

# ZŁAM TEN KOD Z PYTHONEM

JAK TWORZYĆ, TESTOWAĆ  
I ŁAMAĆ SZYFRY

AL SWEIGART



Tytuł oryginału: Cracking Codes with Python: An Introduction to Building and Breaking Ciphers

Tłumaczenie: Agnieszka Górczyńska

ISBN: 978-83-283-7495-9

Copyright © 2018 by Al Sweigart. Title of English-language original:  
Cracking Codes with Python: An Introduction to Building and Breaking Ciphers,  
ISBN: 978-1-59327-822-9, published by No Starch Press.

Polish-language edition copyright © 2021 by Helion S.A. All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<https://ftp.helion.pl/przyklady/zlamtk.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/zlamtk>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

# Spis treści

O autorze	4
O korektorach merytorycznych	4
Podziękowania	17

## **WPROWADZENIE ..... 19**

Kto powinien przeczytać tę książkę?	20
Co znajdziesz w tej książce?	21
Jak używać tej książki?	23
Wpisywanie kodu źródłowego	23
Sprawdzanie pod kątem błędów	23
Konwencje zastosowane w książce	24
Zasoby w internecie	24
Pobieranie i instalowanie Pythona	24
Instalacja Pythona w systemie Windows	24
Instalacja Pythona w systemie macOS	25
Instalacja Pythona w systemie Ubuntu	25
Pobieranie pliku pyperclip.py	25
Uruchamianie środowiska IDLE	26
Podsumowanie	27

## **1**

### **PAPIER JAKO NARZĘDZIE KRYPTOGRAFICZNE ..... 29**

Co to jest kryptografia?	30
Kod a szyfr	30
Szyfr Cezara	32
Krążek szyfrowania	32
Szyfrowanie wiadomości za pomocą krążka szyfrowania	33
Deszyfrowanie za pomocą krążka szyfrowania	34
Szyfrowanie i deszyfrowanie z użyciem arytmetyki	35
Dlaczego podwójne szyfrowanie nie działa?	36
Podsumowanie	36

## **2**

### **PROGRAMOWANIE W POWŁOCE INTERAKTYWNEJ ..... 39**

Kilka prostych wyrażeń matematycznych	40
Wartości całkowite i wartości zmiennoprzecinkowe	41
Wyrażenia	41
Kolejność wykonywania działań	42
Obliczanie wartości wyrażeń	42

Przechowywanie wartości w zmiennych .....	43
Nadpisywanie zmiennej .....	45
Nazwy zmiennych .....	46
Podsumowanie .....	47

### 3

#### **CIĄGI TEKSTOWE I TWORZENIE PROGRAMÓW ..... 49**

Praca z tekstem przy użyciu wartości w postaci ciągu tekstowego .....	50
Konkatenacja ciągu tekstowego za pomocą operatora + .....	51
Replikacja ciągu tekstowego przy użyciu operatora * .....	52
Pobieranie znaków z ciągu tekstowego przy użyciu indeksów .....	53
Wyświetlanie wartości za pomocą funkcji print() .....	56
Wyświetlanie znaków sterujących .....	57
Apostrof i cudzysłów .....	58
Tworzenie programów w edytorze pliku IDLE .....	59
Kod źródłowy programu typu Witaj, świecie! .....	59
Sprawdzanie kodu źródłowego za pomocą narzędzia Online Diff Tool .....	61
Użycie środowiska IDLE w celu późniejszego uzyskania dostępu do programu .....	62
Zapisywanie programu .....	62
Uruchamianie programu .....	63
Otwieranie wcześniej zapisanych programów .....	64
W jaki sposób działa program Witaj, świecie!? .....	64
Komentarze .....	64
Wyświetlanie wskazówek dla użytkownika .....	65
Pobieranie danych wejściowych od użytkownika .....	65
Zakończenie programu .....	66
Podsumowanie .....	66

### 4

#### **SZYFR ODWROTNY ..... 69**

Kod źródłowy programu wykorzystującego szyfr odwrotny .....	70
Przykładowe uruchomienie programu .....	70
Definiowanie komentarzy i zmiennych .....	71
Określanie długości ciągu tekstowego .....	72
Wprowadzenie do pętli while .....	73
Boolowski typ danych .....	73
Operatory porównania .....	74
Blok kodu .....	76
Konstrukcja pętli while .....	77
„Rośnięcie” ciągu tekstowego .....	78
Usprawnianie programu za pomocą funkcji input() .....	81
Podsumowanie .....	82

### 5

#### **SZYFR CEZARA ..... 85**

Kod źródłowy programu wykorzystującego szyfr Cezara .....	86
Przykładowe uruchomienie programu .....	87
Importowanie modułu i przypisywanie zmiennych .....	88
Stałe i zmienne .....	89

Pętla for .....	90
Przykład pętli for .....	90
Pętla while będąca odpowiednikiem pętli for .....	91
Konstrukcja if .....	92
Przykład użycia polecenia if .....	92
Polecenie else .....	92
Polecenie elif .....	93
Operatory in i not in .....	94
Metoda find() .....	95
Szyfrowanie i deszyfrowanie symboli .....	96
Obsługa zawinięcia .....	97
Obsługa symboli spoza zbioru symboli .....	98
Wyświetlanie i kopiowanie skonwertowanego ciągu tekstowego .....	98
Szyfrowanie innych symboli .....	99
Podsumowanie .....	100

## 6

<b>ŁAMANIE SZYFRU CEZARA ZA POMOCĄ ATAKU BRUTE FORCE .....</b>	<b>103</b>
Kod źródłowy programu wykorzystującego szyfr odwrotny .....	104
Przykładowe uruchomienie programu .....	105
Definiowanie zmiennych .....	106
Iteracja z użyciem funkcji range() .....	106
Deszyfrowanie wiadomości .....	108
Stosowanie formatowania ciągu tekstowego do wyświetlenia klucza i deszyfrowanej wiadomości .....	109
Podsumowanie .....	110

## 7

<b>SZYFROWANIE ZA POMOCĄ SZYFRU PRZESTAWIENIOWEGO .....</b>	<b>113</b>
Sposób działania szyfru przestawieniowego .....	113
Ręczne szyfrowanie wiadomości .....	114
Tworzenie programu szyfrującego .....	116
Kod źródłowy programu wykorzystującego szyfr kolumnowy .....	117
Przykładowe uruchomienie programu .....	118
Samodzielne definiowanie funkcji za pomocą polecenia def .....	118
Definiowanie funkcji pobierającej argumenty .....	119
Zmiana parametru istniejącego tylko wewnątrz funkcji .....	120
Definiowanie funkcji main() .....	121
Przekazywanie klucza i wiadomości jako argumentów .....	122
Typ danych listy .....	123
Ponowne przypisywanie elementów na liście .....	124
Lista list .....	125
Stosowanie funkcji len() i operatora in z listą .....	126
Konkatenacja listy i replikacja za pomocą operatorów + i * .....	127
Algorytm szyfrowania przestawieniowego .....	127
Rozszerzone operatory przypisania .....	128
Iteracja currentIndex przez wiadomość .....	129
Metoda join() .....	131
Wartość zwrotna i polecenie return .....	132
Przykład polecenia return .....	132
Zwrot szyfrogramu .....	133

Zmienna <code>__name__</code> .....	133
Podsumowanie .....	134
<b>8</b>	
<b>DESZYFROWANIE WIADOMOŚCI CHRONIONEJ SZYFREM PRZESTAWIENIOWYM .....</b>	<b>137</b>
Łamanie szyfru przestawieniowego za pomocą kartki i ołówka .....	138
Kod źródłowy programu deszyfrującego wiadomość chronioną szyfrem przestawieniowym .....	139
Przykładowe uruchomienie programu .....	141
Importowanie modułów i definiowanie funkcji <code>main()</code> .....	141
Deszyfrowanie wiadomości za pomocą klucza .....	142
Funkcje <code>round()</code> , <code>math.ceil()</code> i <code>math.floor()</code> .....	142
Funkcja <code>decryptMessage()</code> .....	143
Operatory boolowskie .....	145
Dostosowywanie wartości zmiennych <code>column</code> i <code>row</code> .....	148
Wywoływanie funkcji <code>main()</code> .....	150
Podsumowanie .....	150
<b>9</b>	
<b>TWORZENIE PROGRAMU DO TESTOWANIA INNYCH PROGRAMÓW .....</b>	<b>153</b>
Kod źródłowy programu do testowania innych programów .....	154
Przykładowe uruchomienie programu .....	155
Importowanie modułów .....	156
Generowanie liczb pseudolosowych .....	156
Tworzenie losowo wybranego ciągu tekstowego .....	158
Powielanie ciągu tekstowego losowo wybraną liczbę razy .....	158
Zmienna <code>listy</code> używa odwołania .....	159
Przekazywanie odwołania .....	162
Stosowanie funkcji <code>copy.deepcopy()</code> do powielenia listy .....	162
Funkcja <code>random.shuffle()</code> .....	163
Losowe mieszanie ciągu tekstowego .....	163
Testowanie poszczególnych wiadomości .....	164
Sprawdzanie poprawności szyfrowania i zakończenie programu .....	165
Wywoływanie funkcji <code>main()</code> .....	166
Testowanie programu .....	166
Podsumowanie .....	167
<b>10</b>	
<b>SZYFROWANIE I DESZYFROWANIE PLIKÓW .....</b>	<b>169</b>
Pliki zwykłego tekstu .....	170
Kod źródłowy programu wykorzystującego szyfr przestawieniowy do szyfrowania pliku .....	170
Przykładowe uruchomienie programu .....	171
Praca z plikami .....	172
Otwieranie pliku .....	172
Zapisywanie i zamykanie pliku .....	173
Odczyt danych z pliku .....	174
Funkcja <code>main()</code> programu .....	175
Sprawdzanie istnienia pliku .....	175
Funkcja <code>os.path.exists()</code> .....	176
Sprawdzanie za pomocą funkcji <code>os.path.exists()</code> istnienia pliku danych wejściowych .....	176

Stosowanie metod ciągu tekstowego do zapewnienia większej elastyczności danych wejściowych .....	177
Metody ciągu tekstowego upper(), lower() i title() .....	177
Metody ciągu tekstowego startswith() i endswith() .....	177
Stosowanie metod ciągu tekstowego w programie .....	178
Odczyt pliku danych wejściowych .....	179
Pomiar czasu operacji szyfrowania i deszyfrowania .....	179
Moduł time i funkcja time.time() .....	179
Stosowanie funkcji time.time() w programie .....	180
Zapis danych wyjściowych do pliku .....	181
Wywoływanie funkcji main() .....	181
Podsumowanie .....	182

## 11

<b>PROGRAMOWE WYKRYWANIE JĘZYKA ANGIELSKIEGO .....</b>	<b>183</b>
Jak komputer może zrozumieć język angielski? .....	184
Kod źródłowy modułu do wykrywania języka angielskiego .....	186
Przykładowe uruchomienie programu .....	187
Polecenia i definiowanie stałych .....	187
Typ danych w postaci słownika .....	188
Różnice między słownikiem i listą .....	189
Dodawanie lub modyfikowanie elementów słownika .....	190
Stosowanie funkcji len() ze słownikiem .....	191
Stosowanie operatora in ze słownikiem .....	191
Wyszukiwanie elementów w słowniku odbywa się szybciej niż na liście .....	192
Stosowanie pętli for w słowniku .....	192
Implementacja pliku słownika .....	193
Metoda split() .....	193
Podział słownika na poszczególne słowa .....	194
Zwrot danych słownika .....	194
Zliczanie liczby słów angielskich w wiadomości .....	195
Błąd dzielenia przez zero .....	196
Zliczanie dopasowań słów w języku angielskim .....	196
Funkcje float(), int() i str() oraz dzielenie całkowite .....	197
Określanie proporcji angielskich słów w wiadomości .....	198
Usuwanie znaków innych niż litery .....	198
Metoda append() typu listy .....	199
Tworzenie ciągu tekstowego liter .....	200
Wykrywanie słów angielskich .....	200
Stosowanie argumentów domyślnych .....	200
Obliczanie wartości procentowych .....	201
Podsumowanie .....	203

## 12

<b>ŁAMANIE SZYFRU PRZESTAWIENIOWEGO .....</b>	<b>205</b>
Kod źródłowy programu umożliwiającego złamanie szyfru przestawieniowego .....	206
Przykładowe uruchomienie programu .....	207
Importowanie modułów .....	208
Wielowierszowy ciąg tekstowy ujęty w potrójny cudzysłów .....	208
Wyświetlanie wyniku deszyfrowania wiadomości .....	209

Pobranie deszyfrowanej wiadomości .....	210
Metoda strip() ciągu tekstowego .....	212
Stosowanie metody strip() ciągu tekstowego .....	213
Nieudana próba deszyfrowania wiadomości .....	213
Wywoływanie funkcji main() .....	214
Podsumowanie .....	214

### 13

<b>MODUŁ ARYTMETYKI MODULARNEJ DLA SZYFRU AFINICZNEGO .....</b>	<b>215</b>
Arytmetyka modularna .....	216
Operator reszty z dzielenia .....	217
Wyszukiwanie dzielników do obliczenia największego wspólnego dzielnika .....	218
Przypisanie wielokrotne .....	220
Algorytm Euklidesa do wyszukiwania największego wspólnego dzielnika .....	221
Sposób działania szyfrów multiplikatywnego i afinicznego .....	222
Wybór poprawnego klucza multiplikatywnego .....	223
Szyfrowanie z użyciem szyfru afinicznego .....	224
Deszyfrowanie szyfru afinicznego .....	225
Określanie odwrotności modularnej .....	226
Operator dzielenia całkowitego .....	226
Kod źródłowy modułu cryptomath .....	227
Podsumowanie .....	228

### 14

<b>PROGRAMOWANIE SZYFRU AFINICZNEGO .....</b>	<b>231</b>
Kod źródłowy programu wykorzystującego szyfr afiniczny .....	232
Przykładowe uruchomienie programu .....	233
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	234
Generowanie i weryfikowanie kluczy .....	236
Typ danych w postaci krotki .....	236
Sprawdzanie pod kątem słabych kluczy .....	237
Ile kluczy może mieć szyfr afiniczny? .....	238
Tworzenie funkcji szyfrującej .....	240
Tworzenie funkcji deszyfrującej .....	241
Generowanie losowych kluczy .....	242
Wywoływanie funkcji main() .....	243
Podsumowanie .....	244

### 15

<b>ŁAMANIE SZYFRU AFINICZNEGO .....</b>	<b>245</b>
Kod źródłowy programu umożliwiającego złamanie szyfru afinicznego .....	245
Przykładowe uruchomienie programu .....	247
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	248
Funkcja odpowiedzialna za złamanie szyfru afinicznego .....	249
Operator wykładniczy .....	249
Obliczanie całkowitej liczby kluczy, których można użyć .....	250
Polecenie continue .....	251
Stosowanie polecenia continue do pominięcia kodu .....	252
Wywoływanie funkcji main() .....	253
Podsumowanie .....	254



## 16

<b>PROGRAMOWANIE PROSTEGO SZYFRU PODSTAWIENIOWEGO .....</b>	<b>255</b>
Jak działa prosty szyfr podstawieniowy? .....	256
Kod źródłowy programu wykorzystującego szyfr podstawieniowy .....	257
Przykładowe uruchomienie programu .....	259
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	259
Metoda sort() listy .....	261
Funkcje opakowujące .....	262
Funkcja translateMessage() .....	263
Metody isupper() i islower() ciągu tekstowego .....	265
Zachowywanie wielkości liter dzięki metodzie isupper() .....	266
Generowanie losowego klucza .....	267
Wywoływanie funkcji main() .....	268
Podsumowanie .....	268

## 17

<b>ŁAMANIE PROSTEGO SZYFRU PODSTAWIENIOWEGO .....</b>	<b>271</b>
Stosowanie wzorca słowa do deszyfrowania .....	272
Znajdowanie wzorca słowa .....	272
Wyszukiwanie potencjalnych liter odszyfrowujących .....	273
Omówienie procesu łamania szyfru .....	275
Moduł wzorca słowa .....	275
Kod źródłowy programu wykorzystującego szyfr podstawieniowy .....	276
Przykładowe uruchomienie programu .....	280
Importowanie modułów i stałych .....	280
Wyszukiwanie znaków za pomocą wyrażeń regularnych .....	281
Konfigurowanie funkcji main() .....	281
Wyświetlanie użytkownikowi wyniku operacji łamania szyfru .....	282
Tworzenie mapowania szyfrogramu .....	283
Tworzenie pustego mapowania .....	283
Dodawanie liter do mapowania .....	283
Łączenie dwóch mapowań .....	285
W jaki sposób działają funkcje pomocnicze mapowania liter? .....	286
Wyszukiwanie zdeszyfrowanych liter w mapowaniu .....	290
Testowanie funkcji removeSolvedLettersFromMapping() .....	292
Funkcja hackSimpleSub() .....	292
Metoda replace() ciągu tekstowego .....	294
Deszyfrowanie wiadomości .....	295
Deszyfrowanie w powłoce interaktywnej .....	296
Wywoływanie funkcji main() .....	297
Podsumowanie .....	298

## 18

<b>PROGRAMOWANIE SZYFRU VIGENÈRE'A .....</b>	<b>299</b>
Stosowanie wielu liter kluczy w szyfrze Vigenère'a .....	300
Dłuższe klucze szyfru Vigenère'a są znacznie bezpieczniejsze .....	302
Wybór klucza uniemożliwiającego atak słownikowy .....	303
Kod źródłowy programu wykorzystującego szyfr Vigenère'a .....	303
Przykładowe uruchomienie programu .....	305
Importowanie modułów i stałych oraz definiowanie funkcji main() .....	305
Tworzenie ciągu tekstowego za pomocą procesu dołączania do listy .....	306

Szyfrowanie i deszyfrowanie wiadomości .....	307
Wywoływanie funkcji main() .....	310
Podsumowanie .....	310

## 19

<b>ANALIZA CZĘSTOTLIWOŚCI .....</b>	<b>313</b>
Analiza częstotliwości występowania liter w tekście .....	314
Dopasowywanie częstotliwości występowania liter .....	316
Obliczanie wyniku dopasowania częstotliwości dla prostego szyfru podstawieniowego .....	316
Obliczanie wyniku dopasowania częstotliwości dla prostego szyfru przestawieniowego .....	317
Stosowanie analizy częstotliwości do złamania szyfru Vigenère'a .....	318
Kod źródłowy programu obliczającego wynik dopasowania częstotliwości .....	319
Przechowywanie liter w kolejności ETAOIN .....	321
Zliczanie liter w wiadomości .....	321
Pobieranie pierwszego elementu składowego krotki .....	323
Układanie liter według częstotliwości ich występowania w wiadomości .....	323
Zliczanie liter za pomocą funkcji getLetterCount() .....	324
Tworzenie słownika częstotliwości wystąpień i listy liter .....	324
Sortowanie liter w odwrotnej kolejności ETAOIN .....	325
Sortowanie list słownika według częstotliwości występowania .....	330
Tworzenie listy sortowanych liter .....	332
Obliczanie wyniku dopasowania częstotliwości dla wiadomości .....	332
Podsumowanie .....	334

## 20

<b>ŁAMANIE SZYFRU VIGENÈRE'A .....</b>	<b>335</b>
Atak słownikowy w celu złamania szyfru Vigenère'a metodą brute force .....	336
Kod źródłowy programu umożliwiającego złamanie szyfru Vigenère'a za pomocą ataku słownikowego .....	336
Przykładowe uruchomienie programu .....	337
Informacje o programie do łamania szyfru Vigenère'a za pomocą ataku słownikowego .....	337
Stosowanie metody Kasiskiego do ustalenia długości klucza .....	338
Odszukanie powtarzających się sekwencji .....	338
Pobieranie dzielników liczb określających odstęp .....	339
Pobieranie każdej n-tej litery ciągu tekstowego .....	341
Stosowanie analizy częstotliwości do złamania poszczególnych podkluczy .....	342
Przeprowadzanie ataku brute force na możliwe klucze .....	344
Kod źródłowy programu umożliwiającego złamanie szyfru Vigenère'a .....	344
Przykładowe uruchomienie programu .....	349
Importowanie modułów i definiowanie funkcji main() .....	350
Wyszukiwanie powtarzających się sekwencji .....	351
Obliczanie dzielników odstępów .....	354
Usuwanie duplikatów za pomocą funkcji set() .....	355
Usuwanie powtarzających się dzielników i sortowanie listy .....	355
Wyszukiwanie najczęściej występujących dzielników .....	356
Określanie prawdopodobnej długości klucza .....	358
Metoda listy extend() .....	358
Rozszerzanie słownika repeatedSeqSpacings .....	359
Pobieranie dzielników z factorsByCount .....	360
Pobieranie liter szyfrowanych za pomocą tego samego podklucza .....	360

Próba deszyfrowania z użyciem potencjalnych długości klucza .....	362
Argument w postaci słowa kluczowego key funkcji print() .....	364
Uruchamianie programu w trybie cichym lub wyświetlania informacji użytkownikowi .....	365
Wyszukiwanie możliwych kombinacji podkluczy .....	365
Wyświetlanie deszyfrowanego tekstu z użyciem właściwej wielkości liter .....	369
Zwrot deszyfrowanej wiadomości .....	370
Opuszczanie pętli po znalezieniu potencjalnego klucza .....	371
Atak brute force na wszystkie długości klucza .....	371
Wywoływanie funkcji main() .....	372
Modyfikowanie stałych programu .....	373
Podsumowanie .....	373

## 21

<b>SZYFR Z KLUCZEM JEDNORAZOWYM .....</b>	<b>375</b>
Niemożliwy do złamania szyfr z kluczem jednorazowym .....	376
Tworzenie klucza o długości odpowiadającej długości wiadomości .....	376
Zapewnianie prawdziwej losowości klucza .....	378
Dlaczego klucza jednorazowego można użyć tylko raz? .....	379
Dlaczego dwukrotnie użyty klucz jednorazowy to szyfr Vigenère'a? .....	379
Podsumowanie .....	380

## 22

<b>WYSZUKIWANIE I GENEROWANIE LICZB PIERWSZYCH .....</b>	<b>381</b>
Co to jest liczba pierwsza? .....	382
Kod źródłowy modułu liczb pierwszych .....	384
Przykładowe uruchomienie modułu .....	386
Sposób działania algorytmu próbnego dzielenia .....	386
Implementacja algorytmu próbnego dzielenia .....	388
Sito Eratostenesa .....	389
Generowanie liczb pierwszych za pomocą sita Eratostenesa .....	391
Algorytm pierwszości Rabina-Millera .....	392
Wyszukiwanie ogromnych liczb pierwszych .....	393
Generowanie ogromnych liczb pierwszych .....	395
Podsumowanie .....	395

## 23

<b>GENEROWANIE KLUCZY DLA SZYFRU KLUCZA PUBLICZNEGO .....</b>	<b>397</b>
Kryptografia klucza publicznego .....	398
Problem z uwierzytelnieniem .....	400
Podpis cyfrowy .....	400
Uważaj na atak MITM .....	401
Etapy generowania kluczy publicznego i prywatnego .....	402
Kod źródłowy programu generującego klucze kryptografii klucza publicznego .....	403
Przykładowe uruchomienie programu .....	404
Tworzenie funkcji main() .....	406
Generowanie kluczy za pomocą funkcji generateKey() .....	406
Obliczanie wartości e .....	407
Obliczanie wartości d .....	407
Zwracanie kluczy .....	408
Tworzenie plików kluczy za pomocą funkcji makeKeyFiles() .....	408

Wywoływanie funkcji main() .....	410
Hybrydowe systemy kryptograficzne .....	411
Podsumowanie .....	411

## 24

<b>PROGRAMOWANIE SZYFRU KLUCZA PUBLICZNEGO .....</b>	<b>413</b>
Jak działa kryptografia klucza publicznego? .....	414
Tworzenie bloku .....	414
Konwersja ciągu tekstowego na blok .....	415
Matematyka szyfrowania i deszyfrowania za pomocą kryptografii klucza publicznego .....	416
Konwersja bloku na ciąg tekstowy .....	418
Dlaczego nie można złamać szyfru wykorzystującego kryptografię klucza publicznego? .....	420
Kod źródłowy programu wykorzystującego kryptografię klucza publicznego .....	421
Przykładowe uruchomienie programu .....	425
Konfiguracja programu .....	426
Wybór trybu pracy programu .....	426
Konwersja ciągu tekstowego na bloki za pomocą funkcji getBlocksFromText() .....	428
Funkcje min() i max() .....	428
Przechowywanie bloków w blockInt .....	429
Stosowanie funkcji getTextFromBlocks() do deszyfrowania wiadomości .....	431
Stosowanie metody insert() listy .....	432
Łączenie listy message i tworzenie na jej podstawie jednego ciągu tekstowego .....	432
Tworzenie funkcji encryptMessage() .....	433
Tworzenie funkcji decryptMessage() .....	433
Odczytywanie kluczy publicznego i prywatnego z ich plików .....	434
Zapisywanie szyfrogramu do pliku .....	435
Deszyfrowanie danych z pliku .....	437
Wywoływanie funkcji main() .....	439
Podsumowanie .....	439

## A

<b>DEBUGOWANIE KODU PYTHONA .....</b>	<b>441</b>
Na czym polegają działania debugera? .....	441
Usuwanie błędów z programu wykorzystującego szyfr odwrotny .....	443
Definiowanie punktu przerwania .....	445
Podsumowanie .....	447

## B

<b>ODPOWIEDZI DO ĆWICZEŃ .....</b>	<b>449</b>
Rozdział 1. ....	449
Rozdział 2. ....	450
Rozdział 3. ....	451
Rozdział 4. ....	452
Rozdział 5. ....	453
Rozdział 6. ....	454
Rozdział 7. ....	455
Rozdział 8. ....	457
Rozdział 9. ....	459
Rozdział 10. ....	459
Rozdział 11. ....	460

Rozdział 12. ....	461
Rozdział 13. ....	462
Rozdział 14. ....	462
Rozdział 15. ....	463
Rozdział 16. ....	463
Rozdział 17. ....	464
Rozdział 18. ....	464
Rozdział 19. ....	465
Rozdział 20. ....	465
Rozdział 21. ....	466
Rozdział 22. ....	466
Rozdział 23. ....	466



# 1

## Papier jako narzędzie kryptograficzne

*„Dzin szyfrowania opuścił butelkę”.*  
— Jan Koum, założyciel WhatsApp



ZANIM PRZYSTĄPIMY DO TWORZENIA PROGRAMÓW SZYFRUJĄCYCH, WARTO ZAPOZNAĆ SIĘ Z PROCESEM SZYFROWANIA I DESZYFROWANIA WIADOMOŚCI Z UŻYCIEM PAPIERU I OŁÓWKA. TO POMOŻE ZROZUMIEĆ, JAK działa szyfrowanie i jakie operacje matematyczne są stosowane do wygenerowania zabezpieczonych wiadomości. Z tego rozdziału dowiesz się, co oznacza kryptografia i czym kod różni się od szyfru. Następnie użyjesz prostego szyfru zwanego szyfrem Cezara, aby zaszyfrować i odszyfrować wiadomość za pomocą papieru i ołówka.

### TEMATY OMÓWIONE W TYM ROZDZIALE:

- co to jest kryptografia;
- kody i szyfry;
- szyfr Cezara;
- krążki szyfrowania;
- kryptografia i arytmetyka;
- podwójne szyfrowanie.

# Co to jest kryptografia?

W przeszłości każdy, kto musiał udostępniać tajne informacje innym osobom, np. szpieg, żołnierz, haker, pirat, sprzedawca, tyran czy działacz polityczny, korzystał z kryptografii, aby informacje te nie wpadły w niepowołane ręce. *Kryptografia* to nauka stosowania tajnych kodów. Jeżeli chcesz się dowiedzieć, na czym polega idea kryptografii, spójrz na dwa przedstawione tutaj fragmenty kodu:

nyr N .vNwz5uNz5Ns6620Nz0N3z2v	!NN2 Nuwv,N9,vNN!vNrBN3zyN4vN
N yvNwz9vNz5N6!9Nyvr9	N6 Qvv0z6nvN.7N0yv4N 4 zzvNN
y0QNnvNwv tyNz	vyN,NN99z0zz6wz0y3vv26 9
Nw964N6!9N5vzxsy690,N.vN2z5u-	w296vyNNrrNyQst.560N94Nu5y
3vNz Nr Ny64v,N .vNt644!5ztr vNz	rN5nz5vv5t6v63zNr5.
N 6N6 yv90,Nr5uNz Nsvt64v0N	N75sz6966NNvw6 zu0 wtNxs6t
yvN7967v9 BN6wNr33Q N-m63 rz9v	49NrN3Ny9Nvzy!

Tekst w lewej kolumnie to tajna wiadomość, która została *zaszyfrowana*, czyli zmieniona na tajny kod. Pozostaje ona całkowicie nieczytelna dla każdego, kto nie wie, jak należy ją *deszyfrować*, czyli zmienić z powrotem na pierwotną postać. Tekst w prawej kolumnie to ciąg przypadkowo wybranych znaków, bez ukrytego znaczenia. Szyfrowanie pozwala na ukrycie wiadomości przed osobami, które nie potrafią jej odszyfrować, nawet jeśli w ich ręce wpadnie zaszyfrowana wiadomość. *Wiadomość zaszyfrowana wygląda dokładnie tak samo jak pozbawiony sensu ciąg znaków i cyfr.*

*Kryptograf* stosuje i analizuje tajne kody. Oczywiście te tajne wiadomości nie zawsze takimi pozostają. Z kolei *kryptoanalityk*, nazywany również *hakerem* (lub *lamaczem kodów*), może złamać kod i odczytać zaszyfrowane wiadomości. Z tej książki dowiesz się, jak szyfrować i deszyfrować wiadomości za pomocą różnych technik. Jednak wiedza ta nie jest na tyle niebezpieczna, aby zaprowadzić Cię za kratki.

## Kod a szyfr

W przeciwieństwie do szyfru *kod* ma być doskonale zrozumiały i dostępny publicznie. Kod zastępuje wiadomość symbolami, które każdy powinien potrafić odczytać i zamienić na postać pierwotnej wiadomości.

Na początku XIX wieku powstał doskonale znany kod dzięki opracowaniu telegrafu, który umożliwił niemal natychmiastową komunikację przewodową między kontynentami. Wysłanie wiadomości za pomocą telegrafu było znacznie szybsze niż skorzystanie z którejś z dotychczasowych możliwości, np. posłańca na koniu lub listu napisanego na papierze. Telegraf pozwalał wysyłać jedynie dwa rodzaje impulsów elektrycznych: krótki, nazywany kropką, i długi, nazywany kreską.

Aby skonwertować litery alfabetu na owe kropki i kreski, potrzebny był system kodowania pozwalający zamienić słowa na impulsy elektryczne. Proces konwersji słów na kropki i kreski oraz ich przesyłania przez telegraf jest nazywany *kodowa-*



niem, proces odwrotny zaś nosi nazwę *dekodowania*. Kod używany podczas kodowania i dekodowania wiadomości przekazywanych poprzez telegraf (a później także radiowo) został opracowany przez Samuela Morse'a i Alfreda Vaila i nosi nazwę kodu Morse'a. Przedstawiłem go w tabeli 1.1.

Tabela 1.1. Międzynarodowy kod Morse'a

Litera	Kod	Litera	Kod	Liczba	Kod
A	• —	N	— •	1	• — — — —
B	— • • •	O	— — —	2	• • — — —
C	— • — •	P	• — — •	3	• • • — —
D	— • • •	Q	— — • —	4	• • • • —
E	•	R	• — •	5	• • • • •
F	• • — •	S	• • •	6	— • • • •
G	— — •	T	—	7	— — • • •
H	• • • •	U	• • —	8	— — — • •
I	• •	V	• • • —	9	— — — — •
J	• — — —	W	• — —	0	— — — — —
K	— • —	X	— • • —		
L	• — • •	Y	— • — —		
M	— —	Z	— — • •		

Dzięki generowaniu kropek i kresek za pomocą jednoprzyciskowego telegrafu jego operator mógł przekazywać, niemalże natychmiast, wiadomości do osoby znajdującej się na drugim końcu świata. (Jeżeli chcesz dowiedzieć się więcej na temat kodu Morse'a, zajrzyj na stronę [https://pl.wikipedia.org/wiki/Kod\\_Morse'a](https://pl.wikipedia.org/wiki/Kod_Morse'a)).

W przeciwieństwie do kodu szyfr to specjalny rodzaj kodu przeznaczony do zapewnienia tajności przekazywanym wiadomościom. Szyfr można wykorzystać do zmiany całkowicie jawnego tekstu, np. w języku angielskim, nazywanego *zwykłym tekstem*, w pozbawiony sensu ciąg liter i znaków, nazywany *szyfrogramem*, zawierający tajną wiadomość. Szyfr to zbiór reguł dotyczących konwersji między postaciami zwykłego tekstu i szyfrogramu. Te reguły często podczas komunikacji wykorzystują do szyfrowania i deszyfrowania tajny klucz, który jest znany tylko komunikującym się stronom. W niniejszej książce poznasz kilka szyfrów oraz utworzysz kilka stosujących je programów do szyfrowania i deszyfrowania wiadomości. Jednak najpierw zobacz, jak wiadomość można zaszyfrować ręcznie za pomocą papieru i ołówka.

# Szyfr Cezara

Pierwszym szyfrem przedstawionym w tej książce będzie szyfr Cezara. Jego nazwa pochodzi od nazwiska Juliusza Cezara, który stosował go ponad 2000 lat temu. Dobra wiadomość jest taka, że szyfr ten jest prosty i dość łatwy do opanowania. Złą wiadomością jest to, że owa prostota oznacza możliwość łatwego złamania tego szyfru przez kryptoanalityka. Mimo to nadal jest użyteczny do zastosowania w celach edukacyjnych.

Szyfr Cezara działa poprzez zastąpienie każdej litery wiadomości inną literą, po przesunięciu całego alfabetu o określoną liczbę pozycji. Przykładowo Juliusz Cezar zastępował litery wiadomości przez przesunięcie liter alfabetu o trzy — każda litera wiadomości była zastępowana literą oddaloną od niej w alfabecie o trzy pozycje.

W omawianym przykładzie każde wystąpienie litery A w wiadomości będzie zastąpione literą D, każde wystąpienie litery B będzie zastąpione literą E itd. Gdy Cezar musiał przesunąć literę znajdującą się na końcu alfabetu, np. Y, wówczas przechodził na początek alfabetu i przesunął ją o wymaganą liczbę pozycji, co oznacza, że np. literę Y zastąpi litera B. W tym podrozdziale ręcznie zaszyfrujesz wiadomość za pomocą szyfru Cezara.

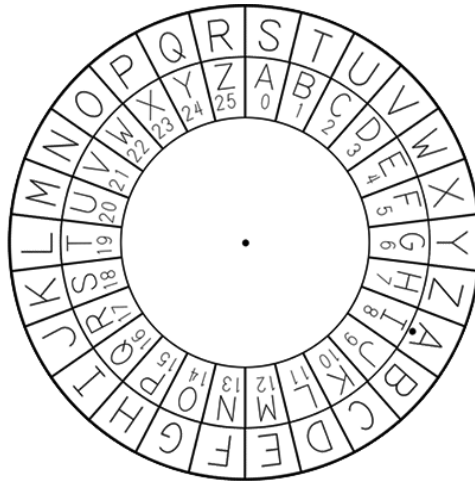
## Krażek szyfrowania

Aby ułatwić sobie zadanie konwersji wiadomości w postaci zwykłego tekstu na szyfrogram z użyciem szyfru Cezara, można skorzystać z tzw. *krażka szyfrowania*, nazywanego również *dyskiem szyfrowania*. Krażek szyfrowania składa się z dwóch pierścieni liter, a każdy z nich jest podzielony na 26 slotów (po jednym dla każdej litery alfabetu). Pierścień zewnętrzny przedstawia alfabet zwykłego tekstu, natomiast pierścień wewnętrzny przedstawia litery szyfrogramu. W pierścieniu wewnętrznym litery są ponumerowane od 0 do 25. Te liczby stanowią tzw. *klucz szyfrowania*, który w omawianym przykładzie określa wielkość przesunięcia pierścienia zewnętrznego względem litery A w pierścieniu wewnętrznym. Ponieważ tarcza jest okrągła, przesunięcie o wartość klucza większą niż 25 powoduje powrót na początek alfabetu. Dlatego klucz 26 odpowiada przesunięciu o zero slotów, klucz 27 odpowiada przesunięciu o jeden slot itd.

Na stronie <https://inventwithpython.com/cipherwheel/> znajduje się internetowa wersja krażka szyfrowania. Pokazałem ją również na rysunku 1.1. Użycie internetowej wersji krażka szyfrowania wymaga jego kliknięcia myszą i przesunięcia kursora aż do otrzymania żądanej konfiguracji. Następnie należy ponownie kliknąć myszą i tym samym zakończyć możliwość obracania pierścienia zewnętrznego.

W witrynie internetowej poświęconej tej książce znajdziesz krażek szyfrowania w wersji przeznaczonej do wydruku. Wytnij oba koła i ulóż je, mniejsze na środku większego. Następnie w środku wbij szpilkę, aby móc obracać koła.

W ten sposób będziesz mógł szyfrować wiadomości za pomocą krażka szyfrowania.



Click wheel to rotate.

S T U V W X Y Z A B C D E F G H I J K L M N O P Q R  
 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  
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

Rysunek 1.1 Internetowa wersja krążka szyfrowania

## Szyfrowanie wiadomości za pomocą krążka szyfrowania

Jeżeli chcesz zaszyfrować wiadomość, najpierw musisz ją zapisać na kartce. W omawianym przykładzie zaszyfrujemy następującą wiadomość: THE SECRET PASSWORD IS ROSEBUD. Następnie obracaj pierścień wewnętrzny, aż dopasujesz sloty z pierścieniem zewnętrznym. Zwróć uwagę na kropkę znajdującą się pod literą A w pierścieniu zewnętrznym. Zanotuj liczbę w slotcie pierścienia wewnętrznego odpowiadającą tej literze A. To jest nasz klucz szyfrowania.

Przykładowo na rysunku 1.1 litera A w pierścieniu zewnętrznym znajduje się nad liczbą 8 w pierścieniu wewnętrznym. Wykorzystamy ten klucz szyfrowania, aby przygotować zaszyfrowaną wersję naszej wiadomości (zobacz rysunek 1.2).

T	H	E	S	E	C	R	E	T	P	A	S	S	W	O	R	D	I	S	R	O	S	E	B	U	D
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
B	P	M	A	M	K	Z	M	B	X	I	A	A	E	W	Z	L	Q	A	Z	W	A	M	J	C	L

Rysunek 1.2. Szyfrowanie wiadomości za pomocą szyfru Cezara i klucza szyfrowania o wartości 8

Każdą literę wiadomości odszukaj w pierścieniu zewnętrznym i zastąp odpowiadającą jej literą z pierścienia wewnętrznego. W omawianym przykładzie pierwsza litera wiadomości to *T* (w słowie *THE*). Po odszukaniu litery *T* w pierścieniu zewnętrznym zobaczysz, że odpowiadającą jej literą pierścienia wewnętrznego jest *B*. Dlatego w tajnej wiadomości litera *T* zawsze będzie zastępowana literą *B*. (Jeżeli używasz innego klucza szyfrowania, każde wystąpienie litery *T* zostanie zastąpione

inną literą). Następną literą w naszej wiadomości to H, która ma być zastąpiona literą P. Kolejną literą to E, którą należy zastąpić literą M. Każda litera pierścienia zewnętrznego zawsze powoduje zaszyfrowanie tej samej litery pierścienia wewnętrznego. Aby zaoszczędzić nieco czasu, po wyszukaniu pierwszej litery T i sprawdzeniu, że po zaszyfrowaniu to litera B, można od razu zastąpić wszystkie wystąpienia litery T w wiadomości literą B i dzięki temu tylko jednokrotnie sprawdzać daną literę.

Po zaszyfrowaniu pierwotna wiadomość THE SECRET PASSWORD IS ROSEBUD będzie miała postać BPM AMKZMB XIAAEWZL QA ZWAMJCL. Zwróć uwagę na to, że znaki inne niż litery, np. spacja, pozostają bez zmian.

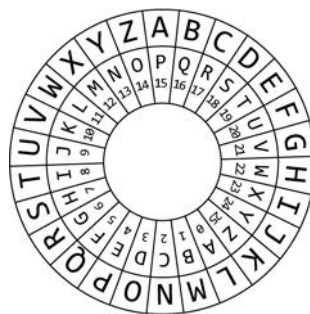
Teraz tę wiadomość można wysłać do kogoś (lub zachować ją dla siebie), a nikt nie będzie w stanie jej odczytać, dopóki nie otrzyma klucza szyfrowania. Nie ujawniaj nikomu klucza szyfrowania. Szyfrogram będzie mógł zostać odczytany przez każdego, kto wie, że kluczem szyfrowania jest 8.

## Deszyfrowanie za pomocą krążka szyfrowania

Aby deszyfrować szyfrogram, trzeba rozpocząć od pierścienia wewnętrznego i następnie przejść do pierścienia zewnętrznego. Załóżmy, że otrzymaliśmy taki oto szyfrogram: IWT CTL EPHHLDGS XH HLDGSUXHW. Nie będziesz w stanie odszyfrować tej wiadomości, jeśli nie znasz klucza (lub nie jesteś sprytnym hakerem). Na szczęście kolega poinformował Cię, że użył klucza szyfrowania 15. Krążek szyfrowania dla tego klucza pokazałem na rysunku 1.3.

Teraz możesz wyrównać literę A w pierścieniu zewnętrznym (tę z kropką na dole) z literą w pierścieniu wewnętrznym, która ma liczbę 15 (jest to P). Odszukaj pierwszą literę szyfrogramu, I, w pierścieniu wewnętrznym i zobacz, jaka odpowiada jej litera w pierścieniu zewnętrznym (jest to T). Drugą literą szyfrogramu, W, po deszyfrowaniu to H. Procedurę deszyfrowania kontynuuj dla wszystkich liter szyfrogramu, a otrzymasz wiadomość THE NEW PASSWORD IS SWORDFISH w postaci zwykłego tekstu (zobacz rysunek 1.4).

Jeżeli do deszyfrowania zostanie użyty niepoprawny klucz, np. 16, wówczas wiadomość będzie miała postać SGD MDV OZRRVNQC HR RVNQCENRG, która



Rysunek 1.3. Krążek szyfrowania z ustawionym kluczem 15

I	W	T		C	T	L		E	P	H	H	L	D	G	S		X	H		H	L	D	G	S	U	X	H	W
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
T	H	E		N	E	W		P	A	S	S	W	O	R	D		I	S		S	W	O	R	D	F	I	S	H

Rysunek 1.4. Deszyfrowanie wiadomości za pomocą szyfru Cezara i klucza szyfrowania o wartości 15

jest bez sensu. Dopóki nie zostanie zastosowany prawidłowy klucz, deszyfrowana wiadomość jest ciągiem przypadkowych liter.

## Szyfrowanie i deszyfrowanie z użyciem arytmetyki

Krażek szyfrowania to wygodne narzędzie przeznaczone do szyfrowania i deszyfrowania wiadomości za pomocą szyfru Cezara. Jednak operacje szyfrowania i deszyfrowania można przeprowadzać również z wykorzystaniem arytmetyki. W tym celu litery alfabetu od A do Z trzeba zapisać za pomocą liczb od 0 do 25. W takim przypadku 0 odpowiada literze A, 1 odpowiada literze B itd., aż do liczby 25, która odpowiada literze Z. Te przypisania pokazałem na rysunku 1.5.

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Rysunek 1.5. Numerowanie od 0 do 25 liter alfabetu

Ten kod „litera na liczbę” można wykorzystać do przedstawienia liter. Jest to potężna koncepcja, ponieważ pozwala przeprowadzać operacje matematyczne na literach. Przykładowo jeśli litery CAT odpowiadają liczbom 2, 0 i 19, można do tych liczb dodać 3 i tym samym otrzymać 5, 3 i 22. Te nowe liczby odpowiadają literom FDW, jak pokazałem na rysunku 1.5. „Dodaliśmy” jedynie 3 do słowa *cat*. W dalszej części książki utworzymy program, aby to komputer za nas wykonywał odpowiednie operacje matematyczne.

Jeżeli chcesz skorzystać z arytmetyki do szyfrowania za pomocą szyfru Cezara, odszukaj liczbę odpowiadającą literze przeznaczonej do szyfrowania, a następnie dodaj do niej wartość klucza. W efekcie otrzymasz sumę reprezentującą zaszyfrowaną literę. Przykładowo szyfrujemy wiadomość HELLO. HOW ARE YOU? z użyciem klucza 13. (Jako klucza można użyć dowolnej liczby z przedziału od 1 do 25). Zaczynamy od odszukania litery H, która odpowiada liczbie 7. Po dodaniu do niej klucza, 13, otrzymujemy wartość 20. Skoro liczbie 20 odpowiada litera U, zaszyfrowane H to U.

Podobnie szyfrujemy literę E (4), dodajemy wartość klucza i otrzymujemy liczbę 17, której odpowiada litera R. Zaszyfrowane E to zatem litera R. Dalej postępujemy według tej samej procedury.

Proces działa świetnie aż do litery O, której odpowiada liczba 14. Po dodaniu klucza, 13, otrzymujemy liczbę 27, a lista kończy się na wartości 25. Jeżeli suma litery i klucza jest większa lub równa 26, wówczas trzeba od niej odjąć 26. W omawianym przykładzie mamy  $27 - 26 = 1$ . Literą odpowiadającą liczbie 1 jest B, więc litera O po zaszyfrowaniu będzie literą B. Po zaszyfrowaniu całej wiadomości otrzymamy szyfrogram: URYYB. UBJ NER LBH?.

Aby deszyfrować szyfrogram, wartość klucza trzeba odejmować, a nie dodawać. Literze B odpowiada liczba 1. Po odjęciu klucza, 13, otrzymujemy -12. Podobnie jak w przypadku reguły „odejmij 26” stosowanej przy szyfrowaniu, gdy podczas deszyfrowania wynik jest mniejszy niż 0, trzeba do niego dodać 26. Skoro  $-12 + 26 = 14$ , to literze B szyfrogramu odpowiada litera O.

**UWAGA** *Jeśli nie wiesz, jak dodawać i odejmować liczby ujemne, więcej informacji na ten temat znajdziesz na stronie [https://inventwithpython.com/chapter12.html#\\_Toc405145134](https://inventwithpython.com/chapter12.html#_Toc405145134).*

Jak widać, do odszyfrowania wiadomości zabezpieczonej szyfrem Cezara nie musisz korzystać z krążka szyfrowania. Potrzebujesz jedynie papieru, ołówka i nieco prostej arytmetyki.

## Dlaczego podwójne szyfrowanie nie działa?

Być może sądzisz, że dwukrotne zaszyfrowanie wiadomości z użyciem dwóch różnych kluczy może podwoić siłę szyfrowania. Jednak to nieprawda w przypadku szyfru Cezara (a także wielu innych szyfrów). Ponadto wynik podwójnego szyfrowania jest taki sam jak w przypadku zwykłego pojedynczego szyfrowania. Spróbujmy dwukrotnie zaszyfrować wiadomość, aby się przekonać, dlaczego tak się dzieje.

Przykładowo jeśli zaszyfrujemy słowo KITTEN z użyciem klucza 3, będziemy dodawać 3 do liczby reprezentującej literę wiadomości w postaci zwykłego tekstu i otrzymamy szyfrogram NLWWHQ. Gdy teraz ten ciąg tekstowy zaszyfrujemy ponownie, ale z użyciem klucza 4, otrzymamy szyfrogram RPAALU, ponieważ dodajemy 4 do liczby reprezentującej literę wiadomości w postaci zwykłego tekstu. Jednak ma to taki sam efekt jak jednokrotne zaszyfrowanie słowa KITTEN kluczem 7.

W przypadku większości szyfrów wielokrotne szyfrowanie nie przekłada się na jego większą siłę. A jeśli zaszyfrujesz tekst za pomocą dwóch kluczy o sumie wynoszącej 26, wynikowy szyfrogram będzie miał dokładnie taką samą postać jak wiadomość w postaci zwykłego tekstu!

## Podsumowanie

Szyfr Cezara i inne szyfry były na przestrzeni wieków stosowane do szyfrowania tajnych wiadomości. Gdyby ktoś chciał zaszyfrować długi komunikat, np. całą książkę, taka operacja przeprowadzana ręcznie mogła zająć wiele dni lub tygodni. Tutaj z pomocą może przyjść programowanie. Komputer umożliwi zaszyfrowanie i deszyfrowanie ogromnych ilości tekstu w czasie poniżej sekundy!

Aby wykorzystać komputer do szyfrowania, należy go *zaprogramować*, czyli nakażać mu wykonanie dokładnie tych samych kroków, ale w postaci zrozumiałej dla maszyny. Na szczęście poznanie języka programowania takiego jak Python jest znacznie łatwiejsze niż nauczanie się języka obcego, np. japońskiego lub hiszpańskiego. Ponadto nie trzeba mieć zbyt dużej wiedzy z zakresu matematyki — wystarczy umiejętność dodawania, odejmowania i mnożenia. Musisz mieć do dyspozycji komputer i tę książkę!

Przejdźmy teraz do rozdziału 2., z którego dowiesz się, jak używać powłoki interaktywnej Pythona w celu wykonywania pojedynczych wierszy kodu.

## ĆWICZENIA PRAKTYCZNE

Odpowiedzi do ćwiczeń zostały zamieszczone w dodatku B.

1. Zszyfruj wyrażenia za pomocą podanego klucza.
  - a) Able to pick with equal skill a right-hand pocket or a left, klucz 4: AMBIDEXTROUS
  - b) A machine which makes a Frenchman shrug his shoulders with good reason, klucz 17: GUILLOTINE
  - c) Your irreverence toward my deity, klucz 21: IMPIETY
2. Deszyfruj szyfrogramy za pomocą podanego klucza.
  - a) P RDHIJBT HDBTIXBTH LDGC QN HRDIRWBTC XC PBTGXR PCH XC HRDIAPCS, klucz 15: ZXAI
  - b) E VMZEP EWTMVERX XS TYFPMG LRSRVW, klucz 4: MQTSWXSV
3. Zszyfruj zdanie This is a silly example z użyciem klucza 0.
4. Oto kilka słów i ich szyfrogramów. Jaki klucz został użyty do zaszyfrowania poszczególnych słów?
  - a) ROSEBUD – LIMYVOX
  - b) YAMAMOTO – PRDRDFKF
  - c) ASTRONOMY – HZAYVUVTF
5. Jakie zdanie zostało zaszyfrowane kluczem 8 i deszyfrowane kluczem 9?  
UMMSVMAA: Cvkwuuvw xibqmvkm qv xtivvqvo i zmdmvom bpib qa ewzbp epqtm





# PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion**



## A TERAZ STWÓRZ ALGORYTM SZYFRU IDEALNEGO!

Szyfrowanie do niedawna było wiązane z bezpieczeństwem publicznym. Najbezpieczniejsze implementacje podlegały takim samym rządowym regulacjom jak przemysł zbrojeniowy. Do dzisiaj rządy i różnego rodzaju służby dążą do uzyskania możliwości odczytywania zaszyfrowanych danych. Tymczasem silna kryptografia jest podstawą globalnej ekonomii, zapewnia codzienną ochronę milionom użytkowników i większości organizacji. A to nie wszystko. Algorytmy szyfrujące, ich implementacja czy programowe łamanie szyfrów to równocześnie fascynująca dziedzina wiedzy i pole do zabawy, ćwiczeń oraz eksperymentowania z programowaniem.

Ta książka jest przeznaczona dla osób, które nie umieją programować, ale chciałyby zapoznać się z kryptografią. Omówiono tu podstawowe koncepcje programowania w Pythonie, który dziś jest uważany za najlepszy język dla początkujących koderów. Pokazano, jak tworzyć, testować i łamać programy implementujące szyfry klasyczne, takie jak przestawieniowy i Vigenère'a, by stopniowo przejść do znacznie bardziej zaawansowanych zagadnień, w tym kryptografii klucza publicznego. Każdy program przedstawiono w postaci pełnego kodu źródłowego, wyjaśniono także wiersz po wierszu jego działanie. Dzięki tej książce można się zarówno nauczyć

zasad kryptografii, jak i zdobyć umiejętności pisania kodu szyfrującego i deszyfrującego w Pythonie.

### Znajdziesz tutaj między innymi:

- wprowadzenie do programowania w Pythonie: pętle, zmienne, kontrola przepływu działania programu
- omówienie technik szyfrowania stosowanych przed wynalezieniem komputerów
- różne algorytmy szyfrowania z wykorzystaniem Pythona
- testowanie programów szyfrujących i deszyfrujących
- szyfrowanie i deszyfrowanie plików
- łamanie szyfrów techniką *brute force* czy analiza częstotliwości

**Al Sweigart** jest zawodowym programistą, projektantem UI i autorem cenionych książek technicznych. Jego ulubionym językiem programowania jest Python. Uważany za utalentowanego nauczyciela kodowania, twórca wielu samouczków w tej dziedzinie. Obecnie mieszka w San Francisco.

**Helion**

helion.pl

HELION SA  
ul. Kościuszki 1c  
44-100 Gliwice  
tel.: 32 230 98 63  
helion@helion.pl

Sprawdź nasze szkolenia!

SZKOLENIA



AKADEMIA IT & BUSINESS

HELIONSZKOLENIA.PL

KOD KORZYŚCI  
Sięgnij po więcej! ▶



ISBN 978-83-283-7495-9



9 788328 374959

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 89,00 zł

