



Math Functions & Methods

How to use Math Module

How to use Math Module

Ruby

```
include Math
```

C

```
#include <math.h>
```

How to use Math Module

Java

```
import java.lang.Math.*;
```

Python

```
import math
#or
import cmath # สำหรับจัดการจำนวนเชิงซ้อนจะมีวงจรจำนวนเชิงซ้อนต่อท้าย
#print(cmath.sqrt(9)) # Output: (3+0j)
#สำหรับเอกสารนี้อธิบายเพียง math
```

Math Functions & Methods

Constants

π e

Methods

$F(x)$

Math Constants

Math.constants in Ruby

```
puts Math.constants
```

Output:

DomainError

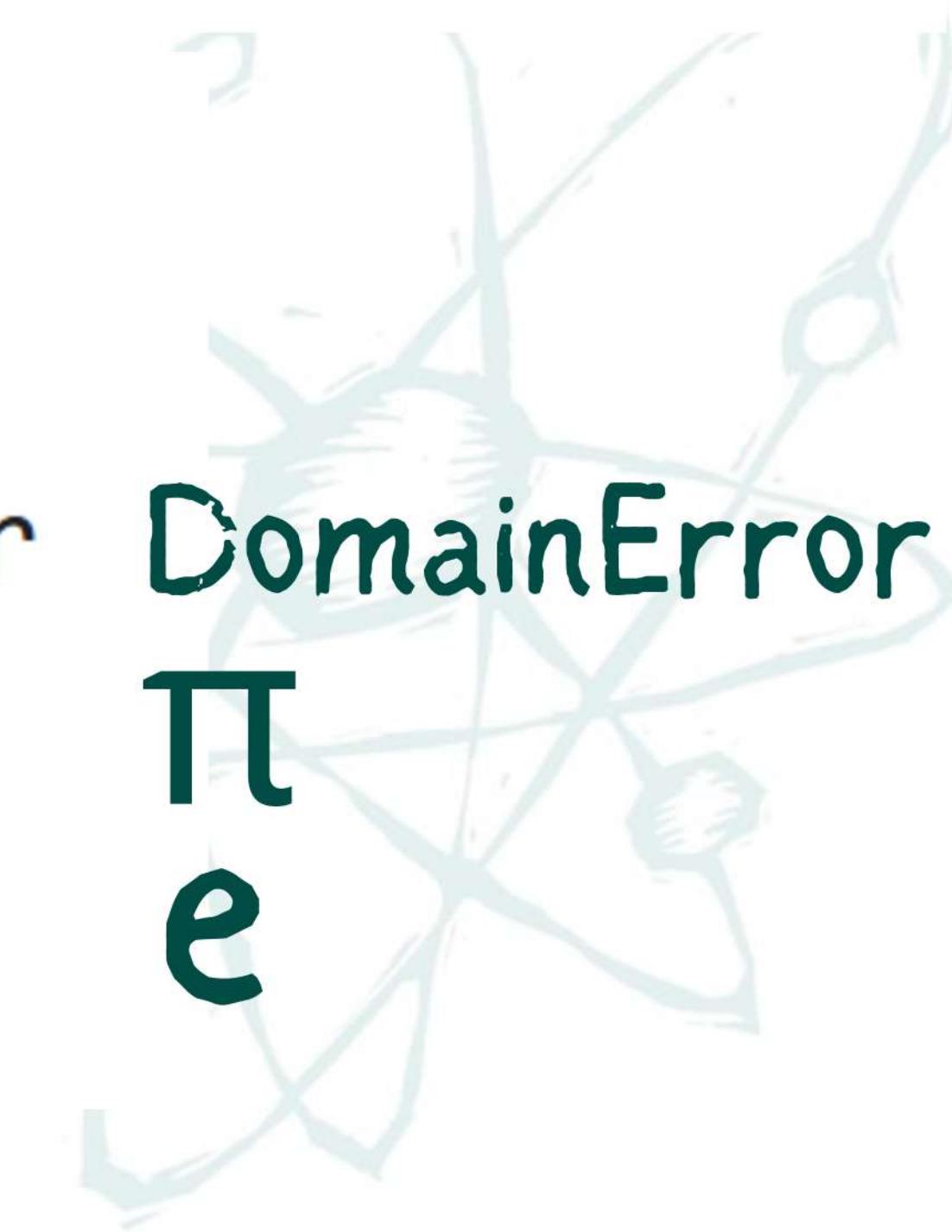
PI

E

DomainError

π

e



Math Constants in Ruby

เรียกด้วย `Math::` ได้เนื่องจากต้องการ `include Math`

```
Math::PI    # => 3.141592653589793  
Math::E     # => 2.718281828459045
```

หาก `include Math` แล้วสามารถ `PI` สำหรับใช้ค่าพาย หรือ `E` สำหรับค่าอยเลอร์

```
include Math  
PI    # => 3.141592653589793  
E     # => 2.718281828459045
```

Math Constants in C

ค่าคงที่ทางคณิตศาสตร์ไม่ถูกกำหนดใน C มาตรฐาน

หากต้องการใช้ค่าคงที่ทางคณิตศาสตร์จะต้องเพิ่ม

```
#define _USE_MATH_DEFINES  
#include <math.h>
```

```
#define _USE_MATH_DEFINES // for C  
#include <math.h>
```

เมื่อเพิ่มเรียบร้อยสามารถใช้ `M_PI` สำหรับค่าพาย และ `M_E` สำหรับค่าอยเลอร์

```
M_PI // 3.14159265358979323846  
M_E // 2.71828182845904523536
```



Math Constants in Java

ในภาษาเขียนโปรแกรม Java ไม่จำเป็นต้อง `import java.lang.math;` เพราะ `java.lang` จะถูก import โดยอัตโนมัติ เปรียบเสมือนมี `import java.lang.*;` ที่จุดเริ่มของ compiler กันกีหลังจาก package ได้ ๆ

`Math.PI` สำหรับใช้ค่าพาย และ `Math.E` สำหรับใช้ค่าอยเลอร์

`Math.PI` // 3.141592653589793

`Math.E` // 2.718281828459045

Math Constants in Python

Module Math เป็น module มาตรฐานของ Python
ต้อง **import math** ก่อน

```
import math
```

เมื่อเพิ่มเรียบร้อยสามารถใช้
math.pi สำหรับค่าพาย และ **math.e** สำหรับค่าอยเลอร์

```
math.pi # 3.141592653589793  
math.e # 2.718281828459045
```



Math Functions & Methods

Constants

π e

Methods

$F(x)$

Math Methods in Ruby

[P01.Trigonometric Functions](#)

[P02.Inverse Trigonometric Functions](#)

[P03.Hyperbolic Trigonometric Functions](#)

[P04.Inverse Hyperbolic Trigonometric Functions](#)

[P05.Exponentiation and Logarithmic Functions](#)

[P06.Fraction and Exponent Functions](#)

[P07.Root Functions](#)

[P08.Error Functions](#)

[P09.Gamma Functions](#)

[P10.Hypotenuse Function](#)

Math Methods

Math Methods in Ruby

เรียกด้วย `Math`. เมื่อ `include Math` ไม่ต้องการ `include Math`

```
Math.sin(0.0) # => 0.0  
Math.cos(0.0) # => 1.0
```

หาก `include Math` และสามารถใช้ เมื่อ `include Math` ได้โดยตรง

```
include Math  
sin(0.0) # => 0.0  
cos(0.0) # => 1.0
```



Math Methods in C

เมื่อ #include <math.h> และสามารถเรียกใช้ฟังก์ชันคณิตศาสตร์ได้เลย

```
sin(M_PI/4) // 0.707107  
cos(M_PI/4) // 0.707107  
tan(M_PI/4) // 1.000000
```

ตั้งแต่ C99 เป็นต้นไป หากใช้ฟังก์ชันคณิตศาสตร์ที่เติม f หรือ l ต่อท้าย จะสามารถระบุชนิดค่าส่งคืนที่ต้องการได้

```
sinf( float arg ); // return float  
sin( double arg ); // return double  
sinl( long double arg ); // return long double
```



Math Methods in Java

ในภาษาเขียนโปรแกรม Java ไม่จำเป็นต้อง `import java.lang.math;` เพราะ `java.lang` จะถูก import โดยอัตโนมัติ เปรียบเสมือนมี `import java.lang.*;` ที่จุดเริ่มของ compiler กันกีหลังจาก package ได้ ๆ

ใช้ฟังก์ชันทางคณิตศาสตร์ด้วย `Math`. เมื่อเดินทางคณิตศาสตร์

```
Math.sin(Math.PI/4) // 0.7071067811865475  
Math.cos(Math.PI/4) // 0.7071067811865475  
Math.tan(Math.PI/4) // 0.9999999999999999
```



Math Methods in Python

Module Math เป็น module มาตรฐานของ Python
ต้อง **import math** ก่อน

```
import math
```

เมื่อเพิ่มเรียบร้อยสามารถใช้ฟังก์ชันคณิตศาสตร์ได้โดย
math.เมธอดที่ต้องการ

```
math.sin(math.pi/4) # 0.7071067811865475
math.cos(math.pi/4) # 0.7071067811865476
math.tan(math.pi/4) # 0.9999999999999999
```



Functions Math

ทุกฟังก์ชันคณิตศาสตร์จะมี Domain และ Range

Domain ช่วงค่าเลขที่ใส่ฟังก์ชันได้

Range ช่วงค่าเลขที่ส่งกลับมา

ฟังก์ชันตรีโกณมิติ ฟังก์ชันตรีโกณมิติพกผัน ฟังก์ชันตรีโกณมิติไอเพอร์โบลิก
ฟังก์ชันตรีโกณมิติไอเพอร์โบลิกพกผัน ทั้งหมดใช้ระบบเรเดียน



Trigonometric Functions in Ruby

- **cos(x)** : ส่งคืนค่า cosine ของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [-1.0, 1.0]
- **sin(x)** : ส่งคืนค่า sine ของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [-1.0, 1.0]
- **tan(x)** : ส่งคืนค่า tangent ของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** (-INFINITY, INFINITY)



Examples:

```
cos(-PI)    # => -1.0
cos(-PI/2)   # => 6.12303176911886e-17 # 0.0000000000000001
cos(0.0)     # => 1.0
cos(PI/2)    # => 6.12303176911886e-17 # 0.0000000000000001
cos(PI)      # => -1.0
```

Examples:

```
sin(-PI)    # => -1.2246063538223773e-16 # -0.0000000000000001
sin(-PI/2)   # => -1.0
sin(0.0)     # => 0.0
sin(PI/2)    # => 1.0
sin(PI)      # => 1.2246063538223773e-16 # 0.0000000000000001
```

Examples:

```
tan(-PI)    # => 1.2246467991473532e-16 # -0.0000000000000001
tan(-PI/2)   # => -1.633123935319537e+16 # -16331239353195370.0
tan(0.0)     # => 0.0
tan(PI/2)    # => 1.633123935319537e+16 # 16331239353195370.0
tan(PI)      # => -1.2246467991473532e-16 # -0.0000000000000001
```

Trigonometric Functions in C

`sin(double x)`

`cos(double x)`

`tan(double x)`

```
sin(M_PI/4) // 0.707107
cos(M_PI/4) // 0.707107
tan(M_PI/4) // 1.000000
```



Trigonometric Functions in Java

Math.sin(double x)

Math.cos(double x)

Math.tan(double x)

```
Math.sin(Math.PI/4) // 0.7071067811865475
Math.cos(Math.PI/4) // 0.7071067811865475
Math.tan(Math.PI/4) // 0.9999999999999999
```



Trigonometric Functions in Python

```
math.sin( double x )
```

```
math.cos( double x )
```

```
math.tan( double x )
```

```
math.sin(math.pi/4) # 0.7071067811865475
math.cos(math.pi/4) # 0.7071067811865476
math.tan(math.pi/4) # 0.9999999999999999
```



Inverse Trigonometric Functions in Ruby

- **acos(x)** : ส่งคืนค่า arccos ของ x
 - **Domain:** [-1, 1]
 - **Range:** [0, PI]
- **atan(x)** : ส่งคืนค่า arctan ของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [-PI/2, PI/2]
- **asin(x)** : ส่งคืนค่า arcsin ของ x
 - **Domain:** [-1, 1]
 - **Range:** [-PI/2, PI/2]
- **atan2(y, x)** : ส่งคืนค่า arctan ของ y และ x
 - **Domain of y:** (-INFINITY, INFINITY)
 - **Domain of x:** (-INFINITY, INFINITY)
 - **Range:** [-PI, PI]





มหาวิทยาลัยมหิดล

Examples:

```
acos(-1.0) # => 3.141592653589793 # PI  
acos(0.0)  # => 1.5707963267948966 # PI/2  
acos(1.0)  # => 0.0
```

Examples:

```
asin(-1.0) # => -1.5707963267948966 # -PI/2  
asin(0.0)  # => 0.0  
asin(1.0)  # => 1.5707963267948966 # PI/2
```

Examples:

```
atan(-Float::INFINITY) # => -1.5707963267948966 # -PI/2
atan(-PI)               # => -1.2626272556789115
atan(-PI/2)             # => -1.0038848218538872
atan(0.0)               # => 0.0
atan(PI/2)              # => 1.0038848218538872
atan(PI)                # => 1.2626272556789115
atan(Float::INFINITY)   # => 1.5707963267948966 # PI/2
```

Examples:

```
atan2(-1.0, -1.0) # => -2.356194490192345 # -3*PI/4
atan2(-1.0, 0.0)  # => -1.5707963267948966 # -PI/2
atan2(-1.0, 1.0)  # => -0.7853981633974483 # -PI/4
atan2(0.0, -1.0)  # => 3.141592653589793 # PI
```

Inverse Trigonometric Functions in C

asin(double x)

acos(double x)

atan(double x)

atan2(double y, double x)

```
asin(0.900000) // 1.119770
acos(0.900000) // 0.451027
atan(0.900000) // 0.732815
atan2(7.000000, -7.000000) //2.356194
```



Inverse Trigonometric Functions in Java

Math.asin(double x)

Math.acos(double x)

Math.atan(double x)

Math.atan2(double y, double x)

```
Math.asin(0.900000) // 1.1197695149986342
Math.acos(0.900000) // 0.45102681179626236
Math.atan(0.900000) // 0.7328151017865066
Math.atan2(7.000000,-7.000000) // 2.356194490192345
```



Inverse Trigonometric Functions in Python

```
math.asin( double x )
```

```
math.acos( double x )
```

```
math.atan( double x )
```

```
math.atan2( double x )
```

```
math.asin(0.9) # 1.1197695149986342
math.acos(0.9) # 0.45102681179626236
math.atan(0.9) # 0.7328151017865066
math.atan2(7, -7) # 2.356194490192345
```



Hyperbolic Trigonometric Functions in Ruby

- **cosh(x)** : ส่งคืนค่า \cosh ของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [1, INFINITY]
- **sinh(x)** : ส่งคืนค่า \sinh ของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** (-INFINITY, INFINITY)
- **tanh(x)** : ส่งคืนค่า \tanh ของพารามิเตอร์ที่ใส่
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [-1, 1]



Examples:

```
cosh(-Float::INFINITY) # => Infinity  
cosh(0.0)               # => 1.0  
cosh(Float::INFINITY)   # => Infinity
```

Examples:

```
sinh(-Float::INFINITY) # => -Infinity  
sinh(0.0)               # => 0.0  
sinh(Float::INFINITY)   # => Infinity
```

Examples:

```
tanh(-Float::INFINITY) # => -1.0  
tanh(0.0)               # => 0.0  
tanh(Float::INFINITY)   # => 1.0
```



Hyperbolic Trigonometric Functions in C

`sinh(double x)`

`cosh(double x)`

`tanh(double x)`

```
sinh(M_PI/4) // 0.868671
cosh(M_PI/4) // 1.324609
tanh(M_PI/4) // 0.655794
```



มหาวิทยาลัยมาหิดล
Mahidol University

Hyperbolic Trigonometric Functions in Java

```
Math.sinh( double x )
```

```
Math.cosh( double x )
```

```
Math.tanh( double x )
```

```
Math.sinh(Math.PI/4) // 0.8686709614860095
```

```
Math.cosh(Math.PI/4) // 1.3246090892520057
```

```
Math.tanh(Math.PI/4) // 0.6557942026326724
```



Hyperbolic Trigonometric Functions in Python

math.sinh(double x)

math.cosh(double x)

math.tanh(double x)

```
math.sinh(math.PI/4) # 0.8686709614860095
math.cosh(math.PI/4) # 1.3246090892520057
math.tanh(math.PI/4) # 0.6557942026326724
```



Inverse Hyperbolic Trigonometric Functions in Ruby

- **acosh(x)** : ส่งคืนค่า arccosh ของ x

- **Domain:** `[1, INFINITY)`

- **Range:** `[0, INFINITY)`

- **asinh(x)** : ส่งคืนค่า arcsinh ของ x

- **Domain:** `(-INFINITY, INFINITY)`

- **Range:** `(-INFINITY, INFINITY)`

- **atanh(x)** : ส่งคืนค่า arctanh ของ x

- **Domain:** `[-1, 1]`

- **Range:** `(-INFINITY, INFINITY)`



Examples:

```
acosh(1.0)      # => 0.0
acosh(Float::INFINITY) # => Infinity
```

Examples:

```
asinh(-Float::INFINITY) # => -Infinity
asinh(0.0)      # => 0.0
asinh(Float::INFINITY) # => Infinity
```



Examples:

```
atanh(-1.0) # => -Infinity
atanh(0.0)  # => 0.0
atanh(1.0)  # => Infinity
```



Inverse Hyperbolic Trigonometric Functions in C

asinh(double x)

acosh(double x)

atanh(double x)

```
asinh(0.868671) // 0.785398
acosh(1.324609) // 0.785398
atanh(0.655794) // 0.785398
```



Inverse Hyperbolic Trigonometric Functions in Java

คำเตือน: ไม่มีการใช้ asinh acosh atanh ในภาษาเขียนโปรแกรม Java ต้องใช้สูตรบดิททางคณิตศาสตร์

```
asinh = Math.log(x + Math.sqrt(x * x + 1))
```

```
acosh = Math.log(x + Math.sqrt(x * x - 1))
```

```
atanh = 0.5 * Math.log((1 + x) / (1 - x))
```

```
public static double asinh(double x) {  
    return Math.log(x + Math.sqrt(x * x + 1));  
}  
  
public static double acosh(double x) {  
    // โดเมนของ acosh(x) คือ x >= 1.  
    if (x < 1) {  
        return Double.NaN;  
    }  
    return Math.log(x + Math.sqrt(x * x - 1));  
}  
  
public static double atanh(double x) {  
    // โดเมนของ atanh(x) คือ -1 < x < 1.  
    if (x <= -1 || x >= 1) {  
        return Double.NaN;  
    }  
    return 0.5 * Math.log((1 + x) / (1 - x));  
}
```

```
asinh(0.868671) // 0.7853981924731886  
acosh(1.324609) // 0.7853980606519705  
atanh(0.655794) // 0.7853978078604246
```

Inverse Hyperbolic Trigonometric Functions in Python

`math.asinh(double x)`

`math.acosh(double x)`

`math.atanh(double x)`

```
math.asinh(0.868671) # 0.7853981924731888  
math.acosh(1.324609) # 0.7853980606519706  
math.atanh(0.655794) # 0.7853978078604246
```



Exponentiation & Logarithmic Functions in Ruby

- **exp(x)** : ส่งคืนค่า e ยกกำลังด้วย x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [0, INFINITY]
- **expm1(x)** : ส่งคืนค่า e ยกกำลังด้วย x จากนั้นลบด้วย 1
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** [-1.0, INFINITY)
- **log(x, base=Math::E)** : ส่งคืนค่าลอการิทึมของ x ฐาน $base$ (หารูร่วมชาร์ต (e))
 - **Domain:** [0, INFINITY)
 - **Range:** (-INFINITY, INFINITY)

Examples:

```
exp(-Float::INFINITY) # => 0.0
exp(-1.0)             # => 0.36787944117144233 # 1.0/E
exp(0.0)              # => 1.0
exp(0.5)              # => 1.6487212707001282 # sqrt(E)
exp(1.0)              # => 2.718281828459045 # E
exp(2.0)              # => 7.38905609893065 # E**2
exp(Float::INFINITY)  # => Infinity
```

Examples:

```
expm1(-Float::INFINITY) # => 0.0
expm1(-1.0)             # => -0.6321205588285577 # 1.0/E - 1
expm1(0.0)              # => 0.0
expm1(0.5)              # => 0.6487212707001282 # sqrt(E) - 1
expm1(1.0)              # => 1.718281828459045 # E - 1
expm1(2.0)              # => 6.38905609893065 # E**2 - 1
expm1(Float::INFINITY)  # => Infinity
```

Examples:

```
log(0.0)               # => -Infinity
log(1.0)               # => 0.0
log(E)                 # => 1.0
log(Float::INFINITY)   # => Infinity

log(0.0, 2.0)           # => -Infinity
log(1.0, 2.0)           # => 0.0
log(2.0, 2.0)           # => 1.0
```

```
log(0.0, 10.0)          # => -Infinity
log(1.0, 10.0)          # => 0.0
log(10.0, 10.0)         # => 1.0
```

- **log10(x)** : ส่งคืนค่าลอการิทึมของ x ฐาน 10

- **Domain:** $[0, \text{INFINITY})$
- **Range:** $(-\text{INFINITY}, \text{INFINITY})$

- **log1p(x)** : ส่งคืนค่าลอการิทึมของ $x+1$ ฐานธรรมชาติ (e)

- **Domain:** $[-1, \text{INFINITY})$
- **Range:** $(-\text{INFINITY}, \text{INFINITY})$

- **log2(x)** : ส่งคืนค่าลอการิทึมของ x ฐาน 2

- **Domain:** $[0, \text{INFINITY})$
- **Range:** $(-\text{INFINITY}, \text{INFINITY})$

Examples:

```
log10(0.0)      # => -Infinity  
log10(1.0)      # => 0.0  
log10(10.0)     # => 1.0  
log10(Float::INFINITY) # => Infinity
```

Examples:

```
log1p(-1.0)      # => -Infinity  
log1p(0.0)      # => 0.0  
log1p(E - 1)     # => 1.0  
log1p(Float::INFINITY) # => Infinity
```

Examples:

```
log2(0.0)      # => -Infinity  
log2(1.0)      # => 0.0  
log2(2.0)      # => 1.0  
log2(Float::INFINITY) # => Infinity
```



Exponentiation & Logarithmic Functions in C

คำเตือน: โดย log สามารถใช้ได้เพียง log ฐานธรรมชาติเท่านั้น หากต้องการใช้ฐานอื่นจำเป็นต้องใช้สมบัติของ log $\log_b(a) = \log(a) / \log(b)$;

`exp(double x)`

```
exp(2.000000) // 7.389056
```

`log(double x)`

```
log(2.000000) // 0.693147
```

`log10(double x)`

```
log10(2.000000) // 0.301030
```

ตั้งแต่ C99 เป็นต้นไป สามารถใช้ expm1 log1p log2 ได้

expm1(double x)

log1p(double x)

log2(double x)

```
expm1(2.000000) // 6.389056
log1p(M_E - 1) // 1.000000
log2(2.000000) // 1.000000
```



Exponentiation & Logarithmic Functions in Java

Math.exp(double x)

Math.expm1(double x) `Math.exp(2.000000) // 7.38905609893065`

`Math.expm1(2.000000) // 6.38905609893065`

Math.log(double x) `Math.log(2.000000) // 0.6931471805599453`

`Math.log10(2.000000) // 0.3010299956639812`

Math.log1p(double x) `Math.log1p(Math.E - 1) // 1.0`



มหาวิทยาลัยมาหิดล
มหาวิทยาลัยมาหิดล

คำเตือน: โดย \log สามารถใช้ได้เพียง \log ฐานธรรมชาติเท่านั้น หากต้องการใช้ฐานอื่นจำเป็นต้องใช้สมบัติของ \log $\log_b(a) = \log(a) / \log(b)$;

คำเตือน: "ไม่มีคำสั่ง \log_2 ในภาษาเขียนโปรแกรม Java" ใช้สมบัติ \log ข้างต้นเขียนเมธอดแทน

Exponentiation & Logarithmic Functions in Python

math.exp(double x)

math.expm1(double x)

math.log(double x, int *base=math.e*)

math.log10(double x)

math.log1p(double x)

math.log2(double x)

```
math.exp(2.00000) # 7.38905609893065
math.expm1(2.00000) # 6.38905609893065
math.log(2.00000) # 0.6931471805599453
math.log10(2.00000) # 0.3010299956639812
math.log1p(math.e - 1) # 1.0
math.log2(2.00000) # 1.0
```



Fraction & Exponent Functions in Ruby

IEEE-Standard Floating Point

```
x = fraction * 2**exponent
```

fraction คือ เศษส่วน

exponent คือ เลขยกกำลัง



- **frexp(x)** : ส่งคืน array เศษส่วนและเลขชี้กำลังจาก x ที่กำหนด

Examples:

```
frexp(-Float::INFINITY) # => [-Infinity, -1]
frexp(-2.0)             # => [-0.5, 2]
frexp(-1.0)             # => [-0.5, 1]
frexp(0.0)              # => [0.0, 0]
frexp(1.0)              # => [0.5, 1]
frexp(2.0)              # => [0.5, 2]
frexp(Float::INFINITY)  # => [Infinity, -1]
```

- **ldexp(fraction, exponent)** : ส่งคืนค่าการคำนวณเศษส่วนและเลขชี้กำลังที่กำหนด

Examples:

```
ldexp(-Float::INFINITY, -1) # => -Infinity
ldexp(-0.5, 2)              # => -2.0
ldexp(-0.5, 1)              # => -1.0
ldexp(0.0, 0)               # => 0.0
ldexp(-0.5, 1)              # => 1.0
ldexp(-0.5, 2)              # => 2.0
ldexp(Float::INFINITY, -1)  # => Infinity
```

ถ้า **exponent** เป็นพจนิยม
ทำงานเหมือนจำนวนเต็ม

| | |
|------------------|-----------|
| ldexp(-0.5, 2.2) | # => -2.0 |
| ldexp(-0.5, 1.2) | # => -1.0 |
| ldexp(0.0, 0.3) | # => 0.0 |
| ldexp(-0.5, 1.2) | # => 1.0 |
| ldexp(-0.5, 2.3) | # => 2.0 |



Fraction & Exponent Functions in C

```
frexp( double x, int *exponent )
```

และ exponent ใน C ต้องระบุเป็น int

```
ldexp( double fraction, int exponent )
```

```
int i;  
double fraction = frexp(-1.0, &i); // fraction = -0.500000, i=1  
  
double ld = ldexp(-0.5,1); // ld = -1.000000
```



Fraction & Exponent Functions in Java

สามารถใช้ Idexp ฝ่านคำสั่งด้านล่าง

`Math.scalb(double d, int scaleFactor)` // เหมือนกับ Idexp ใน Ruby

ไม่มีคำสั่ง frexp ในภาษา Java ต้องใช้เขียนขึ้นมาเองความสัมพันธ์ทางด้านล่าง

```
int exponent = (int) Math.floor(Math.log(Math.abs(number)) / Math.log(2)) +1;
```

// exponent ของ frexp ในภาษาเขียนโปรแกรม Ruby

```
double fraction = x / Math.pow(2, exponent) // fraction ของ frexp ในภาษาเขียน  
โปรแกรม Ruby
```



```

class Main {

    public static double[] frexp(double number) {

        // ถ้า number เป็น NaN (Not a Number) จะคืนค่าเป็น NaN และเลขชี้กำลังเป็น 0
        if (Double.isNaN(number)) {
            return new double[]{Double.NaN, 0};
        }

        // ถ้า number เป็น 0.0 จะคืนค่าเป็น 0.0 และเลขชี้กำลังเป็น 0
        if (number == 0.0) {
            return new double[]{0.0, 0};
        }

        // ถ้า number เป็นอนันต์บวก (+Infinity) จะคืนค่าเป็น +Infinity และ
        // เลขชี้กำลังเป็น -1
        if (number == Double.POSITIVE_INFINITY) {
            return new double[]{Double.POSITIVE_INFINITY, -1};
        }

        // ถ้า number เป็นอนันต์ลบ (-Infinity) จะคืนค่าเป็น -Infinity และเลขชี้กำลัง
        // เป็น -1
        if (number == Double.NEGATIVE_INFINITY) {
            return new double[]{Double.NEGATIVE_INFINITY, -1};
        }

        int exponent = (int) Math.floor(Math.log(Math.abs(number)) /
Math.log(2)) +1;
        double fraction = number / Math.pow(2, exponent);
        return new double[]{fraction, exponent};
    }
}

```

```

public static void main(String[] args) {
    double number = -1.0;
    double[] result = frexp(number); // [-0.5, 1.0]
    System.out.println(result[0]); // -0.5
    System.out.println(result[1]); // 1.0
}

```

ผลลัพธ์:

-0.5

1.0

Fraction & Exponent Functions in Python

```
math.frexp(double x)
```

```
math.ldexp(double x, int i)
```

```
math.frexp(-1) # (-0.5, 1)  
math.ldexp(-0.5,1) # -1.0
```



Root Functions in Ruby

- **cbrt(x)** : ส่งคืนรากที่สามของ x
 - **Domain:** (-INFINITY, INFINITY)
 - **Range:** (-INFINITY, INFINITY)



- **sqrt(x)** : ส่งคืนรากที่สองของ x
 - **Domain:** (0, INFINITY)
 - **Range:** (0, INFINITY)



Examples:

```
cbrt(-Float::INFINITY) # => -Infinity  
cbrt(-27.0)           # => -3.0  
cbrt(-8.0)            # => -2.0  
cbrt(-2.0)            # => -1.2599210498948732  
cbrt(1.0)             # => 1.0  
cbrt(0.0)              # => 0.0  
cbrt(1.0)             # => 1.0  
cbrt(2.0)             # => 1.2599210498948732  
cbrt(8.0)              # => 2.0  
cbrt(27.0)             # => 3.0  
cbrt(Float::INFINITY) # => Infinity
```

Examples:

```
sqrt(0.0)             # => 0.0  
sqrt(0.5)             # => 0.7071067811865476  
sqrt(1.0)             # => 1.0  
sqrt(2.0)             # => 1.4142135623730951  
sqrt(4.0)             # => 2.0  
sqrt(9.0)             # => 3.0  
sqrt(Float::INFINITY) # => Infinity
```

Root Functions in C

`cbrt` ใช้ได้ตั้งแต่ C99 เป็นต้นไป

```
cbrt( double x )
```

```
cbrt(8.0) // 2.000000
```

```
sqrt( double x )
```

```
sqrt(4.0) // 2.000000
```



Root Functions in Java

Math.cbrt(double x)

```
Math.cbrt(8.0) // 2.0
```

Math.sqrt(double x)

```
Math.sqrt(4.0) // 2.0
```

Root Functions in Python

```
math.cbrt(double x)
```

```
math.cbrt(8.0) # 2.0
```

```
math.sqrt(double x)
```

```
math.sqrt(4.0) # 2.0
```



Error Functions in Ruby

- **erf(x)** : ส่งคืนค่าฟังก์ชัน Gauss error ของ x

- **Domain:** (-INFINITY, INFINITY)
- **Range:** [-1, 1]

Examples:

```
erf(-Float::INFINITY) # => -1.0  
erf(0.0)             # => 0.0  
erf(Float::INFINITY) # => 1.0
```

- **erfc(x)** : ส่งคืนค่าของฟังก์ชัน complementary error ของ x หรือคือ $1 - \text{erf}(x)$
Gauss error ของ x $\text{erfc}(x) = 1 - \text{erf}(x)$

- **Domain:** (-INFINITY, INFINITY)
- **Range:** [0, 2]

Examples:

```
erfc(-Float::INFINITY) # => 2.0  
erfc(0.0)              # => 1.0  
erfc(Float::INFINITY) # => 0.0
```

Error Functions in C

erf erfc ใช้ได้ตั้งแต่ C99 เป็นต้นไป

```
erf( double x )
```

```
erf(0.0) // 0.000000
```

```
erfc( double x )
```

```
erfc(0.0) // 1.000000
```



Error Functions in Java

มี erf erfc ในภาษา Java แต่จะต้อง `import org.apache.commons.math3.special.Erf;`

เพิ่มไลบรารี [Apache Commons Math](#) ↗ เข้าไปในโปรเจกต์ฝาน Maven หรือ Gradle

สำหรับ Maven: เพิ่มโค้ดนี้ลงในไฟล์ `pom.xml` ของคุณ

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-math3</artifactId>
    <version>3.6.1</version>
</dependency>
```

เมื่อเพิ่มไลบรารีเรียบร้อยแล้ว สามารถเรียกใช้เมธอด `erf()` และ `erfc()` ได้โดยตรง



```
import org.apache.commons.math3.special.Erf;

public class ErfExample {
    public static void main(String[] args) {
        double x = 1.0;

        // คำนวณ Error Function (erf)
        double erfValue = Erf.erf(x);
        System.out.println("erf(" + x + ") = " + erfValue);

        // คำนวณ Complementary Error Function (erfc)
        double erfcValue = Erf.erfc(x);
        System.out.println("erfc(" + x + ") = " + erfcValue);

        // erfc(x) = 1 - erf(x)
        System.out.println("1 - erf(" + x + ") = " + (1 - erfValue));
    }
}
```

ผลลัพธ์ของโค้ด:

```
erf(1.0) = 0.84270079294971497151
erfc(1.0) = 0.157299207050285028488
1 - erf(1.0) = 0.15729920705028488
```



Error Functions in Python

erf erfc ถูกเพิ่มเข้ามาใน Python version 3.2

```
math.erf( double x )
```

```
math.erf(0.0) # 0.0
```

```
math.erfc( double x )
```

```
math.erfc(0.0) # 1.0
```

Gamma Functions in Ruby

- **gamma(x)** : ส่งคืนค่าฟังก์ชัน gamma ของ x
 - **Domain:** (-INFINITY, INFINITY]
 - **Range:** [-INFINITY, INFINITY]
- **Igamma(x)** : ส่งคืนค่า array ค่าฟังก์ชันลอการิทึมของ abs(gamma(x)) และค่า -1 หรือ 1 ขึ้นอยู่กับ gamma(x) หากเป็นบวกคืนค่า 1 หากเป็นลบคืนค่า -1 ดังสมการข้างล่าง (Log-Gamma) [Math.log(Math.gamma(x).abs), Math.gamma(x) < 0 ? -1 : 1]
 - **Domain:** (-INFINITY, INFINITY]
 - **Range of first element:** (-INFINITY, INFINITY]
 - **Range of second element** เป็น -1 หรือ 1



Examples:

Examples:

```
gamma(-2.5)      # => -0.9453087204829431
gamma(-1.5)      # => 2.3632718012073513
gamma(-0.5)      # => -3.5449077018110375
gamma(0.0)        # => Infinity
gamma(1.0)        # => 1.0
gamma(2.0)        # => 1.0
gamma(3.0)        # => 2.0
gamma(4.0)        # => 6.0
gamma(5.0)        # => 24.0
```

```
lgamma(-4.0)    # => [Infinity, -1]
lgamma(-3.0)    # => [Infinity, -1]
lgamma(-2.0)    # => [Infinity, -1]
lgamma(-1.0)    # => [Infinity, -1]
lgamma(0.0)     # => [Infinity, 1]

lgamma(1.0)     # => [0.0, 1]
lgamma(2.0)     # => [0.0, 1]
lgamma(3.0)     # => [0.6931471805599436, 1]
lgamma(4.0)     # => [1.7917594692280545, 1]

lgamma(-2.5)   # => [-0.05624371649767279, -1]
lgamma(-1.5)   # => [0.8600470153764797, 1]
lgamma(-0.5)   # => [1.265512123484647, -1]
lgamma(0.5)    # => [0.5723649429247004, 1]
lgamma(1.5)    # => [-0.12078223763524676, 1]
lgamma(2.5)    # => [0.2846828704729205, 1]
```

Gamma Functions in C

`tgamma` `lgamma` ใช้ได้ตั้งแต่ C99 เป็นต้นไป

`tgamma(double x) //` เมื่อเทียบกับ `gamma(x)` ในภาษาโปรแกรม Ruby

```
tgamma(0.5) // 1.772454
```

`lgamma(double x) //` ถ้าเป็นทศนิยมติดลบจะคืนค่าติดลบโดยไม่แยกเป็น array เมื่อเทียบกับภาษาโปรแกรม Ruby

```
lgamma(0.5) // 0.572365
```

Gamma Functions in Java

มี gamma Igamma ในภาษา Java แต่จะต้อง import

```
org.apache.commons.math3.special.Gamma;
```

เพิ่มไลบรารี [Apache Commons Math](#) ↗ เข้าไปในโปรเจกต์ผ่าน Maven หรือ Gradle

สำหรับ Maven: เพิ่มโค้ดนี้ลงในไฟล์ `pom.xml` ของคุณ

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-math3</artifactId>
    <version>3.6.1</version>
</dependency>
```

เมื่อเพิ่มไลบรารีเรียบร้อยแล้ว สามารถเรียกใช้เมธอด `gamma()` และ `logGamma()` ได้โดยตรง



```
import org.apache.commons.math3.special.Gamma;

public class GammaExample {
    public static void main(String[] args) {
        double x = 5.0;

        // คำนวณ Gamma Function ( $\Gamma(x)$ )
        double gammaValue = Gamma.gamma(x);
        System.out.println("Gamma(" + x + ") = " + gammaValue);

        // คำนวณ Log-Gamma Function ( $\ln(\Gamma(x))$ )
        double logGammaValue = Gamma.logGamma(x);
        System.out.println("logGamma(" + x + ") = " + logGammaValue);

        // ตรวจสอบความถูกต้อง:  $e^{\ln(\Gamma(x))}$  ควรเท่ากับ  $\Gamma(x)$ 
        System.out.println("e^logGamma(" + x + ") = " +
Math.exp(logGammaValue));
    }
}
```

ผลลัพธ์ของโค้ด:

Gamma(5.0) = 24.0
logGamma(5.0) = 3.1780538303479458
e^logGamma(5.0) = 24.000000000000004

Gamma Functions in Python

tgamma lgamma ຖຸກເພີ່ມໄວໃນ Python version 3.2

```
math.gamma( double x )
```

```
math.gamma(0.5) # 1.7724538509055159
```

```
math.lgamma( double x )
```

```
math.lgamma(0.5) # 0.5723649429247004
```



Hypotenuse Functions in Ruby

- **hypot(x, y)** : ส่งคืนค่า $\sqrt{x^2 + y^2}$ สำหรับค่า x และ y
 - **Domain of x:** (-INFINITY, INFINITY)
 - **Domain of y:** (-INFINITY, INFINITY)
 - **Range:** [0, INFINITY)

Examples:

```
hypot(0.0, 1.0)          # => 1.0
hypot(1.0, 1.0)          # => 1.4142135623730951 # sqrt(2.0)
hypot(3.0, 4.0)          # => 5.0
hypot(5.0, 12.0)         # => 13.0
hypot(1.0, sqrt(3.0))   # => 1.9999999999999998 # Near 2.0
```



Hypotenuse Functions in C

```
hypot( double x, double y )
```

```
hypot(1,1) // 1.414214
```

Hypotenuse Functions in Java

```
Math.hypot( double x, double y)
```

```
Math.hypot(1, 1) // 1.4142135623730951
```



Hypotenuse Functions in Python

math.hypot(double x, double y)

```
math.hypot(1, 1) # 1.4142135623730951
```



DomainError in Ruby

Main::DomainError

เกิดขึ้นเมื่อฟังก์ชันทางคณิตศาสตร์ได้รับพารามิเตอร์นอกค่าจำกัดความทางคณิตศาสตร์

เช่น \cos จะส่งคืนค่าในช่วง $[-1, 1]$ ฟังก์ชันผกผัน \arccos จึงถูกกำหนดบนช่วงนั้นเท่านั้น:

`Math.acos(42)`

ผลลัพธ์ :

`Math::DomainError: อาร์กิวเมนต์ตัวเลขอยู่นอกโดเมน - "acos"`



DomainError in C

เมื่อ `#include <errno.h>`
สามารถใช้ `errno`
ดู `error` ต่างๆ ได้

`errno` จาก header `errno`
เปลี่ยนเป็น `EDOM`
เมื่อ `DomainError` เกิดขึ้น

```
#include <stdio.h>
#include <math.h>
#include <errno.h>

int main() {
    double invalid_value = 2.0;

    double result = asin(invalid_value);
    if (errno == EDOM) {
        printf("Arcsine of %f is not defined.\n", invalid_value);
    } else {
        printf("Arcsine of %f = %f radians\n", invalid_value, result);
    }

    return 0;
}
```

ผลลัพธ์: Arcsine of 2.000000 is not defined.

```
#include <stdio.h>
#include <errno.h>
#include <math.h>

int main () {
    double val;

    errno = 0;
    val = sqrt(-10);
    if(errno == EDOM) {
        printf("Invalid value \n");
    } else {
        printf("Valid value\n");
    }

    errno = 0;
    val = sqrt(10);
    if(errno == EDOM) {
        printf("Invalid value\n");
    } else {
        printf("Valid value\n");
    }

    return(0);
}
```

หากใช้ `errno` ต้อง `errno = 0` เพราะ
`errno` ไม่กำหนดค่ากลับอัตโนมัติ

ผลลัพธ์:

Invalid value
Valid value



DomainError in Java

ไม่ผิดพลาดการ แต่ค่าของฟังก์ชัน
จะ return NaN ต้องเช็ค error เอง
สามารถใช้ร่วมกับ
IllegalArgumentException
เพื่อแจ้งเตือน error ได้ด้วยตัวเอง

ผลลัพธ์:

```
public class Main{  
    public static void main(String[] args) {  
        double result = Math.sqrt(-1.0);  
        System.out.println(result); // Output: NaN  
        if (Double.isNaN(result)) {  
            throw new IllegalArgumentException("Input must be >= 0.0");  
        }  
    }  
}
```

Nan
ERROR!
Exception in thread "main" java.lang.IllegalArgumentException: Input
must be >= 0.0
at Erf.main(Main.java:6)

DomainError in Python



ใช้ ValueError เพื่อใส่ค่านอกโดเมน
จะเจ้ง math domain error

```
import math  
math.sqrt(-10)
```

ผลลัพธ์:

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ValueError: math domain error
```

Quiz

จงเขียนโปรแกรมการหาค่าของ sin cos tan จาก Input ของผู้ใช้ช่วงค่า [0,1) คุณเก็บค่าพายในภาษา Ruby C Java Python

Ruby

```
puts "Input in [0,1): "
n = Float(gets.chomp)
if(n>=0 && n<1)
    angle = n * Math::PI
    puts "Angle: #{angle} radians"
    puts Math.sin(angle)
    puts Math.cos(angle)
    puts Math.tan(angle)
else
    puts "Input not in [0,1)"
end
```

C

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>

int main() {
    double n, angle, sin_val, cos_val, tan_val;

    printf("Input in [0,1): \n");
    if (scanf("%lf", &n) != 1) {
        printf("\n*** ERROR: Invalid input. Enter a number. ***\n");
        return 1;
    }

    if (n>=0 && n<1) {
        angle = n * M_PI;
        printf("Angle: %f radians\n", angle);
        printf("%f\n",sin(angle));
        printf("%f\n",cos(angle));
        printf("%f\n",tan(angle));
    }else{
        printf("Input not in [0,1)");
    }
    return 0;
}
```

Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Input in [0,1]: ");
        double n = sc.nextDouble();
        if(n>=0 && n<1){
            double angle = n * Math.PI;
            System.out.printf("Angle: %f radians\n",angle);
            System.out.println(Math.sin(angle));
            System.out.println(Math.cos(angle));
            System.out.println(Math.tan(angle));
        }else{
            System.out.println("Input not in [0,1]");
        }
    }
}
```

Python

```
import math

n = float(input("Input in [0,1]: "))
if(n>=0 and n<1):
    angle = n * math.pi
    print(f"\nAngle: {angle} radians")
    print(math.sin(angle))
    print(math.cos(angle))
    print(math.tan(angle))
else:
    print(f"\nInput not in [0,1]")
```

Reference

การเพิ่มโมดูลคณิตศาสตร์ในภาษาต่าง ๆ

Ruby

Docs.ruby-lang. (n.d.). *module Math*. Retrieved from
<https://docs.ruby-lang.org/en/master/Math.html>

C

Cppreference. (n.d.). *Standard library header <math.h>*. Retrieved from
<https://en.cppreference.com/w/c/header/math.html>

Java

Oracle. (2025, July 14). *Class Math*. Retrieved from

[https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/
lang/Math.html](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html)

Python

Python Software Foundation. (2025, September 2).

math — Mathematical Functions. Retrieved from

<https://docs.python.org/3/library/math.html>

ค่าคงที่ทางคณิตศาสตร์

Ruby

Flanagan, D., & Matsumoto, Y. (2008, P.322-323).

The Ruby Programming Language. O'Reilly.

Techotopia. (2016, October 27). *Ruby Math Functions and Methods.*

Retrieved from

https://www.techotopia.com/index.php/Ruby_Math_Functions_and_Methods

GeeksforGeeks. (2022, September 13). *Ruby / Math Module.* Retrieved from

<https://www.geeksforgeeks.org/ruby/ruby-math-module/>

Ruby-doc. (n.d.). *Math (Ruby 2.2.0 Core API).* Retrieved from

<https://ruby-doc.org/core-2.2.0/Math.html>

C

Learn.microsoft.com. (2023, January 2). *Math constants*. Retrieved from
[https://learn.microsoft.com/en-us/cpp/c-runtime-library/math-constants?
view=msvc-170](https://learn.microsoft.com/en-us/cpp/c-runtime-library/math-constants?view=msvc-170)

Ashwin. (2015, October 26). *Math constants for C and C++*. Retrieved from
<https://gist.github.com/ashwin/3e2b6e1881d3e0ddb5dc>

Java

Oracle. (2025, July 14). *Class Math*. Retrieved from

[https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/
lang/Math.html](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html)

Byron Kiourtzoglou. (2012, November 11). *Using Math Constants*.

JavaCodeGeeks. Retrieved from

[https://examples.javacodegeeks.com/java-development/core-java/
math/using-math-constants/](https://examples.javacodegeeks.com/java-development/core-java/math/using-math-constants/)

GeeksforGeeks. (2025, July 23). *Import Statement in Java*. Retrieved from
<https://www.geeksforgeeks.org/java/import-statement-in-java/>

Oracle .(2015, February 13). *Packages*. Retrieved from
<https://docs.oracle.com/javase/specs/jls/se8/html/jls-7.html#jls-7.3>

Python

GeeksforGeeks. (2025, July 12). *Constants of Maths in Python*. Retrieved from
<https://www.geeksforgeeks.org/python/constants-of-maths-in-python/>

Python Software Foundation (2025, September 3).
math — Mathematical Functions. Retrieved from
<https://docs.python.org/3/library/math.html>

ຝຶກໍ່ຊັ້ນທາງຄນິຕະລາສຕົວ

Ruby

Docs.ruby-lang. (n.d.). *module Math*. Retrieved from

<https://docs.ruby-lang.org/en/master/Math.html>

Techotopia. (2016, October 27). *Ruby Math Functions and Methods*.

Retrieved from

https://www.techotopia.com/index.php/Ruby_Math_Functions_and_Methods

GeeksforGeeks. (2022, September 13). *Ruby | Math Module*. Retrieved from

<https://www.geeksforgeeks.org/ruby/ruby-math-module/>

Ruby-doc. (n.d.). *Math (Ruby 2.2.0 Core API)*. Retrieved from

<https://ruby-doc.org/core-2.2.0/Math.html>

C

TutorialsPoint. (n.d.). *C - Math Functions*. Retrieved from

https://www.tutorialspoint.com/cprogramming/c_math_functions.htm

Cppreference. (n.d.). *Standard library header <math.h>*. Retrieved from

<https://en.cppreference.com/w/c/header/math.html>

GeeksforGeeks. (2023, April 3). *C Library math.h Functions*. Retrieved from

<https://www.geeksforgeeks.org/c-c-library-math-h-functions/>

Java

GeeksforGeeks. (2025, July 23). *Import Statement in Java*. Retrieved from
<https://www.geeksforgeeks.org/java/import-statement-in-java/>

Oracle .(2015, February 13). *Packages*. Retrieved from
<https://docs.oracle.com/javase/specs/jls/se8/html/jls-7.html#jls-7.3>

Oracle. (2025, July 14). *Class Math*. Retrieved from
<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html>

W3schools. (n.d.). *Java Math Methods*. Retrieved from
https://www.w3schools.com/java/java_ref_math.asp

GeeksforGeeks. (2025, July 23). *Java Math Class*. Retrieved from
<http://geeksforgeeks.org/java/java-math-class/>

Geekster. (n.d.). *Math Class In Java*. Retrieved from
<https://www.geekster.in/articles/math-class-in-java/>

Weisstein, E. (2025, August 28). *Inverse Hyperbolic Functions*.
Wolfram. Retrieved from
<https://mathworld.wolfram.com/InverseHyperbolicFunctions.html>

Maven Central. (2014). *commons-math3*. Retrieved from
<https://central.sonatype.com/artifact/org.apache.commons/commons-math3?smo=true>

Oracle. (n.d.). *Class Gamma*. Retrieved from
<https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/special/Gamma.html>

Oracle. (n.d.). *Class Erf*. Retrieved from
<https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/special/Erf.html>

Python

W3schools. (n.d.). *Python math Module*. Retrieved from
https://www.w3schools.com/python/module_math.asp

Python Software Foundation (2025, September 3).
math — Mathematical Functions. Retrieved from
<https://docs.python.org/3/library/math.html>

ข้อผิดพลาดของໂດເນັ້ນ

Ruby

Docs.ruby-lang. (n.d.). *class Math::DomainError*. Retrieved from

<https://docs.ruby-lang.org/en/master/Math/DomainError.html>

GitHub. (2025, June 3). *Exception: Math::DomainError*. Retrieved from

https://msp-greg.github.io/ruby_3_1/Core/Math/DomainError.html

C

Cppreference. (n.d.). *Standard library header <errno.h>*. Retrieved from
<https://en.cppreference.com/w/c/header/errno.html>

Cppreference. (n.d.). *sqrt, sqrtf, sqrtl*. Retrieved from
<https://en.cppreference.com/w/c/numeric/math/sqrt.html>

Cppreference. (n.d.). *errno*. Retrieved from
<https://en.cppreference.com/w/c/error/errno.html>

TutorialsPoint. (n.d.). *C library - EDOM Macro*. TutorialsPoint. Retrieved from
https://www.tutorialspoint.com/c_standard_library/c_macro_edom.htm

Java

Oracle. (2025, July 14). *Class Math*. Retrieved from

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/Math.html>

Oracle. (2025, July 14). *Class IllegalArgumentException*. Retrieved from

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/IllegalArgumentException.html>

Python

Pankaj Kumar.(2022, August 4). *Python ValueError Exception Handling Examples.* DigitalOcean. Retrieved from

<https://www.digitalocean.com/community/tutorials/python-valueerror-exception-handling-examples>

Ramos, L. P. (2025, March 3). *ValueError*. RealPython. Retrieved from

<https://realpython.com/ref/builtin-exceptions/valueerror/>

Python Software Foundation (2025, September 3). *Built-in Exceptions.*

Retrieved from

<https://docs.python.org/3/library/exceptions.html#ValueError>