🆘 Morse Code De-/Encoder (Console)

This project is intended to:

- Practice the complete process from **problem analysis to implementation**
- Apply basic **Python** programming concepts learned in the Programming Foundations module
- Demonstrate the use of **console interaction, data validation, and file processing**
- Produce clean, well-structured, and documented code
- Prepare students for **teamwork and documentation** in later modules
- Use this repository as a starting point by importing it into your own GitHub account.
- Work only within your own copy — do not push to the original template.
- Commit regularly to track your progress.

📖 Documentation

📝 Analysis

Problem
People are curious about Morse code, but manually encoding and decoding messages is error-prone and tedious. There's a need for a simple tool that translates between text and Morse code quickly and reliably.

Scenario
A user wants to encode messages into Morse code to practice or send signals, or decode received Morse code messages. They use this application in a console, inputting either plain text or Morse code, and receive the corresponding translation instantly. The program also keeps a history of conversions for reference.

User stories:

1. As a user, I want to enter text and receive the Morse code translation.
2. As a user, I want to enter Morse code and get the corresponding text.
3. As a user, I want to see error messages if I input invalid Morse code.
4. As a user, I want all conversions to be saved automatically in a JSON history file.

Use cases:

- Encode text to Morse code
- Decode Morse code to text
- Save conversion history to `morse_history.json`
- Display meaningful error messages for invalid input

☑️ Project Requirements

Each app must meet the following three criteria in order to be accepted:

1. Interactive app (console input)
2. Data validation (input checking)

3. File processing (read/write)

---

1. Interactive App (Console Input)

The application interacts with the user via the console. Users can:

- Choose to encode text to Morse code or decode Morse code to text
- Input text or Morse code
- View the translation immediately in the console
- Exit the program

```
print("\nBitte wähle eine Option:")
print("1 = Text →  Morse")
print("2 = Morse →  Text")
print("q = Beenden")

choice = input("\nDeine Wahl: ").strip()
```

The program now uses **nested input loops** so that invalid input in a chosen mode does **not return to the main menu** immediately, but instead asks the user again until valid input is provided.

---

2. Data Validation

The application validates all user input to ensure correctness:

- **Empty input handling:** Skips processing if the user enters nothing.

```
if not text:
    print("Bitte Text eingeben!")
```

```
if not morse:
    print("Bitte Morse-Code eingeben!")
```

- **User input validation:** Checks each Morse code sequence or character depending on the chosen mode and prints a warning for invalid inputs.

```
if char in TO_MORSE_DICT:
    encoded_chars.append(TO_MORSE_DICT[char])
else:
    print(f"Fehler: '{char}' kann nicht in Morse-Code dargestellt werden!")
    return None
```

```
if code in TO_TEXT_DICT:
    decoded_chars.append(TO_TEXT_DICT[code])
else:
    print(f"Fehler: '{code}' ist kein gültiger Morse-Code!")
    return None
```

- **Menu choice validation:** Ensures that only valid options (1, 2, q) are processed.

```
if not choice:
    print("Keine Eingabe, bitte nochmal.")
    continue
```

```
if choice == "1":
    ...

elif choice == "2":
    ...

elif choice.lower() == "q":
    ...

else:
    print("Ungültige Eingabe, bitte nochmal.")
```

These checks prevent crashes and guide the user to provide correct input, fulfilling the validation requirement.

---

3. File Processing

The program reads and writes conversion history using a JSON file:

- **Output file:** `morse_history.json` — stores a history of all conversions with timestamps, input, and output.

```
[
    {
        "input": "SOS",
        "output": "... --- ...",
        "timestamp": "2025-10-01T14:57:17"
    }
]
```

- Reading the JSON file checks for existing data and prevents errors with corrupted files:

```
try:
    data = json.load(f)
except json.JSONDecodeError:
    data = []
```

- Writing appends new entries and ensures proper formatting.

⚙️ Implementation

Technology

- Python 3.x
- Environment: Any IDE or GitHub Codespaces
- No external libraries required (only Python libraries)

🗂 Repository Structure

```
3-25HS.W-B-WI-GrPro_morse-code-converter/
├── .gitignore               # files git should ignore
├── main.py                  # main program logic (console application)
├── morse_history.json       # JSON file storing conversion history
└── README.md                # project description and documentation
```

How to Run

1. Open the repository in your IDE or terminal
2. Run the program:

```
py main.py
```

3. Follow the on-screen prompts to encode or decode messages.

Libraries Used

- `json`: For reading/writing the conversion history
- `pathlib`: For handling the JSON file path
- `datetime`: For adding timestamps to each conversion

👥 Team & Contributions

Name	Contribution
Janis Huser	Implemented text-to-Morse encoding

Name	Contribution
Fabian Jäggi	Implemented Morse-to-text decoding
Dominik Suter	File handling, saving conversion history & error handling

📝 License