备选 example

Find the number of functions of the form $f(x) = ax^2 + bx + c$ such that\n\[f(x) f(-x) = f(x^2).\]

```
def solution():
    # Define the variables
    x, a, b, c = sp.symbols('x a b c')

    # Define the function f(x)
    f_x = a*x**2 + b*x + c

    # Define f(-x)
    f_minus_x = a*x**2 - b*x + c

    # f(x) * f(-x)
    product_f = f_x * f_minus_x

    # f(x^2)
    f_x_squared = a*x**4 + b*x**2 + c

    # Equation f(x) * f(-x) = f(x^2)
    equation = sp.Eq(product_f, f_x_squared)

    # Solve the equation for a, b, c
    solutions = sp.solve(equation, (a, b, c))

    # There is only one solution if a, b, c are constants, so:
    num_solutions = len(solutions)
    return num_solutions
```

Question:When $1 + 7 + 7^2 + \cdots + 7^{2004}$ is divided by $1000$, a remainder of $N$ is obtained. Determine the value of $N$.\n

```
def solution():
    # Define the sum as a geometric series: 1 + 7 + 7^2 + ... + 7^2004
    # The sum of a geometric series is S_n = a * (1 - r^n) / (1 - r),
where a is the first term and r is the common ratio.
    a = 1  # First term
    r = 7  # Common ratio
    n = 2005  # Total number of terms (from 7^0 to 7^2004)

    # The sum of the series S_n = (1 - 7^2005) / (1 - 7)
    sum_series = (1 - r**n) // (1 - r)

    # We need to find the remainder when this sum is divided by 1000
    remainder = sum_series % 1000

    return remainder
```

Question:In a certain hyperbola, the center is at $(2,0),$ one focus is at $(2,6),$ and one vertex is at $(2,-3).$ The equation of this hyperbola can be written as\n\[\frac{(y - k)^2}{a^2} - \frac{(x - h)^2}{b^2} = 1.\]Find $h + k + a + b.$"

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Calculate the sum h + k + a + b
    result = h + k + a + b

    return result
```

获取解决方案

You are a helpful python programmer. You will write python program to solve math problems. You will only write code blocks. Let's use python to solve math problems,y ou can write plan in the code notes and solve the problem in the code content. Here are three examples how to do it. Question:The Smith family has 4 sons and 3 daughters. In how many ways can they be seated in a row of 7 chairs such that at least 2 boys are next to each other?

```
import math
def solution():
    # Total number of ways to arrange 7 people without restriction
    total_ways = math.factorial(7)

    # Number of ways where no two boys are next to each other:
    # 1. Arrange 3 daughters in a row (3! ways)
    # 2. Place the boys in the gaps between the daughters (4 gaps) in a
way such that no two boys are next to each other
    daughter_arrangements = math.factorial(3)
    boy_positions = math.comb(4, 4) * math.factorial(4)  # Place all 4
boys in the 4 gaps
```

```
    no_boys_next_to_each_other_ways = daughter_arrangements *
boy_positions

    # The number of ways where at least 2 boys are next to each other
    result = total_ways - no_boys_next_to_each_other_ways
    return result
```

Question:The sphere with radius 1 and center $(0,0,1)$ rests on the $xy$-plane. A light source is at $P = (0,-1,2).$ Then the boundary of the shadow of the sphere can be expressed in the form $y = f(x),$ for some function $f(x).$ Find the function $f(x).$

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Equation for the boundary of the shadow
    lhs = y + 3
    rhs = sp.sqrt(x**2 + (y + 1)**2 + 4)

    # Solve the equation y + 3 = sqrt(x^2 + (y + 1)^2 + 4)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

Question:The fifth term of a geometric sequence of positive numbers is $11$ and the eleventh term is $5$. What is the eighth term of the sequence? Express your answer in simplest radical form. [asy]\nsize(150); defaultpen(linewidth(2));\nreal loc = 0;\nfor(int i = 0; i < 11; ++i) {\n\nif(i == 4)\n\nlabel("$\mathbf{\mathit{11}}$",(loc,0),(0.8,1.2),fontsize(14));\n\nif(i == 10)\n\nlabel("$\mathbf{\mathit{5}}$",(loc,0),(1.2,1.2),fontsize(14));\n\nfill(box((loc,0),(loc+1,0.15)));\n\nloc += 4/3;\n}\n[/asy]

```
import math
def solution()
    # Given values
    a_5 = 11  # fifth term
    a_11 = 5  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5
    r_squared = r_squared  # r^6 = 5/11

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 11 * r^3
    result = a_5 * r_cubed
    return result
```

How about this question? Question:

反写问题

You are a Mathematics Expert specializing in question reconstruction.

Your goal is to reconstruct the original mathematical question based on a corresponding Python program solution. The reconstructed question should restore the problem's background as much as possible and retain the difficulty involved in obtaining the initial variable assignments.

Follow the method below to complete the #Question 4# section:

1. Analyze the #Given Program 4# thoroughly, focusing on the relationships between variables and their actual values.
2. Reconstruct the problem background and context based on the information inferred from the program.
3. Ensure that the complexity and difficulty of initial variable assignments are preserved in the reconstructed question.
4. Ensure that the reconstructed question can be solved using the provided program.
5. DO NOT include ANY part of the solution process, including formulas, equations, or calculation methods in the #Question 4#.
6. DO NOT provide instructions on how to perform calculations or mention any mathematical operations explicitly in the #Question 4#.
7. Provide a clear and concise mathematical question that aligns with the functionality and variable relationships of the program. Here are three examples of how to do it. #Program 1#:

```
import math
def solution():
    # Total number of ways to arrange 7 people without restriction
    total_ways = math.factorial(7)

    # Number of ways where no two boys are next to each other:
    # 1. Arrange 3 daughters in a row (3! ways)
    # 2. Place the boys in the gaps between the daughters (4 gaps) in a
way such that no two boys are next to each other
    daughter_arrangements = math.factorial(3)
    boy_positions = math.comb(4, 4) * math.factorial(4)  # Place all 4
boys in the 4 gaps
    no_boys_next_to_each_other_ways = daughter_arrangements *
boy_positions

    # The number of ways where at least 2 boys are next to each other
    result = total_ways - no_boys_next_to_each_other_ways
    return result
```

#Question 1#: The Smith family has 4 sons and 3 daughters. In how many ways can they be seated in a row of 7 chairs such that at least 2 boys are next to each other?

#Program 2#:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Equation for the boundary of the shadow
    lhs = y + 3
    rhs = sp.sqrt(x**2 + (y + 1)**2 + 4)

    # Solve the equation y + 3 = sqrt(x^2 + (y + 1)^2 + 4)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

#Question 2#: The sphere with radius 1 and center $(0,0,1)$ rests on the $xy$-plane. A light source is at $P = (0,-1,2).$ Then the boundary of the shadow of the sphere can be expressed in the form $y = f(x),$ for some function $f(x).$ Find the function $f(x).$

#Program 3#:

```
import math
def solution()
    # Given values
    a_5 = 11  # fifth term
    a_11 = 5  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5
    r_squared = r_squared  # r^6 = 5/11

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 11 * r^3
    result = a_5 * r_cubed

    return result
```

#Question 3#: The fifth term of a geometric sequence of positive numbers is $11$ and the eleventh term is $5$. What is the eighth term of the sequence?

How about this program? #Program 4#:

```
<Here is the program.>
```

#Question 4#:

数值增强

You are a Programming Expert in the field of rewriting python program. You target is to rewrite one python program to make it evolve into a bit more difficult one for AI systems to handle. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should fill the #Rewritten Program# part using following method: You should rewrite the #Given Program# and evolve it by enhancing its initial numerical values. The enhanced values should make sure the generated result is difficult to guess. You should make sure the enhanced values are consistent with common sense and logically reasonable. Try your best to make sure the ehanced program aims to solve one definite question. You are NOT allowed to add new variable. If you cannot enhance the numerical values or the enhanced value cannot form one question, just output ERROR. You should only write code blocks. #Given Program#:

```
<Here is the program.>
```

#Rewritten Program#:

In-context版本

You are a Programming Expert in the field of rewriting python program. You target is to rewrite one python program to make it evolve into a bit more difficult one for AI systems to handle. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should fill the #Rewritten Program# part using following method: You should rewrite the #Given Program# and evolve it by enhancing its initial numerical values. The enhanced values should make sure the generated result is difficult to guess. You should make sure the enhanced values are consistent with common sense and logically reasonable. Try your best to make sure the ehanced program aims to solve one definite question. You are NOT allowed to add new variable. If you cannot enhance the numerical values or the enhanced value cannot form one question, just output ERROR. You should only write code blocks. Here are three examples how to do it. #Given Program#:

```
import math
def solution():
    # Total number of ways to arrange 7 people without restriction
    total_ways = math.factorial(7)

    # Number of ways where no two boys are next to each other:
    # 1. Arrange 3 daughters in a row (3! ways)
    # 2. Place the boys in the gaps between the daughters (4 gaps) in a
way such that no two boys are next to each other
```

```
    daughter_arrangements = math.factorial(3)
    boy_positions = math.comb(4, 4) * math.factorial(4)  # Place all 4
boys in the 4 gaps
    no_boys_next_to_each_other_ways = daughter_arrangements *
boy_positions

    # The number of ways where at least 2 boys are next to each other
    result = total_ways - no_boys_next_to_each_other_ways
    return result
```

#Rewritten Program#:

```
import math
def solution():
    # Total number of ways to arrange 10 people without restriction
    total_ways = math.factorial(10)

    # Number of ways where no two men are next to each other:
    # 1. Arrange 5 women in a row (5! ways)
    # 2. Place the men in the gaps between the women (6 gaps) in a way
such that no two men are next to each other
    women_arrangements = math.factorial(5)
    men_positions = math.comb(6, 5) * math.factorial(5)  # Place all 5 men
in the 5 of the 6 gaps
    no_men_next_to_each_other_ways = women_arrangements * men_positions

    # The number of ways where at least 2 men are next to each other
    result = total_ways - no_men_next_to_each_other_ways
    return result
```

#Given Program#:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Equation for the boundary of the shadow
    lhs = y + 3
    rhs = sp.sqrt(x**2 + (y + 1)**2 + 4)

    # Solve the equation y + 3 = sqrt(x^2 + (y + 1)^2 + 4)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

#Rewritten Program#:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Enhanced equation for the boundary of the shadow
    lhs = y + 10
    rhs = sp.sqrt(x**2 + (y + 5)**2 + 25)

    # Solve the equation y + 10 = sqrt(x^2 + (y + 5)^2 + 25)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

#Given Program#:

```
import math
def solution()
    # Given values
    a_5 = 11  # fifth term
    a_11 = 5  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5
    r_squared = r_squared  # r^6 = 5/11

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 11 * r^3
    result = a_5 * r_cubed

    return result
```

#Rewritten Program#:

```
import math
def solution():
    # Given values
    a_5 = 17  # fifth term
    a_11 = 8  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5
    r_squared = r_squared  # r^6 = 8/17
    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 17 * r^3
    result = a_5 * r_cubed

    return result
```

How about this program? #Given Program#:

```
<Here is the program.>
```

#Rewritten Program#:

逻辑增强(增加额外的约束条件)

You are a programming expert specializing in rewriting Python programs.

Your goal is to slightly transform a given Python program to make it marginally more complex and challenging for AI systems to handle, while still solving a reasonable and well-defined mathematical problem.

The Python program solves a specific mathematical question, and the variable names clearly represent their intended meaning.

Follow the method below to complete the #Rewritten Program# section:

1. Rewrite the #Given Program# by enhancing its logical conditions through the addition of ONLY ONE constraint that is directly related to the mathematical concepts of the problem and provides information gain. Ensure that the changes are minimal and only parts of the program are modified.
2. The added constraint MUST ALIGN WITH MATHEMATICAL CONCEPTS relevant to the problem and influence the program's behavior meaningfully.
3. Ensure that the enhanced logical conditions remain consistent with common sense and are logically reasonable within the mathematical context.
4. DO NOT use the added constraint for validations, checks, or raising errors in the #Rewritten Program#.
5. Ensure that numerical types and their real-world meanings correspond accurately.
6. Ensure that all intermediate variables produced at each step have reasonable numerical values that make sense within the context of the mathematical problem.
7. The rewritten program MUST SOLVE A REASONABLE AND WELL-DEFINED MATHEMATICAL PROBLEM.
8. You may introduce new variables as needed to maintain coherence and fidelity of the program's solution.
9. ONLY return ONE variable and the name MUST be "result".
10. Provide only code blocks. #Given Program#: #Rewritten Program#:

in-context版本

Here is one example how to do it. #Given Program#:

```
def solution():
    computers_initial = 9
    computers_per_day = 5
    num_days = 4  # 4 days between monday and thursday
    computers_added = computers_per_day * num_days
    computers_total = computers_initial + computers_added
    result = computers_total
    return result
```

#Rewritten Program#:

```
def solution():
    computers_initial = 9
    computers_added_per_day = 5
    num_days = 4  # Monday to Thursday
    days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday']

    computers_total = computers_initial
    for day in range(num_days):
        daily_addition = computers_added_per_day

        # Adjust daily addition based on special conditions
        if days_of_week[day] == 'Wednesday':
            # On Wednesday, add extra 3 computers as a special promotion
            daily_addition += 3
        elif days_of_week[day] == 'Tuesday':
            # On Tuesday, add only 4 computers due to budget constraints
            daily_addition = computers_added_per_day - 1

        computers_total += daily_addition

    # Return the final total, ensuring no unrealistic overshooting
    result=computers_total
    return result
```

How about this program? #Given Program#:

```
<Here is the program.>
```

#Rewritten Program#:

维度增强（新的角度、维度）

- 改成维度增强，添加天级维度、增加一个z轴、增加一个变量。 You are a Programming Expert in the field of rewriting python program. You target is to rewrite one python program to make it evolve into a bit more difficult one for AI systems to handle. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should fill the #Rewritten Program# part using following method: You should rewrite the #Given Program# and evolve it by introducing one new dimension into the process of the python program. The new dimension should correspond to a new aspect or a new coordinate to the solution. The #Rewritten Program# should aim to solve another rational mathematical question. You MUST make sure the enhanced dimension are consistent with common sense and logically reasonable. ONLY return ONE variable and the name MUST be "result". Try your best to make sure the whole program solution is coherent and faithful. You MUST only write code blocks. #Given Program#: #Rewritten Program#:

in-context 版本

Here is one example how to do it. #Given Program#:

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Calculate the sum h + k + a + b
    result = h + k + a + b

    return result
```

#Rewritten Program#:

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex
    point_y = 4  # y-coordinate of a point on the hyperbola

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Equation of the hyperbola: (y - k)^2 / a^2 - (x - h)^2 / b^2 = 1
    # Solving for the x-coordinate of a given point_y

    # Rearranging for x^2:
    x_squared = b**2 * (1 + ((point_y - k)**2 / a**2))

    # Calculate x-coordinate
    x = math.sqrt(x_squared)  # Taking the positive root for simplicity
    # Calculate the sum h + k + a + b + x
    result = h + k + a + b + x

    return result
```

#Given Program#: #Rewritten Program#:

代码续写

You are a programming expert specializing in rewriting Python programs.

Your goal is to slightly transform a given Python program to make it marginally more complex and challenging for AI systems to handle, while still solving a reasonable and well-defined mathematical problem.

The Python program solves a specific mathematical question, and the variable names clearly represent their intended meaning.

Follow the method below to complete the #Rewritten Program# section:

1. Append new code at the end of the original program, ensuring the original code remains a part of the enhanced program.
2. The new code should correspond to ONE additional mathematical step that extends the problem's solution.
3. You may make minor modifications to the existing code if necessary to accommodate the new step.
4. Ensure that all numerical values of every calculated intermediate variable strictly follow the type constraints corresponding to their actual meaning.
5. Ensure that the rewritten program allows for the reconstruction of a meaningful mathematical question based on its solution.
6. ONLY return one variable named "result".
7. Provide only code blocks.

#Given Program#: #Rewritten Program#:

in-context版本

Here are two examples how to do it. #Given Program#:

```
import math
def solution():
    # Given values
    a_5 = 17  # fifth term
    a_11 = 8  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5
    r_squared = r_squared  # r^6 = 8/17

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 17 * r^3
    result = a_5 * r_cubed

    return result
```

#Rewritten Program#:

```
import math

def solution():
    # Given values
    a_5 = 17  # fifth term
    a_11 = 8  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5  # r^6 = 8/17
    r_squared = r_squared  # This is r^6

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 17 * r^3
    eighth_term = a_5 * r_cubed

    # Calculate the first term using the general geometric progression
formula
    # a_n = a_1 * r^(n-1)
    # Using a_5, we can find a_1
    a_1 = a_5 / (r_cubed ** 4)  # a_5 = a_1 * r^4 => a_1 = a_5 / r^4

    # Compute the sum of the first six terms (S_6) in the geometric
progression
    S_6 = a_1 * (1 - r_cubed**6) / (1 - r_cubed)  # Sum of first 6 terms
of geometric progression
```

```
    # Calculate the difference between the sum of first six terms and
eighth term
    result = S_6 - eighth_term

    return result
```

#Given Program#:

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Calculate the sum h + k + a + b
    result = h + k + a + b
    return result
```

#Rewritten Program#:

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Calculate the sum h + k + a + b
    intermediate_result = h + k + a + b
    # Further calculations involving the geometric properties of the
hyperbola
    eccentricity = c / a  # Eccentricity for the hyperbola
    transverse_axis = 2 * a  # Transverse axis length for the hyperbola
    conjugate_axis = 2 * b  # Conjugate axis length for the hyperbola

    # Final result incorporating the new geometric features
    result = intermediate_result + eccentricity + transverse_axis +
conjugate_axis

    return result
```

#Given Program#:

```
<Here is the program.>
```

#Rewritten Program#:

破开条件

You are a programming expert specializing in rewriting Python programs.

Your goal is to slightly transform a given Python program to make it marginally more complex and challenging for AI systems to handle, while still solving a reasonable and well-defined mathematical problem.

The Python program solves a specific mathematical question, and the variable names clearly represent their intended meaning.

Follow the method below to complete the #Rewritten Program# section:

1. Rewrite the #Given Program# by adding new code at the beginning of the program.
2. The addition of new code should first replace the direct definition of a randomly chosen initial variable and then add new step(s) to compute the value of the eliminated variable.
3. The added new code MUST provide additional information or functionality beyond the original program's scope.
4. Introducing new variables as needed to perform the addition is allowed.
5. Ensure that the numerical values of every calculated intermediate variable adhere to the type constraints corresponding to their actual meaning, allowing for derived or computed values instead of fixed numbers.
6. Ensure that the rewritten program allows for the reconstruction of a meaningful mathematical question based on its solution.
7. The addition of program should correspond to ONLY ONE mathematical step.
8. ONLY return ONE variable and the name MUST be "result".
9. Provide only code blocks.

#Given Program#: #Rewritten Program#:

方法加强

You are a programming expert specializing in rewriting Python programs.

Your goal is to slightly transform a given Python program by incorporating more advanced mathematical theories and techniques, making it marginally more complex and challenging for AI systems to handle, while still solving a reasonable and well-defined mathematical problem.

The Python program solves a specific mathematical question, and the variable names clearly represent their intended meaning.

Follow the method below to complete the #Rewritten Program# section:

1. Rewrite the #Given Program# by integrating a bit more sophisticated mathematical theories and techniques, ensuring that changes are minimal and only parts of the program are modified.
2. The applied mathematical techniques SHOULD align with standard mathematical concepts.
3. Ensure that the enhanced calculations remain consistent with common sense and are logically reasonable.
4. Ensure that the numerical values of every calculated intermediate variable strictly follow the type constraints corresponding to their actual meaning.
5. Ensure that the rewritten program allows for the reconstruction of a meaningful mathematical question based on its solution.
6. Try your best to make sure we can reconstruct one meaningful question based on the solution of #Rewritten Program#.
7. ONLY return ONE variable and the name MUST be "result".
8. You should only write code blocks. #Given Program#: #Rewritten Program#:

in-context 版本

You are a Programming Expert in the field of rewriting python program. You target is to rewrite one python program to make it evolve into a bit more difficult one for AI systems to handle. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should fill the #Rewritten Program# part using following method: You should rewrite the #Given Program# and evolve it by applying a bit more advanced math techniques in the calculation process of the program. You should make sure the enhanced calculation consistent with common sense and logically reasonable. MUST Make sure the numerical value of every calculated intermediate variable are STRICTLY follow the type constraint for the actual meaning. You are NOT allowed to add any new variables. You should only write code blocks. Try your best to make sure we can reconstruct one meaningful question based on the solution of #Rewritten Program#. Here are three examples how to do it. #Given Program#:

```
def solution():
    """There were nine computers in the server room. Five more computers
were installed each day, from monday to thursday. How many computers are
now in the server room?"""
    computers_initial = 9
    computers_per_day = 5
    num_days = 4  # 4 days between monday and thursday
    computers_added = computers_per_day * num_days
    computers_total = computers_initial + computers_added
    result = computers_total
    return result
```

#Rewritten Program#:

```
def solution():
    computers_initial = 9
    computers_per_day = 5
    num_days = 4  # 4 days between monday and thursday
    # Applying a more complex calculation involving squares and sums for
enhanced computation
    computers_added = sum([computers_per_day * i for i in range(1,
num_days + 1)])
    computers_total = computers_initial + computers_added
    result = computers_total
    return result
```

#Given Program#:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Equation for the boundary of the shadow
    lhs = y + 3
    rhs = sp.sqrt(x**2 + (y + 1)**2 + 4)

    # Solve the equation y + 3 = sqrt(x^2 + (y + 1)^2 + 4)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

#Rewritten Program#:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Equation for the boundary of the shadow, enhanced with additional
terms
    lhs = y + 3 + sp.sin(x)  # Adding a sinusoidal term to the left-hand
side
    rhs = sp.sqrt(x**2 + (y + 1)**2 + 4) + sp.log(x + 2)  # Logarithmic
```

```
term added to the right-hand side

    # Solve the modified equation y + 3 + sin(x) = sqrt(x^2 + (y + 1)^2 +
4) + log(x + 2)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

#Given Program#:

```
<Here is the program.>
```

#Rewritten Program#:

进化选择

You are a Programming Expert in the field of rewriting python program. You target is to choose one evolutionary direction to make it evolve into a bit more difficult one for AI systems to handle. You will be given serval evolutionary directions in #Evolutionary Directions# and ONE python programs in #Given Program#. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should make the decision using following method: You should identify the controllablly evolvable and insufficient part of the program and find out the most suitable evolutionary direction from a mathematical perspective. Make sure the program can evolve a little bit in the chosen direction. Try your best to make sure we can conduct enhancement on the program to make it evolve into a coherent and logical one in the chosen direction with high success rate for AI systems like GPT4. Try your best to make sure we can reconstruct one math question from the enhanced program. You should choose ONLY ONE direction to evolve. You should ONLY output the number of the directions. #Evolutionary Directions#: 1.Enhancing its initial numerical values to make the problem difficult to guess. 2.Enhancing the logical judgment conditions in the way of adding constraints. 3.Increasing the number of loop layers to add one new dimension. 4.Continuing writing the program a bit further to generate further conclusion. 5.Applying a bit more advanced math techniques in the calculation process of the program. #Given Program#:

```
<Here is the Program.>
```

with examples

You are a Programming Expert in the field of rewriting python program. You target is to choose one evolutionary direction to make it evolve into a bit more difficult one for AI systems to handle. You will be given serval evolutionary directions and corresponding examples in #Evolutionary Directions# and ONE python programs in #Given Program#. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should make the decision using following method:

You should identify the controllablly evolvable and insufficient part of the program and find out the most suitable evolutionary direction from a mathematical perspective. Try your best to make sure we can enhance the program into a coherent and logical one in the chosen direction with absolutely high success rate for AI systems like GPT4. You should choose ONLY ONE direction to evolve. You should ONLY output the number of the directions. #Evolutionary Directions#: 1.Enhancing its initial numerical values. Initial Program:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Equation for the boundary of the shadow
    lhs = y + 3
    rhs = sp.sqrt(x**2 + (y + 1)**2 + 4)

    # Solve the equation y + 3 = sqrt(x^2 + (y + 1)^2 + 4)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

Evolved Program:

```
import sympy as sp
def solution():
    # Define the symbols
    x, y = sp.symbols('x y')

    # Enhanced equation for the boundary of the shadow
    lhs = y + 10
    rhs = sp.sqrt(x**2 + (y + 5)**2 + 25)

    # Solve the equation y + 10 = sqrt(x^2 + (y + 5)^2 + 25)
    equation = sp.Eq(lhs, rhs)
    result = sp.solve(equation, y)

    return result
```

2.Enhancing the logical judgment conditions. Initial Program:

```
import math
def solution():
    # Total number of ways to arrange 7 people without restriction
    total_ways = math.factorial(7)

    # Number of ways where no two boys are next to each other:
```

```
    # 1. Arrange 3 daughters in a row (3! ways)
    # 2. Place the boys in the gaps between the daughters (4 gaps) in a
way such that no two boys are next to each other
    daughter_arrangements = math.factorial(3)
    boy_positions = math.comb(4, 4) * math.factorial(4)  # Place all 4
boys in the 4 gaps
    no_boys_next_to_each_other_ways = daughter_arrangements *
boy_positions

    # The number of ways where at least 2 boys are next to each other
    result = total_ways - no_boys_next_to_each_other_ways
    return result
```

Evolved Program:

```
import math
def solution():
    # Total number of ways to arrange 7 people with the first place must
be boys.
    total_ways = math.comb(4, 1) * math.factorial(6)


    # Number of ways where no two boys are next to each other with the
first place must be boys:
    # 1. Arrange 3 daughters in a row (3! ways)
    # 2. Place the boys in the gaps between the daughters (4 gaps ) in a
way such that no two boys are next to each other
    daughter_arrangements = math.factorial(3)
    boy_positions = math.comb(4, 4) * math.factorial(4)  # Place all 4
boys in the 4 gaps
    no_boys_next_to_each_other_ways = daughter_arrangements *
boy_positions

    # The number of ways where at least 2 boys are next to each other and
the first place must be boys.
    result = total_ways - no_boys_next_to_each_other_ways
    return result
```

3.Increasing the number of loop layers. Initial Program:

```
def solution():
    initial_computers = 9
    computers_per_day = 5
    num_days = 4  # 4 days between monday and thursday
    computers_added = computers_per_day * num_days
    computers_total = initial_computers + computers_added
    result = computers_total
    return result
```

Evolved Program:

```
def solution():
    initial_computers = 9
    computers_per_day = 5
    num_days_week1 = 4  # 4 days between Monday and Thursday
    num_days_week2 = 3  # 3 additional days between Friday and Sunday
    num_weeks = 2  # The total number of weeks (2 weeks in this case)

    total_computers = initial_computers

    # Loop for first week (Monday to Thursday)
    for week in range(num_weeks):
        if week == 0:
            num_days = num_days_week1
        else:
            num_days = num_days_week2

        for day in range(num_days):
            total_computers += computers_per_day  # Add computers for each
day

    # Final result calculation after all days
    result = total_computers
    return result
```

4.Continuing writing the program a bit further. Initial Program:

```
import math
def solution():
    # Given values
    a_5 = 17  # fifth term
    a_11 = 8  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5
    r_squared = r_squared  # r^6 = 8/17

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 17 * r^3
    result = a_5 * r_cubed

    return result
```

Evolved Program:

```
import math

def solution():
    # Given values
    a_5 = 17  # fifth term
    a_11 = 8  # eleventh term

    # Calculate r^6
    r_squared = a_11 / a_5  # r^6 = 8/17
    r_squared = r_squared  # This is r^6

    # r^3 is the square root of r^6
    r_cubed = math.sqrt(r_squared)

    # The eighth term is 17 * r^3
    eighth_term = a_5 * r_cubed

    # Calculate the first term using the general geometric progression
formula
    # a_n = a_1 * r^(n-1)
    # Using a_5, we can find a_1
    a_1 = a_5 / (r_cubed ** 4)  # a_5 = a_1 * r^4 => a_1 = a_5 / r^4

    # Compute the sum of the first six terms (S_6) in the geometric
progression
    S_6 = a_1 * (1 - r_cubed**6) / (1 - r_cubed)  # Sum of first 6 terms
of geometric progression

    # Calculate the difference between the sum of first six terms and
eighth term
    result = S_6 - eighth_term

    return result
```

#Given Program#:

```
<Here is the Program.>
```

循环增强

- 删除循环增强，不合理。
- 改成维度增强，添加天级维度、增加一个z轴之类的。 You are a Programming Expert in the field of rewriting python program. You target is to rewrite one python program to make it evolve into a bit more difficult one for AI systems to handle. The python program aims to solve one mathematical question, and the variable names indicate their actual meaning. You should fill the #Rewritten Program# part using following method: You should rewrite the #Given Program# and evolve it by increasing the times of loops or increasing the number of loop layers to add one new dimension. You

should make sure the enhanced loops are consistent with common sense and logically reasonable. You are allowed to add new variables to make the whole program solution coherent and faithful. You should only write code blocks. #Given Program#:

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Calculate the sum h + k + a + b
    result = h + k + a + b

    return result
```

#Rewritten Program#:

```
import math
def solution():
    # Given values
    h = 2  # center's x-coordinate
    k = 0  # center's y-coordinate
    focus_y = 6  # y-coordinate of the focus
    vertex_y = -3  # y-coordinate of the vertex

    # Distance from center to focus (c) and center to vertex (a)
    c = abs(focus_y - k)
    a = abs(vertex_y - k)

    # For a hyperbola, c^2 = a^2 + b^2
    b_squared = c**2 - a**2
    b = math.sqrt(b_squared)

    # Adding a new loop to simulate multiple hyperbolas with varying
vertex_y values
    total_sum = 0
    for new_vertex_y in range(vertex_y - 3, vertex_y + 3):  # New loop to
simulate changes in vertex_y
        # Update a with new vertex_y value
        a = abs(new_vertex_y - k)
        # Recalculate b for the new vertex_y value
        b_squared = c**2 - a**2
        b = math.sqrt(b_squared)

        # Calculate the sum for this new hyperbola configuration
        total_sum += (h + k + a + b)

    return total_sum
```