

SUPINFO Academic Dept.

**Arithmetic and Cryptography**

Mini - Project

Programming cryptographic algorithms

Version 1.0

Last update: 19/02/2016

Use: Students/Staff

Author: Laurent GODEFROY

CONTENTS

0 introduction 3

1 FIGURE OF THE BOOK 4

1.1 Generalities on this figure 4

1.2 Python implementation 5

2 PLAYFAIR FIGURE 6

2.1 Generalities on this figure 6

2.2 Python implementation 7

3 COLON FIGURE 9

3.1 Generalities on this figure 9

3.2 Python implementation 10

# introduction

This examination is to be carried out in groups of two students. In the only case where the number of students per class is odd, one and only one group of three is allowed

Any form of plagiarism or use of codes available on the internet or any other format, even partially, is strictly prohibited and will be penalized with a 0 and will be referenced as "cheater" and will be summoned to a disciplinary committee.

You must send your project by email to your trainer before Sunday 20 mars 2016 at 23:59 local time. Beyond this date and time your score will be 0. You will compress your source code and images in a ".zip" archive format and name respecting **ID-1ARI-votreCampus-MP** convention (example: 66280 -1ARI-Tours-MP). The subject of the mail will have the same name. You also put in carbon copy 1ARI@supinfo.com email address. If you do not put it address a copy any subsequent claim will be acceptable.

This mini-project will give rise to defenses which will take place the week of 21 mars 2016. Your passage schedule will be communicated to you by your campus.

The defenses are also in groups of two. They will last 20 minutes during which you will show to your examiner the smooth running of your program by making the demonstration. If you have not implemented the entire project, you will expose the functional parts.

To support your presentation, you must prepare a PowerPoint file in which you will explain the most important and significant points of the code. It will not be necessary to send this file to your reviewer, they will discover it the day of the defense. A statement specifying all of this will be sent to early February.

Another note, **do not hesitate to use the forum to exchange about this project; I opened a thread for this purpose**.

We will inevitably have to use the graphic Pygame bookshop in the part 2. The use of another bookshop will not be taken into consideration.

If desired, we can add functions and procedures to those requested.

**Important Note:** sub-programs written in one part can of course be imported into another.

# Figure of the book

## Generalities on this figure

**Important note:** no code is requested in this paragraph which is only explanatory.

The purpose of this part is to implement the figure of the book, also called the system dictionary, which is a figure of homophonic substitution.

To perform an encryption, you must provide a key, which is here a fairly long text, historically a fixed page of a book.

We will encrypt each letter of the plaintext by randomly taking indications of the key letter. Indexing is done by not taking into account spaces, accented letters, punctuation marks, etc.

If some letters are not included in the key, we will encrypt them with the numbers following the last index.

**Example:** we will take as key the presentation text of SUPINFO campus in Tours.

« The SUPINFO institution of higher education in Tours, located in the city center, offers a more appropriate workspace and relaxation area. The existence of a SUPINFO Campus in Tours allows the young inhabitants of Tours but also the young people of Indre-et-Loire to study the computer science and digital technology while staying near their family.

The administrative team of the Campus makes it a point of honor to accompany the SUPINFO students throughout their program and takes to heart to place them in the best learning conditions to lead each of them to the success. »

We want to encrypt "SUPINFO". The 'S' may be encrypted by 6,7,16,26,34,45, ..., 449 or 450. The letter 'U' by 27,32,35,43, ..., 443 or 448. Etc.

So a possible encryption of "SUPINFO" is 26 144 112 330 159 140 143.

Note that the letters 'K', 'W', 'X' and 'Y' did not appear in the text served to establish the key. They would have been encrypted in 454, 455, 456 and 457 since the last index of the text was 453.

The decryption is immediate.

## Python Implementation

**It is strongly recommended that you read the entire of this part before beginning to code. The work required is highlighted with a blue color.**

**In a file which will be called "chiffreDuLivre.py" implement the following subprograms:**

* A function taking a string parameter, which returns it after you delete the spaces, punctuation marks, accented characters, etc...
* A function taking a string parameter, which returns a dictionary whose keys are the capital letters of the alphabet. The value associated with a key will be the list of the indications of this key chain. To calculate the indications, we will not take into account spaces, punctuation marks, accentuated letters, etc.
* A function parameter taking a letter and a key (example: a dictionary of the previous form), which returns a number encrypting the letter according to the digit Book algorithm.
* A function taking a parameter and a key text (example: one of the preceding form dictionary), and returns a list of numbers encrypting the text according to the book figure algorithm.
* Similar to the previous two functions but this time allowing the decryption.

**Using these procedures and functions:**

* Decipher the text 'figure 1.txt' with key 'cle1.txt'. We will access the data in these text files with the usual playback functions, and not via copy / paste in the script.
* Encrypt a text of your choice with a key of your choice and also attach the key and the encrypted text to your ".txt" project files.

**Two small questions:**

* What do you think of the safety of this algorithm?
* What are the main qualities and defects?

**Bonus:**

* Improve the above to take into consideration the spaces and accented letters without having to remove them.

# PLAYFAIR FIGURE

## Generalities on this figure

**Important note:** no code is asked in this paragraph which is only explanatory.

The purpose of this part is to implement the Playfair figure, which is a polygramic substitution figure. Let us recall how it works.

To perform encryption, you must provide a key, which here is a breakdown of 25 letters of the alphabet (all the letters except the 'w') in a square of 5 boxes of 5. For example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s | e | v | m | a |
| t | k | p | y | n |
| l | x | z | b | h |
| d | q | i | c | u |
| j | f | o | r | g |

In the text to be encrypted we are going to replace first of all the ' w ' by ' v ', then we are going to cut the text in blocks of two letters. If a bigram obtained is composed of two identical letters, we will insert an 'x' between of them. Once this is done, if the number of letters of the text is odd, we will add at the end an 'x' to be the last bigram.

**For example,** the "hello world" string will be transformed into "he lx lo vo Services dx".

Here are the rules of an encryption bigram:

* If the two letters to be encrypted are on the opposite summits of a rectangle, the letters will be encrypted on the other two summits. So the bigram "he" will be encrypted in "XA". Note that the first of the two letters is always encrypted on the same line as the first of the two letters in the clear. And "eh" would be encrypted in "AX".
* If both letters are located on the same line we can take the two letters located immediately to their right. If both letters are located on the same column, we wll take the two letters located just below them. For example "lx" will be encrypted in "XZ" and "vo" will be encrypted in "PV".

**Following of the example:** the string "hello world" transformed at first in "he lx lo vo rl dx" will be encrypted with the previous key in "XA XZ ZJ PV JB QL". That we can reshape by using the usual conventions: "XAXZZ JPVJB QL".

The decryption is obtained by applying these rules "upside down".

## Python implementation

**It is strongly recommended that you read the entire of this part before beginning to code. The work required is highlighted with a blue color.**

**In a file which will be called "chiffreDePlayfayr.py" implement the following subprograms:**

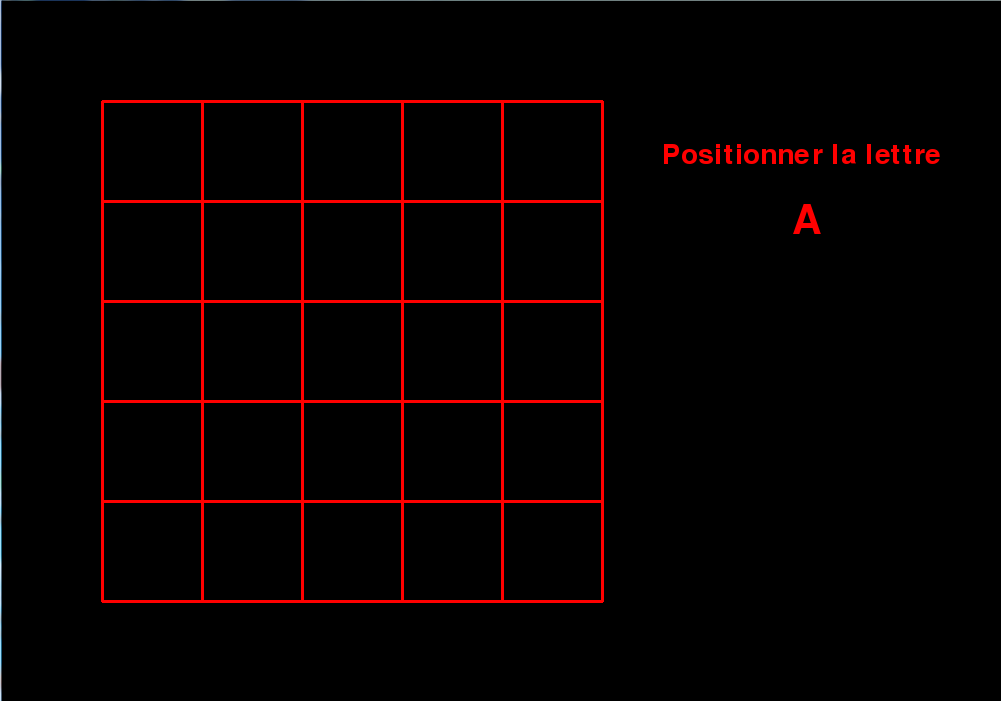
* A function taking a parameter string, and returns it after replacing each of the "w" (if there are any) by a 'v'.
* A function taking a string parameter, which returns it after inserting wisely the 'x' in order to avoid having two identical letters bigrams.
* A function taking a string parameter, which returns after adding an 'x' at the end if the number of letters is odd.
* A function taking a parameter string consisting of 25 lowercase letters of the alphabet without 'w' (each of the 25 letters appearing only once). It returns a list with two dimensions of 5 rows and 5 columns, filled row by row with the string parameter. Example "sevmatkpynlxzbhdqicujforg" string will be the square of the previous subsection.
* A function taking a parameter list with two dimensions of 5 rows and 5 columns, and a character which will be assumed present once and only once in the list. It returns the indications of rows and character columns in the list.
* A function that takes as parameters two pairs of indices between 0 and 4, and returns two pairs of indices between 0 and 4, calculated according to the rules of Playfair figure. For example, if [2,4], [0,1] are recognized as parameters the function will return [2,1], [0,4]. In the case of [0,2], [4,2] we obtain [1,2], [0,2].
* A function taking a parameter bigram and a key (example a list of the previous form), and returns the encrypted bigram according to Playfayr algorithm.
* A function taking a parameter and a key text (example a list of the previous form), and returns the encrypted text according to the Playfayr algorithm. We will state it according to the usual conventions.
* See how to use / change the above functions to perform the decryption of a bigram and a text.

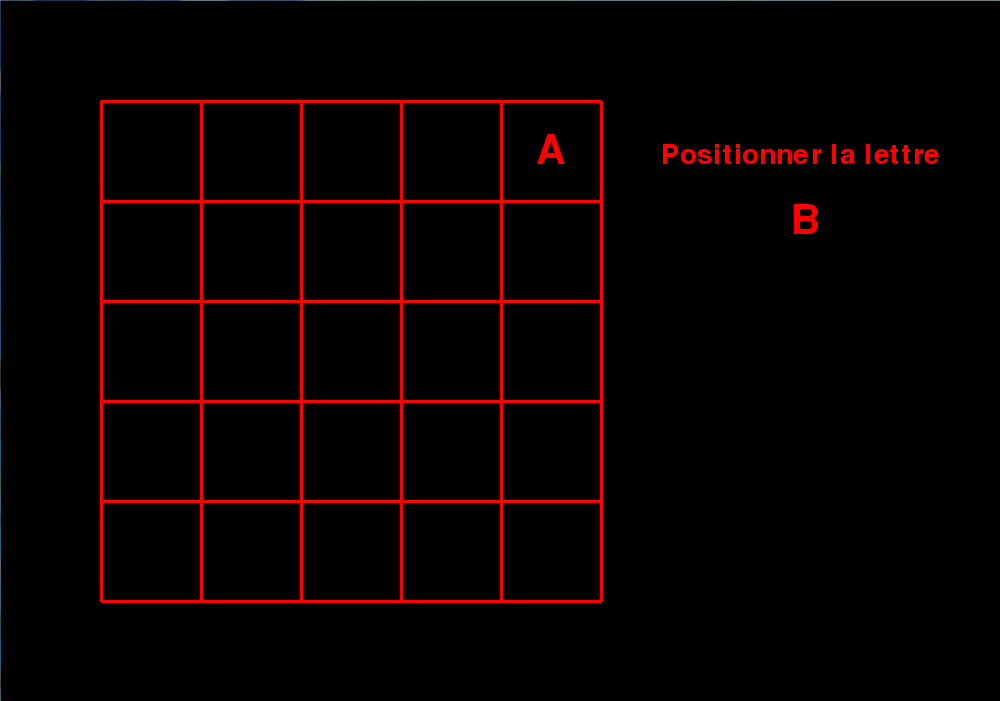
**Using these procedures and functions:**

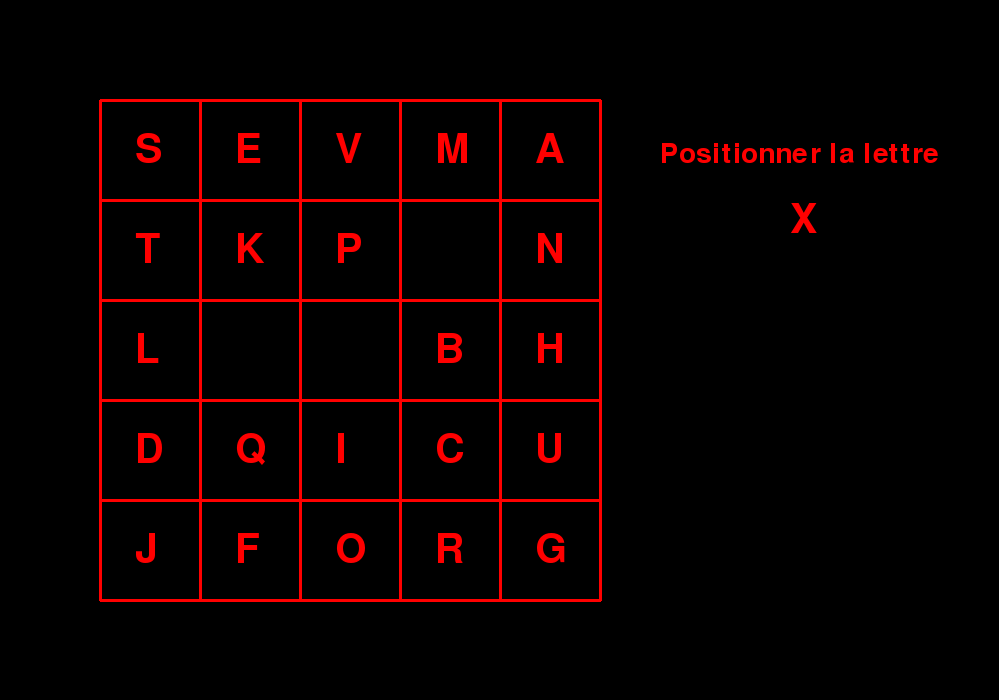
* Decipher the text 'figure 2.txt' with key 'cle2.txt'. We will access the data in these text files with the usual playback functions, and not via copy / paste in the script.
* Encrypt a text of your choice with a key of your choice and also attach the key and the encrypted text to your ".txt" project files.

**Complement (not optional): a GUI to enter the key**

Implement a graphical interface to enter the key letter by letter, by clicking on the desired position. Here is an example:







**Two small questions:**

* What do you think of the safety of this algorithm?
* What are the main qualities and defects?

# Colon FIGURE

## Generalities on This figure

**Important note:** no code is asked in this paragraph which is only explanatory.

The purpose of this section is to implement the Colon figure, which is a figure of tomogrammique substitution. Let us recall how it works

To perform an encryption, you must provide a key, which is here a integer, for example 7 and a distribution of 25 letters of the alphabet (all the letters except the 'w') in a square of 5 boxes of 5 .

**For example:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s | e | v | m | a |
| t | k | p | y | n |
| l | x | z | b | h |
| d | q | i | c | u |
| j | f | o | r | g |

In the text to be encrypted we are first going to replace all the 'w' with 'v'.

We will then encrypt each of the letters of the text by a bigram by doing this: we start by identifying the I number of the line of the j letter and column number of the letter. Then form a bigram with the i-th element of the first column and the j-th element of the last row.

**Example:** 'h' is encrypted in "LG", "e" is encrypted in "SF", 'l' is encrypted in "LJ**".**

To encrypt a text, we will encrypt every letter in bigrams that are written on two lines, the first letter of the bigram on the first line and the second on the second. Then onto 7 characters (remember that the integer 7 is part of the key) of the first line, and 7 of the second, then another 7 of the first etc.

**Example**: to encrypt "jumping jack flash" First we get

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| clair | j | u | m | p | i | n | g |  | j | a | c | k | f | l | a |  | s | h |
| 1ère | J | D | S | T | D | T | J |  | J | S | D | T | J | L | S |  | S | L |
| 2ème | J | G | R | O | O | G | G |  | J | G | R | F | F | J | G |  | J | G |

Where from the encryption "JDSTDTJ JGROOGG JSDTJLS JGRFFJG SLJG" you can get using the usual conventions: "JDSTD TJJGR OOGGJ SDTJL SJGRF FJGSL JG".

The decryption is obtained by applying these rules, "upside down" we will reconstitute each of the two lines, then each of the bigrams, which finally converts into letters.

## Python Implémentation

**It is strongly recommended that you read the entire of this part before beginning to code. The work required is highlighted with a blue color.**

**In a file which will be called "chiffreDeCollon.py" implement the following subprograms:**

* A function taking a string parameter, and returns a list with two dimensions of 5 rows and 5 columns, filled row by row with the string parameter. It removes any duplicate items, and if the string is "too short" we will complete the square with the missing letters of the alphabet taken in the right order. For example, if you pass "academic service" parameter, we obtain the square:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s | e | r | v | i |
| c | a | d | m | q |
| u | b | f | g | h |
| j | k | l | n | o |
| p | t | x | y | z |

* A function taking a key and a letter parameter (ie a list of the form shown in section 2.2), and returns the encrypted letter in the form of a bigram according to the Colon algorithm.
* A function taking a key and a letter parameter (ie a list of the form shown in section 2.2 and an integer), and returns the encrypted text according to the Colon algorithm. We will state it according to the usual conventions.
* A function taking a bigram and a key parameter (ie a list of the form shown in section 2.2), and returns the letter obtained by decrypting the bigram according to the Colon algorithm.
* A function taking a key and text parameter (ie a list of the form shown in section 2.2 and an integer), and returns the decrypted text according to the Colon algorithm.

**Using these procedures and functions:**

* Decrypt the text 'figure 3.txt' with the key generated from the word "academic service" and the integer 7. It will access the data in this text file with the usual playback functions, and not via a copy / paste in the script.
* Encrypt a text of your choice with a key of your choice and also attach the key and the encrypted text to your project in the form of ".txt" files.

**Two small questions:**

* What do you think of the safety of this algorithm?
* What are the main qualities and defects?