Szyfr Cezara w języku Python

Szyfr Cezara w języku Python

Źródło: Pixabay, domena publiczna.

Poznaliśmy już szyfr Cezara, czyli klasyczny szyfr przesuwający (podstawieniowy). Jego działanie polega na zastąpieniu każdej z liter tekstu jawnego odpowiadającą jej literą, oddaloną o określoną liczbę miejsc w alfabecie.

W tym e-materiale dowiesz się, w jaki sposób można napisać algorytm szyfru Cezara w języku Python.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych e-materiałach z tej serii:

- Implementacja szyfru Cezara w języku C++,
- Implementacja szyfru Cezara w języku Java.

Więcej zadań? Sięgnij do Szyfr Cezara – zadania maturalne.

> **Twoje cele**
>
> - Prześledzisz algorytm szyfrowania tekstu za pomocą szyfru Cezara.
> - Przeanalizujesz algorytm odszyfrowania tekstu utajnionego, za pomocą szyfru Cezara.
> - Napiszesz w języku Python program szyfrujący oraz odszyfrowujący tekst.
> - Wyjaśnisz działanie funkcji na pojedynczych znakach i na napisach.

Przeczytaj

Problem 1
Napisz program szyfrujący i deszyfrujący tekst przy użyciu szyfru Cezara. Przetestuj jego działanie dla wiadomości „JESIENNIE” i dla klucza równego 3.

Specyfikacja:

Dane:

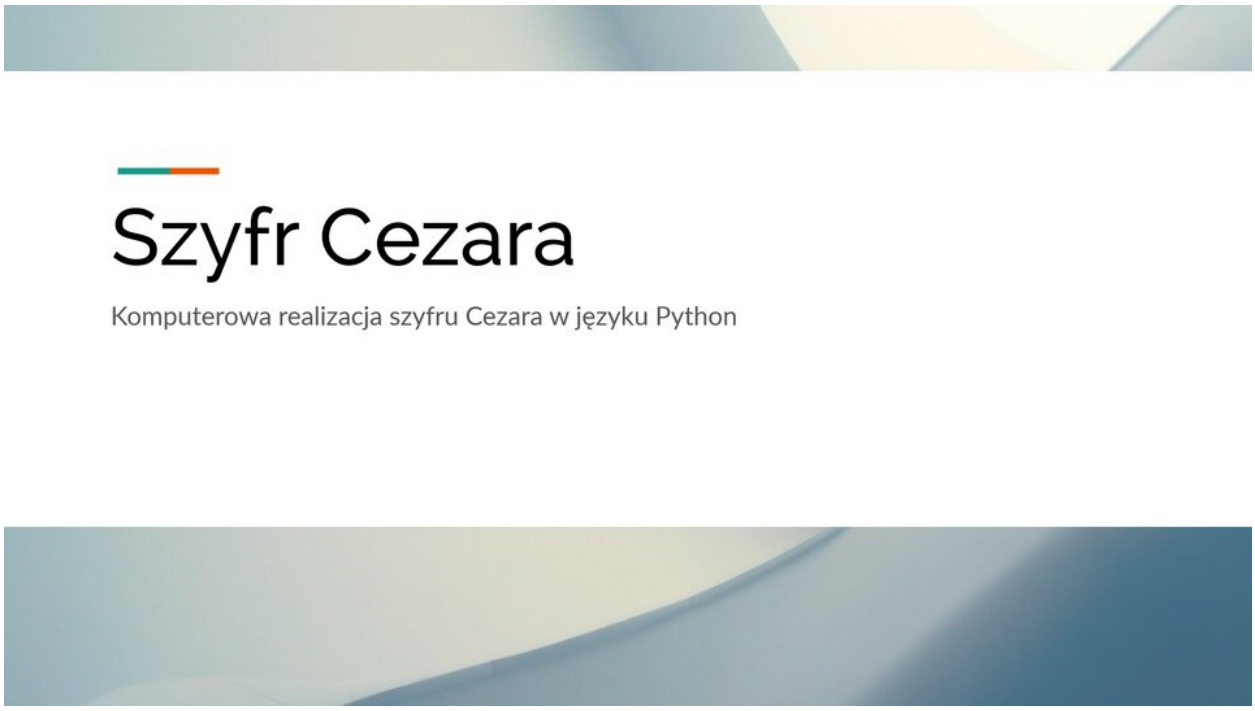
- `klucz` – liczba całkowita; wartość klucza szyfrowania
- `wiadomosc` – ciąg znaków; wiadomość do zaszyfrowania zapisana dużymi literami

Wynik:

Program na wyjściu standardowym zwróci w jednej linii wiadomość zaszyfrowaną, a w drugiej – deszyfrowaną.

Polecenie 1
Porównaj swoje rozwiązanie z przedstawionym w filmie.

Trwa wczytywanie danych ..



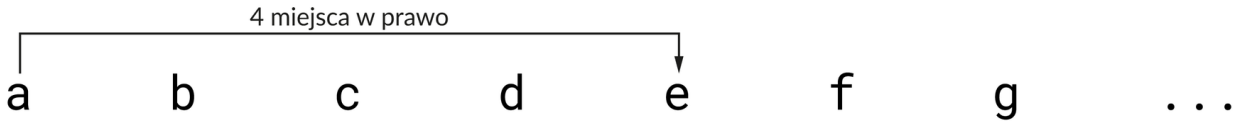
Film dostępny pod adresem /preview/resource/R18Dq6UwlViqt

Nagranie filmowe dotyczące szyfru Cezara - implementacji algorytmu szyfrowania i deszyfrowania wiadomości zapisanej alfabetem łacińskim.

Na czym polega szyfr Cezara?

Szyfr Cezara jest szyfrem podstawieniowym, służącym do utajniania tekstów. Oznacza to, że każda litera w szyfrowanym ciągu znaków zastępowana jest inną, oddaloną w alfabecie o pewną stałą liczbę miejsc. Odległość, o którą oddalone są litery zastępowana i zastępująca, nazywa się kluczem szyfru.

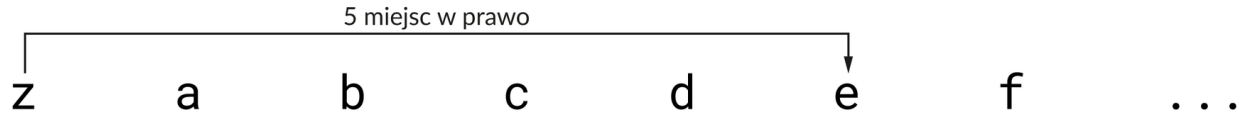
Posłużmy się przykładem. Przyjmijmy, że używając klucza równego 4 chcemy zaszyfrować literę „a". Musimy ją więc zastąpić literą położoną o 4 miejsca dalej w alfabecie:



Po zaszyfrowaniu litera „a” stanie się literą „e”. Tekst zapisany w taki sposób jest w stanie odkodować tylko osoba znająca metodę szyfrowania i rozmiar klucza.

Ponieważ alfabet łaciński składa się z 26 liter, a zastosowanie klucza równego zero nie powoduje zastąpienia jednych znaków innymi, możemy zakodować tekst tylko na 25 sposobów. W rezultacie szyfr Cezara jest dość łatwy do złamania, nawet gdy nie znamy użytego klucza.

Spróbujmy teraz zaszyfrować literę „z”, używając klucza 5. Ponieważ „z” jest ostatnią literą alfabetu, nie możemy jej zastąpić inną, położoną o choćby jedno miejsce dalej. W takiej sytuacji zaczynamy odliczanie od początku alfabetu:



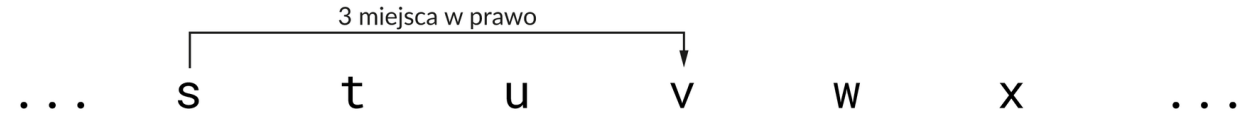
Litera „z” po zaszyfrowaniu zamieni się zatem w literę „e”.

Jak już wspominaliśmy, klucz może przybrać tylko 25 wartości, aby jedne litery w tekście zostały zastąpione innymi. Wynika to z liczby znaków alfabetu łacińskiego. Dlaczego jednak nie zastosować klucza o wartości większej niż 25?

Otóż jeżeli użyjemy klucza o wartości 27, tekst wciąż będzie szyfrowany, ale osiągniemy taki sam wynik jak w przypadku zastosowania klucza o wartości 1. Klucz równy 28 odpowiada kluczowi wynoszącemu 2 itd. W rezultacie mamy do dyspozycji tylko 25 kluczy.

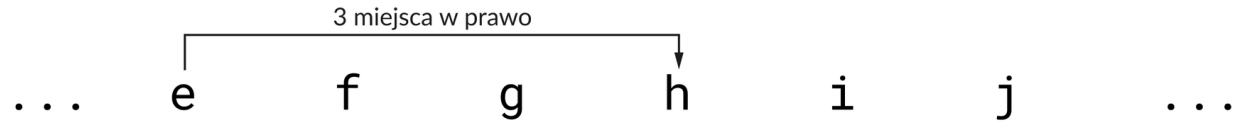
Szyfr Cezara – przykład zastosowania

Spróbujmy zaszyfrować słowo „ser”, używając klucza równego 3. Zaczniemy od pierwszej litery, czyli „s”:



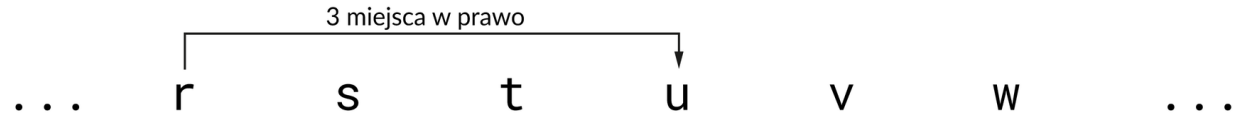
Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Litera „s” zostanie zastąpiona przez „v”. Następna jest litera „e”:



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

W tym przypadku nowym znakiem jest „h”. Pozostało nam jeszcze zaszyfrować literę „r”:



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Ostatecznie słowo „ser”, zakodowane za pomocą szyfru Cezara o kluczu równym 3, zmienia się w wyraz „vhu”. Jeżeli każdy jego znak zastąpimy literą o trzy pozycje wcześniejszą (czyli przesuniemy je „w lewo” o wartość klucza), odszyfrujemy ciąg, otrzymując ponownie słowo „ser”.

Szyfr Cezara – pseudokod

Skoro wiemy już jak działa szyfr Cezara, spróbujmy zapisać jego algorytm w pseudokodzie, który później przełożymy na język Python. Zaczniemy od zdefiniowania kilku zmiennych:

```
wyraz = "informatyka"
zaszyfrowanyWyraz = ""
klucz = 4
```

Następną czynnością jest odczytanie poszczególnych liter szyfrowanego wyrazu (musimy bowiem zakodować każdy znak osobno):

```
```

```
wyraz = "informatyka"
zaszyfrowanyWyraz = ""
klucz = 4

dla i = 0, 1, ..., długość(wyraz) wykonuj
```

W jaki sposób zastąpimy („przesuniemy w prawo") litery składające się na wyraz? Przypomnijmy sobie, czym jest kod ASCII. Każdy zapisany w tym systemie znak ma przyporządkowany odpowiednik liczbowy. Litery są uporządkowane alfabetycznie („a" ma przypisaną wartość 97, zaś „z" – 122). Wystarczy zatem dodać wartość klucza do liczby odpowiadającej literze w kodzie ASCII. Otrzymamy nowy kod, który odpowiada zaszyfrowanej literze.

Niestety, napotkamy wówczas pewien problem. Co się stanie, gdy – przykładowo – będziemy chcieli przesunąć literę „z" o 3 miejsca dalej?

Liczbowy zapis „z" w kodzie ASCII to 122. Po dodaniu do niego liczby 3, nie otrzymamy litery „c" – jej wartość w kodzie ASCII wynosi 99. Zamiast tego uzyskamy symbol } (prawy nawias klamrowy) – właśnie jemu w kodzie ASCII odpowiada liczba 125.

Aby rozwiązać taki problem, musimy najpierw odjąć od sumy (wynoszącej w opisywanym przypadku 125) wartość litery „a" w kodzie ASCII. Następnie obliczamy resztę z dzielenia otrzymanej różnicy przez 26. Na koniec do wyniku dodajemy ponownie wartość litery „a" w kodzie ASCII.

W rezultacie po otrzymaniu kodu ASCII spoza zakresu odpowiadającego literom łacińskim, znak zostanie przeniesiony na początek alfabetu:

```
wyraz = "informatyka"
zaszyfrowanyWyraz = ""
klucz = 4

dla i = 0, 1, ..., długość(wyraz) wykonuj
    przesunietaLitera = wartośćASCII(wyraz[i]) + klucz
```

```
    poprawkaNaKoniecAlfabetu = (przesunietaLitera - wartośćASCII(
    zaszyfrowanaLitera = znakZASCII(poprawkaNaKoniecAlfabetu + wa
```

> W omówionym pseudokodzie pojawiają się dwie funkcje: `wartośćASCII()` i `znakZASCII()`. Pierwsza z nich zwraca wartość kodu ASCII podanego znaku. Druga zwraca znak o podanym kodzie ASCII.

Pozostaje jeszcze zastosować konkatenację w celu dopisania zaszyfrowanej litery do wyrazu:

```
wyraz = "informatyka"
zaszyfrowanyWyraz = ""
klucz = 4

dla i = 0, 1, ..., długość(wyraz) wykonuj
    przesunietaLitera = wartośćASCII(wyraz[i]) + klucz
    poprawkaNaKoniecAlfabetu = (przesunietaLitera - wartośćASCII(
    zaszyfrowanaLitera = znakZASCII(poprawkaNaKoniecAlfabetu + wa
    zaszyfrowanyWyraz = zaszyfrowanyWyraz + zaszyfrowanaLitera
```

W kolejnej sekcji przełożymy pseudokod na instrukcje w języku Python.

Słownik

> **kod ASCII**
>
> 7-bitowy system kodowania znaków, w którym każdy z obsługiwanych symboli jest reprezentowany przez liczbę; 7 bitów umożliwia przechowanie informacji o znakach o kodach z zakresu 0-127; używany m.in. we współczesnych komputerach oraz sieciach komputerowych

> **konkatenacja**
>
> łączenie ze sobą łańcuchów znaków

kryptoanaliza

analiza systemu kryptograficznego w celu wyciągnięcia informacji przez niego utajnionych

kryptografia

nauka zajmująca się szyfrowaniem, czyli zamianą treści jawnej i dostępnej publicznie, na treść tajną, która może zostać odczytana tylko przez upoważnione jednostki

szyfr Cezara

metoda utajniania tekstu, polegająca na zmianie liter z tekstu na litery położone w alfabecie określoną liczbę miejsc dalej

Prezentacja multimedialna

Problem 1
Napisz program szyfrujący podany tekst (przy ustalonym kluczu szyfrowania). Przetestuj jego działanie dla wyrazu `informatyka` oraz klucza o wartości 4.

Specyfikacja:

Dane:

- `klucz` – liczba całkowita; wartość klucza szyfrowania
- `wyraz` – ciąg znaków; wiadomość do zaszyfrowania zapisana małymi literami

Wynik:

Program na wyjściu standardowym wyświetli tekst zaszyfrowany przy pomocy ustalonego klucza szyfrowania.

Polecenie 1
Zapoznaj się z prezentacją. Porównaj z nią swoje rozwiązanie.

Zastanów się, jak zmienić program, aby tekst był szyfrowany poprawnie bez względu na wielkość liter.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Napisz program, który stosując konkatenację połączy wszystkie ciągi znaków z listy `ciagi`, a następnie wyświetli utworzone zdanie. Przetestuj jego działanie dla podanej listy:

```
ciagi = ["Dura ", "lex", ", ", "sed ", "lex", "."]
```

Specyfikacja:

Dane:

- `ciagi` – lista łańcuchów znaków

Wynik:

Program wyświetla połączone ciągi znaków.

Program na wyjściu standardowym zwróci wyświetla połączone ciągi znaków.

Ćwiczenie 2

Napisz program, który dodaje kody ASCII wszystkich liter w ciągu `ciag`, a następnie wyświetla znak odpowiadający otrzymanej sumie w kodzie ASCII. Przetestuj jego działanie dla danego ciągu:

```
ciag = "$%&"
```

Specyfikacja:

Dane:

- `ciag` – ciąg znaków

Wynik:

Program na wyjściu standardowym zwróci znak odpowiadający otrzymanej sumie w kodzie ASCII.

Ćwiczenie 3

Napisz program, który zaszyfruje dane słowo `wyraz`, używając szyfru Cezara z kluczem o danej wartości. Przetestuj działanie programu dla tekstu jawnego `programowanie` oraz klucza równego 7.

Specyfikacja:

Dane:

- `wyraz` – ciąg znaków; tekst jawny
- `klucz` – liczba całkowita; wartość klucza szyfrowania

Wynik:

Program na wyjściu standardowym wyświetla tekst zaszyfrowany szyfrem Cezara o kluczu danej wartości.

Ćwiczenie 4

Napisz program, który odszyfruje ciąg `zdanie_do_odszyfrowania`, zakodowany szyfrem Cezara z kluczem o danej wartości. Pamiętaj, że tekst nie może zawierać znaków diakrytycznych.

Przetestuj działanie programu dla podanych danych:

```
zdanie_do_odszyfrowania = "KNSNX HTWTSFY TUZX"
klucz = 5
```

Specyfikacja:

Dane:

- `zdanie_do_odszyfrowania` – łańcuch znaków
- `klucz` – liczba całkowita

Wynik:

Program wyświetla odszyfrowany łańcuch znaków.

Program na wyjściu standardowym zwróci odszyfrowaną wiadomość z wykorzystaniem klucza o wartości 5.

Dla nauczyciela

Autor: Maurycy Gast
Przedmiot: Informatyka

Temat: Szyfr Cezara w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

2) stosuje przy rozwiązywaniu problemów z różnych dziedzin algorytmy poznane w szkole podstawowej oraz algorytmy:

b) na tekstach: porównywania tekstów, wyszukiwania wzorca w tekście metodą naiwną, szyfrowania tekstu metodą Cezara i przestawieniową,

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):

- Prześledzisz algorytm szyfrowania tekstu za pomocą szyfru Cezara.
- Przeanalizujesz algorytm odszyfrowania tekstu utajnionego, za pomocą szyfru Cezara.
- Napiszesz w języku Python program szyfrujący oraz odszyfrowujący tekst.
- Wyjaśnisz działanie funkcji na pojedynczych znakach i na napisach.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

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

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. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Szyfr Cezara w języku Python". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu i programowania, np. :
   – jak nazywamy dziedzinę zajmującą się szyfrowaniem?
   – gdzie w życiu codziennym możemy się spotkać z szyfrowaniem danych?
   – jakie znacie rodzaje szyfrów?
   Chętni uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. Uczniowie analizują przykład z sekcji „Przeczytaj" i powtarzają zaprezentowane rozwiązanie na swoim komputerze.

2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna”, wybrany uczeń czyta treść polecenia nr 1: „Zapoznaj się z prezentacją, a następnie napisz program deszyfrujący podany mu wyraz (przy ustalonym kluczu szyfrowania).” i omawia przykładowe rozwiązanie postawionego problemu.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują ćwiczenia nr 1–2 z sekcji „Sprawdź się”. Nauczyciel sprawdza poprawność wykonanych zadań, omawiając je wraz z uczniami.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie” i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Wybrany uczeń podsumowuje zajęcia, zwracając uwagę na nabyte umiejętności, omawia ewentualne problemy podczas rozwiązania ćwiczeń z programowania w języku Python.

Praca domowa:

1. Uczniowie proponują alternatywny sposób rozwiązania problemu postawionego w sekcji „Prezentacja multimedialna”.
2. Uczniowie wykonują ćwiczenie 3 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 „Prezentacja multimedialna” można wykorzystać jako materiał służący powtórzeniu materiału.