N\}$ contains a monochromatic arithmetic progression of length k. In other words: no matter how you color the first $N$ integers with r colors, you cannot avoid producing a progression $a, a+d, a+2d, \ldots, a+(k-1)d$ whose elements all share the same color. | **9**: W(2,3) = 9 Any coloring of {1,...,9} with two colors will contain a monochromatic arithmetic progression of three terms (e.g., 1–5–9 or 2–5–8). For n = 8 it can still be avoided, but for n = 9 the property is unavoidable ⇒ 9 is a Van der Waerden number. OEIS A005346 |

## Conjectures and Equation-based Numbers

Numbers appearing in or related to classical conjectures and Diophantine equations.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Egyptian m/n** | Searches m/n as a sum of k unit fractions. $\frac{4}{n} = \frac{1}{x} + \frac{1}{y}$ Use egyptian_m_value and egyptian_k_value in profile, default: m = 4, k = 2 | **11**: $\frac{4}{11} = \frac{1}{3} + \frac{1}{33}$ (for m = 1, n = 11, k = 4) OEIS A192881 |
| **Erdős–Straus** | Tests whether $\frac{4}{n} = \frac{1}{x} + \frac{1}{y} + \frac{1}{z}$ has a solution in positive distinct integers. Conjectured true for all n ≥ 2. | **5**: $\frac{4}{5} = \frac{1}{2} + \frac{1}{4} + \frac{1}{20}$ OEIS A073101 |
| **Goldbach conjecture** | Every even integer > 2 can be written as the sum of two primes. | **8**: 3 + 5 = 8 OEIS A002372 |
| **Legendre prime interval** | *Legendre's conjecture* states that for every n > 0 there exists at least one prime p such that $n^2 < p < (n+1)^2$. | **5**: n = 5, $n^2 = 25$, $(n+1)^2 = 36$, in that range we have two primes: 29 and 31. OEIS A014085 |
| **Lemoine's (Levy's)** | Every odd integer > 5 can be written as p + 2q with primes p, q. | **53**: 47 + 2 × 3 = 53 (has 5 different solutions in total.) OEIS A194828 |
| **Sierpiński** | *Sierpiński's conjecture* asserts that for every integer n ≥ 2, the Diophantine equation 5/n = 1/x + 1/y + 1/z + 1/w has a solution in positive integers. | **9**: $\frac{5}{9} = \frac{1}{3} + \frac{1}{9} + \frac{1}{10} + \frac{1}{90}$ OEIS N/A |
| **Weak Goldbach (ternary)** | Every odd integer ≥ 7 can be expressed as the sum of three primes. | **91**: 3 + 5 + 83 = 91 (91 has 83 different solutions in total.) OEIS A068307 |

| | | |
|---|---|---|
| **Znám chain** | A proper Znám set {$a_1$,...,$a_k$} satisfies $\sum_{i=1}^{k}\frac{1}{a_i} = 1 - \frac{1}{(a^1a^2 \ldots a_k)}$<br>The Znám chain of length k finds such a set.<br>k range: proper version $k \in [5,10]$; improper version: $k \in [2,10]$ and allows the largest element to equal the product of the others plus 1 (i.e., $a_{\max} = \prod_{i\neq\max} a_i + 1$). | **6**: $S = \frac{1}{2}+\frac{1}{3}+\frac{1}{7}+\frac{1}{47}+\frac{1}{403}+\frac{1}{19403} = 1 - \frac{1}{P}$<br>*P* = 2 × 3 × 7 × 47 × 403 × 19403 = 15435513366<br>OEIS A075441 |

## Digit-based Numbers

Numbers defined by patterns in their digits (usually base 10).

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Automorphic number** | An *Automorphic number* is an integer whose square ends in the same digits as the number itself.<br>$n^2$ ends with n. | **76**: $76^2$ = 5776<br>OEIS A003226 |
| **Brazilian number** | A *Brazilian number* can be represented as a repdigit (repeating copies of the same digit) in some base b with 2 ≤ b ≤ n – 2. | **72**: In base 11, 72 is a repdigit of length 2 with digit 6: 66<br>$6 \times R_2(11), where\ R_2(11) = \frac{11^2-1}{11-1} = \frac{120}{10} = 12$<br>n = 6 × 12 = 72<br>OEIS A125134 |
| **Disarium number** | A *Disarium number* is a number that is equal to the sum of its digits each raised to the power of their respective positions (counting from left to right, starting at 1).<br>It has been proven that no term can have more than 22 digits, and a complete search up to $10^{22}$ shows that there are only 20 Disarium numbers with the last being: 12157692622039623539. | **135**: $1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135$<br>⇒ 135 is a Disarium number.<br>OEIS A032799 |
| **Dudeney number** | A *Dudeney number* (sometimes spelled Dudeny number) has the cube of the sum of digits equal to the number itself.<br>( (sum of digits)$^3$ = n )<br>Only six such numbers are known. | **512**: 5 + 1 + 2 = 8, $8^3$ = 512<br>⇒ 512 is a Dudeney number.<br>OEIS A061209 |
| **Evil number** | An *Evil number* is a non-negative integer that has an even number of 1-bits in its binary (base-2) representation. | **9**: $9_{10} = 1001_2$<br>The binary form has two 1's, which is an even number.<br>⇒ 9 is an Evil number.<br>OEIS A001969 |
| **Factorion** | A *Factorion* is a number that is equal to the sum of the factorials of its digits. | **145**: 1! + 4! + 5! = 1 + 24 + 120 = 145 ⇒ 145 is a Factorion.<br>OEIS A014080 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Fibonacci-concatenation** | A *Fibonacci concatenation* number is a number formed by writing several consecutive Fibonacci numbers next to each other (seed 1,1 or 0,1). | **11235813**: Fibonacci numbers, (seed 1,1) 1, 1, 2, 3, 5, 8, 13, ... Concatenated digits: 11235813 ⇒ is a Fibonacci-concatenation. OEIS A019523 |
| **Grafting number** | *Grafting numbers* are integers that, when expressed in base 10, will appear in their own square root before or directly after the decimal point (ignoring leading 0's and including trailing 0's). | **765**: $\sqrt{765} \approx \mathbf{27.65}863$ the square root contains the digits 7. 65 around the decimal point ⇒ 765 is a grafting number. OEIS A232087 |
| **Happy number** | A *Happy number* is a number that eventually reaches 1 when you repeatedly replace it by the sum of the squares of its digits. | **19**: : $1^2 + 9^2 = 82, 8^2 + 2^2 = 68, 6^2 + 8^2 = 100, 1^2 + 0^2 + 0^2 = 1$ ⇒ 19 is a Happy number. OEIS A007770 |
| **Harshad number (Niven number)** | A *Harshad number* is an integer that is divisible by the sum of its digits in a given base (usually base 10). | 18: Sum of digits: 1 + 8 = 9, 18 ÷ 9 = 2 ⇒ 18 is a Harshad number. OEIS A005349 |
| **Hoax number** | A *Hoax number* is a composite number whose digit sum equals the sum of the digits of its distinct prime factors. | **319**: Prime factors: 11 × 29 Sum of factor digits: 1 + 1 + 2 + 9 = 13. Sum of digits of 319: 3 + 1 + 9 = 13 ⇒ 319 is a Hoax number. OEIS A019506 |
| **Kaprekar number** | A *Kaprekar number* is a non-negative integer whose square can be split into two parts that, when added together, give back the original number. | **45**: $45^2$ = 2025, split into 20 and 25: 20 + 25 = 45 ⇒ 45 is a Kaprekar number. OEIS A006886 |
| **Lychrel number** | A *Lychrel number* is a natural number that is suspected never to form a palindrome through the reverse-and-add process. NumClass sets the limit to 1000 iterations. After reaching the limit the number is considered to be a Lychrel number candidate. | **196**: 196 + 691 = 887, 887 + 788 = 1675, 1675 + 5761 = 7436, 7436 + 6347 = 13783, etc. No palindrome found after 1000 iterations ⇒ 196 is a Lychrel number candidate. OEIS A023108 |
| **Moran number** | A *Moran number* is a positive integer that is divisible by the sum of its digits, and when divided by that sum, the quotient is a prime number. | **133**: Sum of digits: 1 + 3 + 3 = 7. 133 ÷ 7 = 19, 19 is prime ⇒ 133 is a Moran number. OEIS A001101 |
| **Narcissistic number Armstrong number** | A Narcissistic number is a number that is equal to the sum of its own digits each raised to the power of the number of digits. | **153**: has 3 digits: $1^3 + 5^3 + 3^3$ = 1 + 125 + 27 = 153 ⇒ 153 is a Narcissistic number. OEIS A005188 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Odious number** | An *Odious number* is a non-negative integer that has an odd number of 1-bits in its binary (base-2) representation. | **11**: $11_{10} = 1011_2$<br>The binary form has three 1's, which is an odd number.<br>⇒ 11 is an Odious number.<br>OEIS A000069 |
| **Palindrome** | A *Palindrome* reads identically forward and backward in decimal notation. | **13631**: Reads the same forward and backward ⇒ 13631 is a Palindrome.<br>OEIS A002113 |
| **Palindromic Harshad number** (intersection) | A *Palindromic Harshad number* is a Harshad number that is also a palindrome in decimal form. | **111**: Sum of digits: 1 + 1 + 1 = 3<br>111 ÷ 3 = 37 ⇒ 111 is a Harshad number and palindromic.<br>OEIS A082232 |
| **Repdigit** | A *Repdigit* (short for *repeated digit*) is a positive integer composed of repeated occurrences of the same digit in a given base — most often base 10.<br>In other words, all its digits are identical. A Repdigit in base b can be written as:<br>$n = d \times \frac{b^k - 1}{b - 1}$<br>Where d is the repeated digit and k is the number of times the digit repeats. (One digit numbers are included by definition.) | **777**: In base 10: number 7, 3 digits:<br>$7 \times \frac{10^3-1}{10-1} = 7 \times \frac{999}{9} = 7 \times 111 = 777$<br>⇒ 777 is a Repdigit (digit 7 repeated 3 times).<br>OEIS A010785 |
| **Self number** (Devlali number) (Colombian number) | A *Self number* is a positive integer that cannot be generated by any other number through the function $f(n) = n +$ sum of digits of $n$<br>In other words, there is no integer $m$ such that<br>$m +$ sum of digits of $m = n$<br>If such an $m$ does not exist, then it "creates itself," because no other number produces it. | **20**: Try to find $m$ such that $m +$ sum of digits of $m = 20$.<br>• 15 + (1 + 5) = 21<br>• 14 + (1 + 4) = 19<br>• 13 + (1 + 3) = 17<br>• 12 + (1 + 2) = 15<br>None produce 20 ⇒ 20 is a Self number.<br>OEIS A003052 |
| **Self-descriptive number** (Self-documenting number) | A *Self-descriptive number* is an integer that describes itself — each digit in the number tells how many times a particular digit appears in the whole number.<br>In base 10, the digit in position *i* (counting from 0 on the left) represents how many times the digit *i* occurs in the entire number. | **6210001000**:<br>Position 0 1 2 3 4 5 6 7 8 9<br>Digit 6 2 1 0 0 0 1 0 0 0<br>Check:<br>There are six 0's, two 1's, one 2, zero 3's, zero 4's, zero 5's, one 6, zero 7's, 8's and 9's .<br>All conditions hold ⇒ 6210001000 is a Self-descriptive number.<br>OEIS A108551 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Smith number** (Joke number) | A *Smith number* is a composite number whose sum of digits equals the sum of the digits of its prime factors (counted with multiplicity). In other words, when you factorize the number into primes, and add up all the digits appearing in those factors, the total matches the sum of the digits of the original number. | **666**: Prime factors: $2 \times 3 \times 3 \times 37$, Sum of digits: $6 + 6 + 6 = 18$. Sum of digits of all prime factors (with multiplicity): $(2) + (3 + 3) + (3 + 7) = 2 + 3 + 3 + 3 + 7 = 18 \Rightarrow 666$ is a Smith number. OEIS A006753 |
| **Sum of 2 palindromes** | Can be expressed as the sum of two palindromic numbers in base 10. | **167**: 66 + 101 = 167 OEIS A035137 |
| **Sum of 3 palindromes** | Can be expressed as the sum of three palindromic numbers in base 10. It is proved that every integer is a sum of three palindromes. | **500**: 77 + 151 + 272 = 500 OEIS A261132 |

## Diophantine Representations

Numbers expressible as sums of integer powers, forming classical Diophantine identities.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Sum of 2 cubes** | Can be written as $n = a^3 + b^3$ for integers a, b. | **28**: $1^3 + 3^3 = 1 + 27 = 28$ OEIS A003325 |
| **Sum of 2 squares** | Can be written as $n = a^2 + b^2$ for integers a, b. | **13**: $2^2 + 3^2 = 9 + 4 = 13$ OEIS A001481 |
| **Sum of 3 cubes** | Can be written as $n = a^3 + b^3 + c^3$ for integers a, b, c. | **7**: $(-105)^3 + 32^3 + 104^3 = 7$ OEIS A060464 |
| **Sum of 3 squares** | Can be written as $n = a^2 + b^2 + c^2$ for integers a, b, c. | **29:** $2^2 + 3^2 + 4^2 = 4 + 9 + 16 = 29$ OEIS A000378 |

## Dynamical Sequences

Numbers generated or studied through iterative or recursive rules.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Collatz (3n + 1) sequence** | Defined by the rule: n → n / 2 if n is even, else n → 3n + 1; conjectured that every positive integer eventually reaches 1. | **6**: sequence $6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$ 8 steps, 6 even, 2 odd, odd/even $1{:}3 \approx 0.333$ $(25.0\%)$, peak value $16$. OEIS A006370 |
| **Generalized Collatz (5n + 1)** | Defined by the rule: n → n / 2 if n is even, else n → 5n + 1 | **10**: sequence $10 \to 5 \to 26 \to 13 \to 66 \to 33 \to 166 \to 83 \to 416 \to 208 \to 104 \to 52 \to 26$. Loops back to 26, peak value $416$. OEIS A232711 |
| **Generalized Collatz (7n + 1)** | Defined by the rule: n → n / 2 if n is even, else n → 7n + 1 | **10**: sequence: $10 \to 5 \to 36 \to 18 \to 9 \to 64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$, peak value $64$. OEIS A133421 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Generalized Collatz (an + b)** | Defined by the rule: n → n / 2 if n is even, else n → an + b, a and b can be specified in the profile settings. (Default: 3n+3) | **10**: For default values a=3 and b=3: sequence: 10 → 5 → 18 → 9 → 30 → 15 → 48 → 24 → 12 → 6 → 3 → 12. Loops back to 12, peak value 48.<br>OEIS N/A |
| **Ducci (4-digit)** sequence | A Ducci sequence starts from the last four digits of the input number. These four digits form a tuple $(a_1, a_2, a_3, a_4)$, and each step produces a new tuple of the absolute differences of adjacent entries (with wrap-around): $(a1,a2,a3,a4) \to (\lvert a1-a2\rvert, \lvert a2-a3\rvert, \lvert a3-a4\rvert, \lvert a4-a1\rvert)$.<br>A number is Ducci-terminating if the sequence eventually reaches $(0,0,0,0)$. | **91234**: last for digits: 1234<br>$(1,2,3,4) \to (1,1,1,3)$<br>$(1,1,1,3) \to (0,0,2,2)$<br>$(0,0,2,2) \to (0,2,0,2)$<br>$(0,2,0,2) \to (2,2,2,2)$<br>$(2,2,2,2) \to (0,0,0,0)$<br>The sequence reaches all zeros after five steps, so 1234 is Ducci-terminating.<br>OEIS N/A |
| **Fibonacci mod n** (Pisano period) | The *Pisano period* π(n) is the length of the repeating cycle in the Fibonacci sequence taken modulo n. Formally, if $F_0 = 0$, $F_1 = 1$, $F_k = F_{k-1} + F_{k-2}$ (mod n),<br>then the Pisano period π(n) is the smallest positive integer *m* such that $F_m \equiv 0$ (mod n) and $F_{m+1} \equiv 1$ (mod n).<br>The sequence of Fibonacci numbers modulo n always eventually repeats, and the period depends on n. | **11**: Fibonacci sequence mod 11: residues: [0, 1, 1, 2, 3, 5, 8, 2, 10, 1, 0] → repeat every 10 times, so the Pisano period $\pi(11) = 10$<br>OEIS A001175 |
| **Happy number sequence** | Starting from any positive integer *n*, replace the number by the sum of the squares of its decimal digits. Repeat this process iteratively:<br>$n \to \Sigma\,(\text{digits}(n))^2$<br>If this sequence eventually reaches 1, the original number is called a happy number. | **19**:<br>$1^2 + 9^2 = 82$<br>$8^2 + 2^2 = 68$<br>$6^2 + 8^2 = 100$<br>$1^2 + 0^2 + 0^2 = 1 \Rightarrow$ 19 is a happy number sequence, if not it's an unhappy number sequence.<br>OEIS A007770 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Kaprekar routine** | Start with a positive integer, arrange its digits in descending and ascending order to form two numbers, and subtract the smaller from the larger.<br>Repeat this process with each new result.<br>For most numbers with a fixed number of digits, this process quickly reaches a fixed point or a short loop, known as a Kaprekar constant.<br>• For 3-digit numbers → 495<br>• For 4-digit numbers → 6174<br>If all digits are identical (e.g., 1111), the process collapses to 0. | **3524**:<br>$5432 - 2345 = 3087$<br>$8730 - 0378 = 8352$<br>$8532 - 2358 = 6174$<br>→ The sequence $3524 \rightarrow 3087 \rightarrow 8352 \rightarrow 6174$ reaches the Kaprekar constant $6174$, after which $7641 - 1467 = 6174$ repeats indefinitely.<br>OEIS A099009 |
| **Reverse-and-add sequence** | Starting from any positive integer *n*, reverse its decimal digits and add the result to the original number:<br>n → n + reverse(n)<br>Repeat this process iteratively.<br>If the sequence eventually becomes a palindrome (a number that reads the same forward and backward), *n* is said to reach a palindrome.<br>Some numbers never appear to reach a palindrome — these are known as Lychrel candidates (for example, 196). | **349**:<br>$349 + 943 = 1292$<br>$1292 + 2921 = 4213$<br>$4213 + 3124 = 7337$<br>→ 7337 is a palindrome after 3 steps.<br>OEIS A033865 |

## Fun Numbers

Numbers notable for cultural, historical, or humorous reasons rather than mathematical properties.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Fun number** | Recognized number of interest in computing, science fiction, or pop culture.<br>The database currently contains 84 of such numbers. | **42**: The Answer to the Ultimate Question of Life, the Universe, and Everything (Douglas Adams, The Hitchhiker's Guide to the Galaxy)<br>OEIS N/A |

## Mathematical Curiosities

Numbers exhibiting aesthetic, linguistic, or self-referential properties.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Additive sequence** | Number whose digits can be split into parts forming an additive sequence where each term equals the sum of the two preceding ones. | **437**: $4+3=7 \Rightarrow 437 \Rightarrow 437$ is an Additive sequence.<br>OEIS A033627 (listing numbers that are NOT an Additive sequence.) |
| **Binary-interpretable number** | A decimal number that represents a valid binary numeral when its digits are read as bits. | **1011**: Base 10 number 1011 can be interpreted as a number in base 2: $1+2+8=11$ in base 10.<br>OEIS N/A |
| **Boring number** | The first number with the least number of classifications in NumClass using the default profile. | **1911**: The first number found having only 4 classifications.<br>OEIS N/A |
| **Cyclic permutation number** | Integer whose cyclic rotations are successive multiples of the original number. | **142857**:<br>$142857 \times 1 = 142857$<br>$142857 \times 2 = 285714$<br>$142857 \times 3 = 428571$<br>$142857 \times 4 = 571428$<br>$142857 \times 5 = 714285$<br>$142857 \times 6 = 857142$<br>Each product is a cyclic rotation of the digits of 142857 ⇒ 142857 is a cyclic number<br>OEIS A180340 |
| **Cyclops number** | Has an odd number of digits with exactly one zero in the middle position. | **11011**: Odd number and single zero in the middle.<br>OEIS A134808 |
| **Digit-reversal constant** | Appears as a fixed result in a digit-reversal subtraction procedure, reverse, subtract, reverse, add. This always converges to 1089. | **532**: Reverse: 235, Subtract: $532-235=297$, Reverse: 792, Add: $297+792=1089$.<br>OEIS N/A |
| **Eban number** | Lacks the letter “e” when its English name (short-scale) is spelled out. In English, every odd number contains an "e," so all eban numbers are even. | **32**: (thirty-two). Does not contain the letter “e” ⇒ 32 is an eban number.<br>OEIS A006933 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Fibonacci-base palindrome** | A *Fibonacci-base palindrome* is a number whose Zeckendorf Fibonacci-base bitstring reads the same forwards and backwards.<br>The Fibonacci-base bitstring is derived from the Zeckendorf decomposition by writing one bit for each Fibonacci number $F_k$, from the largest index used down to $F_2$:<br>• **1** indicates that $F_k$ is included in the decomposition,<br>• **0** indicates that it is not.<br>A number is Fibonacci-base palindromic if this bitstring is symmetric. | **12**: 12 = 8 + 3 + 1 = F6 + F4 + F2<br>The corresponding Fibonacci-base bitstring is: bits(F6..F2) = 10101<br>The bitstring reads the same forwards and backwards ⇒ 12 is a Fibonacci-base palindrome.<br>OEIS N/A |
| **Harshad in all bases** | A number is a *Harshad number* in all bases if it is divisible by the sum of its digits in every base $b \geq 2$ when written in that base.<br>Formally, an integer $n$ is *all-base Harshad* if: $n \equiv \left(\text{moddigit-sum}_b(n)\right)$ for every base b≥2. Only a few numbers satisfy this for all bases: 1, 2, 3 and 4. | **3**: Check bases $b = 2,3,4,5,\ldots$<br>• Base 2: $3_{10} = 11_2$, digit sum $= 1 + 1 = 2$<br>$3 \equiv 0 \pmod{2}$→ ✓<br>• Base 3: $3_{10} = 10_3$, digit sum $= 1 + 0 = 1$<br>$3 \equiv 0 \pmod{1}$→ ✓<br>• Base 4: $3_{10} = 3_4$, digit sum = $3$, $3 \equiv 0 \pmod{3}$→ ✓<br>• Base 5 and higher:<br>$3_{10} = 3_b$, digit sum = 3<br>$3 \equiv 0 \pmod{3}$→ ✓<br>⇒ 3 is Harshad in every base.<br>OEIS A005349 (Harshad) |
| **Kaprekar constant (3-digit)** | The fixed point of the 3-digit Kaprekar routine (495). | **495**: Identifies this number as the 3-digit Kaprekar constant, see Kaprekar routine.<br>OEIS A099009 |
| **Kaprekar constant (4-digit)** | The fixed point of the 4-digit Kaprekar routine (6174). | **6174**: Identifies this number as the 4-digit Kaprekar constant, see Kaprekar routine.<br>OEIS A099009 |
| **LCM-prefix number** | An *LCM-prefix number* is a positive integer that equals the least common multiple of the first *k* positive integers for some integer $k \geq 1$: $n = \text{lcm}\,(1,2,\ldots,k)$<br>Equivalently, it is the smallest positive integer divisible by every integer from 1 through *k*.<br>LCM-prefix numbers grow in jumps: the value only increases when a new prime or a higher prime power is required. As a result, the same number may equal $\text{lcm}(1..k)$ for several consecutive values of *k*. | **2520**: 2520 = $2^3 \times 3^2 \times 5 \times 7$<br>No higher prime powers are needed until $k = 11$, so the value remains unchanged for multiple prefixes:<br>$\text{lcm}\,(1..9) = \text{lcm}\,(1..10) = 2520$<br>OEIS A003418 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Lucky number of Euler** (Heegner number) | Checks if n is a 'lucky number of Euler' (Heegner number). These are the unique positive integers n for which Q(√−n) has class number 1, giving rise to the near-integer phenomenon e^{π√n} ≈ integer. | **43**: $e^{\pi\sqrt{43}} \approx 884736743.999777\ldots$ rounded: $884736744$ OEIS A003173 |
| **Munchausen number** | Equals the sum of its digits, each raised to the power of itself. $0^0$ is undefined as a real-number limit, but in digit functions we need every digit to have a well-defined self-power. Two conventions exist: $0^0=1$ (classical): only 1 and 3435 exists. $0^0=0$ (modern digit-function rule), 1, 3435 and 438579088 exists. NumClass uses the modern digit rule. | **3435**: Take its digits: $3, 4, 3, 5$. Compute the sum of each digit raised to its own power: $3^3 = 27, 4^4 = 256, 3^3 = 27, 5^5 = 3125$ Added: $27 + 256 + 27 + 3125 = 3435$ The sum equals the original number ⇒ 3435 is a Münchhausen number. OEIS A046253 |
| **Octal-interpretable number** | Decimal number that represents a valid octal numeral when read as digits in base 8. | **773**: Base 10 number 773 can be interpreted an octal numeral: $3 \times 8^0 + 7 \times 8^1 + 7 \times 8^2 = 3 + 56 + 448 = 507$ in base 10. OEIS N/A |
| **Pandigital number (0–9)** | Contains each digit 0 through 9 exactly once. | **8376201459**: each digit 0-9 appears only once. OEIS A050278 |
| **Pandigital number (1–9)** | Contains each digit 1 through 9 exactly once. | **918573624**: each digit 1-9 appears only once. OEIS A050289 |
| **Polydivisible number** (magic numbers) | A *polydivisible number* is a positive integer whose prefixes are successively divisible by their length. Let the decimal digits of a number be $d_1 d_2 \ldots d_k$. For each prefix formed by the first $i$ digits: $(d_1 d_2 \ldots d_i)$ is divisible by $i$ for all $i = 1,2,\ldots,k$. In other words, when reading the number from left to right, each growing prefix is evenly divisible by the number of digits it contains. | **360**: Check each prefix: • 3 is divisible by 1 • 36 is divisible by 2 • 360 is divisible by 3 Since all prefixes satisfy the divisibility condition, 360 is polydivisible. OEIS A144688 |
| **Smarandache–Wellin number** | Formed by concatenating the first n prime numbers written in base 10. | **2357**: The first 4 primes 2, 3, 5, 7 concatenated. OEIS A019518 |
| **Strobogrammatic number** | Looks the same when rotated 180°, using digits like 0, 1, 6, 8, and 9. | **68189**: Reads the sum up-side-down. OEIS A000787 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Vampire number** | Composite number factorable into two equal-length factors containing exactly the same digits as the original number. | **1260**: $1260 = 21 \times 60$, possible "fangs": 21 and 60.<br>Check digit lengths:<br>• 1260 has 4 digits<br>• Fangs must have 2 digits each → ✓<br>• 21 and 60 both have 2 digits → ✓<br>Check digit multiset<br>Digits of 1260: $\{1, 2, 6, 0\}$<br>Digits of 21 and 60 combined:<br>• 21 → $\{2, 1\}$, 60 → $\{6, 0\}$<br>• Combined → $\{1, 2, 6, 0\}$<br>They match exactly → ✓<br>Zero rule: Fangs may not both end in 0.<br>21 ends with 1, 60 ends with 0<br>Only one ends with zero → ✓<br>⇒ 1260 is a vampire number with fangs 21 and 60.<br>OEIS A014575 |

# Named Sequences

Numbers that occur in named integer sequences with recursive or combinatorial definitions.

<table>
<tr><th>Classifier</th><th>Definition / Description</th><th>Examples / OEIS</th></tr>
<tr><td>2-term Zeckendorf number</td><td>A 2-term Zeckendorf number is a number whose Zeckendorf decomposition consists of exactly two Fibonacci numbers.<br>Because Zeckendorf decompositions use only non-consecutive Fibonacci numbers, such a number can be written uniquely as:<br>$n = F_a + F_b$with $a \geq b + 2$</td><td>11: $11 = 8 + 3 = F_6 + F_4$<br>The Zeckendorf decomposition uses exactly two Fibonacci terms, so 11 is a 2-term Zeckendorf number.<br>OEIS N/A</td></tr>
<tr><td>Busy Beaver number<br>(Rado's sigma function)</td><td>Maximum number of steps that a halting Turing machine with n states can execute before halting.<br>BB(n) is noncomputable.</td><td>6: Maximum number of steps for a 2-state, 2-symbol Busy Beaver that achieves it.<br>(blank = 0, start in state A at cell 0; symbols {0,1}; halting state H):<br><table>
<tr><th>State</th><th>Read</th><th>Write</th><th>Move</th><th>Next</th></tr>
<tr><td>A</td><td>0</td><td>1</td><td>R</td><td>B</td></tr>
<tr><td>A</td><td>1</td><td>1</td><td>L</td><td>B</td></tr>
<tr><td>B</td><td>0</td><td>1</td><td>L</td><td>AH</td></tr>
<tr><td>B</td><td>1</td><td>1</td><td>R</td><td></td></tr>
</table>
Execution trace (from blank tape):<br>1. A reads 0 → write 1, R, go B (head to +1)<br>2. B reads 0 → write 1, L, go A (head to 0)<br>3. A reads 1 → write 1, L, go B (head to −1)<br>4. B reads 0 → write 1, L, go A (head to −2)<br>5. A reads 0 → write 1, R, go B (head to −1)<br>6. B reads 1 → write 1, R, halt (head to 0)<br>It halts after 6 steps, so $S(2) = 6$, and it leaves 4 ones on the tape, so $\Sigma(2) = 4$.<br>OEIS A060843</td></tr>
<tr><td>Carol number</td><td>A Carol number is a special kind of number defined by the formula<br>$C_n = 2^n - 2^{\lfloor n/2 \rfloor} - 1$<br>for integer $n \geq 1$, meaning it's 2 less than the square of one less than a power of two.</td><td>47: For $n = 3$:<br>$C_3 = 4^3 - 2^4 = 64 - 16 - 1 = 47$<br>⇒ 47 is a Carol number.<br>OEIS A093112</td></tr>
<tr><td>Cullen number</td><td>Number of the form $n = k \cdot 2^k + 1$ for some integer $k \geq 1$.</td><td>25: Take $k = 3$:<br>$C_3 = 3 \times 2^3 + 1 = 3 \times 8 + 1 = 25$<br>⇒ 25 is a Cullen number.<br>OEIS A002064</td></tr>
</table>

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Erdős–Woods number** | Length of an interval in which every integer shares a non-trivial common factor with the first or last element.<br>k is Erdős–Woods if there are integers $a$ and $b = a + k$ such that for every $a < n < b$ we have $\gcd(n, a) > 1$or $\gcd(n, b) > 1$. | **16**: Endpoints:<br>$a = 2184 = 2^3 \times 3 \times 7 \times 13$<br>$b = 2200 = 2^3 \times 5^2 \times 11$<br>Property: For every integer $n$ with $2184 < n < 2200$, we have:<br>• either $\gcd(n, 2184) > 1$,<br>• or $\gcd(n, 2200) > 1$.<br>⇒ 16 is an Erdős–Woods number.<br>OEIS A059756 |
| **Factorial number** | Integer that equals k! for some integer k ≥ 0. | 720: $720 = 6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1$, $720 \div 2 \div 3 \div 4 \div 5 \div 6 = 1$<br>⇒ 720 is a factorial number.<br>OEIS A000142 |
| **Fibonacci number** | Member of the sequence defined by $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$. | 34: Fibonacci sequence:<br>0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...<br>⇒ 34 is the 9th Fibonacci number.<br>OEIS A000045 |
| **Hamming number**<br>(regular number)<br>(Ugly number)<br>(5-smooth number) | Number whose prime factors (and are limited to 2, 3, and 5. | **90**: $90 = 2 \times 3^2 \times 5$, contains no other prime factors than 2, 3 or 5 ⇒ 90 is a Hamming number.<br>OEIS A051037 |
| **Keith number**<br>(Repfigit number) | Number that appears in its own digit-sequence recurrence: each term is the sum of the previous d digits, where d is the number of digits in n.<br>Let $n$ have decimal digits $d_1, d_2, \dots, d_k$.<br>Form a sequence starting with these digits and for all later terms:<br>$a_m = a_{m-1} + a_{m-2} + \cdots + a_{m-k}$.<br>If the number $n$ itself appears later in this sequence, then $n$ is a Keith number.<br>Keith numbers are very rare because the digit-based recurrence almost never matches the number again. | **197**: start with the digits: 1, 9, 7. Initial terms:<br>$a_1 = 1, a_2 = 9, a_3 = 7$<br>Next: $a_4 = a_1 + a_2 + a_3 = 1 + 9 + 7 = 17$<br>Next: $a_5 = a_2 + a_3 + a_4 = 9 + 7 + 17 = 33$<br>Next: $a_6 = a_3 + a_4 + a_5 = 7 + 17 + 33 = 57$<br>Next: $a_7 = a_4 + a_5 + a_6 = 17 + 33 + 57 = 107$<br>Next: $a_8 = a_5 + a_6 + a_7 = 33 + 57 + 107 = 197$<br>Now the sequence looks like:<br>1, 9, 7, 17, 33, 57, 107, 197, ...<br>And since the sequence reaches 197 ⇒ 197 is a Keith number.<br>OEIS A007629 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Lucas number** | A *Lucas number* is a number in the Lucas sequence, which is very similar to the Fibonacci sequence but starts with different initial values. The Lucas numbers $L_n$ are defined by: $L_0 = 2, L_1 = 1,$ $L_n = L_{n-1} + L_{n-2}$ for $n \geq 2$. So the first values are: 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, ... | **18**: Take $n = 6$: $\mathrm{L6} = \mathrm{L5} + \mathrm{L4} = 11 + 7 = 18$ ⇒ 18 is the 6th Lucas number. OEIS A000032 |
| **Lucky number** | Generated by the “lucky sieve”: repeatedly removing every kth number from the natural numbers. Start with a list of all positive odd integers: 1, 3, 5, 7, 9, 11, 13, 15, 17, ... Because the second number in the list is 3, remove every 3rd position (not every number divisible by 3): 1, 3, 7, 9, 13, 15, 19, ... next surviving number is 3, remove every 3rd remaining number position. 1, 3, 7, 9, 13, 15, 21 next surviving number is 7, remove every 7th number position and continue infinitely. A number that is never removed is called a *Lucky number*. | **13**: The sequence left after the sieve is: 1, 3, 7, 9, 13, 15, 21, 25, 31, 33, 37, ..., 13 is in the sequence. ⇒ 13 is a Lucky number. OEIS A000959 |
| **Padovan number** | The Padovan sequence P(n) is defined as: P(0) = P(1) = P(2) = 1, P(n) = P(n – 2) + P (n – 3) for n ≥ 3. Resulting in the following sequence: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, ... | **21**: P(12) = P (10) + P(9) = 12 + 9 = 21 ⇒ 21 is a Padovan number. OEIS A000931 |
| **Pell number** | A *Pell number* is defined recursively by $P_n = 2P_{n-1} + P_{n-2}$, with $P_0 = 0$, $P_1 = 1$, forming the sequence: 0, 1, 2, 5, 12, 29, 70, 169, 408, 985, ... | **12**: $P_4 = 2P_3 + P_2 = 2 \times 5 + 2 = 12$ ⇒ 12 is the 4th Pell number OEIS A000129 |
| **Sparse Zeckendorf representation** | A number has a *sparse Zeckendorf representation* if its Zeckendorf decomposition uses only a small number of Fibonacci terms**.** The maximum allowed number of terms is configurable. If the number of Fibonacci terms in the Zeckendorf decomposition does not exceed this limit, the number is classified as *sparse*. | **12**: For a maximum of 3 terms: $12 = 8 + 3 + 1 = F_6 + F_4 + F_2$ The decomposition uses three Fibonacci terms ⇒12 has a sparse Zeckendorf representation. OEIS N/A |
| **Taxicab number** (Hardy–Ramanujan number) | Can be expressed as the sum of two positive cubes in at least two distinct ways. | **1729**: $1729 = 1^3 + 12^3 = 9^3 + 10^3$. OEIS A011541 |
| **Tribonacci number** | Each term equals the sum of the previous three terms, starting with 0, 0, 1, so the first values are 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, ... | **44**: $T_8 = T_5 + T_6 + T_7 = 7 + 13 + 24 = 44$ ⇒ 44 = Tribonacci number $T_8$. OEIS A000073 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Ulam number** | Term of the Ulam sequence: each is the smallest integer that can be written uniquely as a sum of two distinct earlier terms in exactly one way, starting with 1 and 2. Resulting in the following sequence: 1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 16, ... | **6**: 6 = 2 + 4, so 6 can be written as a sum of two earlier Ulam numbers and no other distinct pair of earlier Ulam numbers gives 6.<br>⇒ 6 is an Ulam number.<br>OEIS A002858 |
| **Woodall number** (Riesel number) | Number of the form $n = k \cdot 2^k - 1$ for some integer k ≥ 1. | **63**: Take k=4:<br>$W_4 = 4 \times 2^4 - 1 = 416 - 1 = 63$ ⇒ 25 is a Woodall number.<br>OEIS A003261 |

## Polygonal and Figurate Numbers

Numbers representable as geometric shapes such as triangles, squares, or polygons.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Centered hexagonal number** | Number of the form $n = 3 \cdot k(k - 1) + 1$ for some integer k ≥ 1; represents a centered hexagon pattern. | **37**: Take k=4: $H_4 = 3 \times 4 \times (4-1)+1 = 37$<br>⇒ 37 is a Centered hexagonal number.<br>OEIS A003215 |
| **Centered square number** | Number of the form $n = 2 \cdot k(k - 1)\ 1 +$ for some integer k ≥ 1; geometrically forms a centered square (an odd square). | **41**: Take k=5: $S_4 = 2 \times 5 \times (5-1)+1 = 41$ ⇒ 41 is a Centered square number.<br>OEIS A001844 |
| **Centered triangular number** | Number of the form $n = (3/2) \cdot k(k - 1) + 1$ for some integer k ≥ 1. | **46**: Take k=6:  Take k=6: $T_4 = 3/2 \times 6 \times (6-1)+1 = 46$<br>⇒ 46 is a Centered triangular number.<br>OEIS A005448 |
| **Harshad hexagonal number** (intersection) | Hexagonal number that is also divisible by the sum of its decimal digits. $H_n = n(2n - 1)$<br>$H_n \div \text{digit\_sum}(H_n) \in \mathbb{Z}$. | **190:** Take $n = 10$:<br>$H_{10} = 10(2 \times 10 - 1) = 10 \times 19 = 190$.<br>Digit sum: $1 + 9 + 0 = 10$.<br>$190 \div 10 = 19$ (integer)<br>⇒ 190 is both hexagonal and a Harshad number.<br>OEIS N/A |
| **Harshad triangular number** (intersection) | Triangular number that is also divisible by the sum of its decimal digits.<br>$T_n = \frac{k(k+1)}{2}$ | **36**: Take n=8<br>$T_8 = \frac{8 \times 9}{2} = 36$.<br>Digit sum: 3 + 6 + 9.<br>Check the Harshad property:<br>$36 \div 9 = 4$ (integer)<br>⇒ 36 is both triangular and a Harshad number.<br>OEIS A076713 |
| **Hexagonal number** | Number of the form $n = k(2k - 1)$ for some integer k ≥ 1. | **66**: Take k=6, n = 6(2×6-1) = 66<br>⇒ 66 is a hexagonal number<br>OEIS A000384 |

| Classifier | Definition / Description | Examples / OEIS |
| --- | --- | --- |
| **Lehmer number** | Number equal to $(a^k - 1) / (a - 1)$ for integers $2 \le a \le 10$, $k > 1$; corresponds to k consecutive 1's in base a. | **31**: Take a=2, k=5, $n = (2^5-1) \div (2-1) = 31$ ⇒ 31 is a Lehmer number. OEIS N/A |
| **Palindromic triangular number** (intersection) | Triangular number of the form $T_n = \frac{k(k+1)}{2}$ whose decimal representation is a palindrome. | **171**: Take k=18, $T_n = \frac{18(18+1)}{2}$ = 171 and is a palindrome. OEIS A003098 |
| **Pentagonal number** | Pentagonal number of the form $P_n = \frac{k(3k-1)}{2}$ for some integer k ≥ 1. | **22**: Take k=4, $P_n = \frac{4(3 \times 4 - 1)}{2} = 22$ ⇒ 22 is a pentagonal number OEIS A000326 |
| **Pronic number** (Oblong number) (Heteromecic number) | Number of the form n = k(k + 1); the product of two consecutive integers. | **4**: 4(4+1) = 4 × 5 = 20 ⇒ 4 is a Pronic number. OEIS A002378 |
| **Repunit** | Number consisting solely of digit 1, e.g., 1, 11, 111, etc. | **111**: Number solely of digit 1 ⇒ 111 is a repunit. OEIS A002275 |
| **Star number** (Centered hexagram number) (Centered 12-gonal number) | Number of the form $S_n = 6k(k-1) + 1, k \ge 1$. Represents a centered hexagram or 12-gonal pattern. | **3**: $S_3 = 6 \times 3 \times 2 + 1 = 36 + 1 = 37$. ⇒ 37 is a star number. OEIS A003154 |
| **Tetrahedral number** | Number of the form $T_n = \frac{k(k+1)(k+2)}{6}$ Represents a 3D pyramid with a triangular base. | **4**: $T_4 = \frac{4 \times 5 \times 6}{6} = \frac{120}{6} = 20.$ ⇒ 20 is a tetrahedral number. This corresponds to a tetrahedron made of:<br>• 1 point on top<br>• 3 points in the second layer<br>• 6 in the third<br>• 10 in the base<br>And $1 + 3 + 6 + 10 = 20$. OEIS A000292 |
| **Triangular number** | Number of the form $T_n = \frac{k(k+1)}{2}$ for some integer k ≥ 1. | **15**: 5(5+1) ÷ 2 = 15 OEIS A000217 |

## Primes and Prime-related Numbers

Numbers that are prime or defined through relationships to prime forms or patterns.

| Classifier | Definition / Description | Examples / OEIS |
| --- | --- | --- |
| **Absolute prime** (Permutable prime) | Prime that remains prime under all permutations of its digits. | **113**: 113, 131, 311 are all prime. OEIS A003459 |
| **Automorphic prime** (intersection) | Prime whose square ends with the same digits as the prime itself. | **5**: $5^2 = 25$ and 5 is prime. OEIS A052228 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Balanced prime** | A prime that is exactly halfway between the previous and the next primes. | **53**: Primes before and after 53 are 47 and 59, average (47+59) ÷ 2 = 53, the gap between both primes are equal (6) ⇒ 53 is a balanced prime.<br>OEIS A006562 |
| **Bell prime** (intersection) | Prime that divides a Bell number, which counts set partitions. | **877**: Is a Bell number and a prime.<br>OEIS A051130 |
| **Both-truncatable prime** | Prime that remains prime when digits are removed from either the left or the right. | **313**: 313, 31, 13 and 3 are all primes.<br>OEIS A020994 |
| **Brazilian prime** (intersection) | Prime representable as a repdigit in some base b with $2 \leq b \leq n - 2$. | **8191**: Is a base-2 repunit $1111111111111_2$ and prime.<br>OEIS A377886 |
| **Catalan prime** | Prime that is also a Catalan number. (See Catalan number for definition) The only prime Catalan numbers are 2 and 5. | **5**: Is a Catalan number and a prime.<br>OEIS A000108 |
| **Chen prime** | Prime p for which p + 2 is either prime or a semiprime (product of two primes). | **89**: 89 + 2 = 91 = 7 × 13, a semiprime. 89 is also a prime.<br>OEIS A109611 |
| **Circular prime** | Prime that remains prime under all cyclic rotations of its digits. | **197**: 197, 971 and 791 are all prime.<br>OEIS A068652 |
| **Cousin prime** | Prime that forms a pair with another prime differing by 4. | **967**: 967 and 971 differ by 4 ⇒ 967 and 971 are Cousin primes.<br>OEIS A160440 (Smaller member)<br>OEIS A046132 (Larger member) |
| **Cullen prime** (intersection) | Prime of the form $p = k \cdot 2^k + 1$ for some integer $k \geq 1$. | **393050634124102232869567034555427371542904833**: Let k = 141, $141 \times 2^{141} + 1 = p$ and is also prime.<br>OEIS A050920 |
| **Cyclops prime** (intersection) | Prime with an odd number of digits and a single zero in the middle. | **12049**: Odd number with single zero in the middle and is prime.<br>OEIS A134809 |
| **Disarium prime** (intersection) | Prime that also satisfies the Disarium property (sum of digits^position = n). Only 5 Disarium primes exist: 2, 3, 5, 7 and 89. | **89**: $89 = 8^1 + 9^2 = 8 + 81$ and 89 is also prime.<br>OEIS N/A |
| **Emirp** | Prime that yields a different prime when its digits are reversed.<br>(The word emirp = prime in reverse). | **13**: 13 and 31 are prime.<br>OEIS A006567 |
| **Factorial prime** | Prime of the form k! +1 or k! - 1 for some integer $k \geq 1$. | **5039**: 5039 = 7! – 1<br>⇒ 5039 is a factorial prime<br>OEIS A002981 (k! + 1)<br>OEIS A002982 (k! - 1) |
| **Factorion prime** (intersection) | Prime that equals the sum of the factorials of its digits.<br>2 is the only factorion prime. | **2**: 2! = 2 and 2 is a prime.<br>OEIS N/A |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Fermat prime** | Prime of the form<br>$F_n = 2^{2^n} + 1, n \geq 0.$<br>No Fermat primes with n ≥ 5 are known. | **65537**: Take n=4:<br>$F_4 = 2^{2^4} + 1 = 2^{16} + 1 = 65\,536 + 1 = 65\,537$ and is also prime.<br>⇒ 65537 is a Fermat prime $F_4$.<br>OEIS A019434 |
| **Fibonacci prime** (intersection) | Prime that is also a term of the Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... | **89**: 89 is in the Fibonacci sequence and is prime.<br>OEIS A005478 |
| **Gaussian prime** | Prime element in the ring of Gaussian integers<br>$\mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}.$ | **13**: Take 3 + 2i, it's norm is $N(3+2i) = 3^2 + 2^2 = 9 + 4 = 13$ and 13 is a rational prime, the Gaussian integer is irreducible<br>⇒ 13 is a Gaussian prime.<br>OEIS A055025 |
| **Good prime** | A prime $p_n$ is a *good prime* if it dominates *all* symmetric products of earlier and later primes:<br>$p_n^2 > p_{n-i} \times p_{n+i}$<br>for all $1 \leq i \leq n-1$.<br><br>Equivalently, the local prime gaps around $p_n$ grow "faster than expected" when compared to all previous symmetric gaps. | **29**: it's index is n = 10, and checks show:<br>$29^2 > 2331, 29^2 > 19 \times 37, 29^2 > 17 \times 41, \ldots$<br>holding for all valid $i$.<br>Thus 29satisfies the global good-prime condition.<br>OEIS A028388 |
| **Good prime (local)** | A prime $p_n$ is a *local good prime* if it exceeds the geometric mean of its immediate neighbors:<br>$p_n^2 > p_{n-1} \times p_{n+1}.$<br>This is the nearest-neighbor version of the good-prime condition.<br>Every global good prime is automatically a local good prime, but the converse need not hold. | **17**:<br>$17^2 = 289$ and $13 \times 19 = 247$, and since 289 > 247, the condition is satisfied.<br>OEIS A046869 |
| **Happy palindromic** prime (intersection) | Prime that is both happy and palindromic in decimal representation. | **383**:<br>• Happy: repeatedly summing the squares of its digits eventually reaches 1.<br>• Palindromic: Forward and backward reads the same.<br>• Prime: 383 is a prime number.<br>OEIS A364479 |
| **Happy prime** (intersection) | Prime that remains after repeatedly summing squares of digits until reaching 1. | **79**:<br>• Happy: repeatedly summing the squares of its digits eventually reaches 1.<br>• Prime: 383 is a prime number.<br>OEIS A035497 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Harshad prime** (intersection) | Prime in base 10 that is also divisible by the sum of its digits. The only Harshad primes are: 2, 3, 5 and 7. | **7**: 7 ÷ 7 = 1<br>The only divisors of a prime are 1 and n, so no Harshad primes (base 10) can exist with 2 or more digits.<br>OEIS N/A |
| **Isolated prime** | Prime not adjacent to any twin prime (neither p−2 nor p+2 is prime). | **89**: 89 is not divisible by 2, 3, 5 and 7 (divisibility up to $\sqrt{89}$)<br>89 – 2 = 87, 89 + 2 = 91, both 87 and 91 are not prime.<br>OEIS A007510 |
| **Keith prime** | Prime that is also a Keith number under the base-10 digit-recurrence definition. Only 6 Keith primes are known. All base-10 Keith numbers up to 36 digits have been determined and none of those are prime. | **197**: It has 3 digits, so $k = 3$.<br>Start the sequence with its digits: $a_1 = 1, a_2 = 9, a_3 = 7$<br>Now build the sequence, each term = sum of previous 3:<br>• $a_4 = a_1 + a_2 + a_3 = 1 + 9 + 7 = 17$<br>• $a_5 = a_2 + a_3 + a_4 = 9 + 7 + 17 = 33$<br>• $a_6 = a_3 + a_4 + a_5 = 7 + 17 + 33 = 57$<br>• $a_7 = a_4 + a_5 + a_6 = 17 + 33 + 57 = 107$<br>• $a_8 = a_5 + a_6 + a_7 = 33 + 57 + 107 = 197$<br>197 reappears as $a_8$ and is prime ⇒ 197 is a Keith prime.<br>OEIS A048970 |
| **Left-truncatable prime** | Prime that remains prime after removing leading digits one at a time and no digits are zero. | **239**: 239, 39 and 9 are all prime.<br>OEIS A024785 |
| **Lucas prime** (intersection) | Prime that is also a term in the Lucas sequence: 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, … (See Lucas number) | **199**: 199 is in the Lucas sequence L(11) and is prime.<br>OEIS A005479 |
| **Lucky prime** (intersection) | Prime that is also a lucky number from the lucky sieve. | **223**: 223 = L(43) and prime.<br>OEIS A031157 |
| **Mersenne prime** (intersection) | Prime of the form $2^p - 1$ where p is prime. | **8191**: Take p=13,<br>$2^{13} - 1 = 8191$ ⇒ 8191 is a Mersenne prime.<br>OEIS A000668 |
| **Motzkin prime** | Prime that is also a Motzkin number. (See Motzkin number).<br>Only 4 Motzkin primes are known. | **15511**: 15511 is a Motzkin number and the 3rd known Motzkin prime.<br>OEIS A092832 |
| **Narcissistic prime** (intersection) | Prime that is also a narcissistic number (sum of digits^count = n). | **28116440335967**: 14 digits,<br>$28116440335967 = 2^{14} + 8^{14} + 1^{14} + 1^{14} + 6^{14} + 4^{14} + 4^{14} + 0^{14} + 3^{14} + 3^{14} + 5^{14} + 9^{14} + 6^{14} + 7^{14}$<br>and is also prime.<br>OEIS A145380 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Padovan prime** (intersection) | Prime that is also a Padovan number. (Sequence 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, 28, 37, ...) | **37**: 37 is P(14) and is also prime.<br>OEIS A100891 |
| **Palindromic prime** (intersection) | Prime whose decimal representation reads the same forward and backward. | **787**: reads the same forward and backward and is prime.<br>OEIS A002385 |
| **Pell prime** (intersection) | Prime that is also a Pell number. (See Pell number definition) | 29: $P_5 = 2P_4 + P_3 = 2 \times 12 + 5 = 29$ and is also prime.<br>OEIS A096650 |
| **Pierpont prime** (Class 1 prime) | Prime of the form $2^a \cdot 3^b + 1$ for integers a, b ≥ 0. | **97**: $2^5 \times 3^1 + 1 = 97$<br>OEIS A005109 |
| **Prime triplet member** | Member of the only prime triplet P, P+2, P+4 (3, 5, 7). | **5**: 5 belongs to the prime triplet 3, 5, 7<br>OEIS N/A |
| **Primorial number** | A *primorial number* is the product of all prime numbers up to a given prime $p$.<br>The primorial of $p$ is written as $p\#$ and defined as: $p\# = 2 \cdot 3 \cdot 5 \cdots p$.<br>Thus, the sequence begins with: 2, 6, 30, 210, 2310, ...<br>Primorials are the prime analogue of factorials: just as $n! = 1 \cdot 2 \cdot 3 \cdots n$, the primorial multiplies all primes up to $p$. These numbers grow quickly and appear in many areas of number theory, including prime gaps and sieve methods. | **30**: 30=2·3·5=5#.<br>OEIS A002110 |
| **Primorial prime** | Prime of the form p# ± 1, where p# is the product of the first n primes. | **30029**: Take p=13, 13# = 2 × 3 × 5 × 7 × 11 × 13 = 20030 – 1 = 30029 and is also prime.<br>OEIS A018239 p# + 1<br>OEIS A006794 p# -1 (index) |
| **Proth prime** | A *Proth prime* is a prime of the form $n = k \cdot 2^m + 1$<br>with k odd and $2^m > k$. | **13**: Take k=3 and m = 2 so $2^2 = 4 > 3$, $n = 3 \times 2^2 + 1 = 13$. 13 is prime ⇒ 13 is a Proth prime.<br>OEIS A080076 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Ramanujan prime** | Smallest R satisfying π(x) − π(x/2) ≥ n for all x ≥ R where π(x) is the prime-counting function.<br>It is known (and tabulated) that the Ramanujan primes begin 2, 11, 17, 29, 41, 47, 59, 67, 71, 97, 101, 107, 127, ... | **127**: From the sequence $127 = R_{13}$.<br>We must have exactly 13 primes in the interval $(127/2,127] = (63.5,127]$.<br>Primes between 64 and 127 are: 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127<br>There are 13 of them, so $\pi(127) - \pi(63.5) = \pi(127) - \pi(63) = 13$.<br>Thus, at $x = 127$ the inequality holds with equality for $n = 13$.<br>By the known theory and tables of Ramanujan primes, for every $x \geq 127$ we have $$\pi(x) - \pi\left(\frac{x}{2}\right) \geq 13,$$ and 127 is the smallest integer with this property ⇒ 127 is Ramanujan prime $R_{13}$.<br>OEIS A104272 |
| **Repunit prime** (intersection) | Prime composed entirely of digit '1's. | **11**: Entirely composed of the digit 1.<br>OEIS A004022 |
| **Right-truncatable prime** | Prime that remains prime after removing trailing digits one at a time and no digits are zero. | **739**: 739, 73 and 7 are all prime.<br>OEIS A024770 |
| **Safe prime** | Prime p for which (p − 1)/2 is also prime. | **47**: 47 is prime, (47-1)/2 = 33 and 33 is also prime ⇒ 47 is a save prime.<br>OEIS A005385 |
| **Semiprime** | Product of exactly two (not necessarily distinct) primes. | **33**: 33 = 3 × 11 = product of two primes.<br>OEIS A001358 |
| **Sexy prime** | Prime that forms a pair with another prime differing by 6. | **461**: 461 differs by 6 with 467 ⇒ 461 is a sexy prime.<br>OEIS A046117 (upper of a pair)<br>OEIS A023201 (lesser of a pair) |
| **Smarandach-Wellin prime** (intersection) | Prime composed of the concatenation of first n primes (written in base 10). | **23**: First primes 2, 3 and 23 are all prime.<br>OEIS A069151 |
| **Sophie Germain prime** | Prime p such that 2p + 1 is also prime. | **23**: 23 is prime and 2×23 + 1 = 47 is also prime.<br>OEIS A005384 |
| **Strong prime** | Prime p for which $p_{-1} + p_{+1} < 2p$, indicating a narrow prime gap. | **11**: Surrounding primes: 7 + 13 = 20 < 2 × 11 = 22, 11 > (7 + 13) / 2 ⇒ Strong prime.<br>OEIS A051634 |
| **Super prime** (Prime-indexed prime) | Prime whose position in the ordered list of primes is itself prime. | 59: 59 is the 17th prime, and 17 is also prime.<br>OEIS A006450 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Thin prime** (Broughan–Zhou) | A *Thin prime* is an odd prime $p$ such that $p + 1$ is either a power of 2 or a prime times a power of 2, i.e. $p + 1 = 2^k$ or $p + 1 = q\, 2^k$ with q prime, k ≥ 1. | **67**: is prime and odd: p+1 = 68. 68 = 17 × $2^2$<br>A prime $p$ is thin if:<br>• $p + 1 = 2^k$ (pure power of 2), **or**<br>• $p + 1 = q \cdot 2^k$ where $q$ is prime.<br>$k = 2, q = 17$ and 17 is prime. So $p + 1$ is a prime times a power of 2. That matches the second case exactly ⇒ 67 is a thin prime.<br>OEIS A192869 |
| **Triangular prime** (intersection) | Prime that is also a triangular number.<br>$T_k = \dfrac{k(k+1)}{2}$.<br>The only triangular prime number is 3. | 3: Take k = 2:<br>$T_2 = 2 \times (2 + 1) / 2 = 3$ ⇒ 3 is a Triangular prime.<br>OEIS N/A |
| **Twin prime** | Prime that forms a pair with another prime differing by 2. | **17**: 17 is prime and 19 is also prime.<br>OEIS A001359 (lesser)<br>OEIS A006512 (greater) |
| **Ulam prime** (intersection) | Prime that is also a term in the Ulam sequence: 1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 26, 28, 36, 38, 47, ... | **47**: 47 is prime and in the Ulam sequence.<br>OEIS A068820 |
| **Wieferich prime** | Prime p satisfying $2^{(p-1)} \equiv 1 \pmod{p^2}$.<br>Only two are known: 1093 and 3511. If others exist, they must be larger than $2^{64}$, Proven by the PrimeGrid project, finished 2022. | **1093**: 1093 is prime and $2^{1092} \equiv 1 \pmod{1093^2}$<br>⇒ 1093 is a Wieferich prime (base 2).<br>OEIS A001220 |
| **Wilson prime** | Prime p satisfying $(p-1)! \equiv -1 \pmod{p^2}$.<br>Only three are known: 5, 13 and 563. Searched up to $2 \times 10^{13}$. | **563**: 563 is prime and<br>$562! \equiv -1 \pmod{563^2} = 316968 = 563^2 - 1$ ⇒ 563 is a Wilson prime.<br>OEIS A007540 |
| **Woodall prime** (intersection) | Prime of the form $p = k \cdot 2^k - 1$ for some integer k ≥ 1. | **23**: 23 is prime and take k = 3:<br>$3 \cdot 2^3 - 1 = 3 \times 8 - 1 = 23$.<br>⇒ 23 is a Woodall prime.<br>OEIS A050918 |
| **Weak prime** | Prime p for which $p_{-1} + p_{+1} > 2p$, indicating a relatively wide prime gap. | **13**: Surrounding primes: 11 + 17 = 28 > 2 × 13 = 26,<br>13 < (11+17) / 2 ⇒ 13 is a Weak prime.<br>OEIS A051635 |

## Pseudoprimes and Cryptographic Numbers

Composite numbers satisfying certain modular properties or used in cryptographic theory.

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Blum integer** | Semiprime n = p·q where p and q are distinct primes satisfying p ≡ q ≡ 3 (mod 4). | **21**: 21 = 3 × 7, 3 and 7 are distinct primes.<br>Prime modulo 4:<br>• $3 \equiv 3 \pmod 4$<br>• $7 \equiv 3 \pmod 4$ because $7 = 4 + 3$<br>Both prime factors satisfy $p \equiv q \equiv 3 \pmod 4 \Rightarrow$ 21 is a Blum integer.<br>OEIS A016105 |
| **Carmichael number** | A composite number $n$ is *Carmichael* if it satisfies Fermat's little theorem for every base that's coprime to it:<br>For all $a$ with $\gcd(a, n) = 1$: $a^{n-1} \equiv 1 (\text{mod } n)$.<br>A convenient way to *prove* a number is Carmichael is Korselt's criterion:<br>A composite integer $n$ is a Carmichael number iff<br>1. $n$ is **squarefree**, and<br>2. For every prime $p$ dividing $n$, we have $p - 1 \mid n - 1$. | **561**: Prime factors:<br>561 = 3 × 11 × 17 (composite)<br>"squarefree" means no prime square divides it.<br>• $3^2 = 9 \nmid 561$ (since $561/9 = 62.\bar{3}$).<br>• $11^2 = 121 \nmid 561$ (since $561/121 \approx 4.63$).<br>• $17^2 = 289 \nmid 561$ (since $561/289 \approx 1.94$).<br>So no $p^2$ with $p \in \{3, 11, 17\}$ divides 561.<br>Therefore, 561 is squarefree.<br>Each prime divisor $p$ of 561 must be $p - 1 \mid 561 - 1 = 560$.<br>• For $p = 3$: $p - 1 = 2$. $560/2 = 280$, so $2 \mid 560$.<br>• For $p = 1$: $1p - 1 = 10$. $560/10 = 56$, so $10 \mid 560$.<br>• For $p = 17$: $p - 1 = 16$. $560/16 = 35$, so $16 \mid 560$.<br>for every prime divisor $p$ of 561, we have $p - 1 \mid 560$.<br>By Korselt's criterion, this implies: 561 is a Carmichael number.<br>Equivalently: for every integer $a$ with $\gcd(a, 561) = 1$, $a^{560} \equiv 1 \pmod{561}$, even though 561 is not prime.<br>OEIS A002997 |
| **Cunningham number** | A number of the form $b^n \pm 1$ for integer bases b.<br>These numbers play an important role in factorization and computational number theory and are catalogued in the Cunningham project. | 511 = $2^9 - 1$ is a Cunningham number.<br>OEIS: A080262 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **Euler–Jacobi pseudoprime** | An *Euler–Jacobi pseudoprime* is a composite odd integer $n$ such that for some base $a$ with $\gcd(a, n) = 1$,<br>$a^{\frac{n-1}{2}} \equiv (\frac{a}{n}) (\mathrm{mod}\, n)$,<br>where $\left(\frac{a}{n}\right)$ is the Jacobi symbol.<br>If $n$ were prime, this would always hold (Euler's criterion), so when a composite $n$ satisfies it, we call it an Euler–Jacobi pseudoprime to base $a$ in {2, 3, 5, 7, 11, 13}. | **1105**:<br>Prime factors: 5 × 13 × 17<br>gcd(2, 1105) = 1, so 1105 is odd, composite and coprime to 2. The Jacobi symbol is multiplicative:<br>$(\frac{2}{1105}) = (\frac{2}{5})(\frac{2}{13})(\frac{2}{17})$.<br>Use the formula for odd prime $p$: $(\frac{2}{p}) =$<br>$\begin{cases} +1 & \text{if } p \equiv \pm 1 \pmod 8), \\ -1 & \text{if } p \equiv \pm 3 \pmod 8). \end{cases}$<br>For $p = 5$: $5 \equiv 5 \equiv -3 \pmod 8) \Rightarrow (\frac{2}{5}) = -1$.<br>• For $p = 13$: $13 \equiv 5 \equiv -3 \pmod 8) \Rightarrow (\frac{2}{13}) = -1$.<br>• For $p = 17$: $17 \equiv 1 \pmod 8) \Rightarrow (\frac{2}{17}) = +1$.<br>$\Rightarrow (\frac{2}{1105}) = (-1) \cdot (-1) \cdot (+1) = +1$. Thus the Jacobi symbol is $(\frac{2}{1105}) = 1$.<br>$2^{\frac{1105-1}{2}} = 2^{552} \pmod{1105} \equiv 1$<br>$2^{\frac{1105-}{2}} \equiv \left(\frac{2}{1105}\right) \pmod{1105}$<br>while being composite ⇒ 1105 is an Euler–Jacobi pseudoprime to base 2.<br>OEIS A047713 |
| **Fermat pseudoprime** | Composite number n satisfying $a^{n-1} \equiv 1 \pmod n$ for at least one base a in {2, 3, 5, 7, 11, 13}. | **341**:<br>341 = 11 × 31 = composite<br>$2^{340}$ mod 341 = 1<br>So although 341 is composite, it passes Fermat's primality test for base 2 ⇒ 341 is a Fermat pseudoprime to base 2.<br>OEIS A001567 (base 2) |
| **Lucas–Carmichael number** | Odd, squarefree composite n such that p + 1 divides n + 1 for every prime divisor p of n. | **399**: Prime factors:<br>399 = 3 × 7 × 19, 399 is odd and composite. n + 1 = 400<br>Check divisibility condition:<br>For p=3, p+1=4, 400/4 = 100<br>For p=7, p+1=8, 400/8 = 50<br>For p=19, p+1=20, 400/20 = 20<br>Thus, for every prime divisor p of 399, p+1 divides 399+1<br>OEIS A006972 |

| Classifier | Definition / Description | Examples / OEIS |
|---|---|---|
| **RSA challenge number** | One of the published RSA Challenge semiprimes (RSA-100, RSA-110, RSA-120) | **1522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006139**: = 37975227936943673922808872755445627854565536638199 × 40094690950920881030683735292761468389214899724061<br>OEIS N/A |
| **Strong pseudoprime** | Composite number n that passes the Miller–Rabin strong probable prime test for at least one base in {2, 3, 5, 7, 11, 13}.<br>For an odd composite $n$, write $n-1=2^s\,d$ with $d$ odd.<br>Then $n$ is a strong pseudoprime to base $a$ if<br>• either $a^d \equiv 1 \pmod{n}$,<br>• or for some $0 \leq r < s,\ a^{2^r d} \equiv -1 \pmod{n}$.<br>(If a number passes this test but we don't yet know it's composite, we call it a *strong probable prime*; if we know it's composite, it's a *strong pseudoprime*.) | **3215031751**: n = 151 × 751 × 28351 (composite)<br>with $d = \frac{n-1}{2}$ = 1607515875<br>$2^d \equiv 1 \pmod{n}$<br>$3^d \equiv -1 \pmod{n}$<br>$5^d \equiv 1 \pmod{n}$<br>$7^d \equiv -1 \pmod{n}$<br>n passes the strong test for bases 2, 3, 5, and 7 even though it is composite<br>⇒ 3215031751 is a strong pseudoprime in this case to bases 2, 3, 5, 7.<br>OEIS A001262 (base 2)<br>OEIS A020229 (base 3)<br>OEIS A020231 (base 5)<br>OEIS A020233 (base 7) |

# Appendix B – Profile Settings

Notes on Defaults and Fallback Values

NumClass settings are organized into named sections (e.g. [FACTORING], [DISPLAY_SETTINGS], [CLASSIFIER.ZUMKELLER]).
A key in one section is entirely independent from a key with the same name in another section.

Global default vs. classifier-specific fallback
Most settings have a single default value shown in this appendix.
However, a small number of settings appear in multiple parts of the code, with different fallback defaults depending on the context.
These are rare but intentional:

- The global default shown in the table is the canonical value.
- Some classifiers implement a local fallback (usually smaller or stricter) that is used only when the key is not defined in the profile.

Rule:
If a value is set in the profile, that value overrides *all* fallback defaults everywhere in NumClass.

```
[_PROFILE_]
```
This section is used to describe the intention of the profile. (Not used in any settings.)

```
name = "all"
description = "Enable every category/classifier; useful for demos and
maximal coverage."
```

```
[CATEGORIES]
```
This section defines which classifier category is enabled (true) or disabled (false).

```
"Arithmetic and Divisor-based"            = true
"Combinatorial and Geometric"             = true
"Conjectures and Equation-based"          = true
"Digit-based"                             = true
"Diophantine Representations"             = true
"Dynamical Sequences"                     = true
"Fun Numbers"                             = true
"Mathematical Curiosities"                = true
"Named Sequences"                         = true
"Polygonal and Figurate Numbers"          = true
"Primes and Prime-related Numbers"        = true
"Pseudoprimes and Cryptographic Numbers" = true
```

```
[INCLUDE_EXCLUDE]
```
Specifies specific classifiers to be included or excluded.
Use the classifier label name in uppercase, replace (subsequent) non-alphanumeric characters with an underscore '_'. Example: label "collatz (3N+1) sequence" becomes "COLLATZ_3N_1_SEQUENCE"
If none are specified, all classifiers will be whitelisted.
Specifying the classifiers true will whitelist these classifiers, all others will be False.
Specifying the classifiers false will blacklist these classifiers, all others will be True.

[BEHAVIOUR]

| Name | Function | Type | Default |
|---|---|---|---|
| FAST_MODE | FAST_MODE = true<br>- NumClass operates in fast mode, heavy or slow classifiers may be skipped.<br>- Step-timeouts are enforced in long-running routines (e.g., aliquot sequences).<br>- Classifier-specific complexity caps are respected (e.g., maximum n thresholds, recursion depth limits, solver time budgets).<br>- Results remain correct, but not all classifications are attempted.<br><br>FAST_MODE = false<br>- NumClass operates in full computation mode, slow or expensive classifiers are allowed to run without limits on large integers or time budget.<br>- This can significantly increase runtime, especially for large integers or combinatorial classifiers. | Bool | true |
| MAX_DIGITS | Maximum allowed number of digits for any integer processed by NumClass. | Int | 100000 |

[FACTORING]

| Name | Function | Type | Default |
|---|---|---|---|
| BITLEN_CAP | Maximum bit-length NumClass will attempt to factor.<br>If a number exceeds this size, factoring is skipped to avoid impossible workloads. | Int | 10000 |
| MAX_TIME_S | Maximum total time (in seconds) NumClass will spend factoring a single number.<br>If the budget runs out, the remaining part is returned as an unfactored composite. | Float | 2.0 |
| SMALL_DIGIT_THRESHOLD | Digit count below which NumClass uses a fast, direct factoring method.<br>Larger numbers use the slower, time-limited “big-n” factoring path. | Int | 80 |
| STEP_LIMITS | Workload limits for the successive stages of the big-n factoring process.<br>Higher values allow deeper searches but use more time. | List | [1000, 10000, 100000, 1000000] |
| USE_ECM | Enable elliptic-curve factoring (ECM) where available.<br>May speed up factoring of certain hard composites at the cost of slightly higher overhead. | Int | False |

[DISPLAY_SETTINGS]

| Name | Function | Type | Default |
|---|---|---|---|
| DIVISOR_PRINT_LIMIT | Maximum number of individual divisors of \|n\| that NumClass will list explicitly.<br>If there are more divisors than this limit, the list is truncated with a "showing first ... of ..." note. | Int | 100 |
| REPEATING_DEC_PRINT_LIMIT | Maximum number of digits used when previewing the decimal expansion of 1/n in base 10.<br>Longer repeating decimals are cut off after these many digits, with an ellipsis if needed; period length logic is unaffected. | Int | 60 |
| SEQUENCE_PRINT_LIMIT | Maximum number of terms from each aliquot-style sequence (aliquot, quasi-aliquot, unitary) that are shown in the statistics block.<br>Longer sequences are truncated after these many steps. | Int | 50 (display), 300 (dynamic helpers) |
| SKIPPED_MAX_LINES | Threshold for showing individual skipped classifications. If the number of skipped items is ≤ this value, all are listed; if it is larger, only a count is shown. 0 means unlimited. | Int | 0 |
| SHOW_ALIQUOT_PROGRESS | Show live, step-by-step progress messages while computing aliquot-type sequences.<br>Turn this off to hide the live "evaluating..." / step trace and only show the final summarized sequence. | Bool | true |
| SHOW_ALIQUOT_SEQUENCE | Show the standard aliquot sequence of n in the "Divisor statistics of \|n\|" section.<br>Turn this off if you only want divisor sums and counts without the full s(n) → s(s(n)) → ... trajectory. | Bool | true |
| SHOW_ALIQUOT_TIME | Include the total computation time of the aliquot sequence in its summary line (after the sequence).<br>If disabled, the sequence and peak info are shown, but the elapsed time is omitted. | Bool | true |
| SHOW_CLASSIFIER_DETAILS | Show a second "Details:" line (or lines) under each classification, with explanations, witnesses, or cycle information.<br>Turn this off for a compact one-line-per-classifier view. | Bool | true |
| SHOW_COLLATZ_PEAK | Controls how the Collatz (3n+1) trajectory peak is displayed in dynamical-sequence output:<br>• "Full" – show the full peak information,<br>• "Shortened" – show an abbreviated summary,<br>• "None" – hide the Collatz peak details entirely. | String | "Full" |

| Name | Function | Type | Default |
|---|---|---|---|
| SHOW_DIGITAL_ROOT_SEQ | For the “Digital root of \|n\|” line, show the full digital-root sequence and additive persistence, not just the final root.<br>If disabled, only the final digital root and persistence are printed. | Bool | true |
| SHOW_DIVISOR_STATS | Enable or disable the entire “Divisor statistics of \|n\|” section (τ(n), divisors, σ/s/q, unitary sums, aliquot sequences, etc.).<br>Turn this off if you only care about basic number statistics and classifications. | Bool | true |
| SHOW_DIVISORS | Show the explicit list of divisors of \|n\| (up to DIVISOR_PRINT_LIMIT) in the divisor statistics section.<br>If disabled, divisor counts and sums are still shown, but the divisor list itself is hidden. | Bool | true |
| SHOW_MULTIPLICATIVE_SEQ | For “Mult. persistence”, show the full multiplicative persistence sequence (n → product of digits → …) instead of just the final persistence value.<br>Disable to keep only the final count. | Bool | true |
| SHOW_QUASI_ALIQUOT_SEQUENCE | Show the quasi-aliquot sequence of n, generated by iterating q(n) = s(n) − 1.<br>Disable to hide this sequence while keeping the main aliquot sequence. | Bool | true |
| SHOW_REPEATING_DEC_COUNT | On the 1/n (base 10) line, show the bracketed period information like [period k] or [rep≥L] where available.<br>If disabled, you still see the decimal preview, but not the explicit period length / lower bound. | Bool | true |
| SHOW_SKIPPED | Show the “Skipped classifications due to constraints or settings” footer listing classifiers that were not evaluated (e.g. time-capped, disabled, or size-limited).<br>Turn this off to omit the skipped-classifier list. | Bool | true |
| SHOW_UNITARY_ALIQUOT_SEQUENCE | Show the unitary aliquot sequence based on s*(n) (sum of proper unitary divisors) in the divisor statistics section.<br>If enabled together with the normal aliquot sequence, NumClass also notes when the unitary and classical sequences coincide. | Bool | false |
| SHOW_ZECKENDORF | Enable or disable display of the Zeckendorf decomposition in the *Number statistics* section. | | true |
| SHOW_ZECKENDORF_BITS | Show the Fibonacci-base bitstring corresponding to the Zeckendorf decomposition. | | true |

| Name | Function | Type | Default |
|---|---|---|---|
| `ZECKENDORF_MAX_TERMS` | Maximum number of Fibonacci terms shown in the Zeckendorf decomposition.<br>If the decomposition uses more terms, a compact summary is displayed instead.<br>A value of 0 disables truncation (show all terms). | | 20 |
| `ZECKENDORF_TRUNC_PREVIEW` | When the Zeckendorf decomposition is truncated, show a short preview of the first and last Fibonacci indices used. | | true |

`[FORMATTING]`

| Name | Function | Type | Default |
|---|---|---|---|
| `ELLIPSIS` | Symbol used when abbreviating long numbers (e.g., 1234567 → 123...567). | String | "..." |
| `ELLIPSIS_SPLIT` | Symbol used when splitting long sequences during truncation — the “big ellipsis” that appears between head and tail parts of a sequence preview. | String | "•••" |
| `NUM_ABBR_ENABLED` | Enable abbreviation of large integers in sequences and peak displays.<br>If disabled, all numbers are printed in full, regardless of size. | Bool | true |
| `NUM_ABBR_HEAD` | How many leading digits to keep when abbreviating a large integer (before the ellipsis). | Int | 10 |
| `NUM_ABBR_TAIL` | How many trailing digits to keep when abbreviating a large integer (after the ellipsis). | Int | 10 |
| `NUM_ABBR_THRESHOLD` | Minimum digit length at which integer abbreviation activates.<br>Numbers shorter than this threshold are printed in full. | Int | 10 |
| `REPEATING_DEC_STYLE` | Visual style used to mark the repeating part of the decimal expansion of 1/n:<br>• "paren" → (123)<br>• "overline" → $\overline{123}$<br>• "underline" → $\underline{123}$<br>This affects only display; period detection is unchanged. | String | "underline" |
| `REPEATING_DEC_MIN_REP_SHOWN` | Minimum number of digits required before NumClass shows a repeating block. Shorter repeats are displayed as non-repeating to avoid noisy output. | Int | 12 |
| `SEQUENCE_ARROW` | Symbol used to join terms in sequences (e.g., 3 → 10 → 5 → 16).<br>Changing it allows stylistic customization such as " -> ", " ⇒ ", etc. | String | " → " |
| `UNITFRAC_MAX_REDUCED` | Maximum value allowed when showing certain unit-fraction decompositions in reduced form.<br>Used as a safety cap to avoid | Int | 1000000 |

| | | | |
|---|---|---|---|
| | producing extremely large denominators in symbolic expansions. | | |

[OUTPUT]

Output file examples:

```
OUTPUT_FILE = ""            # No output file
OUTPUT_FILE = "."           # Write results to <n>.txt in the current folder
OUTPUT_FILE = "results/"    # Write results to folder/file: ./results/<n>.txt
OUTPUT_FILE = "log.txt"     # Append all results to a single file in the current folder
```

## Settings used by multiple classifiers and classifier helpers

[ALIQUOT]

| Name | Function | Type | Default |
|---|---|---|---|
| `MAX_STEPS` | Maximum number of steps (transitions) NumClass will follow in the aliquot, quasi-aliquot, or unitary aliquot sequence.<br>The sequence always includes the starting value, so a full run has max_steps + 1 terms. | Int | 250 (in aspiring, socially aspiring and sociable classifiers, 50 in divisor stats. |
| `STATS_MAX_PEAK_DIGITS` | Maximum allowed digit length of any term in the sequence.<br>If a term grows beyond this size, NumClass stops safely with: “peak digit cap exceeded”.<br>Useful for preventing enormous intermediates (e.g. factorial-like explosions) during explorations. | Int | 10000000 |
| `STATS_MAX_START_DIGITS` | Digit-length threshold above which aliquot statistics are skipped.<br>0 means unlimited. | Int | 0 |
| `STATS_MAX_TIME_S` | A hard limit (in seconds) for the entire sequence.<br>If the total elapsed time exceeds this cap, NumClass stops early with: “sequence time exceeded guard”.<br>This prevents runaway cases (e.g., hard σ(n) evaluations) while still allowing deep sequences. | Float | 0.5 |
| `STEP_TIME_LIMIT` | Per-step soft timeout (in seconds) used only when fast mode is enabled.<br>If a single σ(n) or σ*(n) computation takes longer than this, the sequence stops early with: “step time exceeded guard”.<br>This lets you explore many small aliquot steps very quickly without accidental slow terms. | Float | 0.30 |
| `VALUE_LIMIT` | Maximum allowed size of terms in aliquot-type sequences used by aspiring/sociable classifiers.<br>If a term exceeds this value, those classifiers stop early and treat the run as guarded. | Int | 10000000 |

[COLLATZ_LIKE]

| Name | Function | Type | Default |
|---|---|---|---|
| A_VALUE | Multiplier **a** used in the odd step of the generalized Collatz rule: odd → **a**·n + b. | Int | |
| B_VALUE | Additive constant **b** used in the odd step of the rule: odd → a·n + **b**. | Int | |
| ALLOW_EVEN_A_OR_B | Allow even values for **a** or **b** (default requires both to be odd). | Bool | false |
| A_B_LIMIT | Maximum allowed absolute value of **a** and **b**. | Int | 9 |
| DRIFT_BITS_EPS | Average bit-growth threshold used by the divergence guard.<br>If recent odd-macro-steps grow by more than this many bits on average, the sequence is considered diverging and is stopped. | Float | 0.25 |
| DRIFT_MIN_BITS | Minimum bit-length before the divergence guard is allowed to trigger. Prevents aborting while values are still small. | Int | 4096 |
| DRIFT_WINDOW | Number of recent macro-steps included in the sliding-window average for the divergence detector. | Int | 256 |
| MAX_STEPS | Maximum number of iteration steps allowed.<br>If reached, the sequence stops with reason "steps ≥ MAX_STEPS". | Int | 200000 |
| TIME_BUDGET_S | Maximum time (in seconds) that any single classifier is allowed to spend on its computation.<br>NumClass stops the computation early when exceeding this budget and reports the classifier as skipped. | float \| inf | None |
| VALUE_CAP_BITS | Safety cap on how large values in a generalized Collatz sequence may grow.<br>While A_VALUE and B_VALUE determine the growth rule (odd → a·n+b), this setting prevents runaway bit-length explosions by aborting the sequence once a term exceeds the allowed size. | Int | 20000 |

[COMBINATORIAL]

| Name | Function | Type | Default |
|---|---|---|---|
| TIME_BUDGET_S | Maximum wall-clock time (in seconds) that each combinatorial classifier is allowed to spend when fast mode is enabled.<br>If a check takes longer than this, it stops early and may fall back to an OEIS b-file (Bell, Catalan, Motzkin) or just give up. | Int | Float |

[DIOPHANTINE]

| Name | Function | Type | Default |
|---|---|---|---|
| `ALLOW_ZERO_IN_DECOMP` | Allow decompositions where one of the components is zero (e.g. $n = a^2 + 0^2$).<br>Affects all sum-of-k squares/cubes classifiers. | Bool | false for sums of 2 cubes/squares and 3 squares; true for sums of 3 cubes. |
| `MAX_ABS_FOR_SUM_OF_3_CUBES` | Maximum absolute value allowed for each variable a, b, c when searching solutions to: $n = a^3 + b^3 + c^3$<br>Larger values drastically increase search space; this caps the cube search box to [-B, B]. | Int | 100 |
| `MIN_PALINDROME_DIGITS` | Minimum number of digits a palindrome must have when used in decompositions like:<br>• n = a + b (sum of 2 palindromes)<br>• n = a + b + c (sum of 3 palindromes)<br>If the setting is omitted, no minimum is enforced. | Int | None |
| `MAX_SOL_SUM_OF_2_CUBES` | Maximum number of decompositions shown for $n = a^3 + b^3$.<br>Actual search may find more; only the first *N* canonical solutions are displayed. | Int | 20 |
| `MAX_SOL_SUM_OF_2_PALINDROMES` | Maximum number of decompositions shown for $n = p_1 + p_2$, where $p_i$ are palindromes. If the setting is omitted, show all found (may be many). | Int | None |
| `MAX_SOL_SUM_OF_2_SQUARES` | Maximum number of decompositions shown for $n = a^2 + b^2$<br>(canonical pairs $a \le b$, $a \ge 0$).<br>Does not affect existence checking — only how many are printed. | Int | 20 |
| `MAX_SOL_SUM_OF_3_CUBES` | Maximum number of decompositions shown for $n = a^3 + b^3 + c^3$.<br>NumClass always injects the known "celebrity solutions" even if capped. | Int | 20 |
| `MAX_SOL_SUM_OF_3_PALINDROMES` | Maximum number of decompositions shown for $n = p_1 + p_2 + p_3$, where $p_i$ are palindromes. | Int | None |
| `MAX_SOL_SUM_OF_3_SQUARES` | Maximum number of decompositions shown for $n = a^2 + b^2 + c^2$ (canonical triples).<br>Legendre's conditions still apply before enumeration. | Int | 20 |

## Classifier specific settings

[CLASSIFIER.BELL]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_K | Upper bound on the index $k$when searching for a Bell number $B(k)$. NumClass computes $B(0), B(1), \dots, B(k)$ up to this limit (and the time budget) before giving up and falling back to the OEIS b-file. | Int | 2500 |

[CLASSIFIER.CATALAN]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_K | Maximum index $k$to consider when checking whether n is a Catalan number $C_k$. NumClass iterates $C_0, C_1, \dots, C_k$ up to this cap (and the time budget) before giving up and using the OEIS b-file.<br>This is an index, not a value. | Int | 200000 |

[CLASSIFIER.DUCCI]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_STEPS | Maximum number of Ducci iterations NumClass will perform.<br>If this limit is reached, the computation stops and reports that the step cap was hit. | Int | 256 |

[CLASSIFIER.EGYPTIAN]

Used by Egyptian m/n, Erdős–Straus (4/n, k=3) and Sierpiński (5/n, k=4)

| Name | Function | Type | Default |
|---|---|---|---|
| DISTINCT | Require all denominators to be distinct ($1/d_1 + \dots + 1/d_k$ with all $d_i$ different). | Bool | true |
| K_VALUE | Number of unit fractions in the decomposition. | Int | 2 |
| M_VALUE | Numerator m in the target fraction m/n to be decomposed. | Int | 4 |
| MAX_DENOM | Upper bound on denominators $d_i$ in unit fractions ($1/d_i$). | Int | 10000000 |
| MAX_EXAMPLES | Maximum number of distinct decompositions to return; only the first is shown in details. | Int | 1 |

[CLASSIFIER.GOLDBACH]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_LIST | Maximum number of Goldbach pairs to display in the details (when listing examples). | Int | 10 |
| PROBE_CAP | In fast mode (no sieve, just probing primes), maximum number of candidate primes $p$ to try before giving up. | Int | 500000 |
| SHOW_ALL | Whether to list all Goldbach pairs or only up to MAX_LIST. | Bool | false |

| | | | |
|---|---|---|---|
| SHOW_COUNT | Whether to report the total number of Goldbach pairs for n (e.g., "(12 pairs; examples: ...)"). | Bool | True |
| SIEVE_MAXN | Maximum n for which NumClass will build an explicit sieve to enumerate *all* Goldbach pairs.<br>For n > SIEVE_MAXN, only the fast-probe path is used. | Int | 10000000 |

[CLASSIFIER.FUBINI]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_INDEX | Maximum index $n$to consider when checking whether v is a Fubini (ordered Bell) number $F(n)$.<br>NumClass computes $F(0), F(1), \dots, F(n)$ up to this limit (and the time budget) before stopping. | Int | 5000 |

[CLASSIFIER.GRAFTING]

| Name | Function | Type | Default |
|---|---|---|---|
| ALLOW_STRADDLE | Allow a "straddle" match, where part of $n$'s digits are at the end of the integer part and the rest at the start of the fractional part of $n^{1/p}$. | Bool | true |
| EARLY_STOP_ZEROS | Early-stop heuristic based on leading zeros in the fractional part of $n^{1/p}$.<br>If EARLY_STOP_ZEROS > 0 and the fractional part starts with at least that many zeros for a sufficiently large $n$, higher orders are not tried.<br>Set to 0 to disable this heuristic. | Int | 2 |
| GUARD_EXTRA_DIGITS | Extra safety margin on the number of fractional digits computed when checking for grafting (beyond the length of $n$). Higher values may improve robustness but cost more Decimal work. | Int | 8 |
| HARD_MAX_ORDER | Absolute upper bound on the exponent $p$.<br>Even if adaptive logic suggests a larger value, $p$ never exceeds this | Int | 12 |
| MAX_DIGITS | Maximum number of digits of $n$ for which the grafting test is attempted. Larger numbers are rejected immediately with a "maximum number of digits exceeded" message. | Int | 200 |
| MAX_ORDER | Soft upper bound on the exponent $p$ (root order) to try.<br>The classifier may use a smaller adaptive cap for very small or large numbers of digits. | Int | 6 |

| Name | Function | Type | Default |
|---|---|---|---|
| SIDE | Where the digits of $n$ are allowed to appear around the decimal point of $n^{1/p}$:<br>• "left" – only as a suffix of the integer part,<br>• "right" – only as a prefix of the fractional part,<br>• "either" – allow left, right, or a straddle across the decimal point. Invalid values fall back to "either". | String | "either" |

[CLASSIFIER.K_FULL]

| Name | Function | Type | Default |
|---|---|---|---|
| K_VALUE | Default exponent threshold k for the generic “k-full number” classifier (every prime in the factorization must appear with exponent ≥ k). | Int | 4 |

[CLASSIFIER.LAH]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_INDEX | Maximum index $n$to scan when looking for Lah numbers $L(n, k)$ that equal v. For each {MAX_INDEX}, NumClass scans all $k$ in that row; higher values allow more hits but cost more time. | Int | 10000 |

[CLASSIFIER.LCM_PREFIX]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_DIGITS | Safety cap for very large inputs. If the input integer has more than this many decimal digits, the LCM-prefix classifier skips it (to avoid expensive factorization on huge numbers). | Int | 2000 |
| MAX_K | Safety cap for the inferred witness value k. If the candidate k is larger than this limit, the classifier skips the “check all primes q ≤ k” step (which can be slow for very large k). | Int | 500000 |

[CLASSIFIER.LEGENDRE]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_LIST | Maximum number of primes between $n^2$ and $(n+1)^2$ to display. | Int | 20 |
| SHOW_ALL | Whether to list all primes in the interval $(n^2, (n+1)^2)$ or only up to MAX_LIST. | Bool | false |
| SHOW_COUNT | Whether to show the total number of primes in that interval. | Bool | true |

[CLASSIFIER.LEMOINE]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_LIST | Maximum number of (p, q) pairs to display. | Int | 10 |
| SHOW_ALL | Whether to list all representations n = p + 2q or only up to MAX_LIST. | Bool | false |

| | | | |
|---|---|---|---|
| SHOW_COUNT | Whether to include the total number of Lemoine representations. | Bool | true |

[CLASSIFIER.MOTZKIN]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_INDEX | Maximum index $n$to consider when checking whether v is a Fubini (ordered Bell) number $F(n)$. NumClass computes $F(0), F(1), \dots, F(n)$up to this limit (and the time budget) before stopping. | Int | 5000 |
| MAX_K | Maximum index $k$to consider when checking whether n is a Motzkin number $M_k$. NumClass walks the Motzkin recurrence up to this index (and the time budget) and then falls back to the OEIS b-file if needed. | Int | 200000 |

[CLASSIFIER.PRIMORIAL]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_DIGITS | Maximum decimal digits of \|n\| for which the primorial check is attempted; larger inputs are skipped. | Int | 200 |
| MAX_PRIMES | Maximum number of successive primes the primorial test may multiply; exceeded when p grows too large. | Int | 5000 |

[CLASSIFIER.RANDA] (Reverse-AND-Add)

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_STEPS | Maximum number of reverse-and-add iterations NumClass will perform when building the sequence (n → n + reverse(n)). If no palindrome is found within MAX_STEPS, the classifier stops and reports that no palindrome was reached within the cap. | Int | 1000 |
| PRINT_ALWAYS_IF | If the number of steps needed to reach a palindrome is ≤ this value, NumClass prints the full sequence, even if global sequence limits would normally truncate it. Sequence truncation still respects SEQUENCE_PRINT_LIMIT | Int | 30 |

[CLASSIFIER.ULAM]

| Name | Function | Type | Default |
|---|---|---|---|
| TIME_BUDGET_S | Maximum time (in seconds) that any single classifier is allowed to spend on its computation. NumClass stops the computation early when exceeding this budget and reports the classifier as skipped. | Float | 0.5 |

[CLASSIFIER.WEAK_GOLDBACH]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_LIST | Maximum number of triples p+q+r to display. | Int | 5 |
| SHOW_ALL | Whether to list all ternary representations n = p + q + r found within bounds. | Bool | false |
| SHOW_COUNT | Whether to include the total number of triples in the output. | Bool | true |

[CLASSIFIER.ZECKENDORF]

| Name | Function | Type | Default |
|---|---|---|---|
| SPARSE_MAX_TERMS | Maximum number of Fibonacci terms allowed in a *sparse Zeckendorf representation*.<br>If the Zeckendorf decomposition of a number uses at most this many terms, it is classified as *sparse*.<br>A value of 0 disables the sparse check. | Int | 3 |

[CLASSIFIER.ZNAM]

| Name | Function | Type | Default |
|---|---|---|---|
| MAX_BRANCH_PER_LEVEL | Maximum number of candidate children (branching factor) per search level.<br>Keeps the backtracker from exploding. | Int | 800 |
| MAX_EXAMPLES | Maximum number of Znám sets to keep (only the first is normally printed). | Int | 1 |
| MAX_PRODUCT_BITS | Maximum allowed bit-length of the product $\prod d_i$ during search.<br>Acts as a growth guard for candidate chains. | Int | 512 |
| MAX_WINDOW | Maximum window size for candidate generation when only one element is left to choose (final level). | Int | 200 |
| ODD_ONLY | Restrict the search to odd denominators only. | Bool | False |
| PREFIX_WINDOW | Base window size for candidate generation at intermediate levels (how far above the last element to scan for the next candidate). | Int | 200 |
| REQUIRE_PROPER | Require the chain to be a *proper* Znám set (i.e. $\sum 1/d_i = 1 - 1/\prod d_i$).<br>If true, small k are rejected (k must be 5–10). | Bool | true |
| START | Starting integer for the Znám set search (seed element of the chain). | Int | 2 |
| TIME_BUDGET_S | Maximum time (in seconds) that any single classifier is allowed to spend on its computation.<br>NumClass stops the computation early when exceeding this budget and reports the classifier as skipped. | Float | 0.5 |

[CLASSIFIER.ZUMKELLER]

| Name | Function | Type | Default |
|---|---|---|---|
| BITSET_TARGET_CAP | Zumkeller partitioning requires finding a subset of divisors that sums to half of the total divisor sum, often written as σ(n)/2.<br>BITSET_TARGET_CAP is the maximum allowed value of this half-sum for which the classifier will attempt the bitset dynamic-programming method.<br>If σ(n)/2 is greater than this cap, the bitset would be too large, and the classifier abandons this method and returns skipped. | Int | 10000000 |
| DIVISOR_LIMIT | The maximum number of divisors for which NumClass will attempt any Zumkeller computation.<br>If the number of divisors is greater than DIVISOR_LIMIT, the classifier immediately skips the number, because the divisor set is too large for safe processing. | Int | 5000 |
| SMALL_TAU_MITM | If the number of divisors is less than or equal to this value, NumClass uses the meet-in-the-middle (MITM) subset-sum solver, which is exact and very fast for small sets. | Int | 32 |
| WITNESS_TARGET_CAP | If σ(n)/2 is less than or equal to this value, and a valid partition is found, the classifier will also attempt to construct and display an explicit witness subset whose elements sum to σ(n)/2.<br>If σ(n)/2 is larger than this cap, the classifier will still detect that the number is Zumkeller but will not attempt to produce the actual subset (to avoid heavy work). | Int | 200000 |

# Appendix C – Glossary of Mathematical Terms and Symbols

This glossary provides short, clear explanations of mathematical terms and symbols used throughout the NumClass manual. It is intended for readers who wish to understand key concepts in number theory and how they relate to the classifications.

| Term | Definition / Explanation |
|---|---|
| **Abundancy ratio** | σ(n) / n, a measure of how "divisor-rich" a number is. |
| **Additive persistence** | Number of iterations needed to reach a single digit by repeatedly summing digits. |
| **Aliquot cycle** | A repeating loop in an aliquot sequence. |
| **Aliquot sequence** | Sequence n, s(n), s(s(n)), ... using the aliquot sum. |
| **Aliquot sum s(n)** | Sum of all proper divisors (that is, all positive divisors except n itself): s(n) = σ(n) − n. |
| **Base-10 expansion / repeating decimal** | Decimal representation of 1/n; period = repeating block, preperiod = the non-repeating part. |
| **Binomial coefficient** | Number of ways to choose k items from n without order, written as (n choose k), often denoted $\binom{n}{k}$. C(n,k) = $\binom{n}{k}$ = n! / (k!(n−k)!) |
| **Carmichael function λ(n)** | Smallest m with $a^m$ ≡ 1 (mod n) for all a coprime to n. |
| **Centered figurate number** | Figurate number with a central point and polygonal layers. |
| **Clique** | Set of graph vertices all mutually connected. |
| **Combination** | Selection of k items from n where order does not matter. |
| **Composite number** | Integer > 1 that is not prime. |
| **Concatenation (:)** | NumClass operator joining integers: 12 : 34 → 1234. |
| **Congruence** | a ≡ b (mod n) means n divides (a − b). |
| **Coprime (relatively prime)** | gcd(a, b) = 1. |
| **Cycle (iterative)** | Values that repeat during iteration of a function. |
| **Cycle structure** | Decomposition of a permutation into disjoint cycles. |
| **Digital root** | Repeated sum of digits until a single digit remains (n mod 9, with 9 instead of 0). |
| **Digit reversal** | Reversing decimal digits of n. |
| **Divisor** | d is a divisor of n if d divides n exactly. |
| **Divisor count τ(n)** | Number of positive divisors of n. |
| **Divisor sum σ(n)** | Sum of all positive divisors of n. |
| **Dynamic programming** | Solving problems by storing intermediate results; used in bitset DP and combinatorial classifiers. |
| **Euclidean algorithm** | Method for computing gcd(a, b). |
| **Even / Odd** | Even ↔ divisible by 2; odd ↔ not divisible by 2. |
| **Factorial n!** | Product n × (n−1) × ... × 1. |
| **Fibonacci sequence** | $F_0$=0, $F_1$=1, $F_n$=$F_{n-1}$+$F_{n-2}$; appears in many areas of math. |
| **Fixed point** | Value x satisfying f(x) = x. |
| **Floor ⌊x⌋ / Ceiling ⌈x⌉** | Greatest integer ≤ x / smallest integer ≥ x. |
| **GCD (greatest common divisor)** | Largest integer dividing both a and b. |
| **Graph coloring** | Assigning colors to vertices so adjacent ones differ. |
| **Independent set** | Graph vertices with no edges between them. |
| **Iterative process** | Repeatedly applying a function (e.g., Collatz, Kaprekar). |

| | |
|---|---|
| **Jacobi symbol (a \| n)** | Generalized quadratic-residue symbol. For odd n, (a \| n) behaves like the Legendre symbol when n is prime and is computed from the prime factors of n. Used in primality and pseudoprime tests. |
| **Lattice path** | Path in a grid using permitted step types (used in Catalan/Motzkin). |
| **LCM** | Least common multiple of two integers. |
| **Meet-in-the-middle (MITM)** | Algorithm splitting a problem into halves for faster search (used in Zumkeller). |
| **Modulo (mod)** | Arithmetic where numbers “wrap around” a modulus. |
| **Modular inverse** | $a^{-1}$ such that $a \cdot a^{-1} \equiv 1 \pmod{m}$ when gcd(a,m)=1. |
| **Monochromatic** | All items share the same color in a coloring. |
| **Multiplicative function** | f(ab) = f(a)f(b) whenever gcd(a, b) = 1. |
| **Multiplicative persistence** | Steps needed to reach a single digit via repeated digit multiplication. |
| **Multiplicative order** | Smallest k with $a^k \equiv 1 \pmod{m}$. |
| **OEIS** | Online Encyclopedia of Integer Sequences (https://oeis.org). |
| **Ordered list / ordered subset** | Selection of elements where order matters (used in Lah numbers). |
| **Parity** | Even/odd status of an integer. |
| **Partition (integer)** | Ways to write n as sum of positive integers ignoring order. |
| **Partition (set)** | Division of a set into disjoint non-empty subsets. |
| **Peak value** | Maximum value reached in a finite sequence. |
| **Period / Preperiod** | Repeating + non-repeating lengths of a decimal expansion. |
| **Permutation** | Arrangement of n elements in order. |
| **Polygonal number** | Number forming geometric regular polygons. |
| **Prime factor count Ω(n)** | Total prime factors, with multiplicity. |
| **Prime factor count ω(n)** | Number of distinct prime factors. |
| **Prime factorization** | Expression of n as product of primes. |
| **Primitive root** | g whose powers generate all nonzero residues mod p. |
| **Proper divisors** | Divisors of n excluding n itself. |
| **Pseudoprime** | Composite number passing certain prime tests. |
| **Quasi-aliquot sequence** | Iteration using q(n)=s(n)−1. |
| **Radical rad(n)** | Product of distinct prime factors. |
| **Reciprocal** | 1/n. |
| **Recurrence relation** | Sequence defined via previous terms. |
| **Reduced aliquot sum q(n)** | q(n) = s(n) − 1. |
| **Reduced unitary divisor sum q*(n)** | q*(n) = s*(n) − 1. |
| **Repdigit** | Number consisting of repeated digits. |
| **Repunit** | Repdigit using only digit 1. |
| **Residue class** | Set of all integers $\equiv a \pmod{m}$. |
| **Rough number** | Integer whose *smallest* prime factor exceeds a bound. |
| **Sequence** | Ordered list of numbers defined by rule/iteration. |
| **Set partition** | See Partition (of a set). |
| **Smooth number** | Integer whose *largest* prime factor ≤ bound. |
| **Smoothness parameter u** | $\log n / \log P^{+}(n)$. |
| **Squarefree** | Integer not divisible by any square >1. |
| **Totient ϕ(n)** | Number of integers ≤ n coprime to n. |
| **Totient iteration** | Repeated application of ϕ(n) until 1. |
| **Unit fraction** | Fraction of the form 1/n. |
| **Unitary divisor** | A divisor $d$ of $n$ such that $\gcd(d, n/d) = 1$. |
| **Unitary divisor count τ*(n)** | Number of unitary divisors. |

| | |
|---|---|
| **Unitary divisor sum σ*(n)** | Sum of all unitary divisors. |
| **Weak order (total preorder)** | Total, transitive relation allowing ties; used in Fubini numbers. |
| **Zeckendorf decomposition** | Unique representation of a non-negative integer as a sum of non-consecutive Fibonacci numbers. |
| **Zero divisor (mod m)** | $a \neq 0$ such that $a \cdot b \equiv 0 \pmod{m}$ for some $b \neq 0$. |