

Module doc (index.html) «

Functions

base\_password

check\_password

decoder

encoder

generate

main

Variables

SPECIAL\_CHARS

doc (index.html) Show Private API

module documentation

Основной модуль программы для генерации, проверки и шифрования паролей.

Этот файл объединяет несколько функций:

- генерация пароля (через generate)
- проверка пароля (через check\_password и base\_password)
- шифрование/дешифрование строки (через encoder и decoder)

Программа запускается в интерактивном режиме и предлагает выбрать режим работы.

|          |                |  |
|----------|----------------|--|
| Function | base_password  | Проверка, есть ли пароль в базе популярных паролей.                              |
| Function | check_password | Проверка надежности пароля и наличия повторяющихся символов.                     |
| Function | decoder        | Дешифрует hex-строку, зашифрованную функцией encoder, используя тот же XOR-ключ. |
| Function | encoder        | Шифрует строку с использованием операции XOR и возвращает результат в виде hex.  |
| Function | generate       | Генерация случайного пароля заданной сложности и длины.                          |
| Function | main           | Главная функция программы.   |

|          |               |                     |
|----------|---------------|---------------------|
| Constant | SPECIAL_CHARS | <i>Undocumented</i> |
|----------|---------------|---------------------|

**def base\_password(password: str) -> int :**

Проверка, есть ли пароль в базе популярных паролей.

Аргументы:

password (str): Пароль пользователя для проверки.

Возвращает:

int:

2 – если пароль найден в базе популярных паролей (не надежный)

1 – если пароль не найден в базе (надежный)

**def check\_password(user\_password: str) -> tuple[str, bool] :**

Проверка надежности пароля и наличия повторяющихся символов.

Аргументы:

user\_password (str): Пароль, который нужно проверить.

Возвращает:

tuple[str, bool]:

- Надежность пароля как строка ("очень слабый", "слабый", "средний", "хороший", "очень хороший")

- Строка "есть" или "нет", указывающая, есть ли повторяющиеся символы подряд

**def decoder(encoded\_hex, key) -> str :**

Дешифрует hex-строку, зашифрованную функцией encoder, используя тот же XOR-ключ.

Parameters

-----

encoded\_hex : str

Строка hex-значений, где каждые два символа представляют один байт.

key : int

XOR-ключ, использованный при шифровании. Должен совпадать с ключом encoder.

**def encoder(text, key) -> str :**

Шифрует строку с использованием операции XOR и возвращает результат в виде hex.

Parameters

-----

text : str

Исходная строка, которую нужно зашифровать.

key : int

Целочисленный ключ для XOR-операции. Должен быть в диапазоне 0-255.

**def generate(level: int, length: int) -> tuple[str, int] :**

Генерация случайного пароля заданной сложности и длины.

Аргументы:

level (int): Уровень сложности пароля:

1 - easy (только строчные буквы)

2 - medium (строчные + заглавные буквы + цифры)

3 - hard (medium + специальные символы)

length (int): Длина пароля (количество символов)

Возвращает:  
tuple[str, int]: Сгенерированный пароль и его длину

**def main():**

Главная функция программы.

Предоставляет пользователю выбор одного из трёх режимов работы:

1) Генерация пароля:

- выбор уровня сложности (1–3)
- ввод длины пароля
- автоматическая корректировка некорректного ввода
- генерация пароля через generate()

2) Проверка пароля:

- ввод пароля
- проверка по базе распространённых слабых паролей через base\_password()
- проверка надёжности пароля через check\_password()

3) Шифрование/дешифрование:

- выбор режима (шифрование/дешифрование)
- ввод строки и ключа
- выполнение encoder() или decoder()

Функция не принимает аргументов и запускается автоматически, если файл используется как точка входа.

**SPECIAL\_CHARS:** str =

*Undocumented*

**Value**

'!@#\$%^&\*()\_+-'