1. Permutations

```
def permute(elem, path=[]):
    if not elem:
        print(path)
    else:
        for i in range(len(elem)):
            permute(elem[:i] + elem[i+1:], path + [elem[i]])

def permute_str(s):
    permute(list(s))

def permute_n(nums):
    permute(nums)
print("Permutations of 'ABC':")
permute_str("ABC")

print("\nPermutations of [1, 2, 3]:")
permute_n([1, 2, 3])
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/permutationss.py
Permutations of 'ABC':
['A', 'B', 'C']
['A', 'C', 'B']
['B', 'A', 'C']
['B', 'C', 'A']
['C', 'A', 'B']
['C', 'B', 'A']

Permutations of [1, 2, 3]:
[1, 2, 3]
[1, 3, 2]
[2, 1, 3]
[2, 3, 1]
[3, 1, 2]
[3, 2, 1]
>>>
```

2. Combinations

```
def comb(arr, a):
    if a == 0:
        return [[]]
    if len(arr) <a:
        return []
    if len(arr) == b:
        return [arr]
    result = []
    for i in range(len(arr)):
        f = arr[i]
        rem = arr[i+1:]
        for c in comb(rem, a-1):
            result.append([f] + c)
    return result
a = input("Enter elements of the list separated by spaces: ").split()
b = int(input("Enter the length of combinations: "))
combinations = comb(a,b)
print(combinations)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/combinators.py
Enter elements of the list separated by spaces:
```

3. Subset

```
def subsets(s):
    if len(s) == 0:
        return [[]]
    x = subsets(s[:-1])
    return x + [[s[-1]] + y for y in x]
s = [1, 2, 3,4]
result = subsets(s)
print(result)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/subsets.py
[[], [1], [2], [2, 1], [3], [3, 1], [3, 2], [3, 2, 1], [4], [4, 1], [4, 2], [4, 2, 1], [
4, 3], [4, 3, 1], [4, 3, 2], [4, 3, 2, 1]]
>>>
```

4. Suduko

```python
def solve_sudoku(board):
    def find_empty_cell(board):
        for i in range(9):
            for j in range(9):
                if board[i][j] == 0:
                    return i, j
        return -1, -1

    def is_valid_move(board, row, col, num):
        for i in range(9):
            if board[row][i] == num or board[i][col] == num:
                return False
        box_row, box_col = 3 * (row // 3), 3 * (col // 3)
        for i in range(box_row, box_row + 3):
            for j in range(box_col, box_col + 3):
                if board[i][j] == num:
                    return False
        return True

    def solve(board):
        row, col = find_empty_cell(board)
        if row == -1 and col == -1:
            return True
        for num in range(1, 10):
            if is_valid_move(board, row, col, num):
                board[row][col] = num
                if solve(board):
                    return True
                board[row][col] = 0
        return False

    if solve(board):
        return board
    return None
if __name__ == "__main__":
    board = [
        [5, 3, 0, 0, 7, 0, 0, 0, 0],
        [6, 0, 0, 1, 9, 5, 0, 0, 0],
        [0, 9, 8, 0, 0, 0, 0, 6, 0],
        [8, 0, 0, 0, 6, 0, 0, 0, 3],
        [4, 0, 0, 8, 0, 3, 0, 0, 1],
        [7, 0, 0, 0, 2, 0, 0, 0, 6],
        [0, 6, 0, 0, 0, 0, 2, 8, 0],
        [0, 0, 0, 4, 1, 9, 0, 0, 5],
        [0, 0, 0, 0, 8, 0, 0, 7, 9]
    ]
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/suduko.py
Sudoku board to solve:
[5, 3, 0, 0, 7, 0, 0, 0, 0]
[6, 0, 0, 1, 9, 5, 0, 0, 0]
[0, 9, 8, 0, 0, 0, 0, 6, 0]
[8, 0, 0, 0, 6, 0, 0, 0, 3]
[4, 0, 0, 8, 0, 3, 0, 0, 1]
[7, 0, 0, 0, 2, 0, 0, 0, 6]
[0, 6, 0, 0, 0, 0, 2, 8, 0]
[0, 0, 0, 4, 1, 9, 0, 0, 5]
[0, 0, 0, 0, 8, 0, 0, 7, 9]

Solving Sudoku...

Sudoku solved:
[5, 3, 4, 6, 7, 8, 9, 1, 2]
[6, 7, 2, 1, 9, 5, 3, 4, 8]
[1, 9, 8, 3, 4, 2, 5, 6, 7]
[8, 5, 9, 7, 6, 1, 4, 2, 3]
[4, 2, 6, 8, 5, 3, 7, 9, 1]
[7, 1, 3, 9, 2, 4, 8, 5, 6]
[9, 6, 1, 5, 3, 7, 2, 8, 4]
[2, 8, 7, 4, 1, 9, 6, 3, 5]
[3, 4, 5, 2, 8, 6, 1, 7, 9]
>>>
```

5. Hamiltonian

```python
def hamiltonian_cycle(graph):
    n = len(graph)
    path = [-1] * (n + 1)
    visited = [False] * n

    def is_valid(v, pos):
        return not visited[v] and (pos == 0 or graph[path[pos - 1]][v] == 1)

    def hamiltonian_util(pos):
        if pos == n:
            return graph[path[pos - 1]][path[0]] == 1
        for v in range(n):
            if is_valid(v, pos):
                path[pos] = v
                visited[v] = True
                if hamiltonian_util(pos + 1):
                    return True
                visited[v] = False
        return False

    path[0] = 0
    visited[0] = True
    if hamiltonian_util(1):
        print("Hamiltonian cycle found:")
        print(path[:n])
        return path[:n]
    print("No Hamiltonian cycle exists")
    return None
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr  9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/hamiltonian.py
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/hamiltonian.py
Hamiltonian cycle found:
[0, 1, 2, 4, 3]
>>>
```