Learn Python Programming from Scratch

Topic: Math Functions in Python

1. What are Math Functions?

Math functions are pre-built functions that perform mathematical operations and calculations. Python provides both built-in math functions and an extensive `math` module for more advanced mathematical operations.

Math functions help you:

- Perform complex calculations without writing the logic yourself
- Work with mathematical constants (π, e)
- Handle floating-point precision issues
- Implement scientific and engineering calculations
- Process numerical data efficiently

2. Categories of Math Functions

Built-in Functions (always available):

- `abs()`, `round()`, `pow()`, `min()`, `max()`, `sum()`
- No import required - ready to use immediately

Math Module Functions (require import):

- `sqrt()`, `ceil()`, `floor()`, `sin()`, `cos()`, `log()`, `factorial()`
- Advanced mathematical and trigonometric functions
- Mathematical constants like `π` and `e`

Random Module Functions (for random numbers):

- `random()`, `randint()`, `choice()`, `shuffle()`
- Generate random numbers and make random selections

3. Why Use Math Functions?

Math functions provide:

- **Accuracy**: Tested and optimized implementations
- **Efficiency**: Faster than writing your own calculations
- **Convenience**: Complex operations in single function calls
- **Reliability**: Handle edge cases and special values properly
- **Standards**: Follow mathematical conventions and standards

4. Built-in Math Functions

In [1]:
```
# Built-in Math Functions (No Import Required)

print("=== BASIC NUMERIC FUNCTIONS ===")

# abs() - Absolute value (distance from zero)
print("Absolute Value Examples:")
print(f"abs(-10) = {abs(-10)}")
print(f"abs(10) = {abs(10)}")
print(f"abs(-3.14) = {abs(-3.14)}")
print(f"abs(0) = {abs(0)}")

# round() - Round to nearest integer or specified decimal places
print(f"\nRounding Examples:")
print(f"round(4.6) = {round(4.6)}")           # Round to nearest integer
print(f"round(4.3) = {round(4.3)}")
print(f"round(4.5) = {round(4.5)}")           # Note: rounds to even
print(f"round(3.14159, 2) = {round(3.14159, 2)}")  # Round to 2 decimal places
print(f"round(123.456, 1) = {round(123.456, 1)}")

# pow() - Power function (base ** exponent)
print(f"\nPower Examples:")
print(f"pow(2, 3) = {pow(2, 3)}")             # 2³ = 8
print(f"pow(5, 2) = {pow(5, 2)}")             # 5² = 25
print(f"pow(3, 0) = {pow(3, 0)}")             # Any number to power 0 = 1
print(f"pow(2, -1) = {pow(2, -1)}")           # Negative exponent

print("\n" + "="*50)

# min() and max() - Find minimum and maximum values
print("\n=== MIN/MAX FUNCTIONS ===")

numbers = [45, 23, 78, 12, 89, 34]
print(f"Numbers: {numbers}")
print(f"min(numbers) = {min(numbers)}")
print(f"max(numbers) = {max(numbers)}")

# Can also work with multiple arguments
print(f"min(10, 5, 8, 3) = {min(10, 5, 8, 3)}")
print(f"max(10, 5, 8, 3) = {max(10, 5, 8, 3)}")

# With strings (alphabetical order)
names = ["Alice", "Bob", "Charlie", "David"]
print(f"Names: {names}")
print(f"min(names) = {min(names)}")           # First alphabetically
print(f"max(names) = {max(names)}")           # Last alphabetically

# sum() - Calculate sum of all elements in an iterable
print(f"\nsum(numbers) = {sum(numbers)}")
print(f"sum([1, 2, 3, 4, 5]) = {sum([1, 2, 3, 4, 5])}")

# sum() with start value
print(f"sum([1, 2, 3], 10) = {sum([1, 2, 3], 10)}")  # Start from 10

print("\n=== PRACTICAL EXAMPLES ===")

# Grade calculation
```

```
grades = [85, 92, 78, 96, 88]
total_points = sum(grades)
average_grade = total_points / len(grades)
highest_grade = max(grades)
lowest_grade = min(grades)

print(f"Student Grades: {grades}")
print(f"Total Points: {total_points}")
print(f"Average Grade: {round(average_grade, 1)}")
print(f"Highest Grade: {highest_grade}")
print(f"Lowest Grade: {lowest_grade}")

# Price calculation with absolute difference
original_price = 99.99
sale_price = 79.99
savings = abs(original_price - sale_price)
discount_percent = round((savings / original_price) * 100, 1)

print(f"\nPrice Comparison:")
print(f"Original Price: ${original_price}")
print(f"Sale Price: ${sale_price}")
print(f"You Save: ${savings}")
print(f"Discount: {discount_percent}%")
```

```
=== BASIC NUMERIC FUNCTIONS ===
Absolute Value Examples:
abs(-10) = 10
abs(10) = 10
abs(-3.14) = 3.14
abs(0) = 0

Rounding Examples:
round(4.6) = 5
round(4.3) = 4
round(4.5) = 4
round(3.14159, 2) = 3.14
round(123.456, 1) = 123.5

Power Examples:
pow(2, 3) = 8
pow(5, 2) = 25
pow(3, 0) = 1
pow(2, -1) = 0.5

================================================

=== MIN/MAX FUNCTIONS ===
Numbers: [45, 23, 78, 12, 89, 34]
min(numbers) = 12
max(numbers) = 89
min(10, 5, 8, 3) = 3
max(10, 5, 8, 3) = 10
Names: ['Alice', 'Bob', 'Charlie', 'David']
min(names) = Alice
max(names) = David

sum(numbers) = 281
sum([1, 2, 3, 4, 5]) = 15
sum([1, 2, 3], 10) = 16

=== PRACTICAL EXAMPLES ===
Student Grades: [85, 92, 78, 96, 88]
Total Points: 439
Average Grade: 87.8
Highest Grade: 96
Lowest Grade: 78

Price Comparison:
Original Price: $99.99
Sale Price: $79.99
You Save: $20.0
Discount: 20.0%
```

5. The Math Module

The `math` module provides access to advanced mathematical functions and constants. You need to import it first with `import math`.

In [2]:
```python
# The Math Module - Advanced Mathematical Functions

import math
```

```
print("=== MATHEMATICAL CONSTANTS ===")

# Important mathematical constants
print(f"Pi (π) = {math.pi}")
print(f"Euler's number (e) = {math.e}")
print(f"Infinity = {math.inf}")
print(f"Not a Number = {math.nan}")
print(f"Tau (2π) = {math.tau}")

print("\n=== POWER AND LOGARITHMIC FUNCTIONS ===")

# Square root
print("Square Root Examples:")
print(f"math.sqrt(16) = {math.sqrt(16)}")
print(f"math.sqrt(2) = {math.sqrt(2):.4f}")
print(f"math.sqrt(100) = {math.sqrt(100)}")

# Power functions
print(f"\nPower Functions:")
print(f"math.pow(2, 3) = {math.pow(2, 3)}")
print(f"math.exp(1) = {math.exp(1):.4f}")      # e^1
print(f"math.exp(2) = {math.exp(2):.4f}")      # e^2

# Logarithmic functions
print(f"\nLogarithmic Functions:")
print(f"math.log(math.e) = {math.log(math.e):.4f}")        # Natural log
print(f"math.log10(100) = {math.log10(100)}")              # Base 10 log
print(f"math.log2(8) = {math.log2(8)}")                    # Base 2 log
print(f"math.log(8, 2) = {math.log(8, 2)}")                # Custom base

print("\n=== ROUNDING AND CEILING FUNCTIONS ===")

test_numbers = [4.2, 4.8, -3.2, -3.8, 0.1, 0.9]

print("Number    ceil()    floor()   trunc()")
print("-" * 38)
for num in test_numbers:
    ceil_val = math.ceil(num)
    floor_val = math.floor(num)
    trunc_val = math.trunc(num)
    print(f"{num:6.1f}    {ceil_val:6d}    {floor_val:7d}   {trunc_val:6d}")

print(f"\nExplanation:")
print(f"ceil() - rounds UP to next integer")
print(f"floor() - rounds DOWN to previous integer")
print(f"trunc() - removes decimal part (towards zero)")

print("\n=== TRIGONOMETRIC FUNCTIONS ===")

# Angles in radians (π radians = 180 degrees)
angles_radians = [0, math.pi/6, math.pi/4, math.pi/3, math.pi/2, math.pi]
print("Angle (rad)  Angle (deg)    sin       cos       tan")
print("-" * 55)

for angle in angles_radians:
    degrees = math.degrees(angle)
    sin_val = math.sin(angle)
    cos_val = math.cos(angle)
    tan_val = math.tan(angle) if angle != math.pi/2 else float('inf')
```

```
    print(f"{angle:8.4f}    {degrees:8.1f}°   {sin_val:8.4f} {cos_val:8.4f} {tan_v

# Convert between degrees and radians
print(f"\nAngle Conversion:")
print(f"90 degrees = {math.radians(90):.4f} radians")
print(f"π radians = {math.degrees(math.pi):.1f} degrees")

print("\n=== FACTORIAL AND COMBINATORICS ===")

# Factorial function
print("Factorial Examples:")
for i in range(6):
    print(f"{i}! = {math.factorial(i)}")

# Combinations and permutations (Python 3.8+)
try:
    print(f"\nCombinatorics:")
    print(f"5 choose 2 = {math.comb(5, 2)}")        # Combinations
    print(f"5 permute 2 = {math.perm(5, 2)}")       # Permutations
except AttributeError:
    print(f"\nCombinatorics functions require Python 3.8+")

print("\n=== PRACTICAL APPLICATIONS ===")

# Distance calculation using Pythagorean theorem
def calculate_distance(x1, y1, x2, y2):
    """Calculate distance between two points."""
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

# Circle calculations
def circle_area(radius):
    """Calculate area of a circle."""
    return math.pi * radius**2

def circle_circumference(radius):
    """Calculate circumference of a circle."""
    return 2 * math.pi * radius

# Example calculations
point1 = (0, 0)
point2 = (3, 4)
distance = calculate_distance(*point1, *point2)

radius = 5
area = circle_area(radius)
circumference = circle_circumference(radius)

print(f"Distance between {point1} and {point2}: {distance:.2f}")
print(f"Circle with radius {radius}:")
print(f"  Area: {area:.2f}")
print(f"  Circumference: {circumference:.2f}")
```

```
=== MATHEMATICAL CONSTANTS ===
Pi (π) = 3.141592653589793
Euler's number (e) = 2.718281828459045
Infinity = inf
Not a Number = nan
Tau (2π) = 6.283185307179586

=== POWER AND LOGARITHMIC FUNCTIONS ===
Square Root Examples:
math.sqrt(16) = 4.0
math.sqrt(2) = 1.4142
math.sqrt(100) = 10.0

Power Functions:
math.pow(2, 3) = 8.0
math.exp(1) = 2.7183
math.exp(2) = 7.3891

Logarithmic Functions:
math.log(math.e) = 1.0000
math.log10(100) = 2.0
math.log2(8) = 3.0
math.log(8, 2) = 3.0

=== ROUNDING AND CEILING FUNCTIONS ===
Number    ceil()    floor()   trunc()
---------------------------------------
   4.2         5          4        4
   4.8         5          4        4
  -3.2        -3         -4       -3
  -3.8        -3         -4       -3
   0.1         1          0        0
   0.9         1          0        0

Explanation:
ceil() - rounds UP to next integer
floor() - rounds DOWN to previous integer
trunc() - removes decimal part (towards zero)

=== TRIGONOMETRIC FUNCTIONS ===
Angle (rad)  Angle (deg)    sin       cos       tan
-----------------------------------------------------
  0.0000         0.0°     0.0000   1.0000   0.0000
  0.5236        30.0°     0.5000   0.8660   0.5774
  0.7854        45.0°     0.7071   0.7071   1.0000
  1.0472        60.0°     0.8660   0.5000   1.7321
  1.5708        90.0°     1.0000   0.0000      inf
  3.1416       180.0°     0.0000  -1.0000  -0.0000

Angle Conversion:
90 degrees = 1.5708 radians
π radians = 180.0 degrees

=== FACTORIAL AND COMBINATORICS ===
Factorial Examples:
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
```

```
5! = 120

Combinatorics:
5 choose 2 = 10
5 permute 2 = 20

=== PRACTICAL APPLICATIONS ===
Distance between (0, 0) and (3, 4): 5.00
Circle with radius 5:
  Area: 78.54
  Circumference: 31.42
```

Key Takeaways

- **Built-in functions** (`abs()` , `round()` , `pow()` , `min()` , `max()` , `sum()`) are always available
- **Math module** provides advanced functions - requires `import math`
- **Mathematical constants** like `π` and `e` are available via `math.pi` and `math.e`
- **Trigonometric functions** work with radians (use `math.radians()` to convert from degrees)
- **Logarithmic functions** handle different bases: `log()` , `log10()` , `log2()`
- **Rounding functions**: `ceil()` (up), `floor()` (down), `trunc()` (toward zero)
- **Error handling**: Be careful with domain errors (e.g., `sqrt(-1)` , `log(0)`)

Common Math Operations

Operation	Built-in	Math Module	Example
Absolute value	`abs(x)`	-	`abs(-5) = 5`
Power	`pow(x,y)`	`math.pow(x,y)`	`pow(2,3) = 8`
Square root	-	`math.sqrt(x)`	`math.sqrt(16) = 4`
Rounding	`round(x)`	`math.ceil(x)` , `math.floor(x)`	`round(4.6) = 5`
Min/Max	`min()` , `max()`	-	`max(1,2,3) = 3`
Logarithm	-	`math.log(x)` , `math.log10(x)`	`math.log10(100) = 2`

Practice Exercises

Try these exercises to strengthen your understanding:

1. **Calculator Functions**: Create functions for area/volume calculations using math constants
2. **Grade Statistics**: Use min/max/sum to analyze student grades

3. **Distance Calculator**: Implement distance formula using `math.sqrt()`
4. **Angle Converter**: Build degree-to-radian converter with trigonometric examples
5. **Scientific Calculator**: Combine multiple math functions for complex calculations

Real-World Applications

- **Engineering**: Trigonometry for angles, logarithms for decibel calculations
- **Finance**: Compound interest using `pow()` and `exp()`
- **Graphics**: Coordinate transformations using trigonometric functions
- **Statistics**: Standard deviation, variance calculations
- **Physics**: Motion calculations, wave functions
- **Data Science**: Normalization, scaling, statistical analysis

---

Course Information

Learn Python Programming from Scratch
Author: Prakash Ukhalkar
Topic: Python Fundamentals - Mathematical Functions

Built with ❤️ for the Python learning community