

Listing 1: Example Python Code

```

1  #!/usr/bin/env python3
2  #!/usr/bin/env python3
3
4
5  def multiply_matrices(a: list, b: list) -> list:
6      """
7      Multiply two 2x2 matrices.
8
9      Args:
10         a (list): The first matrix.
11         b (list): The second matrix.
12
13     Returns:
14         list: The product of the two matrices.
15
16     Raises:
17         TypeError: If either matrix is not a list of lists.
18         ValueError: If either matrix is not 2x2 or if the
19                     elements are not numbers.
20     """
21     if not isinstance(a, list) or not isinstance(b, list):
22         raise TypeError("Both matrices must be lists of lists")
23     if len(a) != 2 or len(b) != 2 or len(a[0]) != 2 or
24         len(b[0]) != 2:
25         raise ValueError("Both matrices must be 2x2")
26     if not all(isinstance(row, list) for row in a) or not all(
27         isinstance(row, list) for row in b
28     ):
29         raise TypeError("Both matrices must be lists of lists")
30     if not all(
31         isinstance(element, (int, float)) for row in a for
32         element in row
33     ) or not all(isinstance(element, (int, float)) for row in
34         b for element in row):
35         raise ValueError("All elements of the matrices must be
36             numbers")
37
38     result = [
39         [sum(a_val * b_val for a_val, b_val in zip(a_row,
40             b_col)) for b_col in zip(*b)]
41         for a_row in a
42     ]
43     if not isinstance(result, list) or not all(isinstance(row,
44         list) for row in result):
45         raise TypeError("The result must be a list of lists")
46     if not all(isinstance(element, (int, float)) for row in
47         result for element in row):
48         raise ValueError("All elements of the result must be

```

```

        numbers")
41     return result
42
43
44 def matrix_power(matrix: list, n: int) -> list:
45     """
46     Calculate the nth power of a matrix using a
47         divide-and-conquer approach.
48
49     Args:
50         matrix (list): The matrix to exponentiate.
51         n (int): The exponent.
52
53     Returns:
54         list: The nth power of the matrix.
55
56     Raises:
57         TypeError: If the matrix is not a list of lists or if
58             the exponent is not an integer.
59         ValueError: If the exponent is negative or if the
60             matrix is not 2x2 or if the elements are not
61             numbers.
62     """
63     if not isinstance(matrix, list):
64         raise TypeError("The matrix must be a list of lists")
65     if not isinstance(n, int):
66         raise TypeError("The exponent must be an integer")
67     if n < 0:
68         raise ValueError("Exponent must be a non-negative
69             integer")
70     if len(matrix) != 2 or len(matrix[0]) != 2:
71         raise ValueError("The matrix must be 2x2")
72     if not all(isinstance(row, list) for row in matrix):
73         raise TypeError("The matrix must be a list of lists")
74     if not all(isinstance(element, (int, float)) for row in
75         matrix for element in row):
76         raise ValueError("All elements of the matrix must be
77             numbers")
78
79     if n == 0:
80         result = [[1, 0], [0, 1]]
81     elif n == 1:
82         result = matrix
83     elif n % 2 == 0:
84         half_pow = matrix_power(matrix, n // 2)
85         result = multiply_matrices(half_pow, half_pow)
86     else:
87         half_pow = matrix_power(matrix, n // 2)
88         result = multiply_matrices(multiply_matrices(half_pow,
89             half_pow), matrix)

```

```

82     if not isinstance(result, list) or not all(isinstance(row,
83         list) for row in result):
84         raise TypeError("The result must be a list of lists")
85     if not all(isinstance(element, (int, float)) for row in
86         result for element in row):
87         raise ValueError("All elements of the result must be
88             numbers")
89     return result
90
91 def fibonacci(n: int) -> int:
92     """
93     Calculate the nth Fibonacci number using matrix
94     exponentiation.
95
96     Args:
97         n (int): The index of the Fibonacci number to
98         calculate.
99
100    Returns:
101        int: The nth Fibonacci number.
102
103    Raises:
104        TypeError: If the index is not an integer.
105        ValueError: If the index is negative.
106    """
107    if not isinstance(n, int):
108        raise TypeError("The index must be an integer")
109    if n < 0:
110        raise ValueError("Index must be a non-negative
111            integer")
112    if n == 0:
113        return 0
114    fib_matrix = [[1, 1], [1, 0]]
115    result_matrix = matrix_power(fib_matrix, n - 1)
116    result = result_matrix[0][0]
117    if not isinstance(result, int):
118        raise TypeError("The result must")
119    return result
120
121 if __name__ == "__main__":
122     n = 10
123     print(f"fibonacci({n})={fibonacci(n)}")

```

hello 1