```
from typing import Union, Dict, List, Tuple, Any



def any_to_str(data: Union[str, Dict, List, Tuple, Any]) -> str:
    """Convert any input data type to a nicely formatted string.

    This function handles conversion of various Python data types into a clean string representation.
    It recursively processes nested data structures and handles None values gracefully.

    Args:
        data: Input data of any type to convert to string. Can be:
            - Dictionary
            - List/Tuple
            - String
            - None
            - Any other type that can be converted via str()

    Returns:
        str: A formatted string representation of the input data.
            - Dictionaries are formatted as "key: value" pairs separated by commas
            - Lists/tuples are comma-separated
            - None returns empty string
            - Other types are converted using str()

    Examples:
        >>> any_to_str({'a': 1, 'b': 2})
```

```
    'a: 1, b: 2'
    >>> any_to_str([1, 2, 3])
    '1, 2, 3'
    >>> any_to_str(None)
    ''
"""
try:
    if isinstance(data, dict):
        # Format dictionary with newlines and indentation
        items = []
        for k, v in data.items():
            value = any_to_str(v)
            items.append(f"{k}: {value}")
        return "\n".join(items)

    elif isinstance(data, (list, tuple)):
        # Format sequences with brackets and proper spacing
        items = [any_to_str(x) for x in data]
        if len(items) == 0:
            return "[]" if isinstance(data, list) else "()"
        return (
            f"[{', '.join(items)}]"
            if isinstance(data, list)
            else f"({', '.join(items)})"
        )
```

```
        elif data is None:
            return "None"

        else:
            # Handle strings and other types
            if isinstance(data, str):
                return f"'{data}'"
            return str(data)

    except Exception as e:
        return f"Error converting data: {str(e)}"



# def main():
#     # Example 1: Dictionary
#     print("Dictionary:")
#     print(
#         any_to_str(
#             {
#                 "name": "John",
#                 "age": 30,
#                 "hobbies": ["reading", "hiking"],
#             }
#         )
#     )
```

```
#     print("\nNested Dictionary:")
#     print(
#         any_to_str(
#             {
#                 "user": {
#                     "id": 123,
#                     "details": {"city": "New York", "active": True},
#                 },
#                 "data": [1, 2, 3],
#             }
#         )
#     )

#     print("\nList and Tuple:")
#     print(any_to_str([1, "text", None, (1, 2)]))
#     print(any_to_str((True, False, None)))

#     print("\nEmpty Collections:")
#     print(any_to_str([]))
#     print(any_to_str({}))



# if __name__ == "__main__":
#     main()
```