

Ruzsa Gergely Gábor (BR1GHH)

Óbudai egyetem

Neumann János Informatikai Kar

Playfair-rejtjel kódoló-dekódoló

Felhasználói- és fejlesztői dokumentáció

# Tartalomjegyzék

[Tartalomjegyzék 1](#_Toc113979268)

[A feladat és célok meghatározása 2](#_Toc113979269)

[A Playfair-rejtjel szabályai 2](#_Toc113979270)

[Felhasználói dokumentáció 2](#_Toc113979271)

[Felkészülés a használatra 2](#_Toc113979272)

[A szkript használata 3](#_Toc113979273)

[Paraméterek 3](#_Toc113979274)

[-h paraméter 3](#_Toc113979275)

[-k paraméter 3](#_Toc113979276)

[-d paraméter 4](#_Toc113979277)

[-t paraméter 4](#_Toc113979278)

[Fejlesztői dokumentáció 5](#_Toc113979279)

[Titkosítási folyamat 5](#_Toc113979280)

[Dekódolási folyamat 5](#_Toc113979281)

[Eljárások 5](#_Toc113979282)

[def kodtablaBeolvasas() 5](#_Toc113979283)

[def karakterKeres(k) 5](#_Toc113979284)

[def szovegTisztitas(be) 5](#_Toc113979285)

[def szovegTordeles(be) 5](#_Toc113979286)

# A feladat és célok meghatározása

A feladatunk ebben a félévben az, hogy a *Bevezetés az informatikába* tantárgy keretében kis-ZH mentességért cserébe a listában felsorolt titkosító vagy tömörítő algoritmusok közül valamelyiket implementálni kell. Az én választásom a Playfair-rejtjelre esett, hiszen ezzel már volt előzetesen némi ismeretem, de ettől függetlenül ez is sok utánajárást és tesztelést igényelt.

A feladat pontos követelményeket a szoftverrel kapcsolatban nem állított, így a saját követelményrendszeremnek megfelelően állítottam fel a célokat. A kitűzött céljaim a következők voltak:

* Megfelelően kinéző és használható felhasználói interfész
* Működő kódoló algoritmus
* Működő dekódoló algoritmus
* Háttérellenőrzések a bemenetekhez (pl. tisztasági vizsgálat)
* A páratlan betűkből álló szavak és kifejezések lekezelése

# A Playfair-rejtjel szabályai[[1]](#footnote-1)

A szöveg kódolásához az eredeti szöveget két betűből álló csoportokba kell rendezni (például a "Playfair-rendszer" "PL AY FA IR RE ND SZ ER" lesz), amelyeket a kulcstáblán kell elosztani, majd egymás után alkalmazni az alábbi 4 szabályt az eredeti szöveg minden betűpárjára:

* Ha egy pár mindkét eleme ugyanaz a betű, vagy már csak egy betű maradt az utolsó párba, akkor írjunk egy "X"-et az első betű után, és ezt az új párt kódoljuk. Néhány változat "X" helyett "Q"-t használ, de bármely ritka betű megteszi.
* Ha egy pár mindkét betűje ugyanabban a sorban jelenik meg a kulcstáblán, akkor a tőlük közvetlenül jobbra állóval kell helyettesíteni őket (ha történetesen az egyik betű a sor jobb szélén van, akkor a sor bal szélén álló betűvel kell helyettesíteni).
* Ha egy pár mindkét betűje ugyanabban az oszlopban jelenik meg a kulcstáblán, akkor közvetlenül az alattuk állóval kell helyettesíteni őket (ha az egyik betű az oszlop alján van, akkor az oszlop tetején álló betűvel kell helyettesíteni).
* Ha egy pár betűi nincsenek sem egy sorban, sem egy oszlopban, akkor tekintsük azt a kulcstábla mezőiből felépülő téglalapot, amelynek a két betű a két szemközti csúcsa. A betűket a saját sorukban, a téglalap másik csúcsánál lévő betűkkel helyettesítjük.

# Felhasználói dokumentáció

## Felkészülés a használatra

A szkript Python nyelvben készült, így a használatához szükségünk lesz egy Python értelmezőre, amit könnyedén le tudunk tölteni az internetről a [*https://python.org*](https://python.org)webcímen. A kód minden Python 3.x kiadáson működik, így a legfrissebben is.

Mindemellett szükségünk van arra is, hogy a kódban importált minden függőség (*dependency*) ki legyen elégítve.

* Amennyiben **van** PIP külön telepítve a gépünkre, úgy a *pip install -r requirements.txt* parancsot adjuk ki.
* Amennyiben **nincs** PIP telepítve, úgy a *python -m pip install -r requirements.txt* parancsot adjuk ki!

*(A felhasznált függőségeket a szkript első soraiban, vagy a requirements.txt fájlban olvashatja el.)*

## A szkript használata

Amennyiben az előző fejezetben írtakat sikeresen abszolváltuk, készen is állunk a használatára. A szkript paraméterekkel állítható annak függvényében, hogy kódolni vagy dekódolni szeretnénk. Emellett tartalmaz még két másik kapcsolót melyek nem tartoznak hozzá a kódolás és dekódolás folyamatához, de kiegészítik a felhasználói élményt.

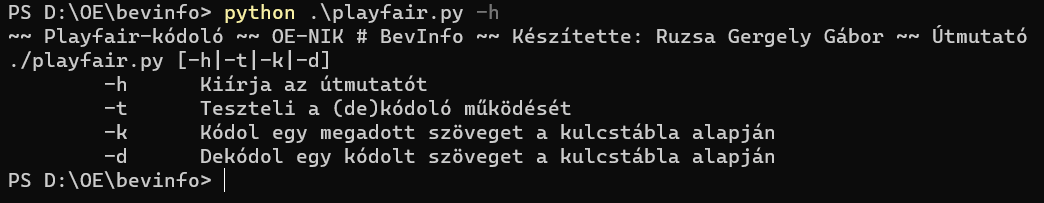
* -h Kiírja a szkripthez tartozó útmutatót
* -t Elindítja a kódoló/dekódoló öntesztet
* -k Elindítja a szkriptet kódoló üzemmódban
* -d Elindítja a szkriptet dekódoló üzemódban

**Figyelem!** A szkript paraméter nélküli elindítása megegyezik azzal, mintha megadtuk volna a „-h” paramétert. Amennyiben ismeretlen, nem implementált paramétert adunk meg, úgy egy hibaüzenet figyelmeztet erre.

## Paraméterek

### -h paraméter

A „-h” paraméter esetén megjelenik egy alapvető útmutató a konzolon, majd bezárul a szkript. Ez akkor hasznos, ha nem jut eszünkbe a többi kapcsoló.



### -k paraméter

Ez a paraméter szolgál arra, hogy a kódoló részét előhozzuk a programnak. A parancs kiadása után a szkript kér egy betűpárt, szót, vagy kifejezést, amit titkosítani szeretnénk.

Tisztításon megy át a bemenet, így teljesen mindegy, hogy ékezettel, írásjelekkel vagy azok nélkül írunk be kis vagy nagy betűket, így is képes lesz átkódolni. Emellett a „jelző betűt” felhasználva kompenzálja a szoftver, ha páratlan számú betűt írunk be (erről későbbiekben olvashat).

A képen szöveg látható

Automatikusan generált leírás

### -d paraméter

Ez a paraméter ugyanazt csinálja, mint a -k paraméter, csak visszafelé, tehát dekódolja a karaktereket. A dekódolásban megadott karaktereket nem „tisztítja meg” a szkript, tehát amennyiben itt nem pontosan nagy betűkkel megkapott rejtjelet olvastatunk be, a szkript egyszerűen csak hibát jelez és bezárul.

A képen szöveg látható

Automatikusan generált leírás

### -t paraméter

Ez a paraméter a szkript ellenőrzésére szolgál. Előre programozott kódolandó szöveg és kódolt szövegek által önellenőrzést folytat le a kódoló és dekódoló metódusokon a szkript. Ezt olyan formán teszi, hogy közben minden lépést (szöveg tisztítás -> szöveg tördelés -> kódolás/dekódolás) láthatóvá tesz a konzolon.

A képen szöveg látható

Automatikusan generált leírás

## Fejlesztői dokumentáció

### Titkosítási folyamat

A szkript több lépcsőben jut el a titkosítás folyamatáig, illetve a kiiratáshoz. A könnyebb megértés érdekében az alábbi folyamatábrát nézzük meg!

<ÁBRA>

### Dekódolási folyamat

A dekódolási folyamat kevesebb lépcsőből áll, mert nincsen bemenetellenőrzés és tisztítás. Továbbiakban a dekódoló algoritmus ugyanaz, mint a kódoló csak visszafelé. Ehhez is készült egy ábra.

<ÁBRA>

### Függvények

#### def kodtablaBeolvasas()

Ez a függvény felelős azért, hogy a kódtáblát beolvassa egy szövegfájlból, melyet egy kétdimenziós tömbben tárolunk le. Egyelőre statikusan bele van kódolva a *kodtabla1.txt* fájlnév, de a jövőben lehetne implementálni, hogy paraméterből vegye át a kódtábla nevét, valamint jelenleg nincs ellenőrizve a karaktertábla tartalma.

#### def karakterKeres(k)

Ez a függvény felelős azért, hogy a program további részei meg tudják találni a kódtáblánkban az egyes betűk indexeit.

A **k** változó a karakter, amit keresni akarunk (tehát char/string típusú bemenetet várunk). A függvény visszatérési értéke **(x,y)** formában a sora és oszlopa a keresett betűnek ahol elhelyezkedik. (Amennyiben nem találja meg a betűt a listában, úgy **(-1, -1)** lesz a visszatérési érték.

#### def szovegTisztitas(be)

A **be** változó a bemeneti értékünk, amit meg kell tisztítani (tehát string). Ebben az eljárásban 3 szűrő követi egymást. Ezek sorrendben:

1. Eltávolítunk belőle minden speciális karaktert (az re könyvtár használatával)
2. Majd az ékezetes betűket ASCII megfelelőkre konvertáljuk (unidecode könyvtárral)
3. Ezek után minden betűt nagybetűsre cserélünk
4. Majd ellenőrizzük, hogy páratlan-e a bevitt szöveg -> ha igen akkor hozzáadunk egy X-et
5. Végezetül ellenőrizzük, hogy két ugyanolyan betű követi-e egymást -> ha igen akkor kicseréljük egy X karakterre

Visszatérési értéke a tisztított szöveg.

#### def szovegTordeles(be)

A **be** változó a bemeneti értékünk, amit meg kell tisztítani (tehát string). Ebben az eljárásban bontjuk fel a *szovegTisztitas()* eljárásból megkapott stringet betűpárokra (2 betű), melyet végül át tudunk kódolni. Minden betűpár egy-egy tömb amit végül hozzáadunk egy másik tömb.

1. Forrás: <https://hu.wikipedia.org/wiki/Playfair-rejtjel> [↑](#footnote-ref-1)