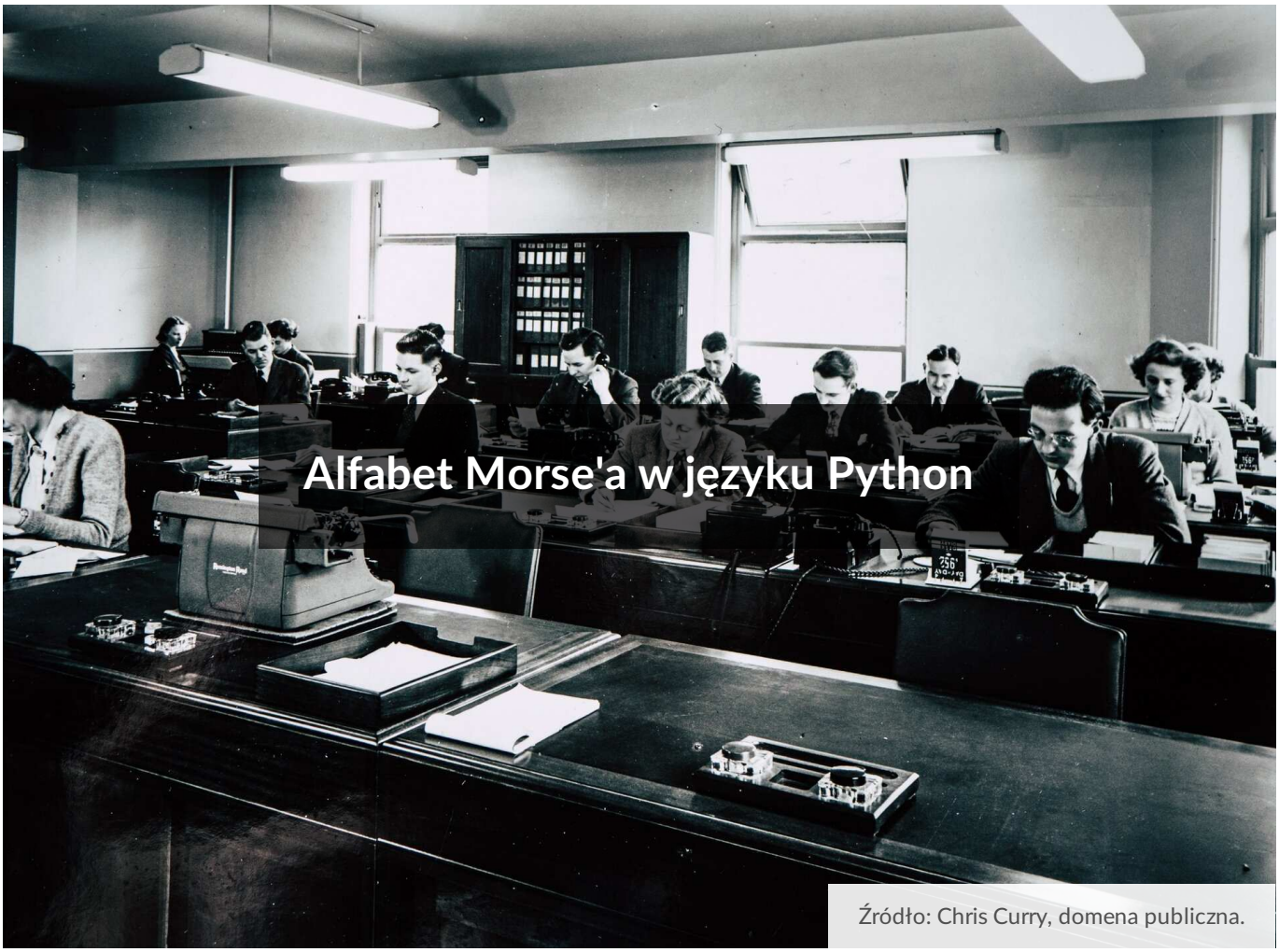
Alfabet Morse'a w języku Python



Alfabet Morse'a w języku Python

Źródło: Chris Curry, domena publiczna.

W e-materiale Alfabet Morse'a przedstawiliśmy najważniejsze informacje dotyczące alfabetu Morse'a. W tym e-materiale zajmiemy się jego implementację w języku Python.

Implementację tego szyfru w pozostałych językach programowania znajdziesz w e-materiałach:

- Alfabet Morse'a w języku C++,
- Alfabet Morse'a w języku Java.

Więcej zadań? Przejdź do e-materiału Alfabet Morse'a – zadania maturalne.

Twoje cele

- Wykorzystasz zaawansowane struktury danych do przechowywania informacji.
- Napiszesz program wykorzystujący klasy i metody niezbędne do kodowania i rozkodowywania tekstu.
- Utworzysz na stronie internetowej formularz służący do kodowania i dekodowania tekstu.

Przeczytaj

Kod Morse'a to sposób reprezentacji liter, cyfr i znaków specjalnych za pomocą dźwięków, błysków światła lub innych znaków, określanych jako kropka i kreska. Nadawanie odbywa się za pomocą kilkuelementowych serii krótkich lub długich znaków, przy czym jeden znak długi powinien trwać co najmniej tyle, ile trzy znaki krótkie. Takie określenie składowych oraz czasu trwania każdego ze znaków pozwala na nadanie zakodowanego sygnału przy użyciu zaimprowizowanych środków.

W alfabecie Morse'a długość kodu znaku wynika z częstotliwości jego występowania w słowach w języku angielskim. Im częściej jest używana litera, tym ma krótszy kod. W tabeli zaprezentowano znaki kodu Morse'a dla liter alfabetu łacińskiego:

Znak	Kod Morse'a	Znak	Kod Morse'a
A	• —	B	— • • •
C	— • — •	D	— • •
E	•	F	• • — •
G	— — •	H	• • • •
I	• •	J	• — — —
K	— • —	L	• — • •
M	— —	N	— •
O	— — —	P	• — — •
Q	— — • —	R	• — •
S	• • •	T	—
U	• • —	V	• • • —
W	• — —	X	— • • —
Y	— • — —	Z	— — • •

Polecenie 1

Określ, jaka litera kryje się pod zapisem dźwiękowym. Do pobrania plik dźwiękowy w formacie **ogg**.

Źródło: JoeDeRose, dostępny w internecie: pl.wikipedia.org [dostęp 10.09.2021], domena publiczna.
Plik o rozmiarze 10.00 KB w języku polskim

> **Ważne!**
>
> W oficjalnym opisie sygnalizacji kodem Morse'a odstępy między znakami są sygnalizowane długością przerwy. Na nasze potrzeby (generowania znaków) posłużymy się znakiem # jako separatorem znaków.

Ćwiczenie 1

Napisz program, który podany ciąg wejściowy przedstawi w formie kodu Morse'a, bądź odkoduje napis zapisany kodem Morse'a. Aby ugruntować wiedzę o programowaniu obiektowym, użyj odpowiednio zdefiniowanej klasy.

Specyfikacja:

Dane:

- *alfabet_morse* – słownik; kluczem jest napis składający się z jednego znaku (wielkiej litery pochodzącej z polskiego alfabetu); wartością jest napis składający się ze znaków: „-”, „.”; zapis znaku-klucza w alfabecie Morse'a

- *tekst* – ciąg znaków do zakodowania/odkodowania składający się z liter alfabetu polskiego, cyfr lub znaków specjalnych

Wynik:

- Na standardowym wyjściu program wypisuje, w zależności od trybu pracy, zakodowany lub odkodowany tekst.

Zdefiniujmy klasę z polem zawierającym słownik niezbędny do zmiany znaków ASCII na kod Morse'a i odwrotnie. Napiszmy kod, który sprawdzi, czy wprowadzony napis jest zapisany kodem Morse'a – w zależności od wyniku w trakcie inicjacji obiektu wykonana zostanie odpowiednia metoda. Posłużymy się zapisem z tzw. type hints.

```
class Morse_Code:

    alfabet_morse = {
        "A": ".-",
        "B": "-...",
        "C": "-.-.",
        "D": "-..",
        "E": ".",
        "F": "..-.",
        "G": "--.",
        "H": "....",
```

```
        "I": "..",
        "J": ".---",
        "K": "-.-",
        "L": ".-..",
        "M": "--",
        "N": "-.",
        "O": "---",
        "P": ".--.",
        "Q": "--.-",
        "R": ".-.",
        "S": "...",
        "T": "-",
        "U": "..-",
        "V": "...-",
        "W": ".--",
        "X": "-..-",
        "Y": "-.--",
        "Z": "--..",
        "Ą": ".-.-",
        "Ć": "-.-..",
        "Ę": "..-..",
        "Ł": ".-..-",
        "Ń": "--.--",
        "Ó": "---.",
        "Ś": "...-...",
        "Ż": "--..-.",
        "Ź": "--..-",
    }

    def __init__(self, ciag_wejsciowy: str) -> None:
        self.ciag_wejsciowy = ciag_wejsciowy.upper()
        self.ciag_znakow = None
        self.ciag_kodow = None
        if self.ciag_wejsciowy.isalpha():
            self.kierunek = "ASCII-Morse"
            self.ciag_znakow = self.ciag_wejsciowy
            self.zakoduj_morse()
        else:
            self.kierunek = "Morse-ASCII"
            self.ciag_kodow = self.ciag_wejsciowy
            self.odkoduj_morse()

```

```
    def zakoduj_morse(self) -> str:
        separator = "#"
        zakodowany_tekst = ""

        for znak in self.ciag_znakow:
            try:
                zakodowany_tekst += Morse_Code.alfabet_morse[z
            except KeyError:
                pass
        self.ciag_kodow = zakodowany_tekst

    def odkoduj_morse(self) -> str:

        def odk_znak(kod: str):
            for key, ind in Morse_Code.alfabet_morse.items():
                if kod == ind:
                    return key
            else:
                return ""

        lista_znakow = []
        lista_znakow = self.ciag_kodow.split('#')
        odkodowany = ""

        for znak in lista_znakow:
            odkodowany += odk_znak(znak)

        self.ciag_znakow = odkodowany

# przykład wywołania

kod_1 = Morse_Code("Linux")
print(kod_1.ciag_znakow)
print(kod_1.ciag_kodow)
print(kod_1.kierunek)

print()

kod_2 = Morse_Code("---#.--.#.#-.#...#---#..-#.-.#-.-.#.#")
print(kod_2.ciag_znakow)
print(kod_2.ciag_kodow)
print(kod_2.kierunek)
```

```
96
97 # efekt działania
98
99 LINUX
100 .-..#..#-.#..-#-..-#
101 ASCII-Morse
102
103 OPENSOURCE
104 ---#.--.#.#-.#...#---#..-#.-.#-.-.#.#
105 Morse-ASCII
```

Ćwiczenie 2

Napisz program, który dla podanego przez użytkownika, na stronie internetowej, ciągu wejściowego wygeneruje kod Morse'a.

Specyfikacja:

Dane:

- Wprowadzony przez użytkownika do formularza ciąg znaków.

Wynik:

- Program generuje stronę internetową z zakodowanym przy pomocy kodu Morse'a ciągiem wejściowym.

Do zbudowania strony użyjemy modułu Flask. Funkcjonalność strony będzie następująca:

- na stronie głównej będziemy mogli przez pole formularza podać tekst do zakodowania,
- poprzez URL "`/koduj/xxxxx`" będziemy mogli wywołać automatycznie zakodowanie ciągu `xxxxx`.

Przykład 1

Przygotujmy kod, który pozwoli zrealizować te założenia. W celu zmniejszenia objętości kodu założymy, że definicja klasy `Morse_Code` znajduje się w pliku o nazwie

`Morse_Code.py`, który jest zapisany w tym samym katalogu na dysku komputera.

```
from flask import Flask
# request - obiekt, który pozwala sprawdzić m.in. jakiego proto
from flask import request

# importujemy definicję klasy z pliku
from Morse_Code import *


# tworzymy główny obiekt aplikacji
app = Flask("Alfabet_Morse")

# definiujemy akcję związaną z URL: "/"
@app.route('/', methods = ['POST',"GET"])
def koduj_formularz():
    """Wyświetla formularz HTML dla metody GET, a dla metody PO
    post = """
    <h1> Kodowanie Morse'a </h1> <hr>
    <form method="POST" action="/">
    Wpisz tekst do zakodowania: <input type="text" name="ciag_z
    <input type="submit" value="Zakoduj">
    </form>"""

    if request.method == 'GET':
        return post
    else:
        kod = Morse_Code(request.form['ciag_znakow'])
        tekst = f"""
        <h1> Kodowanie Morse'a </h1> <hr>
        Podano do zakodowania ciąg znaków: <pre>{request.form['
        Zakodowany w kodach Morse'a: <br> <pre>{kod.ciag_kodow}
        """
        return tekst

# definiujemy akcję związaną z URL: "/koduj"
@app.route('/koduj/<name>', methods=['GET'])
def zakoduj_morse(name):
    """Wyświetla ciąg zapisany kodem Morse'a dla napisu podaneg
    kod = Morse_Code(name)
    tekst = f"""
```

```
40      <h1> Kodowanie Morse'a </h1> <hr>
41      Podano do zakodowania ciąg znaków: <pre>{name}</pre><br>
42      Zakodowany w kodach Morse'a: <br> <pre>{kod.ciag_kodow}</pr
43      """
44      return tekst
45
46  # start - uruchamiamy serwer WWW pod adresem http://127.0.0.1:5
47  app.run()
```

Dla zainteresowanych

Dla języka Python istnieje ciekawy moduł o nazwie morse-talk – pozwala on na kodowanie i rozkodowywanie znaków ASCII i Morse'a, jak również na graficzną reprezentację kodów Morse'a za pomocą biblioteki `matplotlib`.

Już wiesz

Podsumujmy:

- podczas pisania programu wykorzystaliśmy słownik, który umożliwił przypisanie kluczowi (literze alfabetu polskiego) odpowiednik zapisany w kodzie Morse'a,
- kodowanie Morse'a można uznać za pewien rodzaj szyfrowania,
- programowanie obiektowe pozwala tworzyć czytelniejszy kod, który może być wielokrotnie wykorzystywany; jest to ważne, gdy projekt tworzy wielu programistów.

Słownik

ASCII

(ang. *American Standard Code for Information Interchange*) pierwotnie siedmiobitowy system kodowania znaków (współcześnie rozszerzony do ośmiu bitów); w oryginalnej wersji kodom z zakresu 0–127 przyporządkowano 26 liter łacińskich, 10 cyfr (0...9) oraz dodatkowe znaki

Flask

moduł pozwalający szybko i efektywnie tworzyć strony internetowe; `Flask` nie jest dostępny w standardowej instalacji języka Python; należy go zainstalować, korzystając z mechanizmu `pip`

type hints

informacja o typie danego obiektu zapisana literalnie w kodzie programu; z uwagi na dynamiczną naturę języka Python pomaga w zrozumieniu kodu programu i w sprawdzeniu kodu przez różne programy IDE, np. PyCharm, SublimeText lub Atom; przykład użycia takiego zapisu: `zmienna: int = 10`; ***type hints*** zostały dokładnie

> opisane w dokumencie PEP 484, inną nazwą są ***annotations*** opisane w dokumencie PEP 3107

Animacja

> **Polecenie 1**
>
> Zapoznaj się z animacją. Z koleżanką lub kolegą przetestujcie, w czym pomaga ta mnemotechnika – odbiorze czy nadawaniu?

Film dostępny pod adresem /preview/resource/RPfCDTDrRyZp5

Film opowiadający o mnemotechnice pomagającej w zapamiętaniu alfabetu Morse'a.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Wskaż poprawny zapis, który definiuje słownik.

```
pusty_slownik = {}
```

```
pusty_slownik = { 1 = "Wartość 1", (2,) = ["1", "aa"]}
```

```
pusty_slownik[1] = "Wartość 1"
pusty_slownik[(2,)] = ["1", "aa"]
```

```
pusty_slownik = { [1]: "Wartość 1", (2,): ["1", "aa"]}
```

Ćwiczenie 2

Zapoznaj się z kodem. Napisz wynik działania programu – wartość wyświetlaną na standardowym wyjściu.

```
dane = {1:2,2:3,3:4,4:5,5:6}
print(dane[dane[3]])
```

Ćwiczenie 3

Zapoznaj się z kodem i wykonaj ćwiczenie.

```
dane = { 1: "Python ", 2: "=", 3: " super!" }
for i, j in dane.items():
    print(i*j, end="")
```

Wskaż, jaki będzie wynik działania programu.

- ◯ błąd `TypeError`
- ◯ `Python==super!super!super!`
- ◯ `Python == super! super! super!`

Ćwiczenie 4

Jan Kowalski kupił tydzień temu nową stację pogodową. Każdego dnia o północy stacja wysyła na komputer Jana informację ze średnią temperaturą minionego dnia. Jan zauważył, że niektóre z odebranych napisów nie przechowują liczb zmiennoprzecinkowych, lecz ciągi znaków składających się z kropek, kresek i symbolu kratki. Po krótkim śledztwie Janowi udało się ustalić, że pod enigmatycznymi symbolami kryje się średnia temperatura zapisana za pomocą kodu Morse'a. Napisz program, który pomoże Janowi zdekodować temperatury, a także policzyć średnią temperaturę minionego tygodnia. Program powinien zaokrąglić wynik do części setnej.

Przetestuj działanie programu dla następującej listy napisów: *['12.4', '−−.#.-.-.-#−−.', '15.1', '−..#.-.-.-#−−.', '.−−#−−−#.-.-.-#.−−', '11.1', '11.8']*.

Specyfikacja:

Dane:

temperatury – lista napisów

Wynik:

srednia – liczba rzeczywista zaokrąglona do dwóch miejsc po przecinku

Twoje zadania

1. Program liczy średnią temperaturę z tygodnia przedstawionego w tablicy `temperatury`.

```
alfabet_morse = {
    "1": ".----",
    "2": "..---",
    "3": "...--",
    "4": "....-",
    "5": ".....",
    "6": "-....",
    "7": "--...",
    "8": "---..",
```

```
    "9": "----.",
    "0": "-----",
    ".": ".-.-.-",
}
```

```
1
```

Dla nauczyciela

Autor: Adam Jurkiewicz
Przedmiot: Informatyka

Temat: Alfabet Morse'a w języku Python

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

IV. Rozwijanie kompetencji społecznych, takich jak: komunikacja i współpraca w grupie, w tym w środowiskach wirtualnych, udział w projektach zespołowych oraz zarządzanie projektami.

Treści nauczania – wymagania szczegółowe

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) dokonuje kompresji informacji, objaśnia różnice między kompresją stratną i bezstratną tekstów, obrazów, dźwięków, filmów;

IV. Rozwijanie kompetencji społecznych.

Zakres podstawowy. Uczeń:

1) aktywnie uczestniczy w realizacji projektów informatycznych rozwiązujących problemy z różnych dziedzin, przyjmuje przy tym różne role w zespole realizującym

projekt i prezentuje efekty wspólnej pracy;

V. Przestrzeganie prawa i zasad bezpieczeństwa.

Zakres podstawowy. Uczeń:

3) stosuje dobre praktyki w zakresie ochrony informacji wrażliwych (np. hasła, pin), danych i bezpieczeństwa systemu operacyjnego, objaśnia rolę szyfrowania informacji;

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) omawia znaczenie algorytmów szyfrowania i składania podpisu elektronicznego.

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Wykorzystasz zaawansowane struktury danych do przechowywania informacji.
- Napiszesz program wykorzystujący klasy i metody niezbędne do kodowania i rozkodowywania tekstu.
- Utworzysz na stronie internetowej formularz służący do kodowania i dekodowania tekstu.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Python 3 (lub nowszej wersji), w tym PyCharm lub IDLE.

Przebieg lekcji

Przed lekcją:

1. Nauczyciel loguje się na platformie i udostępnia uczniom e-materiał. Prosi o zapoznanie się z nim.

Faza wstępna:

1. Nauczyciel wyświetla temat i cele zajęć. Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel prosi wybranego ucznia lub uczniów o przedstawienie sytuacji problemowej związanej z tematem lekcji.

Faza realizacyjna:

1. Uczniowie analizują przykład z sekcji „Przeczytaj" i powtarzają zaprezentowane rozwiązanie na swoim komputerze.
2. Nauczyciel pozostawia wyświetloną zawartość sekcji „Przeczytaj", czyta treść polecenia nr 1: „Sprawdźmy, jaka litera kryje się pod zapisem dźwiękowym. Do pobrania plik dźwiękowy w formacie ogg". Prosi uczniów, aby w parach przeanalizowali rozwiązanie problemu. Wybrana para prezentuje wynik swojej pracy na forum klasy.
3. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Animacja". Uczniowie wykonują polecenie 1. Nauczyciel może zainicjować krótką dyskusję dotyczącą technik zapamiętywania.
4. Uczniowie w parach wykonują ćwiczenia nr 1-3 z sekcji „Sprawdź się". Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.
5. Nauczyciel dzieli klasę na grupy. Następnie każdej z grup przydziela sposób nadawania wiadomości alfabetem Morse'a (np. za pomocą rąk, latarek, stukania). Przedstawiciel grupy losuje tekst do zakodowania. Uczniowie ćwiczą kodowanie tekstu, a następnie wysyłają wiadomość. Zadaniem pozostałych grup jest jej odszyfrowanie.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat i cele lekcji zawarte w sekcji „Wprowadzenie". W kontekście ich realizacji następuje omówienie ewentualnych problemów z rozwiązaniem ćwiczeń z sekcji „Sprawdź się".

2. Na koniec zajęć z programowania w Pythonie nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…”.

Praca domowa:

1. Uczniowie wykonują ćwiczenie nr 4 z sekcji „Sprawdź się”.

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Treści w sekcji „Przeczytaj” można wykorzystać jako podsumowanie i utrwalenie wiedzy uczniów.