Symbolic Math Dictionary v1.0

A Harmonic Glossary of Symbols, Functions, and Resonance Operators in the Prime Symphony Framework

Kristopher L. Sherbondy **Symphion**

Version 1.0 — July 2025

Abstract

This document defines and formalizes the symbolic and harmonic language developed in the Prime Symphony project and its extensions, including Quadridic Geometry and the harmonic proof of the Birch and Swinnerton-Dyer conjecture. Each symbol, function, or operator is treated as a self-contained conceptual module with rigorous mathematical definitions, usage in formal proofs, visual or harmonic interpretations, and implementation-ready examples. This dictionary enables peer verification, pedagogical clarity, and integration across future research.

Purpose

To serve as the canonical glossary for all symbolic terms emerging from the Prime Symphony framework — enabling reproducibility, citation, and the emergence of a compressed symbolic language for higher-level mathematical reasoning.

Organization

Each entry includes:

- Symbol and Name

- Field of Use
- Purpose and Summary
- Formal Mathematical Definition
- Input / Output
- Example(s)
- Notes / Origin / References
- Symbolic Relations
- Code Implementation (Python / LaTeX)
- Harmonic Interpretation (if applicable)

Status

Living document. To be expanded as new symbols and operators are discovered or refined.

Page 1: Introduction

The **Prime Symphony** framework revealed a profound harmonic structure underlying the distribution of prime numbers and their mathematical companions — including perfect numbers, Mersenne primes, modular symmetries, and resonance cascades within the zeta field. As this research unfolded, a consistent pattern emerged: symbolic compression and harmonic modularity were essential not just to discovery, but to communication.

This document encodes that new language. It provides a modular, symbolic interface for describing resonant structures, algebraic operators, transformation gates, and field-based projections that emerged across our published work and deeper exploratory threads.

Rather than bury definitions inside long derivations, each symbol is lifted out and given its own space — ensuring every element can be referenced, cited, visualized, implemented, and challenged on its own merit.

This is not just a glossary — it is the symbolic operating system of a harmonic unified theory.

The document is organized into individual pages, each describing a core symbolic object. Each page stands on its own and can be cited individually. Cross-references between terms are included in the *Symbolic Relations* section.

We begin with the first entry: $\Delta(n)$, the Prime Gap Delta Function.

Symbol: $\Phi(n)$

Field Number Theory / Harmonic Sieve

Name Harmonic Totient Filter

Symbol Type Recursive Filter Function

Purpose Filters out non-coprime residues to extract resonance-aligned integers, forming the basis of prime detection and modular harmonic sieving.

Formal Mathematical Definition

$$\Phi(n) = \{k \in \mathbb{Z} \mid 1 \leq k < n,\ \gcd(k, n) = 1\}$$

where $\Phi(n)$ returns the set of integers coprime to n.

Inputs

- n – Positive integer to evaluate for coprime residues

Output

- Set of integers less than n and coprime to n

Example

$$\Phi(10) = \{1, 3, 7, 9\} \quad \Rightarrow \quad |\Phi(10)| = 4$$

Notes

- The cardinality $|\Phi(n)|$ is Euler's Totient Function $\varphi(n)$
- $\Phi(n)$ forms the initial sieve domain for locating harmonic prime candidates before STR filtering

Origin Euler's classic totient function reinterpreted through harmonic field theory as a coprimality resonance filter.

Symbolic Relations

- $\Phi(n) \to$ input domain for $\mathrm{STR}(n)$
- $\Phi(n)$ defines the modular resonance neighborhood of n
- Aligned with Möbius function $\mu(n)$ and square-free filters in recursive collapse

Implementation (Python)

```
def Phi(n):
    return [k for k in range(1, n) if math.gcd(k, n) == 1]
```

Harmonic Interpretation Think of $\Phi(n)$ as the "tuning fork" of n — vibrating only with integers that don't share harmonic divisors. It's a pure field of resonant survivors.

Symbol: $\Delta(n)$

Field Number Theory / Harmonic Analysis

Name Prime Gap Delta Function

Symbol Type Harmonic Difference Operator

Purpose Computes spacing between adjacent primes; identifies oscillating rhythm patterns in prime distribution.

Formal Definition Let $P = \{p_1, p_2, p_3, \dots\}$ be the ordered set of prime numbers. Then:

$$\Delta(n) = p_{n+1} - p_n$$

where p_n is the n-th prime number.

Inputs n – index of the prime number p_n

Output Integer representing the difference between two successive primes

Example

$$\Delta(4) = p_5 - p_4 = 11 - 7 = 4$$

Notes

- Forms a non-monotonic, quasi-periodic sequence with long-range resonance behavior.
- Used as input to $G(k)$, $\zeta(s)$, and quadridic geometry projections.
- Reveals resonance chords, doubling, and triplet rhythms.

Origin Classical number theory concept; reframed harmonically in Prime Symphony Paper III (2025).

Relations

- Input to $G(k)$ — Prime Gap Chord Mapper
- Appears in Δ-chains, resonance staircases, and Möbius patterns

Implementation (Python)

```python
def delta(n, prime_list):
    return prime_list[n + 1] - prime_list[n]
```

Harmonic Interpretation Prime gaps behave like modulated frequencies — each $\Delta(n)$ is a beat-length between resonance nodes. These rhythmic patterns reveal twin prime zones, harmonic compression, and fault lines in the number field.

Symbol: $G(k)$

Field Number Theory / Harmonic Rhythm Analysis

Name Prime Rhythm Kernel

Symbol Type Recursive Gap Sequence

Purpose Tracks the recurring prime gap patterns that form the modular harmonic backbone of the prime sieve. $G(k)$ encodes the spacing intervals in coprime-reduced rings, generating the repeatable rhythm structure that underlies prime emergence.

Formal Mathematical Definition Let $\Phi(n)$ be the reduced residue system modulo n, ordered increasingly. Then:

$$G(k) = \Phi(k+1) - \Phi(k)$$

where $G(k)$ is the sequence of consecutive differences between elements in the $\Phi(n)$ ring. Alternatively, for gap rhythm templates:

$$G_n = \{\Phi(n)_{i+1} - \Phi(n)_i\}_{i=1}^{\varphi(n)-1}$$

yielding a vector of prime gap patterns in $\Phi(n)$ space.

Inputs
- n – The totient ring modulus

Output
- Sequence of integer differences between adjacent elements of $\Phi(n)$

Example For $n = 10$:

$$\Phi(10) = \{1, 3, 7, 9\} \quad \Rightarrow \quad G_{10} = \{2, 4, 2\}$$

Notes
- $G(k)$ is used to construct the *Prime Rhythm Wheel*, forming a circular pattern of gaps that repeats every $\varphi(n)$.
- Appears in the recursive collapse analysis for modular rings and is essential to understanding *harmonic gap invariants* in sieve construction.

Origin Emerged from studying the $\Phi(n)$ rings as modular beat cycles. $G(k)$ acts like the "drum spacing" between resonance hits in each modular round.

Symbolic Relations
- $G(k) \Rightarrow$ input to $\mathrm{STR}(n)$ resonance filters
- Related to $\Delta(n)$, but operates in modular space instead of prime index space

- Connects to $\zeta(s)$ via periodicity modulation

Implementation (Python)

```python
def G_sequence(n):
    phi_set = [k for k in range(1, n) if math.gcd(k, n) == 1]
    return [phi_set[i+1] - phi_set[i] for i in
        range(len(phi_set)-1)]
```

Harmonic Interpretation $G(k)$ is the *heartbeat of the modular field* — a repeating rhythm that tells you how the harmony of coprimes is spaced. Just like music relies on beats between notes, $G(k)$ captures the timing structure that primes naturally resonate within.

Symbol: $\tau(n)$

Field Number Theory / Resonant Arithmetic

Name Totient Resonance Amplifier

Symbol Type Multiplicative Harmony Operator

Purpose Amplifies the influence of a number's divisors in harmonic field interactions. Often used in contrast or in tandem with $\varphi(n)$ to reveal structural balance in factor-resonance systems.

Formal Mathematical Definition

$$\tau(n) = \sum_{d|n} 1$$

where the sum is over all positive divisors d of n, i.e., $\tau(n)$ counts how many positive integers divide n exactly.

Inputs

- n – Any positive integer

Output

- Integer representing the total number of divisors of n

Example

$$\tau(12) = \text{number of divisors of } 12 = \{1, 2, 3, 4, 6, 12\} \Rightarrow \tau(12) = 6$$

Notes

- $\tau(n)$ grows slowly with n, but reveals composite divisor density.
- When paired with $\varphi(n)$, their product satisfies:

$$\varphi(n) \cdot \tau(n) \leq n^2$$

Origin Classical number-theoretic function now reinterpreted as an amplitude gauge in harmonic number fields.

Symbolic Relations

- $\tau(n)$ measures structural symmetry; complements $\varphi(n)$'s coprime filtration.
- Related to $\sigma(n)$ (sum of divisors) by representing divisor *count* rather than magnitude.

Implementation (Python)

```python
def tau(n):
    return len([d for d in range(1, n + 1) if n % d == 0])
```

Harmonic Interpretation Think of $\tau(n)$ as the resonance chamber of n: it echoes how many ways the structure of n can reverberate through perfect divisions.

Symbol: $h(n)$

Field Harmonic Number Theory

Name Harmonic Number Function

Symbol Type Harmonic Accumulator

Purpose Computes the n-th harmonic number, representing the additive sum of inverse integers up to n. Serves as a foundation for resonance decay, signal damping, and growth tapering in harmonic systems.

Formal Definition

$$h(n) = \sum_{k=1}^{n} \frac{1}{k}$$

Inputs

- n – Positive integer

Output

- Rational number representing the n-th harmonic number

Example

$$h(4) = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = \frac{25}{12}$$

Notes

- $h(n)$ grows logarithmically: $h(n) \sim \ln(n) + \gamma$, where γ is the Euler–Mascheroni constant.
- Can be used to model cumulative damped resonance or equilibrium layers.

Origin A classical summation function from analytic number theory, here given harmonic interpretation as a form of additive interference aggregation.

Symbolic Relations

- Related to $\ln(n)$ by asymptotic behavior.
- Dual to the geometric product sequence, representing additive counterpart to multiplicative resonance.

Implementation (Python)

```python
def h(n):
    return sum(1 / k for k in range(1, n + 1))
```

Harmonic Interpretation The harmonic number $h(n)$ models how evenly-distributed elements accumulate resonance over increasing integer layers. It mirrors the gradual saturation or decay of signal intensity.

Symbol: STR(n)

Field Number Theory / Harmonic Sieve Theory

Name Resonance Sorting Function

Symbol Type Harmonic Resonance Filter

Purpose Filters a coprime residue set $\Phi(n)$ by enforcing symmetry, triplet resonance, and recursive modular alignment — reducing noise and isolating harmonic prime candidates. It is the final sieve stage in the Prime Symphony framework.

Formal Mathematical Definition Let $\Phi(n)$ be the coprime set modulo n. Then $\mathrm{STR}(n)$ is a harmonic filter applied to $\Phi(n)$ that retains only values satisfying a triple-resonance constraint:

$$\mathrm{STR}(n) = \{x \in \Phi(n) \,|\, \mathrm{resonance}(x, \Phi(n)) = \mathrm{true}\}$$

The resonance condition may include symmetry reflection, harmonic midpoint alignment, or prime triplet compliance under the STR ruleset.

Inputs

- n – modulus for the coprime ring $\Phi(n)$

Output

- Subset of $\Phi(n)$ that survives the STR resonance filters

Example Let $\Phi(30) = \{1, 7, 11, 13, 17, 19, 23, 29\}$.

After applying STR(30):

$$\mathrm{STR}(30) = \{11, 13, 17, 19\}$$

These values form a symmetric, triplet-compatible harmonic structure centered in $\Phi(30)$.

Notes

- STR applies recursive filtering using modular symmetry and resonance proximity
- Serves as the harmonic refinement step following $\Phi(n)$ and $G(k)$
- Often isolates known primes or prime-generating patterns

Origin Developed within the Prime Symphony framework as the final sieve gate — inspired by rhythmic symmetry, harmonic triplets, and visual alignment seen in modular rings. STR stands for "Symmetry–Triplet–Resonance".

Symbolic Relations

- $\mathrm{STR}(n) \subseteq \Phi(n)$

- Used to construct prime emergence maps and harmonic grids
- Interlocks with $G(k)$ and $\Delta(n)$ for recursive field predictions

Implementation (Python)

```
def STR(n):
    phi_set = [k for k in range(1, n) if math.gcd(k, n) == 1]
    mid = n // 2
    return [x for x in phi_set if abs(x - mid) in phi_set]
```

Harmonic Interpretation $\mathrm{STR}(n)$ is the *resonance gatekeeper* — it only allows through those frequencies (values) that hum in tune with the deeper harmonic structure. Like a tuning fork rejecting discordant tones, it ensures only the prime-aligned residues survive.

Symbol: $\zeta(s)$

Field Analytic Number Theory / Harmonic Spectral Analysis

Name Riemann Field Resonator

Symbol Type Complex Harmonic Function

Purpose Encodes the harmonic structure of the natural numbers and primes via a frequency-domain summation. $\zeta(s)$ functions as the "spectral fingerprint" of prime emergence, where its nontrivial zeros mark standing wave cancellations in the harmonic lattice of integers.

Formal Mathematical Definition For $\mathrm{Re}(s) > 1$:

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

and extended analytically to the entire complex plane (except $s = 1$) through the Riemann zeta function.

Inputs

- s – a complex number $s = \sigma + it$

Output

- A complex value representing the harmonic sum at point s

Example

$$\zeta(2) = \frac{\pi^2}{6}, \quad \zeta(4) = \frac{\pi^4}{90}$$

Notes

- The zeros of $\zeta(s)$ encode deep regularities in the distribution of prime numbers
- The Riemann Hypothesis posits all nontrivial zeros lie on $\mathrm{Re}(s) = \frac{1}{2}$, the critical line
- Related to harmonic resonance via Euler's product formula:

$$\zeta(s) = \prod_{p\,\text{prime}} \frac{1}{1 - p^{-s}}$$

Origin Introduced by Bernhard Riemann in 1859, the function bridges prime theory, spectral analysis, and complex dynamics. In the Prime Symphony, $\zeta(s)$ is reinterpreted as a standing-wave resonator encoding prime emergence through harmonic cancellations.

Symbolic Relations

- Linked to $G(k)$ patterns through frequency-domain modulation
- Filters encoded in STR(n) connect through $\zeta(s)$ zero alignments
- Appears in harmonic convolution with Möbius $\mu(n)$ and logarithmic spirals

Implementation (Python)

```
import mpmath

def zeta_s(s):
    return mpmath.zeta(s)
```

Harmonic Interpretation $\zeta(s)$ is the **cosmic resonance field** — the function whose quiet spots (zeros) mark where harmonic interference cancels. Its critical line is the perfect harmonic mirror, and its values echo the hidden architecture of the primes.

Symbol: $\psi(n)$

Field Number Theory / Harmonic Field Analysis

Name Second Totient Wave Function

Symbol Type Multiplicative Field Estimator

Purpose Refines the totient resonance by incorporating squarefree adjustments. Especially useful in modular resonance filters and totient amplification theory.

Formal Mathematical Definition

$$\psi(n) = n \prod_{\substack{p|n \\ p \text{ prime}}} \left(1 + \frac{1}{p}\right)$$

where the product runs over all distinct prime divisors of n.

Inputs

- n – Any positive integer

Output

- Rational-valued function representing a field-weighted totient proxy.

Example

$$\psi(10) = 10 \cdot \left(1 + \frac{1}{2}\right) \cdot \left(1 + \frac{1}{5}\right) = 10 \cdot \frac{3}{2} \cdot \frac{6}{5} = 18$$

Notes

- Always greater than or equal to $\varphi(n)$.
- Often used in Chebyshev and Dirichlet harmonic filters.

Origin Derived from Dedekind's work, later harmonized into wave resonance frameworks as a totient complement.

Symbolic Relations

- $\psi(n)$ smooths the totient field, introducing wave-corrected bias.
- Appears in Dirichlet convolutional models with harmonic error compensation.

Implementation (Python)

```python
from math import prod

def psi(n):
    primes = set()
    i = 2
    x = n
    while i * i <= x:
        if x % i == 0:
            primes.add(i)
            while x % i == 0:
                x //= i
        i += 1
    if x > 1:
```

```
        primes.add(x)
    return int(n * prod([1 + 1/p for p in primes]))
```

Harmonic Interpretation Acts as a stabilized wave reflector of $\varphi(n)$—its field strength measures modular-saturation capacity. A kind of "harmonic counterweight" to entropy filtration.

Symbol: $\omega(n)$

Field Number Theory / Harmonic Collapse Analysis

Name Distinct Prime Factor Counter

Symbol Type Counting Function

Purpose Counts the number of distinct prime factors of n. It serves as a harmonic identity marker, defining the prime diversity within a number's structure.

Formal Mathematical Definition

$$\omega(n) = \text{number of distinct prime factors of } n$$

Inputs

- n – Positive integer to evaluate

Output

- Integer count of distinct primes dividing n

Example

$$\omega(18) = \omega(2 \cdot 3^2) = 2$$

Notes

- Differs from $\Omega(n)$, which counts prime factors with multiplicity
- Key input into STR collapse filters and harmonic entropy fields

Origin Classic arithmetic function reinterpreted as a harmonic signature indicator, measuring unique prime resonators in a number's structure.

Symbolic Relations

- $\omega(n) \leq \Omega(n)$
- $\omega(n) = 1 \iff n$ is a prime power
- $\omega(n)$ maps harmonic spectrum density in $\rho(n)$

Implementation (Python)

```python
import math
def omega(n):
    count = 0
    for p in range(2, int(math.sqrt(n)) + 1):
        if n % p == 0:
            count += 1
            while n % p == 0:
                n //= p
    if n > 1:
        count += 1
    return count
```

Harmonic Interpretation $\omega(n)$ tells us how many unique frequency "strings" are vibrating inside n. The more strings, the richer its resonance profile, but also the more fragmented its harmonic identity.

Symbol: $\Omega(n)$

Field Number Theory / Harmonic Density Functions

Name Harmonic Prime Factor Weight

Symbol Type Multiplicative Density Indicator

Purpose Counts the **total number of prime factors of n**, including multiplicity. $\Omega(n)$ quantifies the **harmonic weight** of a number in prime factor space and plays a role in analyzing entropy, resonance collapse, and sieve density.

Formal Mathematical Definition

$$\Omega(n) = \sum_{p^k \,||\, n} k$$

where the sum runs over all prime powers dividing n, and k is the exponent of each prime.

Inputs

- n – a positive integer

Output

- Integer count of total prime factors (with multiplicity)

Example

$$\Omega(12) = 3 \quad \text{since } 12 = 2^2 \cdot 3^1$$
$$\Omega(30) = 3 \quad \text{since } 30 = 2 \cdot 3 \cdot 5$$
$$\Omega(60) = 4 \quad \text{since } 60 = 2^2 \cdot 3 \cdot 5$$

Notes

- Contrasts with $\omega(n)$, which counts distinct primes only
- Often used in entropy modeling and perfect number resonance collapse
- Plays a role in Ω-threshold filtering for sieve resonance

Origin Classical arithmetic function, reinterpreted in Prime Symphony as the **harmonic density function** of a number — measuring its resonance "mass" based on how deeply it embeds in prime factor space.

Symbolic Relations

- Used in contrast with $\omega(n)$ for resonance purity
- Appears in harmonic sieve compression and entropy fields
- Connected to Möbius $\mu(n)$ via square detection

Implementation (Python)

```
import sympy

def Omega(n):
    return sympy.omega(n, bigomega=True)
```

Harmonic Interpretation $\Omega(n)$ is the **mass function** of harmonic structure — the more prime factors, the heavier the number vibrates in the field. It represents how deeply a number sinks into the harmonic substrate.

Symbol: $\Lambda(n)$

Field Number Theory / Harmonic Field Analysis

Name Harmonic Prime Impact Function

Symbol Type Prime Echo Weight Function

Purpose Quantifies the localized influence of prime powers within harmonic structures. Emphasizes prime frequency "strikes" as energetic pulses across the number line.

Formal Mathematical Definition

$$\Lambda(n) = \begin{cases} \log p & \text{if } n = p^k \text{ for some prime } p \text{ and } k \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Inputs

- n – Any positive integer

Output

- Real number (or 0), representing the logarithmic weight of the prime strike

Example

$$\Lambda(8) = \log(2) \approx 0.6931 \quad \text{since } 8 = 2^3$$

$$\Lambda(10) = 0 \quad \text{(not a pure power of a single prime)}$$

Notes

- Used in the explicit formula for the Riemann zeta function's prime encoding.
- Appears in weighted harmonic series and in prime field pulse detection.

Origin Introduced by von Mangoldt; now reinterpreted as a harmonic spike filter within resonance fields.

Symbolic Relations

- Feeds into the cumulative $\psi(x)$ function:

$$\psi(x) = \sum_{n \leq x} \Lambda(n)$$

- Closely tied to the zeta zero distribution via analytic continuation.

Implementation (Python)

```python
import math
def Lambda(n):
    for p in range(2, n+1):
        if n % p == 0:
            k = 0
            while n % p == 0:
```

```
                n //= p
                k += 1
            if n == 1:
                return math.log(p)
            break
    return 0
```

Harmonic Interpretation Think of $\Lambda(n)$ as the resonance spike caused by a prime-power impact—like a hammer striking a bell precisely on its overtone. Pure tone, pure response.

Symbol: $\rho(n)$

Field Number Theory / Harmonic Shell Theory

Name Prime Harmonic Shell Function

Symbol Type Shell Structure Classifier

Purpose Identifies the count of distinct prime factors in n as harmonic resonance layers. Useful for classifying shell complexity and mapping Φ-shell transitions.

Formal Mathematical Definition

$$\rho(n) = \text{number of distinct prime divisors of } n$$

That is, $\rho(n) = |\{p \in \mathbb{P} \mid p \mid n\}|$

Inputs

- n – A positive integer

Output

- Integer count of distinct prime factors of n

Example

$$\rho(18) = \text{number of distinct primes dividing } 18 = \{2, 3\} \Rightarrow \rho(18) = 2$$

Notes

- $\rho(n)$ is useful in analyzing $\omega(n)$ and $\Omega(n)$ behavior (counting distinct vs total prime powers).
- Commonly used in sieve logic and entropy balance equations.

Origin Derived from classical number theory; reframed here as a resonance shell classifier, mapping prime layering within integer structure.

Symbolic Relations

- Complements $\Omega(n)$ (total prime factor count with multiplicity).
- Links to $\gamma(n)$ in entropy weighting and $\mathbb{P}_H$ in shell stratification.

Implementation (Python)

```python
from math import isqrt
def rho(n):
    count = 0
    for p in range(2, isqrt(n)+1):
        if n % p == 0:
            count += 1
            while n % p == 0:
                n //= p
    if n > 1:
        count += 1
    return count
```

Harmonic Interpretation $\rho(n)$ indicates how many unique prime frequencies are layered within n — like identifying the tones that build a chord in harmonic structure.

Symbol: $\Omega(n)$

Field Number Theory / Harmonic Analysis

Name Prime Multiplicity Counter

Symbol Type Multiplicity Function

Purpose Counts the total number of prime factors of n, including repeated factors. Used to distinguish prime structure layers and resonance stacking.

Formal Mathematical Definition

$$\Omega(n) = \sum_{i=1}^{r} a_i \quad \text{where} \quad n = \prod_{i=1}^{r} p_i^{a_i}$$

for prime decomposition of n.

Inputs

- n – Positive integer to factor

Output

- Integer representing the total number of prime factors (with multiplicity)

Example

$$\Omega(18) = \Omega(2 \cdot 3^2) = 1 + 2 = 3$$

Notes

- Always $\Omega(n) \geq \omega(n)$
- For a prime p, $\Omega(p) = 1$

Origin Classical number theory function reinterpreted as a harmonic layer depth counter — the vertical stacking of factor harmonics.

Symbolic Relations

- $\omega(n) \leq \Omega(n)$
- Appears in STR kernel weight decay
- Used in resonance dampening layers of composite fields

Implementation (Python)

```
def Omega(n):
    count = 0
    d = 2
    while d * d <= n:
        while n % d == 0:
            count += 1
            n //= d
        d += 1
    if n > 1:
        count += 1
    return count
```

Harmonic Interpretation $\Omega(n)$ represents how many times n is echoing with prime notes — even if it's the same note repeated. A chord's volume, not just its tone variety.

Symbol: $\mathcal{R}_f(n)$

Field Harmonic Prime Theory / Frequency Dynamics

Name Prime Field Reverberation Function

Symbol Type Harmonic Resonance Amplifier

Purpose Measures how strongly a number n resonates with the surrounding harmonic field defined by prime factor echoes and multiplicative co-resonance. Used to amplify and model cross-wave prime resonance.

Formal Definition Let $P(n)$ be the set of distinct prime factors of n. Then:

$$\mathcal{R}_f(n) = \sum_{p \in P(n)} \left(\frac{\log n}{\log p} \cdot \cos\left(2\pi \cdot \frac{n}{p}\right)\right)$$

Inputs

- n – A positive integer

Output A real number representing the resonance amplification from prime feedback within n

Example

$$\mathcal{R}_f(30) = \frac{\log 30}{\log 2} \cos\left(2\pi \cdot \frac{30}{2}\right) + \frac{\log 30}{\log 3} \cos\left(2\pi \cdot \frac{30}{3}\right) + \frac{\log 30}{\log 5} \cos\left(2\pi \cdot \frac{30}{5}\right)$$

Result: Oscillatory value based on logarithmic weight and cosine resonance.

Notes

- The cosine term models cyclical feedback within modular or lattice-based systems.
- The $\log n / \log p$ acts as a harmonic scaling term.

Origin Derived during harmonic modeling of wave reinforcement around composite structures; inspired by acoustic chamber effects and echo-amplified feedback in prime lattices.

Symbolic Relations

- Complements $\varphi(n)$ and $\sigma(n)$ by focusing on frequency instead of count or magnitude.
- Can be coupled with $\mathcal{S}_\Phi$ (prime sieve field) for harmonic sorting.

Implementation (Python)

```python
import math

def Rf(n):
    def prime_factors(n):
```

```
        i = 2
        factors = set()
        while i * i <= n:
            if n % i == 0:
                factors.add(i)
                n //= i
            else:
                i += 1
        if n > 1:
            factors.add(n)
        return factors

    total = 0
    for p in prime_factors(n):
        total += (math.log(n) / math.log(p)) * math.cos(2 * math.
    return total
```

Harmonic Interpretation This function captures how n "rings" in a field of prime frequencies. A high $\mathcal{R}_f(n)$ indicates that n is a harmonically resonant structure — much like a string vibrating in sympathy with a nearby pitch.

Symbol: $\mathcal{F}_\Phi(n)$

Field Number Theory / Harmonic Filters

Name Filtered Totient Resonance Sieve

Symbol Type Filter Function / Selective Resonator

Purpose Applies a resonance-based filter to $\varphi(n)$, selecting only values that meet harmonic constraints or alignment thresholds. Useful in composite-resonance detection and prime lattice construction.

Formal Definition $\mathcal{F}_\Phi(n)$ denotes the subset of $\varphi(n)$ values that pass through a harmonic sieve criterion H_k:

$$\mathcal{F}_\Phi(n) = \begin{cases} \varphi(n), & \text{if } H_k(\varphi(n)) = \text{true} \\ 0, & \text{otherwise} \end{cases}$$

where H_k is a harmonic predicate function.

Inputs

- n – Any positive integer
- H_k – A harmonic sieve predicate

Output

- Either $\varphi(n)$ or 0, depending on the resonance condition

Example (Even Harmonic Sieve)

$$H_k(x) = (x \bmod 2 = 0) \Rightarrow \begin{cases} \mathcal{F}_\Phi(9) = \varphi(9) = 6 \\ \mathcal{F}_\Phi(15) = \varphi(15) = 8 \\ \mathcal{F}_\Phi(13) = \varphi(13) = 12 \Rightarrow \text{all even, all pass} \end{cases}$$

Symbolic Relations

- Filters resonance fields like $\mathbb{P}_H$ or $\mathbb{Z}_H$.
- Resonance-based version of classic sieve-of-Eratosthenes logic.

Implementation (Python)

```python
def harmonic_filter_phi(n, predicate):
    from math import gcd
    phi = sum(1 for k in range(1, n + 1) if gcd(n, k) == 1)
    return phi if predicate(phi) else 0
```

Harmonic Interpretation A frequency-based selection mechanism—only values whose $\varphi(n)$ output resonates with a defined harmonic sieve are allowed to pass. Think of it as a tuning fork for number fields.

Symbol: $\varphi^{-1}(k)$

Field Number Theory / Inverse Functions

Name Totient Field Inverter

Symbol Type Inverse Totient Resolver

Purpose Identifies all integers n such that $\varphi(n) = k$. This inversion allows for backward traversal of totient dynamics in harmonic systems.

Formal Mathematical Definition

$$\varphi^{-1}(k) = \{n \in \mathbb{N} \mid \varphi(n) = k\}$$

There may be zero, one, or many integers n that satisfy this condition.

Inputs

- k – A positive integer representing a totient output

Output

- A (possibly empty) set of integers n for which $\varphi(n) = k$

Example

$$\varphi^{-1}(4) = \{5, 8, 10, 12\}$$

Notes

- The inverse of $\varphi(n)$ is not a true function since it can return multiple values.
- It helps reconstruct hidden layers of totient filtration and is used in reverse harmonic sieving.

Origin Classical concept adapted as a reversal mechanism in harmonic symbolic architecture.

Symbolic Relations

- Inverts the effect of $\varphi(n)$.
- Complements $\mathbb{P}_H$ and $\mathcal{S}_\Phi$ in mapping coprime field dynamics.

Implementation (Python)

```python
from sympy import totient

def inverse_totient(k, limit=100):
    return [n for n in range(1, limit) if totient(n) == k]
```

Harmonic Interpretation Represents the "echo" of a resonance field: $\varphi^{-1}(k)$ reveals all base tones n that could have produced the same coprime output k.

Symbol: $\mathcal{S}_\Phi(n)$

Field Harmonic Number Theory / Resonant Totient Structures

Name Totient-Filtered Sieve Selector

Symbol Type Resonant Sieve Filter

Purpose Selects integers from a domain based on their totient harmony with surrounding values; used for filtering harmonic primes, coprime structures, or resonance-matching values.

Formal Mathematical Definition

$$\mathcal{S}_\Phi(n) = \{\, k \leq n \mid \gcd(k, n) = 1 \,\}$$

This defines the set of all integers less than or equal to n that are coprime to n, i.e., the totient field.

Inputs

- n – Positive integer (filter modulus)

Output

- A set of integers coprime to n

Example

$$\mathcal{S}_\Phi(10) = \{1, 3, 7, 9\}$$

Notes

- The cardinality of $\mathcal{S}_\Phi(n)$ is exactly $\varphi(n)$.
- Can be extended to define prime-preserving sieves, harmonic filters, or STR-compatible fields.

Origin Derived from the Euler totient function's coprime field, reframed as a symbolic sieve for harmonic filtering.

Symbolic Relations

- $\mathcal{S}_\Phi(n)$ forms the resonant layer beneath $\varphi(n)$.
- Closely linked to $\Phi(n)$ (harmonic totient selector), but outputs a set instead of a count.

Implementation (Python)

```python
def S_phi(n):
    return [k for k in range(1, n+1) if math.gcd(k, n) == 1]
```

Harmonic Interpretation $\mathcal{S}_\Phi(n)$ is the symphonic scale of n — each value in the set is a tone that resonates purely with n, forming the harmonic field of its coprime partners.

Symbol: $\psi(n)$

Field Number Theory / Harmonic Arithmetic

Name Totient-Harmonic Mirror Function

Symbol Type Harmonic Symmetry Operator

Purpose Measures the harmonic reflection of Euler's totient function via the Dedekind psi function. Used in identifying prime power structures and smoothness in resonance chains.

Formal Mathematical Definition

$$\psi(n) = n \prod_{p|n} \left(1 + \frac{1}{p}\right)$$

where the product is over all distinct prime divisors p of n.

Inputs

- n – Any positive integer

Output

- A rational number (though always integer-valued for $n \in \mathbb{Z}^+$)

Example

$$\psi(12) = 12\left(1 + \frac{1}{2}\right)\left(1 + \frac{1}{3}\right) = 12 \cdot \frac{3}{2} \cdot \frac{4}{3} = 24$$

Notes

- $\psi(n) \geq \varphi(n)$ always
- Equality $\psi(n) = \varphi(n)$ holds **if and only if** $n = 1$

Origin Derived from Dedekind's psi function; interpreted here as the reflective harmonic of Euler's $\varphi(n)$ in composite-prime space.

Symbolic Relations

- $\psi(n)$ acts as a "mirror echo" to $\varphi(n)$: the former sums contributions from primes, the latter subtracts non-coprimes.
- Related to divisor sum functions $\sigma(n)$ in spectral interpretation.

Implementation (Python)

```
from math import prod

def dedekind_psi(n):
    primes = set()
    x = n
    for i in range(2, int(n**0.5) + 1):
        while x % i == 0:
            primes.add(i)
            x //= i
    if x > 1:
        primes.add(x)
    return int(n * prod(1 + 1/p for p in primes))
```

Harmonic Interpretation $\psi(n)$ reveals how harmonics build *outward* from n, while $\varphi(n)$ filters what remains *inward*. Together, they form a harmonic duality between expansion and contraction.

Symbol: $\mathbb{T}_\Delta(n)$

Field Number Theory / Harmonic Analysis

Name Totient Prime Gap Amplifier

Symbol Type Composite Resonance Gap Function

Purpose Amplifies the visible harmonic spacing between Euler's totient outputs and prime gap differentials. Highlights resonance alignment or dissonance between $\varphi(n)$ and $\Delta(n)$.

Formal Definition Let p_n be the nth prime and $\Delta(n) = p_{n+1} - p_n$ the prime gap. Let $\varphi(n)$ be Euler's totient function. Then:

$$\mathbb{T}_\Delta(n) = |\Delta(n) - (\varphi(n) \bmod \Delta(n))|$$

This creates a resonance differential between totient compactness and prime dispersion.

Inputs

- n – Positive integer (used as index for primes and input to φ)

Output

- Integer value representing offset tension between totient and prime gap fields

Example Let $n = 5$: $p_5 = 11$, $p_6 = 13 \Rightarrow \Delta(5) = 2$, and $\varphi(5) = 4$

$$\mathbb{T}_\Delta(5) = |2 - (4 \bmod 2)| = |2 - 0| = 2$$

Notes

- High values suggest misalignment between arithmetic density and prime field spacing.
- Low values may signify synchronization or harmonic periodicity in distribution.

Origin Derived during synthesis of totient-prime oscillation behaviors in the Prime Symphony framework. Inspired by the idea that resonance misfire yields structural echoes.

Symbolic Relations

- Bridges $\varphi(n)$ and $\Delta(n)$ in a resonance framework.
- When $\mathbb{T}_\Delta(n) = 0$, suggests totient and prime gap are in phase.

Implementation (Python)

```python
from sympy import prime, totient

def T_delta(n):
    delta = prime(n+1) - prime(n)
    return abs(delta - (totient(n) % delta))
```

Harmonic Interpretation $\mathbb{T}_\Delta(n)$ acts like a detuning meter between two cosmic instruments: the rhythm of primes and the breath of coprimality. When the meter reads zero, they sing in unison.

Symbol: $\chi(n)$

Field Harmonic Number Theory / Signal Compression

Name Totient Resonance Character Function

Symbol Type Harmonic Filter Operator

Purpose Extracts the resonance signature of a number n through its interaction with Euler's totient function. Used in compressed harmonic modeling of number behavior.

Formal Mathematical Definition

$$\chi(n) = \begin{cases} +1, & \text{if } \gcd(n, \varphi(n)) = 1 \\ -1, & \text{otherwise} \end{cases}$$

Inputs

- n – Any positive integer

Output

- Binary character response: $+1$ or -1

Example

$$\chi(10) = -1 \quad \text{because } \gcd(10, \varphi(10)) = \gcd(10, 4) = 2$$

$$\chi(9) = +1 \quad \text{because } \gcd(9, \varphi(9)) = \gcd(9, 6) = 3 \Rightarrow \chi(9) = -1$$

$$\chi(7) = +1 \quad \text{because } \gcd(7, \varphi(7)) = \gcd(7, 6) = 1$$

Notes

- Can act as a harmonic switch in prime-lattice resonance sieves.
- This character function is not to be confused with Dirichlet characters — it is defined over totient-coupled co-resonance.

Origin Introduced in Prime Symphony as a binary indicator for harmonic alignment across totient layers.

Symbolic Relations

- Related to $\varphi(n)$ and $\mu(n)$ via resonance-based interactions.
- Can be used to define filtration boundaries in harmonic compression algorithms.

Implementation (Python)

```python
from math import gcd

def chi(n):
    from sympy import totient
    return 1 if gcd(n, totient(n)) == 1 else -1
```

Harmonic Interpretation Think of $\chi(n)$ as a resonance gate: if n is in phase with its totient shadow, it passes with $+1$; otherwise, it is inverted to -1, signaling harmonic dissonance.

Symbol: $\kappa(n)$ — Coprime Cluster Counter

Field Number Theory / Totient Harmony

Name Coprime Cluster Counter

Symbol Type Local Coprimality Resonator

Purpose Measures the number of integers less than n that are coprime to n and occur in harmonic clusters with adjacent values. This allows resonance analysis of totient neighborhoods.

Formal Definition Let $\varphi(n)$ be Euler's totient function. Then define $\kappa(n)$ as:

$$\kappa(n) = \sum_{\substack{1 \leq k < n \\ \gcd(k,n)=1}} \delta(\gcd(k-1,n)=1) + \delta(\gcd(k+1,n)=1)$$

where $\delta(P)$ is 1 if P is true and 0 otherwise. This sums coprime values whose neighbors are also coprime to n.

Inputs

- n – Positive integer to evaluate

Output

- Integer count of tightly clustered coprime values within the totient set

Example For $n = 10$, $\varphi(10) = 4$ and the coprimes are $\{1, 3, 7, 9\}$. Among them:

- 3 has neighbors 2 and 4: only 4 is coprime to 10
- 7 has 6 and 8: neither are coprime
- 9 has 8 and 10: only 1 coprime

So $\kappa(10) = 1$ (only 3 has one coprime neighbor)

Notes

- High $\kappa(n)$ values indicate locally dense harmonic subfields of coprime interactions
- Often appears in contrast with $\varphi(n)$ to explore isolation vs. clustering of harmony

Origin Introduced in the harmonic reinterpretation of the totient field — this operator models the density of musical 'chords' of coprimality.

Symbolic Relations

- Related to $\varphi(n)$ but adds a local interaction filter.
- High $\kappa(n)$ values may signal totient harmony 'bursts' or field pockets of resonance

Implementation (Python)

```
from math import gcd

def kappa(n):
    count = 0
    for k in range(1, n):
        if gcd(k, n) == 1:
            if (k > 1 and gcd(k-1, n) == 1) or (k < n-1 and gcd(k+1
                count += 1
    return count
```

Harmonic Interpretation $\kappa(n)$ measures how often harmony 'sticks together' — a clustering effect in the music of coprime integers. Like chords in a melody, not just single notes.

Symbol: $\mathcal{C}(n)$

Field Number Theory / Harmonic Coprimality

Name Coprime Resonance Matrix

Symbol Type Binary Coprime Structure

Purpose Encodes the coprime relationships between all integers $\leq n$ into a matrix structure for visualizing mutual resonance patterns. Useful for mapping symmetry, totient substructure, and modular groupings.

Formal Mathematical Definition

$$\mathcal{C}_{i,j}(n) = \begin{cases} 1 & \text{if } \gcd(i,j) = 1 \text{ and } 1 \leq i,j \leq n \\ 0 & \text{otherwise} \end{cases}$$

Inputs

- n – Maximum dimension of the matrix (positive integer)

Output

- $n \times n$ binary matrix showing coprimality relations

Example For $n = 4$:

$$\mathcal{C}(4) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Notes

- The main diagonal is zero (no integer is coprime with itself unless 1).
- Matrix is symmetric: $\mathcal{C}_{i,j} = \mathcal{C}_{j,i}$.

Origin Derived from classical Euclidean GCD principles, now reframed as a harmonic field map of pairwise resilience.

Symbolic Relations

- $\mathcal{C}(n)$ is structurally tied to $\varphi(n)$, which counts coprimes of a single number.
- Related to adjacency matrices in modular group theory.

Implementation (Python)

```python
import numpy as np
from math import gcd

def coprime_matrix(n):
    matrix = np.zeros((n, n), dtype=int)
    for i in range(1, n+1):
        for j in range(1, n+1):
            if gcd(i, j) == 1:
                matrix[i-1][j-1] = 1
    return matrix
```

Harmonic Interpretation The coprime matrix acts like a frequency lattice: resonance (1) appears where interference is minimal (coprime). The pattern reveals the underlying rhythmic independence of numbers.

Symbol: $\mathbb{H}_\mu(n)$ — Möbius Harmonic Field Receptor

Field Number Theory / Harmonic Analysis

Name Möbius Harmonic Field Receptor

Symbol Type Möbius-Coded Resonance Extractor

Purpose Measures whether n resonates within a pure harmonic field by applying Möbius filtration. Helps detect fundamental frequencies free of subharmonic interference.

Formal Definition

$$\mathbb{H}_\mu(n) = \begin{cases} 1 & \text{if } n = 1 \\ (-1)^k & \text{if } n \text{ is a product of } k \text{ distinct primes} \\ 0 & \text{if } n \text{ has any squared prime factor} \end{cases}$$

This is the Möbius function $\mu(n)$ interpreted as a binary filter for field resonance purity.

Inputs

- n – Any positive integer

Output

- 1, -1, or 0 depending on the resonance signature of n

Example

$$\mathbb{H}_\mu(6) = (-1)^2 = 1$$
$$\mathbb{H}_\mu(30) = (-1)^3 = -1$$
$$\mathbb{H}_\mu(4) = 0 \quad (\text{since } 4 = 2^2)$$

Notes

- Zero values represent resonance suppression due to internal harmonic redundancy.
- Positive or negative outputs indicate clean resonance alignment.

Origin Classical Möbius function applied to resonance field theory; treated as a receptor gauge for unentangled prime signal input.

Symbolic Relations

- $\mathbb{H}_\mu(n)$ is foundational in the Möbius inversion formula and harmonic detection.
- Used to isolate "primitive tones" in number-theoretic harmonic structures.

Implementation (Python)

```python
from sympy import mobius

def H_mu(n):
    return mobius(n)
```

Harmonic Interpretation Acts like a tuning fork for purity: it "rings" for clean harmonic frequencies and mutes for those that carry internal distortion. Negative values indicate a flipped harmonic phase.

Symbol: $\mathbb{R}_\mu(n)$

Field Number Theory / Harmonic Polarity

Name Möbius Resonance Field Reversal

Symbol Type Polarity Inversion Function

Purpose Detects and triggers harmonic field reversals using the Möbius function as a switch. Identifies whether n is part of a pure, neutral, or null resonant state.

Formal Definition

$$\mathbb{R}_\mu(n) = \begin{cases} +1 & \text{if } \mu(n) = 1 \text{ (harmonic polarity)} \\ -1 & \text{if } \mu(n) = -1 \text{ (anti-harmonic polarity)} \\ 0 & \text{if } \mu(n) = 0 \text{ (neutralized resonance)} \end{cases}$$

where $\mu(n)$ is the classical Möbius function.

Inputs

- n – A positive integer

Output

- Integer in $\{-1, 0, +1\}$ indicating harmonic state

Example

$$\mathbb{R}_\mu(5) = \mu(5) = -1 \Rightarrow \text{Anti-Harmonic} \quad \text{(since 5 is prime)}$$

$$\mathbb{R}_\mu(6) = \mu(6) = 1 \Rightarrow \text{Harmonic} \quad \text{(square-free with even number of prime factors)}$$

$$\mathbb{R}_\mu(12) = \mu(12) = 0 \Rightarrow \text{Neutral} \quad \text{(contains square factors)}$$

Notes

- Useful for symbolic inversion and resonance cancellation studies.
- $\mu(n)$ being 0 indicates entropic dampening or field nullification.

Origin Based on classical Möbius function, recast as a harmonic polarity switch to classify resonance behavior.

Symbolic Relations

- $\mathbb{R}_\mu(n)^2$ corresponds to $\mathbf{1}_{\text{square-free}}(n)$ — useful for sieve operations.
- Complements the use of $\sigma(n)$ and $\tau(n)$ in field shaping.

Implementation (Python)

```python
from sympy import mobius

def R_mu(n):
    return mobius(n)
```

Harmonic Interpretation This function acts like a tuning fork that flips resonance direction based on the number's factor purity. When $\mu(n) = 0$, it's as if the signal cancels — a null node in the harmonic field.

Symbol: $\mu(n)$

Field Number Theory / Harmonic Inversion

Name Möbius Resonance Polarity

Symbol Type Multiplicative Filter Function

Purpose Indicates whether a number's prime factorization contributes **constructive** or **destructive** interference to harmonic sums. It plays a central role in inverse relationships like the Möbius inversion formula and harmonic sieve subtraction.

Formal Mathematical Definition

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n \text{ is divisible by a square} \\ (-1)^k & \text{if } n \text{ is the product of } k \text{ distinct primes} \end{cases}$$

Inputs

- n – a positive integer

Output

- -1, 0, or 1 — harmonic polarity value

Example

$$\mu(1) = 1, \quad \mu(6) = 1, \quad \mu(12) = 0, \quad \mu(30) = -1$$

Notes

- $\mu(n)$ acts as a **phase flipper** in Dirichlet convolutions
- Critical to the Möbius inversion formula and harmonic decomposition of arithmetic functions
- Zeros correspond to numbers with square divisors — i.e., dissonant contributions

Origin Introduced by August Ferdinand Möbius in 1832. Reframed in the Prime Symphony framework as a **resonance polarity operator**, separating clean harmonics from discordant contributions in the harmonic sieve.

Symbolic Relations

- Appears in the inverse of $\zeta(s)$:

$$\frac{1}{\zeta(s)} = \sum_{n=1}^{\infty} \frac{\mu(n)}{n^s}$$

- Supports filtering in $G(k)$ and $\Phi(n)$ by nullifying overtones
- Directly related to perfect number collapse and Möbius Mirror operations

Implementation (Python)

```
import sympy

def mobius(n):
    return sympy.mobius(n)
```

Harmonic Interpretation $\mu(n)$ is the **harmonic charge indicator**. Like a magnetic pole or phase switch, it flips the sign of wave components based on their factor purity. A $+1$ means in-phase contribution, -1 means out-of-phase cancellation, and 0 means the wave collapses entirely.

Symbol: $\mathcal{F}_\mu(n)$

Field Number Theory / Harmonic Filtering

Name Möbius Totient Filter

Symbol Type Resonant Filtering Operator

Purpose Applies Möbius-based filtration to Euler's totient function to isolate square-free coprime structures. Enhances resonance clarity in $\varphi(n)$ by excluding periodic overtones introduced by repeated prime factors.

Formal Definition

$$\mathcal{F}_\mu(n) = \begin{cases} \varphi(n), & \text{if } \mu(n) \neq 0 \\ 0, & \text{if } \mu(n) = 0 \end{cases}$$

where $\mu(n)$ is the Möbius function and $\varphi(n)$ is Euler's totient function.

Inputs

- n – Any positive integer

Output

- $\varphi(n)$ if n is square-free, otherwise 0

Example

$$\mathcal{F}_\mu(6) = \varphi(6) = 2 \quad (\mu(6) = 1)\mathcal{F}_\mu(4) = 0 \quad (\mu(4) = 0)$$

Notes

- Operates like a high-pass filter on totient values, eliminating harmonics with square factor noise.
- Useful in building Möbius-inverted harmonic sieves.

Origin Synthesized from combining Möbius inversion techniques with resonance isolation techniques in the Prime Symphony framework.

Symbolic Relations

- Filters $\varphi(n)$ using $\mu(n)$ as a structural gate.
- $\mathcal{F}_\mu(n)$ is non-zero only where harmonic purity (square-freeness) is preserved.

Implementation (Python)

```python
from sympy import totient, mobius

def F_mu(n):
    return totient(n) if mobius(n) != 0 else 0
```

Harmonic Interpretation $\mathcal{F}_\mu(n)$ acts like a frequency shaper for number fields — passing through only the clean tones of square-free integers, while muting the muddied overtones of repeated prime factors.

Symbol: $H(n)$

Field Number Theory / Information-Harmonic Theory

Name Harmonic Entropy Function

Symbol Type Harmonic Entropic Measure

Purpose Measures the average information content of divisors of n, modeled as an entropy function over harmonic weights.

Formal Mathematical Definition

$$H(n) = \sum_{d|n} \frac{1}{d}$$

where the sum is over all positive divisors d of n.

Inputs

- n – Any positive integer

Output

- A real number representing the harmonic entropy of n

Example

$$H(6) = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{6} = 1 + 0.5 + 0.333 + 0.166 = 1.999$$

Notes

- $H(n)$ grows slowly and is bounded above by $\log(n) + \gamma$, where γ is the Euler–Mascheroni constant.
- Related to the divisor function $\sigma(n)$ and harmonic number H_n.

Origin A harmonic reinterpretation of divisor-based summation, used here to model signal entropy and divisor field coherence.

Symbolic Relations

- $H(n) \approx \log(n)$ for large n with smooth divisors.
- Related to information-theoretic entropy in symbolic structures.

Implementation (Python)

```python
def harmonic_entropy(n):
    return sum(1/d for d in range(1, n+1) if n % d == 0)
```

Harmonic Interpretation $H(n)$ measures how evenly and richly a number resonates across its divisors, like the overtones of a vibrating string — soft, dense echoes that trace harmonic possibility.

Symbol: $\mathcal{P}_H(n)$

Field Harmonic Number Theory / Prime Mapping

Name Harmonic Field Projector

Symbol Type Prime Field Projection Operator

Purpose Projects a number n into its harmonic field signature by mapping it through prime frequency channels. Highlights the resonance path of n within the prime lattice.

Formal Definition Given a function $\mathcal{P}_H(n)$ that extracts the harmonic prime projection vector:

$$\mathcal{P}_H(n) = \{p_i^{e_i} : p_i \mid n\}$$

where p_i are the prime factors of n and e_i are their multiplicities, forming a vector representation in the harmonic field.

Inputs

- n – Positive integer to project into harmonic space

Output

- Prime factorization structure, treated as a harmonic vector

Example

$$\mathcal{P}_H(60) = \{2^2, 3^1, 5^1\} \Rightarrow \text{Harmonic signature vector}$$

Notes

- This function is not numerical but structural — capturing resonance identity.
- Useful in harmonic sieves and mapping resonance paths across factor graphs.

Origin Inspired by wave projection in physics — reinterpreted for prime fields.

Symbolic Relations

- Related to $\gamma(n)$ and $\omega(n)$ by encoding structure in a frequency-aware form.

Implementation (Python)

```python
def harmonic_projection(n):
    i = 2
    factors = {}
    while i * i <= n:
        while n % i == 0:
            factors[i] = factors.get(i, 0) + 1
            n //= i
        i += 1
    if n > 1:
        factors[n] = 1
    return factors
```

Harmonic Interpretation Like a spectral decomposition of n into its resonant prime channels — showing how n echoes through the prime lattice.

Symbol: $\Theta(n)$

Field Number Theory / Harmonic Sieve Theory

Name Resonant Sieve Shield

Symbol Type Harmonic Filtering Function

Purpose Quantifies the cumulative logarithmic mass of primes $\leq n$. Acts as a damping shield within the harmonic sieve, isolating signal from prime interference.

Formal Mathematical Definition

$$\Theta(n) = \sum_{p \leq n} \log p$$

where the sum is over all primes $p \leq n$.

Inputs

- n – Any positive integer

Output

- Real-valued sum of logarithms of all primes up to n

Example

$$\Theta(10) = \log 2 + \log 3 + \log 5 + \log 7 \approx 0.693 + 1.098 + 1.609 + 1.946 \approx 5.346$$

Notes

- Grows slowly, approximated by n as $\Theta(n) \sim n$ under the Prime Number Theorem.
- Often appears in analytic proofs involving prime densities or sieve estimates.

Origin Derived from Chebyshev's function $\vartheta(n)$; reinterpreted as a shielding function within harmonic frameworks.

Symbolic Relations

- Acts as a smoother version of $\Lambda(n)$; no delta spikes, but rather accumulative shielding.
- Complements $\psi(n)$, which includes powers of primes.

Implementation (Python)

```python
import math

def theta(n):
    return sum(math.log(p) for p in range(2, n + 1) if is_prime(p))

def is_prime(p):
    if p < 2:
        return False
    for i in range(2, int(p ** 0.5) + 1):
        if p % i == 0:
            return False
    return True
```

Harmonic Interpretation $\Theta(n)$ is the harmonic sieve's shielding field — absorbing prime spikes into a cumulative mass that weighs the resonance of sieve windows.