```
from itertools import permutations

def find_permutations():
    s = input("Enter a string: ")
    return ["".join(p) for p in permutations(s)]

print(find_permutations())
```

```
Enter a string: bhuvan
['bhuvan', 'bhuvna', 'bhuavn', 'bhuanv', 'bhunva', 'bhunav', 'bhvuan', 'bhvuna', 'bhvaun', 'bhvan
```

```
def fibonacci():
    n = int(input("Enter n: "))
    dp = [0, 1] + [0] * (n-1)
    for i in range(2, n+1):
        dp[i] = dp[i-1] + dp[i-2]
    return dp[n]

print(fibonacci())
```

```
Enter n: 3
2
```

```
def find_duplicates():
    arr = list(map(int, input("Enter numbers: ").split()))
    from collections import Counter
    counts = Counter(arr)
    return [num for num, count in counts.items() if count > 1]

print(find_duplicates())
```

```
Enter numbers: 1 2 3 2 3 4
[2, 3]
```

```
def longest_increasing_subsequence():
    arr = list(map(int, input("Enter numbers: ").split()))
    if not arr:
        return 0
    dp = [1] * len(arr)
    for i in range(len(arr)):
        for j in range(i):
            if arr[i] > arr[j]:
                dp[i] = max(dp[i], dp[j] + 1)
    return max(dp)

print(longest_increasing_subsequence())
```

```
Enter numbers: 1 2 3 4 3 2 1
4
```

```
def find_k_largest():
    arr = list(map(int, input("Enter numbers: ").split()))
    k = int(input("Enter k: "))
    return sorted(arr, reverse=True)[:k]

print(find_k_largest())
```

```
Enter numbers: 12 31 21
Enter k: 4
[31, 21, 12]
```

```
def rotate_matrix():
    n = int(input("Enter matrix size: "))
    matrix = [list(map(int, input().split())) for _ in range(n)]
    rotated = list(zip(*matrix[::-1]))
    return [list(row) for row in rotated]

print(rotate_matrix())
```

```
Enter matrix size: 3
1 2 3
4 5 6
7 8 9
[[7, 4, 1], [8, 5, 2], [9, 6, 3]]
```

```
def is_valid_sudoku():
    board = [list(map(int, input().split())) for _ in range(9)]
    def is_valid_block(block):
        nums = [num for num in block if num != 0]
        return len(nums) == len(set(nums))
    for row in board:
        if not is_valid_block(row):
            return False
    for col in zip(*board):
        if not is_valid_block(col):
            return False
    for i in range(0, 9, 3):
        for j in range(0, 9, 3):
            block = [board[x][y] for x in range(i, i+3) for y in range(j, j+3)]
            if not is_valid_block(block):
                return False
    return True

print(is_valid_sudoku())
```

```
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
False
```

```
def stock_market_simulator():
    import random
    stocks = {"AAPL": 100, "GOOGL": 1500, "TSLA": 700}
    portfolio = {}
    for _ in range(5):
        stock = random.choice(list(stocks.keys()))
        change = random.uniform(-5, 5)
        stocks[stock] += change
    return stocks

print(stock_market_simulator())
```

```
{'AAPL': 100, 'GOOGL': 1500.0154059732163, 'TSLA': 696.4526715972945}
```