```python
#!/usr/bin/env python3
"""
base128 - koduje i dekoduje plik tekstowy w formacie Base128.

Port kodu Go (base128.go) na Python, zachowujący identyczne zachowanie
CLI, komunikaty błędów i kody wyjścia.
"""

import sys

BASE = 128


def encode_base128(text: str) -> list[int]:
    """Koduje tekst na listę liczb całkowitych (wartości znaków 0-127)."""
    result = []
    for char in text:
        code = ord(char)
        if code >= BASE:
            print(
                f"base128: znak '{char}' (kod {code}) jest poza zakresem "
                f"Base128 (0-127)",
                file=sys.stderr,
            )
            sys.exit(1)
        result.append(code)
    return result


def decode_base128(numbers: list[int]) -> str:
    """Dekoduje listę liczb całkowitych z powrotem na tekst."""
    chars = []
    for i, code in enumerate(numbers):
        if code < 0 or code >= BASE:
            print(
                f"base128: liczba na pozycji {i} ({code}) jest poza "
                f"zakresem Base128 (0-127)",
                file=sys.stderr,
            )
            sys.exit(1)
        chars.append(chr(code))
    return "".join(chars)

def print_help() -> None:
    BLUE = "\033[34m"
    GREEN = "\033[32m"
    RED = "\033[31m"
    WHITE = "\033[37m"
    PURPLE = "\033[35m"
    ORANGE = "\033[91m"
    YELLOW = "\033[33m"
    RESET = "\033[0m"

    lines = [
        BLUE + r"  __________\_____      ______ ____ ____ _____      ____ ________   ______  " + RESET,
        BLUE + r"  \_____   /  _  \    /     /  _  \|    |   \_____  \    /  _  \ " + RESET,
        BLUE + r"   |    |  _/_\  \  /  ___/  /_\  \|    |   |  |   _/   /  ___/  >  __  < " + RESET,
        BLUE + r"   |    |   \  \____ \  \___\    \|    |___|  |  /    \  \___ \ /  |  | " + RESET,
        BLUE + r"   |_____ /\_____  /\____ >\____ >|_____ \_____/ \_____ /\____ >\_____ /" + RESET,
        BLUE + r"         \/      \/     \/     \/      \/            \/     \/      v2.1\/" + RESET,
        "",
        GREEN + "base128"  + WHITE + " Koduje i dekoduje PLIK w formacie base128." + RESET,
        GREEN + "base128"  + WHITE + " [OPCJA].... [PLIK wejściowy] > [PLIK wyjściowy]" + RESET,
        "",
        GREEN + "PRZYKŁAD kodowania  :"  + WHITE + "  base128 -e tekst.txt > zakodowany.txt" + RESET,
        GREEN + "PRZYKŁAD dekodowania:" + WHITE + " base128 -d zakodowany.txt > odkodowany.txt" +
RESET,
        "",
        RED + "Opcje:" + RESET,
        "",
        RED + "  -d, --decode" + WHITE + "            Odkodowuje dane zakodowane wcześniej za pomocą
Base128." + RESET,
        RED + "  -e, --encode" + WHITE + "            Koduje dane wejściowe do formatu Base128." +
RESET,
        RED + "  -h, --help" + WHITE + "              Wyświetla instrukcje wywołania i listę dostępnych
opcji." + RESET,
        RED + "      --preserve-newlines" + WHITE + " Zachowuje znaki nowej linii podczas kodowania i
dekodowania." + RESET,
        RED + "      --version" + WHITE + "           Wyświetla wersję programu." + RESET,
        RED + "      --copyright" + WHITE + "         Wyświetla informacje o prawach autorskich." +
RESET,
        "",
        PURPLE + " (c) by Lukasz Wojcik 2023, 2025, 2026"+ RESET,
        YELLOW + " https://www.base128.pl"+ RESET,
    ]
    print("\n".join(lines))


def parse_args(argv: list[str]):
    """Prosty parser flag, odzwierciedlający zachowanie pakietu flag z Go."""
    encode_flag = False
    decode_flag = False
    help_flag = False
    version_flag = False
    copyright_flag = False
    preserve_newlines = False
    positional: list[str] = []

    flag_map = {
        "-e": "encode", "--encode": "encode",
        "-d": "decode", "--decode": "decode",
        "-h": "help", "--help": "help",
        "--version": "version",
        "--copyright": "copyright",
        "--preserve-newlines": "preserve_newlines",
    }

    for arg in argv:
        if arg in flag_map:
            name = flag_map[arg]
            if name == "encode":
                encode_flag = True
            elif name == "decode":
                decode_flag = True
            elif name == "help":
                help_flag = True
            elif name == "version":
                version_flag = True
            elif name == "copyright":
                copyright_flag = True
            elif name == "preserve_newlines":
                preserve_newlines = True
        else:
            positional.append(arg)

    return {
        "encode": encode_flag,
        "decode": decode_flag,
        "help": help_flag,
        "version": version_flag,
        "copyright": copyright_flag,
        "preserve_newlines": preserve_newlines,
        "args": positional,
    }


def main() -> None:
    opts = parse_args(sys.argv[1:])

    if opts["help"]:
        print_help()
        sys.exit(0)

    if opts["version"]:
        print("base128 wersja 2.1/2026")
        sys.exit(0)

    if opts["copyright"]:
        print("base128 (C) 2023, 2025, 2026 by Lukasz Wojcik")
        sys.exit(0)

    if opts["encode"] and opts["decode"]:
        print(
            "base128: nie można używać jednocześnie flag -e i -d. "
            "Proszę użyć -e do KODOWANIA lub -d do DEKODOWANIA lub -h "
            "dla POMOCY",
            file=sys.stderr,
        )
        sys.exit(1)

    if not opts["encode"] and not opts["decode"]:
        print(
            "base128: nie określono działania. Proszę użyć -e do "
            "KODOWANIA lub -d do DEKODOWANIA lub -h dla POMOCY",
            file=sys.stderr,
        )
        sys.exit(1)

    # Odczyt danych wejściowych z pliku lub standardowego wejścia.
    args = opts["args"]
    if len(args) == 0 or args[0] == "-":
        try:
            input_text = sys.stdin.read()
        except OSError as err:
            print(f"base128: błąd odczytu ze stdin: {err}", file=sys.stderr)
            sys.exit(1)
    else:
        try:
            with open(args[0], "r", encoding="utf-8", newline="") as f:
                input_text = f.read()
        except OSError as err:
            print(f"base128: błąd odczytu pliku: {err}", file=sys.stderr)
            sys.exit(1)

    if opts["encode"]:
        if not opts["preserve_newlines"]:
            input_text = input_text.replace("\n", "").replace("\r", "")
            input_text = input_text.strip()

        numbers = encode_base128(input_text)
        output = " ".join(str(n) for n in numbers)
        sys.stdout.write(output)

    elif opts["decode"]:
        input_text = input_text.strip()
        if len(input_text) == 0:
            print("base128: brak danych do dekodowania", file=sys.stderr)
            sys.exit(1)

        numbers = []
        for number_str in input_text.split(" "):
            number_str = number_str.strip()
            if number_str == "":
                continue
            try:
                number = int(number_str)
            except ValueError as err:
                print(f"base128: błąd dekodowania liczby: {err}", file=sys.stderr)
                sys.exit(1)
            numbers.append(number)

        output = decode_base128(numbers)
        sys.stdout.write(output)


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