practical1b.py

```python
import random


class PlayfairCipher:
    def __init__(self, keyword: str):
        self.keyword = keyword.upper().replace("J", "I")
        self.alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
        self.bogus = ""
        self.matrix = self._create_matrix()
        self._print_matrix()

    def _create_matrix(self) -> list:
        keyword = "".join(sorted(set(self.keyword), key=self.keyword.index))

        matrix_string = keyword + "".join(
            ch for ch in self.alphabet if ch not in keyword
        )

        return [list(matrix_string[i : i + 5]) for i in range(0, 25, 5)]

    def _prepare_text(self, text: str) -> str:
        text = text.upper().replace("J", "I")
        prepared_text = ""
        i = 0

        while i < len(text):
            if i + 1 < len(text) and text[i] == text[i + 1]:
                prepared_text += text[i] + self.bogus
                i += 1
            else:
                prepared_text += text[i]
                i += 1

        if len(prepared_text) % 2 != 0:
            prepared_text += self.bogus

        return prepared_text

    def _find_index(self, char1: str, char2: str) -> tuple:
        i1, j1 = [
            (i, row.index(char1)) for i, row in enumerate(self.matrix) if char1 in row
        ][0]
        i2, j2 = [
            (i, row.index(char2)) for i, row in enumerate(self.matrix) if char2 in row
        ][0]
        return i1, j1, i2, j2

    def _encrypt_pair(self, char1: str, char2: str) -> str:
        i1, j1, i2, j2 = self._find_index(char1, char2)
        if i1 == i2:
            return self.matrix[i1][(j1 + 1) % 5] + self.matrix[i2][(j2 + 1) % 5]
        elif j1 == j2:
            return self.matrix[(i1 + 1) % 5][j1] + self.matrix[(i2 + 1) % 5][j2]
        else:
            return self.matrix[i1][j2] + self.matrix[i2][j1]

    def _decrypt_pair(self, char1: str, char2: str) -> str:
        i1, j1, i2, j2 = self._find_index(char1, char2)
        if i1 == i2:
            return (
                self.matrix[i1][((j1 - 1) % 5) if (j1 - 1 > -1) else 4]
                + self.matrix[i2][((j2 - 1) % 5) if (j2 - 1 > -1) else 4]
            )
        elif j1 == j2:
            return (
                self.matrix[((i1 - 1) % 5) if (i1 - 1 > -1) else 4][j1]
                + self.matrix[((i2 - 1) % 5) if (i2 - 1 > -1) else 4][j2]
            )
        else:
            return self.matrix[i1][j2] + self.matrix[i2][j1]

    def encrypt(self, plaintext: str) -> str:
        self.bogus = random.choice(
            "".join(ch for ch in self.alphabet if ch not in [*plaintext])
        )
        print(f"Bogus character used: {self.bogus}")
        prepared_text = self._prepare_text(plaintext)
        cipher_text = "".join(
            self._encrypt_pair(prepared_text[i], prepared_text[i + 1])
            for i in range(0, len(prepared_text), 2)
        )
        return cipher_text

    def decrypt(self, ciphertext: str) -> str:
        self.bogus = random.choice(
            "".join(ch for ch in self.alphabet if ch not in [*ciphertext])
        )
        print(f"Bogus character used: {self.bogus}")
        prepared_text = self._prepare_text(ciphertext)
        plain_text = "".join(
            self._decrypt_pair(prepared_text[i], prepared_text[i + 1])
            for i in range(0, len(prepared_text), 2)
        )
        return plain_text.lower().replace("i", "j")

    def _print_matrix(self) -> None:
        print("Matrix:")
        for row in self.matrix:
            print(" ".join(row))


def main() -> None:
    keyword = input("Enter Your KEYWORD in capital: ")
    cipher = PlayfairCipher(keyword)

    while True:
        input_string = input("Enter Your STRING: ").strip()
        if not input_string:
            print("Empty input string. Please enter a valid string.")
            continue

        if input_string.islower():
            result = cipher.encrypt(input_string)
            print("Encrypted:", result)
        elif input_string.isupper():
            result = cipher.decrypt(input_string)
            print("Decrypted:", result)
        else:
            print(
                "Invalid input. Please enter the string in either all lowercase for encryption or all uppercase for decryption."
            )


if __name__ == "__main__":
    main()
```