



```
#Task 1: Basic Dictionary Operations
Person = {'name': 'Alice', 'age': 25, 'city': 'New York'}
Person['email'] = 'alice@example.com'
Person['age'] = 26
del Person['city']
print("Task 1 Output:", Person)
```

 Task 1 Output: {'name': 'Alice', 'age': 26, 'email': 'alice@example.com'}

```
# Task 2: Accessing and Modifying Dictionary Values
fruits = {'apple': 10, 'banana': 5, 'cherry': 15}
print("Task 2 Output - Banana quantity:", fruits['banana'])

fruits['orange'] = 8
fruits['apple'] += 5
del fruits['cherry']

print("Task 2 Output - Final Dictionary:", fruits)
```

 Task 2 Output - Banana quantity: 5
Task 2 Output - Final Dictionary: {'apple': 15, 'banana': 5, 'orange': 8}

```
# Task 3: Word Frequency Counter
sentence = "Hello world hello"
words = sentence.lower().split()
frequency = {}


for word in words:
    frequency[word] = frequency.get(word, 0) + 1

print("Task 3 Output - Word Frequency:", frequency)
```

 Task 3 Output - Word Frequency: {'hello': 2, 'world': 1}

```
#Task 4: Merging Two Dictionaries
def merge_dicts(dict1, dict2):
    result = dict1.copy()
    for key, value in dict2.items():
        result[key] = result.get(key, 0) + value
    return result

dict1 = {'apple': 5, 'banana': 3, 'orange': 7}
dict2 = {'banana': 2, 'orange': 3, 'grape': 4}
print("Task 4 Output - Merged Dictionary:", merge_dicts(dict1, dict2))
```

 Task 4 Output - Merged Dictionary: {'apple': 5, 'banana': 5, 'orange': 10, 'grape': 4}

```
#Task 5: Nested Dictionary Processing
employees = {
    'E001': {'name': 'Alice', 'department': 'HR', 'salary': 50000},
    'E002': {'name': 'Bob', 'department': 'IT', 'salary': 60000},
    'E003': {'name': 'Charlie', 'department': 'Finance', 'salary': 55000}
}

def get_salary(employee_dict, emp_id):
    return employee_dict.get(emp_id, {}).get('salary')

def increase_salary(employee_dict, percentage):
    for emp in employee_dict.values():
        emp['salary'] += emp['salary'] * (percentage / 100)

print("Task 5 Output - Salary of E002:", get_salary(employees, 'E002'))
increase_salary(employees, 10)
print("Task 5 Output - Up
```

◆ What can I help you build?



Task 5 Output - Salary of E002: 60000
 Task 5 Output - Updated Employees: {'E001': {'name': 'Alice', 'department': 'HR', 'salary': 55000.0}, 'E002': {'name': ' '}}

```
# Task 6: Sorting a Dictionary by Value
marks = {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 90}
sorted_marks = dict(sorted(marks.items(), key=lambda x: x[1], reverse=True))
print("Task 6 Output - Sorted Dictionary:", sorted_marks)
```

Task 6 Output - Sorted Dictionary: {'Bob': 92, 'David': 90, 'Alice': 85, 'Charlie': 78}

```
# Task 7: Multiplication Table (1 to 10)
print("Task 7 Output:")
for i in range(1, 11):
    for j in range(1, 11):
        print(f"{i * j:3}", end=" ")
    print()
```

Task 7 Output:

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

```
#Task 8: Transpose of a 2D Matrix
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
rows, cols = len(matrix), len(matrix[0])
transpose = [[0]*rows for _ in range(cols)]

for i in range(rows):
    for j in range(cols):
        transpose[j][i] = matrix[i][j]

print("Task 8 Output - Transposed Matrix:", transpose)
```

Task 8 Output - Transposed Matrix: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

```
#Task 9: Count Prime Numbers in 2D Matrix
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True

matrix = [[2, 4, 5], [7, 9, 11], [13, 16, 19]]
prime_count = sum(is_prime(num) for row in matrix for num in row)
print("Task 9 Output - Total prime numbers:", prime_count)
```

Task 9 Output - Total prime numbers: 6


```
#Task 10: Spiral Order Matrix Traversal
def spiral_order(matrix):
    result = []
    while matrix:
        result += matrix.pop(0)
        if matrix and matrix[0]:
            for row in matrix:
                result.append(row.pop())
    if matrix:
        result += matrix.pop()[::-1]
    if matrix and matrix[0]:
```

```

        for row in matrix[::-1]:
            result.append(row.pop(0))
    return result

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print("Task 10 Output - Spiral Order:", spiral_order(matrix))

```


 Task 10 Output - Spiral Order: [1, 2, 3, 6, 9, 8, 7, 4, 5]

```

# Task 11: BMI Calculator
weight = float(input("Task 11 - Enter weight (kg): "))
height = float(input("Task 11 - Enter height (m): "))
bmi = weight / (height ** 2)

print("Task 11 Output - BMI:", round(bmi, 2))
if bmi < 18.5:
    print("Category: Underweight")
elif bmi < 25:
    print("Category: Normal weight")
elif bmi < 30:
    print("Category: Overweight")
else:
    print("Category: Obesity")

```


 Task 11 - Enter weight (kg): 86
Task 11 - Enter height (m): 144.4
Task 11 Output - BMI: 0.0
Category: Underweight

```

#Task 12: Student Grade Classification
score = int(input("Task 12 - Enter student score: "))
if 90 <= score <= 100:
    grade = "A"
elif 80 <= score < 90:
    grade = "B"
elif 70 <= score < 80:
    grade = "C"
elif 60 <= score < 70:
    grade = "D"
else:
    grade = "F"

status = "Pass" if grade in ["A", "B", "C"] else "Fail"
print("Task 12 Output - Grade:", grade)
print("Task 12 Output - Status:", status)

```


 Task 12 - Enter student score: 88
Task 12 Output - Grade: B
Task 12 Output - Status: Pass

```

#Task 13: Palindrome Words in 2D List
matrix = [
    ["madam", "apple", "racecar"],
    ["level", "hello", "civic"],
    ["world", "deified", "rotor"]
]

print("Task 13 Output - Palindrome Check:")
for row in matrix:
    for word in row:
        if word == word[::-1]:
            print(f"'{word}' is a palindrome")
        else:
            print(f"'{word}' is not a palindrome")

```

 Task 13 Output - Palindrome Check:
'madam' is a palindrome
'apple' is not a palindrome
'racecar' is a palindrome
'level' is a palindrome
'hello' is not a palindrome

```
'civic' is a palindrome
'world' is not a palindrome
'deified' is a palindrome
'rotor' is a palindrome
```

```
#Task 14: Multiplication Table with Even Products Only
```

```
print("Task 14 Output - Even Products Only:")
```

```
for i in range(1, 11):
    for j in range(1, 11):
        product = i * j
        if product % 2 == 0:
            print(f"{product:3}", end=" ")
        else:
            print("  ", end=" ")
    print()
```

```
Task 14 Output - Even Products Only:
      2      4      6      8     10
2    4     6     8    10    12    14    16    18    20
   6     12     18     24     30
4    8    12    16    20    24    28    32    36    40
   10     20     30     40     50
6    12    18    24    30    36    42    48    54    60
   14     28     42     56     70
8    16    24    32    40    48    56    64    72    80
   18     36     54     72     90
10   20    30    40    50    60    70    80    90   100
```

Start coding or [generate](#) with AI.