```python
#basic dictionary operations
mydict = {"name": "Alice", "age": 25, "city": "New York"}
mydict["email"] = "alice@example.com"
mydict["age"] = 26
del mydict["city"]
print(mydict)
```

```
{'name': 'Alice', 'age': 26, 'email': 'alice@example.com'}
```

```python
#Acessing and modifying dictioary values
fruits = {'apple': 10, 'banana': 5, 'cherry': 15}
banana_qty = fruits['banana']
print("Quantity of banana:", banana_qty)
fruits['orange'] = 8
fruits['apple'] += 5
print("Final dictionary:", fruits)
del fruits['cherry']
print("Final dictionary:", fruits)
```

```
Quantity of banana: 5
Final dictionary: {'apple': 15, 'banana': 5, 'cherry': 15, 'orange':
8}
Final dictionary: {'apple': 15, 'banana': 5, 'orange': 8}
```

```python
#counting work frequency
sentence = input("Enter a sentence: ")
words = sentence.lower().split()
word_count = {}
for word in words:
    if word in word_count:
        word_count[word] += 1
    else:
        word_count[word] = 1
print(word_count)
```

```
Enter a sentence:  anclinstephy

{'anclinstephy': 1}
```

```python
#merging two dictionaries
def merge_dicts(dict1, dict2):
    merged_dict = dict1.copy()
    for key, value in dict2.items():
        if key in merged_dict:
            merged_dict[key] += value
        else:
            merged_dict[key] = value
    return merged_dict
dict1 = {'apple': 5, 'banana': 3, 'orange': 7}
dict2 = {'banana': 2, 'orange': 3, 'grape': 4}
```

```
result = merge_dicts(dict1, dict2)
print(result)
```

```
{'apple': 5, 'banana': 5, 'orange': 10, 'grape': 4}
```

```
# 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):
    if emp_id in employee_dict:
        return employee_dict[emp_id]['salary']
    else:
        return "Employee ID not found."
def increase_salary(employee_dict, percentage):
    for emp_id, details in employee_dict.items():
        details['salary'] += details['salary'] * (percentage / 100)
emp_id = 'E002'
print(f"Salary of {emp_id}:", get_salary(employees, emp_id))
increase_salary(employees, 10)
print("Updated employee details after 10% salary increase:")
print(employees)
```

```
Salary of E002: 60000
Updated employee details after 10% salary increase:
{'E001': {'name': 'Alice', 'department': 'HR', 'salary': 55000.0},
'E002': {'name': 'Bob', 'department': 'IT', 'salary': 66000.0},
'E003': {'name': 'Charlie', 'department': 'Finance', 'salary':
60500.0}}
```

```
# sorting a dictionary
marks = {'Alice': 85, 'Bob': 92, 'Charlie': 78, 'David': 90}
sorted_marks = dict(sorted(marks.items(), key=lambda item: item[1],
reverse=True))
print(sorted_marks)
```

```
{'Bob': 92, 'David': 90, 'Alice': 85, 'Charlie': 78}
```

```
# Multiple table
for i in range(1, 11):
    for j in range(1, 11):
        print(f"{i * j:3}", end=" ")
    print()
```

```
  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
```

```python
#Transpose of a 2D Matrix
# Input matrix
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# Get number of rows and columns
rows = len(matrix)
cols = len(matrix[0])

# Create an empty matrix for transpose
transpose = [[0 for _ in range(rows)] for _ in range(cols)]

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

# Print the transposed matrix
print("Transposed Matrix:", transpose)
```

```
Transposed Matrix: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

```python
# Counting Prime Numbers in a 2D Matrix
matrix = [[2, 4, 5], [7, 9, 11], [13, 16, 19]]
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
prime_count = 0
for row in matrix:
    for num in row:
        if is_prime(num):
            prime_count += 1
print("Total prime numbers:", prime_count)
```

```
Total prime numbers: 6
```

```python
#Spiral Order Matrix Traversal
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
spiral_order = []
top, bottom, left, right = 0, len(matrix) - 1, 0, len(matrix[0]) - 1
while top <= bottom and left <= right:
```

```
    for i in range(left, right + 1):
        spiral_order.append(matrix[top][i])
    top += 1
    for i in range(top, bottom + 1):
        spiral_order.append(matrix[i][right])
    right -= 1
    if top <= bottom:
        for i in range(right, left - 1, -1):
            spiral_order.append(matrix[bottom][i])
        bottom -= 1

    if left <= right:
        for i in range(bottom, top - 1, -1):
            spiral_order.append(matrix[i][left])
        left += 1

print("Spiral Order:", spiral_order)
```

```
Spiral Order: [1, 2, 3, 6, 9, 8, 7, 4, 5]
```

```
#Body Mass Index (BMI) Calculation
weight = float(input("Enter weight (kg): "))
height = float(input("Enter height (m): "))
bmi = weight / (height ** 2)
if bmi < 18.5:
    category = "Underweight"
elif 18.5 <= bmi < 25:
    category = "Normal weight"
elif 25 <= bmi < 30:
    category = "Overweight"
else:
    category = "Obesity"
print(f"BMI: {bmi:.2f}")
print(f"Category: {category}")
```

```
Enter weight (kg):  60
Enter height (m):  34.90
```

```
BMI: 0.05
Category: Underweight
```

```
#Student Grade Classification
score = int(input("Enter student score: "))
if 90 <= score <= 100:
    grade = "A"
    status = "Pass"
elif 80 <= score < 90:
    grade = "B"
    status = "Pass"
elif 70 <= score < 80:
    grade = "C"
```

```
    status = "Pass"
elif 60 <= score < 70:
    grade = "D"
    status = "Fail"
else:
    grade = "F"
    status = "Fail"
print(f"Grade: {grade}")
print(f"Status: {status}")
```

```
Enter student score:  23

Grade: F
Status: Fail
```

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

# Print grade and status
print(f"Grade: {grade}")
print(f"Status: {status}")
```

```
Enter student score:  67

Grade: D
Status: Fail
```

```
#Checking Palindromes in a 2D List
matrix = [["madam", "apple", "racecar"], ["level", "hello", "civic"],
["world", "deified", "rotor"]]
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")
```

```
'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
```

```
#Multiplication Table with Even Numbers 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()
```

```
      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
```