Task #1 – Zero-Shot Prompting with Conditional Validation

Objective

Use zero-shot prompting to instruct an AI tool to generate a function that validates an Indian mobile number.

Requirements

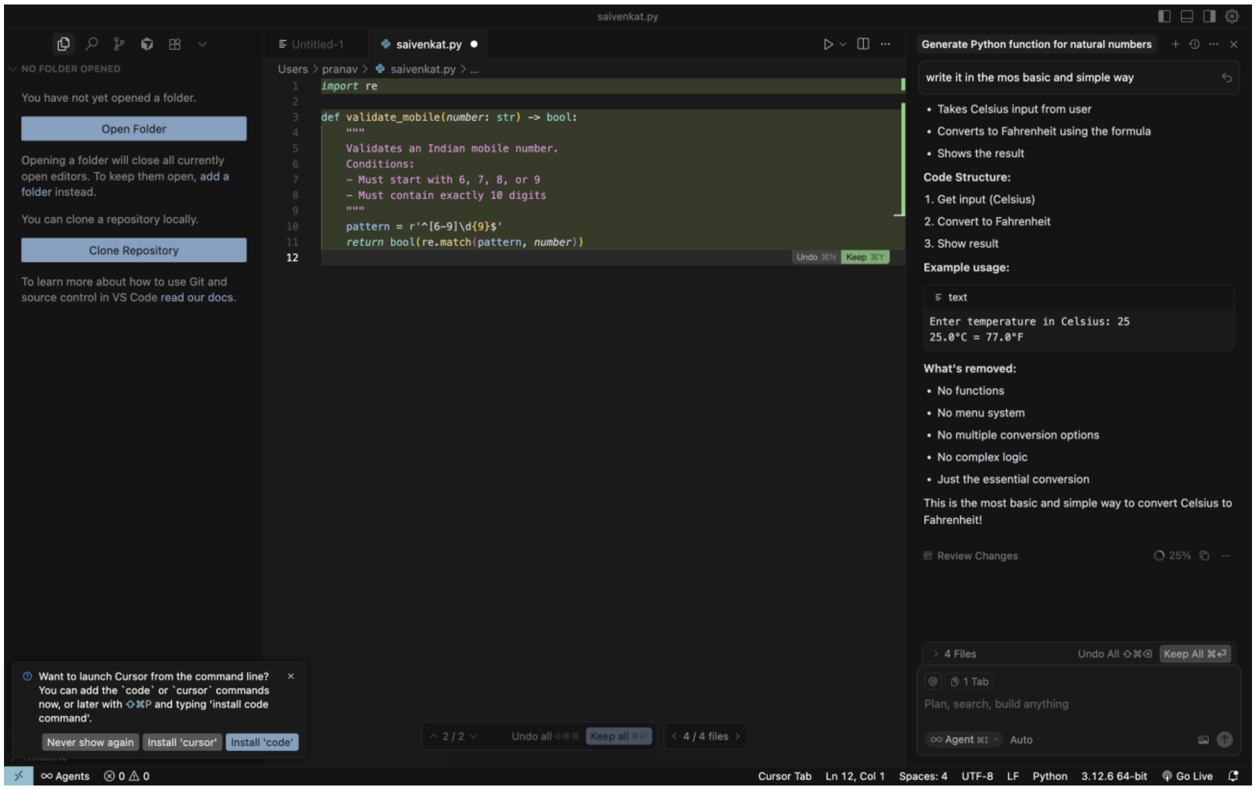
• The function must ensure the mobile number:

o Starts with 6, 7, 8, or 9

o Contains exactly 10 digits

Expected Output

• A valid Python function that performs all required validations without using any input-output examples in the prompt.



NO FOLDER OPENED

You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

Want to launch Cursor from the command line? You can add the `code` or `cursor` commands now, or later with ⇧⌘P and typing 'install code command'.

```python
import re

def validate_mobile(number: str) -> bool:
    """
    Validates an Indian mobile number.
    Conditions:
    - Must start with 6, 7, 8, or 9
    - Must contain exactly 10 digits
    """
    pattern = r'^[6-9]\d{9}$'
    return bool(re.match(pattern, number))
```

write it in the mos basic and simple way

- Takes Celsius input from user
- Converts to Fahrenheit using the formula
- Shows the result

Code Structure:

1. Get input (Celsius)
2. Convert to Fahrenheit
3. Show result

Example usage:

```
Enter temperature in Celsius: 25
25.0°C = 77.0°F
```

What's removed:

- No functions
- No menu system
- No multiple conversion options
- No complex logic
- Just the essential conversion

This is the most basic and simple way to convert Celsius to Fahrenheit!

OUTPUT. :

print(validate_mobile("9876543210")) # Valid

print(validate_mobile("1234567890")) # Invalid

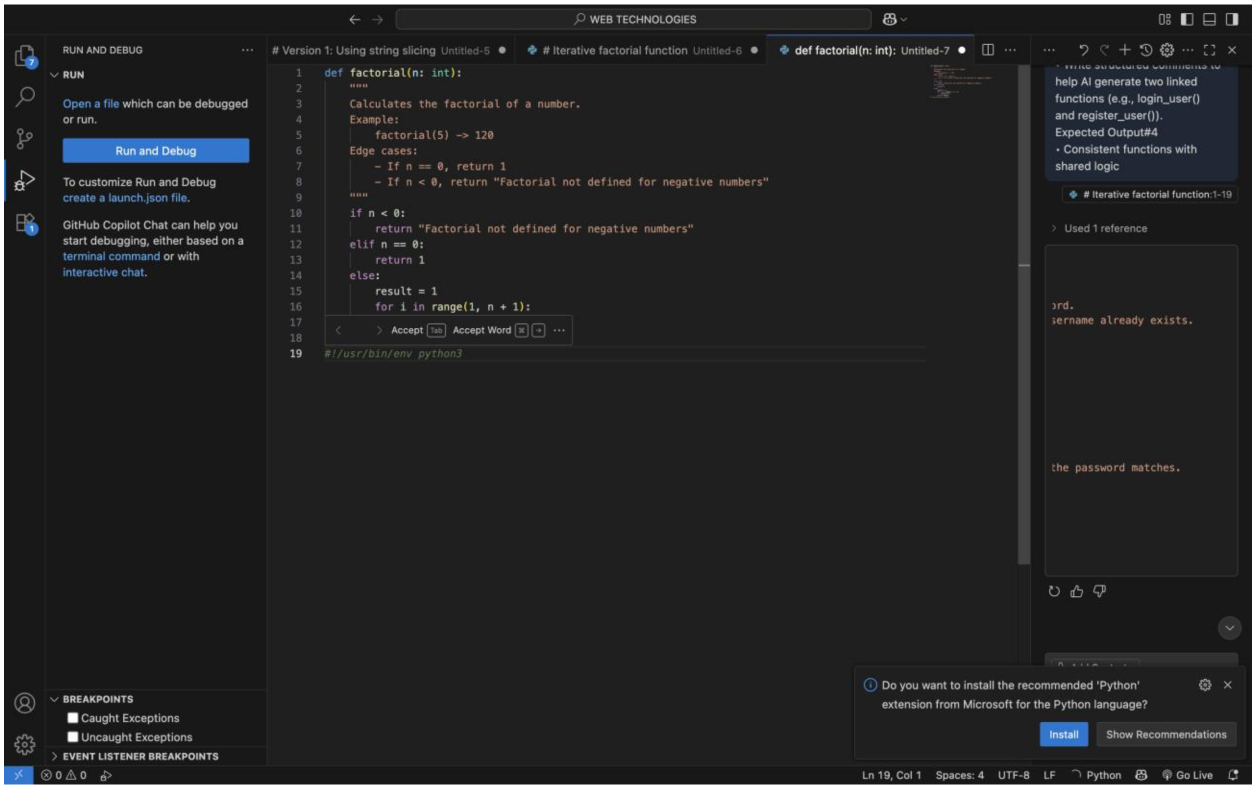
print(validate_mobile("812345678"))   # Invalid (only 9 digits)

True

```
False

False
```

Task #2 – One-Shot Prompting with Edge Case Handling
Objective
Use one-shot prompting to generate a Python function that calculates the factorial of a number.
Requirements
• Provide one sample input-output pair in the prompt to guide the AI.
• The function should handle:
o 0! correctly
o Negative input by returning an appropriate message
Expected Output
• A Python function with correct factorial logic and edge case handling, generated from a single example.



```python
def factorial(n: int):
    """
    Calculates the factorial of a number.
    Example:
        factorial(5) -> 120
    Edge cases:
        - If n == 0, return 1
        - If n < 0, return "Factorial not defined for negative numbers"
    """
    if n < 0:
        return "Factorial not defined for negative numbers"
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):


#!/usr/bin/env python3
```

help AI generate two linked functions (e.g., login_user() and register_user()).
Expected Output#4
• Consistent functions with shared logic

Iterative factorial function:1-19

Used 1 reference

```
ord.
sername already exists.

the password matches.
```

Do you want to install the recommended 'Python' extension from Microsoft for the Python language?

OUTPUT :

print(factorial(5))   # Normal case

print(factorial(0))   # 0!

print(factorial(-3))   # Negative number

120

1

Factorial not defined for negative numbers

Task #3 – Few-Shot Prompting for Nested Dictionary Extraction
Objective
Use few-shot prompting (2–3 examples) to instruct the AI to create a function that parses a nested dictionary representing student information.
Requirements

• The function should extract and return:

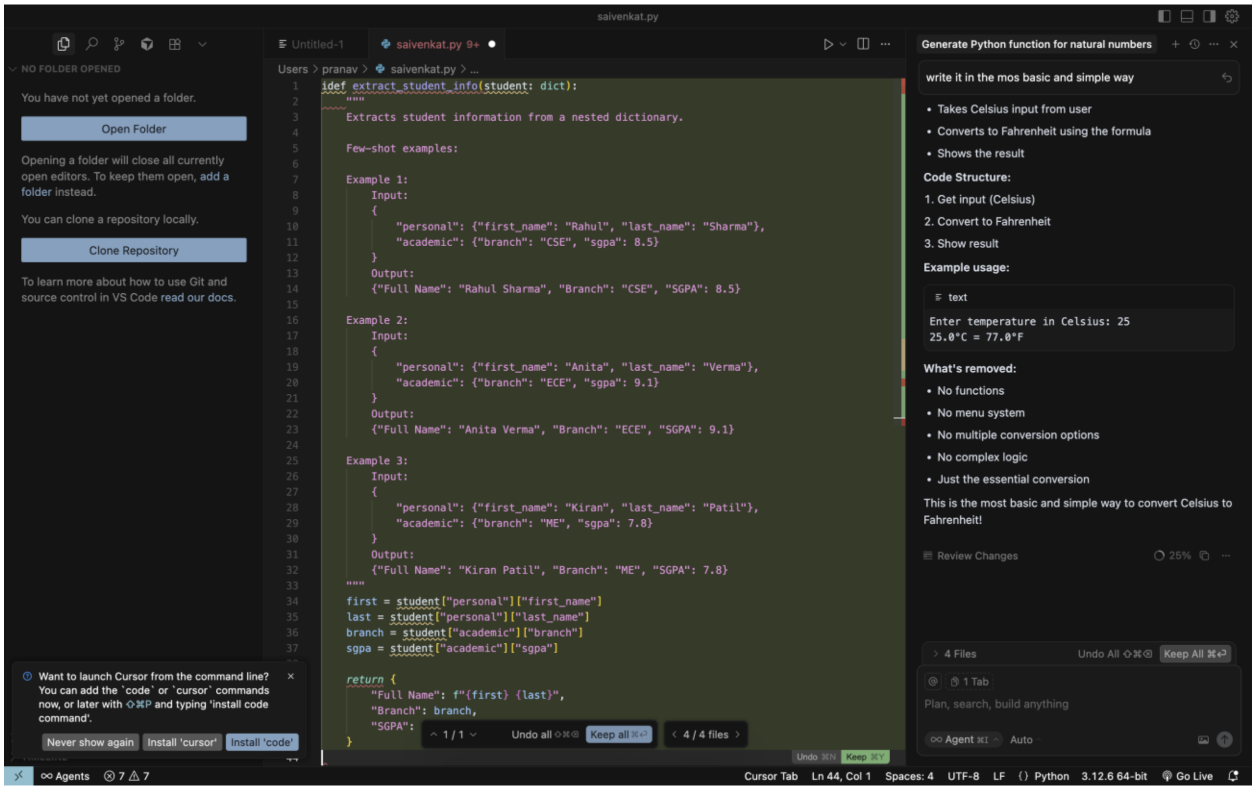
o Full Name

o Branch

o SGPA

Expected Output

• A reusable Python function that correctly navigates and extracts values from nested dictionaries based on the provided examples.



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

Want to launch Cursor from the command line? You can add the `code` or `cursor` commands now, or later with ⇧⌘P and typing 'install code command'.

```python
idef extract_student_info(student: dict):
    """
    Extracts student information from a nested dictionary.

    Few-shot examples:

    Example 1:
        Input:
        {
            "personal": {"first_name": "Rahul", "last_name": "Sharma"},
            "academic": {"branch": "CSE", "sgpa": 8.5}
        }
        Output:
        {"Full Name": "Rahul Sharma", "Branch": "CSE", "SGPA": 8.5}

    Example 2:
        Input:
        {
            "personal": {"first_name": "Anita", "last_name": "Verma"},
            "academic": {"branch": "ECE", "sgpa": 9.1}
        }
        Output:
        {"Full Name": "Anita Verma", "Branch": "ECE", "SGPA": 9.1}

    Example 3:
        Input:
        {
            "personal": {"first_name": "Kiran", "last_name": "Patil"},
            "academic": {"branch": "ME", "sgpa": 7.8}
        }
        Output:
        {"Full Name": "Kiran Patil", "Branch": "ME", "SGPA": 7.8}
    """
    first = student["personal"]["first_name"]
    last = student["personal"]["last_name"]
    branch = student["academic"]["branch"]
    sgpa = student["academic"]["sgpa"]

    return {
        "Full Name": f"{first} {last}",
        "Branch": branch,
        "SGPA":
    }
```

write it in the mos basic and simple way

- Takes Celsius input from user
- Converts to Fahrenheit using the formula
- Shows the result

Code Structure:

1. Get input (Celsius)
2. Convert to Fahrenheit
3. Show result

Example usage:

```text
Enter temperature in Celsius: 25
25.0°C = 77.0°F
```

What's removed:

- No functions
- No menu system
- No multiple conversion options
- No complex logic
- Just the essential conversion

This is the most basic and simple way to convert Celsius to Fahrenheit!

OUTPUT :

student = { "personal": {"first_name": "Rahul", "last_name": "Sharma"}, "academic": {"branch": "CSE", "sgpa": 8.5} } print(extract_student_info(student))

{'Full Name': 'Rahul Sharma', 'Branch': 'CSE', 'SGPA': 8.5}

Task #4 – Comparing Prompting Styles for File Analysis

Objective

Experiment with zero-shot, one-shot, and few-shot prompting to generate functions for CSV file analysis.

Requirements

• Each generated function should:

o Read a .csv file

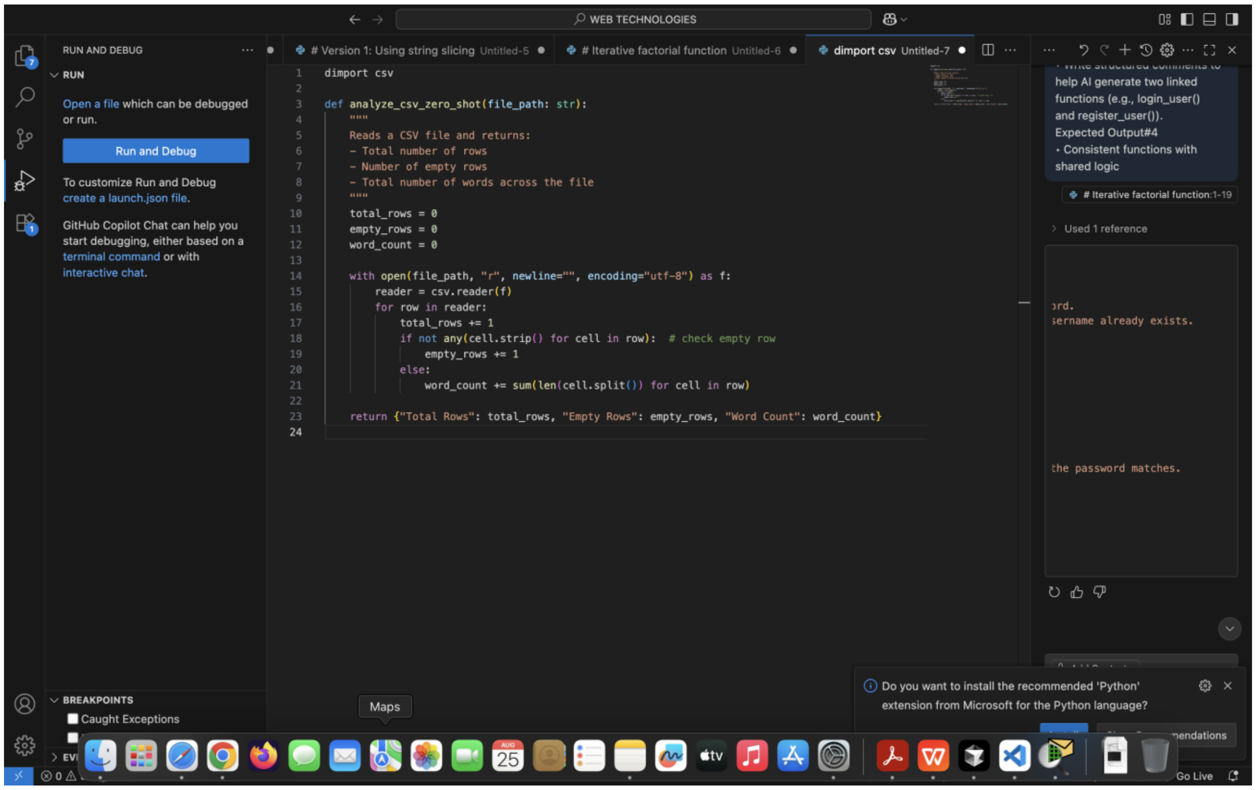
o Return the total number of rows

o Count the number of empty rows

o Count the number of words across the file

Expected Output

• Working Python functions for each prompting style, with a brief reflection comparing their accuracy, clarity, and efficiency.



Open a file which can be debugged or run.

To customize Run and Debug create a launch.json file.

GitHub Copilot Chat can help you start debugging, either based on a terminal command or with interactive chat.

```
dimport csv

def analyze_csv_zero_shot(file_path: str):
    """
    Reads a CSV file and returns:
    - Total number of rows
    - Number of empty rows
    - Total number of words across the file
    """
    total_rows = 0
    empty_rows = 0
    word_count = 0

    with open(file_path, "r", newline="", encoding="utf-8") as f:
        reader = csv.reader(f)
        for row in reader:
            total_rows += 1
            if not any(cell.strip() for cell in row):  # check empty row
                empty_rows += 1
            else:
                word_count += sum(len(cell.split()) for cell in row)

    return {"Total Rows": total_rows, "Empty Rows": empty_rows, "Word Count": word_count}
```

help AI generate two linked functions (e.g., login_user() and register_user()).
Expected Output#4
• Consistent functions with shared logic

Iterative factorial function:1-19

Used 1 reference

```
ord.
sername already exists.

the password matches.
```

Do you want to install the recommended 'Python' extension from Microsoft for the Python language?

Name, Age

Rahul, 21

Anita, 22

print(analyze_csv_zero_shot("sample.csv"))

print(analyze_csv_one_shot("sample.csv"))

print(analyze_csv_few_shot("sample.csv"))

{'Total Rows': 4, 'Empty Rows': 1, 'Word Count': 5}

Task #5 – Few-Shot Prompting for Text Processing and Word Frequency

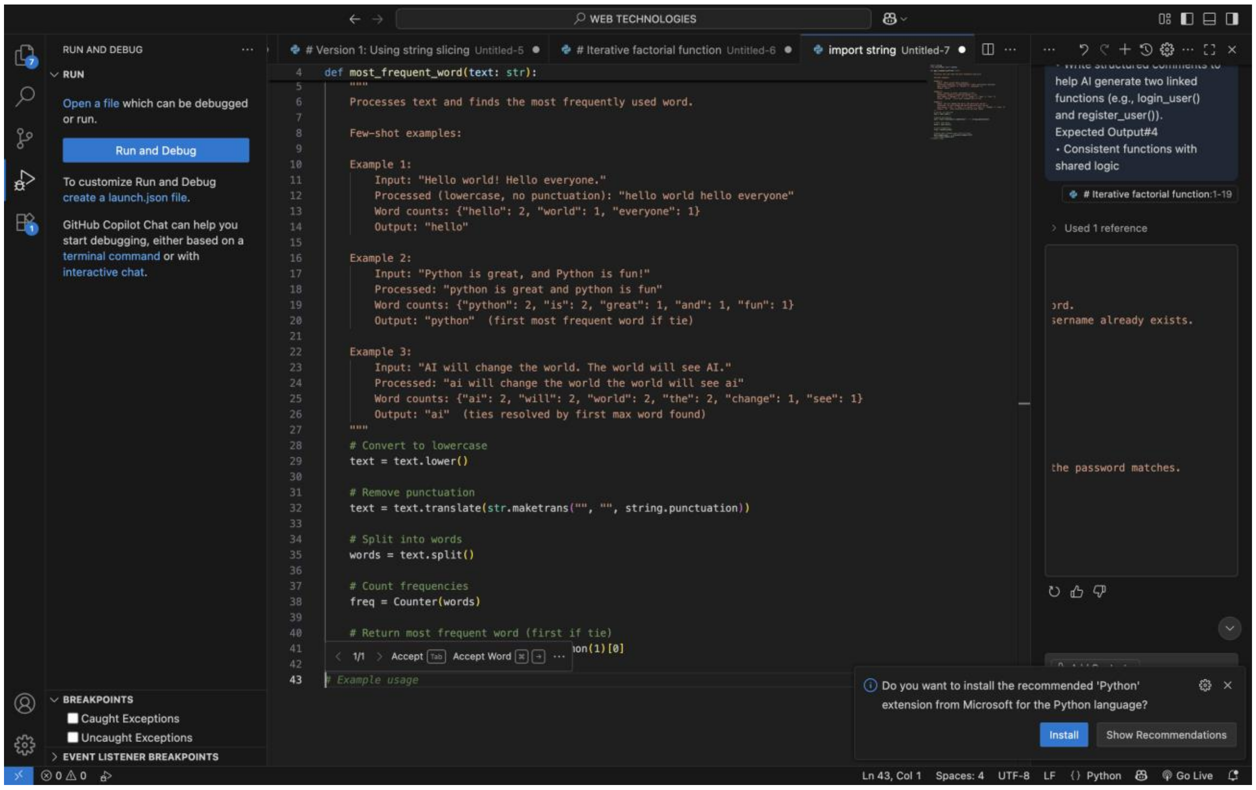
Objective

Use few-shot prompting (with at least 3 examples) to generate a Python function that processes text and analyzes word frequency.

Requirements

The function must:

- Accept a paragraph as input
- Convert all text to lowercase
- Remove punctuation
- Return the most frequently used word


WEB TECHNOLOGIES
RUN AND DEBUG
RUN
Open a file which can be debugged or run.
Run and Debug
To customize Run and Debug create a launch.json file.
GitHub Copilot Chat can help you start debugging, either based on a terminal command or with interactive chat.
BREAKPOINTS
Caught Exceptions
Uncaught Exceptions
EVENT LISTENER BREAKPOINTS
# Version 1: Using string slicing Untitled-5
# Iterative factorial function Untitled-6
import string Untitled-7
# Iterative factorial function:1-19
Used 1 reference
Accept Tab
Accept Word
Install
Show Recommendations
Ln 43, Col 1 Spaces: 4 UTF-8 LF Python Go Live

```python
def most_frequent_word(text: str):
    """
    Processes text and finds the most frequently used word.

    Few-shot examples:

    Example 1:
        Input: "Hello world! Hello everyone."
        Processed (lowercase, no punctuation): "hello world hello everyone"
        Word counts: {"hello": 2, "world": 1, "everyone": 1}
        Output: "hello"

    Example 2:
        Input: "Python is great, and Python is fun!"
        Processed: "python is great and python is fun"
        Word counts: {"python": 2, "is": 2, "great": 1, "and": 1, "fun": 1}
        Output: "python"  (first most frequent word if tie)

    Example 3:
        Input: "AI will change the world. The world will see AI."
        Processed: "ai will change the world the world will see ai"
        Word counts: {"ai": 2, "will": 2, "world": 2, "the": 2, "change": 1, "see": 1}
        Output: "ai"  (ties resolved by first max word found)
    """
    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = text.translate(str.maketrans("", "", string.punctuation))

    # Split into words
    words = text.split()

    # Count frequencies
    freq = Counter(words)

    # Return most frequent word (first if tie)
                                        on(1)[0]

# Example usage
```

help AI generate two linked functions (e.g., login_user() and register_user()).
Expected Output#4
• Consistent functions with shared logic

ord.
ername already exists.

the password matches.

Do you want to install the recommended 'Python' extension from Microsoft for the Python language?

```
print(most_frequent_word("Hello world! Hello everyone."))
print(most_frequent_word("Python is great, and Python is fun!"))
print(most_frequent_word("AI will change the world. The world will see AI."))
```

```
hello
python
ai
```