MP09

UF03

PROPOSED EXERCISES

00. Create a program in Python language, using the imperative procedural programming paradigm, that encrypts messages by replacing vowel-number (the number one by the vowel a, the number two by the vowel e, the number three by the vowel i, etc). Also create the program that performs the reverse process.

Examples of execution (Ubuntu 24.04 and Python 3.12.3):

```
ricard@HP-ProBook:~$ python3 vowels_to_numbers.py
Please provide a message as a command line argument.
ricard@HP-ProBook:~$ python3 vowels_to_numbers.py 'hello world!'
h2ll4 w4rld!
ricard@HP-ProBook:~$ python3 numbers_to_vowels.py
Please provide a message as a command line argument.
ricard@HP-ProBook:~$ python3 numbers_to_vowels.py 'h2ll4 w4rld!'
hello world!
ricard@HP-ProBook:~$ python3 vowels_to_numbers.py XMARKSTHESPOT
xm1rksth2sp4t
ricard@HP-ProBook:~$ python3 numbers_to_vowels.py xm1rksth2sp4t
xmarksthespot
ricard@HP-ProBook:~$
```

The source code of vowels_to_numbers.py is:

```
import sys

def encrypt(message):
    ciphertext = ""
    for letter in message:
        position = "aeiou".find(letter.lower())
        if position > -1:
            ciphertext += str(position + 1)
        else:
            ciphertext += letter.lower()

    return ciphertext

if len(sys.argv) > 1:
    plaintext = sys.argv[1]
    print(encrypt(plaintext))
else:
    print("Please provide a message as a command line argument.")
```

The source code of numbers_to_vowels.py is:

```
import sys

def decrypt(message):
    plaintext = ""
    for letter in message:
        position = "12345".find(letter)
        if position > -1:
            plaintext += "aeiou"[position:position + 1]
        else:
            plaintext += letter.lower()

    return plaintext

if len(sys.argv) > 1:
    ciphertext = sys.argv[1]
    print(decrypt(ciphertext))
else:
    print("Please provide a message as a command line argument.")
```

A program that does the same thing through object oriented programming paradigm is:

```
import sys

class Message:
    def __init__(self, op, te):
        ops = ['e', 'd']
        if op not in ops:
            print("The indicated operation is incorrect.")
            sys.exit()
        if op == ops[0]:
           self.plaintext = te
           self.ciphertext = ""
        if op == ops[1]:
            self.plaintext = ""
            self.ciphertext = te
    def get_plaintext(self):
        return self.plaintext
    def get_ciphertext(self):
        return self.ciphertext
    def set_plaintext(self, p):
        self.plaintext = p
    def set_ciphertext(self, c):
        self.ciphertext = c
    def encrypt_plaintext(self):
        for l in self.get_plaintext():
            p = "aeiou".find(l.lower())
            if p > -1:
                self.set_ciphertext(self.get_ciphertext() + str(p + 1))
            else:
                self.set_ciphertext(self.get_ciphertext() + l)
    def decrypt_ciphertext(self):
        for l in self.get_ciphertext():
            p = "12345".find(l.lower())
            if p > -1:
                self.set_plaintext(self.get_plaintext() + "aeiou"[p:p + 1])
            else:
                self.set_plaintext(self.get_plaintext() + l)

if len(sys.argv) > 2:
    operation, text = sys.argv[1], sys.argv[2]
    msg = Message(operation, text)

    if operation == 'e':
        msg.encrypt_plaintext()
        print(msg.get_ciphertext())
    if operation == 'd':
        msg.decrypt_ciphertext()
        print(msg.get_plaintext())
else:
    print("Please indicate the appropriate arguments.")
```

```
ricard@HP-ProBook:~$ python3 vowel-number.py
Please indicate the appropriate arguments.
ricard@HP-ProBook:~$ python3 vowel-number.py e 'hello world!'
h2ll4 w4rld!
ricard@HP-ProBook:~$ python3 vowel-number.py f 'h2ll4 w4rld!'
The indicated operation is incorrect.
ricard@HP-ProBook:~$ python3 vowel-number.py d 'h2ll4 w4rld!'
hello world!
ricard@HP-ProBook:~$
```

01. Create a program in Python language, using the imperative procedural programming paradigm, that encrypts and decrypts messages using the Atbash encryption method with the English alphabet.

Examples of execution:

```
ricard@HP-ProBook:~$ python3 atbash.py
Please provide a message as a command line argument.
ricard@HP-ProBook:~$ python3 atbash.py wehaveaninfiltratedspy
dvszevzmrmurogizgvwhkb
ricard@HP-ProBook:~$ python3 atbash.py dvszevzmrmurogizgvwhkb
wehaveaninfiltratedspy
ricard@HP-ProBook:~$ python3 atbash.py WEWILLATTACKNEXTTHURSDAY
dvdroozggzxpmvcggsfihwzb
ricard@HP-ProBook:~$ python3 atbash.py dvdroozggzxpmvcggsfihwzb
wewillattacknextthursday
ricard@HP-ProBook:~$
```

02. Create a program in Python language, using the imperative procedural programming paradigm, that encrypts and decrypts messages using the Polybius square cipher with the English alphabet and the following arrangement:

	1	2	3	4	5	6
1	a	b	c	d	e	f
2	g	h	i	j	k	l
3	m	n	o	p	q	r
4	s	t	u	v	w	x
5	y	z	0	1	2	3
6	4	5	6	7	8	9

It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

```
ricard@HP-ProBook:~$ python3 polybius.py
Provide the required arguments.
ricard@HP-ProBook:~$ python3 polybius.py e
Provide the required arguments.
ricard@HP-ProBook:~$ python3 polybius.py e wehaveaninfiltratedspy
45152211441511322332162326423611421514413451
ricard@HP-ProBook:~$ python3 polybius.py d 45152211441511322332162326423611421514413451
wehaveaninfiltratedspy
ricard@HP-ProBook:~$
```

03. Create a program in Python language, using the object oriented programming paradigm, that encrypts and decrypts messages using the Caesar cipher with various alphabets (English, Catalan and Spanish). It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

```
ricard@HP-ProBook:~$ python3 caesar.py eng e thequickbrownfoxjumpsoverthelazydog 1
uifrvjdlcspxogpykvnqtpwfsuifmbazeph
ricard@HP-ProBook:~$ python3 caesar.py eng d uifrvjdlcspxogpykvnqtpwfsuifmbazeph 1
thequickbrownfoxjumpsoverthelazydog
ricard@HP-ProBook:~$ python3 caesar.py eng e thequickbrownfoxjumpsoverthelazydog 3
wkhtxlfneurzqiramxpsvryhuwkhodcbgrj
ricard@HP-ProBook:~$ python3 caesar.py eng d wkhtxlfneurzqiramxpsvryhuwkhodcbgrj 3
thequickbrownfoxjumpsoverthelazydog
ricard@HP-ProBook:~$ python3 caesar.py cat e "tenimunespiainfiltrat" 4
xirmqyriwtmdmrjmpxvdx
ricard@HP-ProBook:~$ python3 caesar.py cat d "xirmqyriwtmdmrjmpxvdx" 4
tenimunespiainfiltrat
ricard@HP-ProBook:~$
```

04. Create a program in Python language, using the imperative procedural programming paradigm, that analyzes the frequency of letters in an encrypted message (original message is written in the English language) using a substitution algorithm discovers the letter e. It must be verified that the arguments of the program execution are acceptable.

Example of execution:

```
ricard@HP-ProBook:~$ python3 caesar.py eng e threisaplace 7 > ciphertext
ricard@HP-ProBook:~$ python3 caesar.py eng e sevenyardseastofthetarget 7 >> ciphertext
ricard@HP-ProBook:~$ python3 caesar.py eng e whatneedstobeprotected 7 >> ciphertext
ricard@HP-ProBook:~$ python3 caesar.py eng e mustnotbeconqueredbytheenemy 7 >> ciphertext
ricard@HP-ProBook:~$ cat ciphertext
aoylpzhwshjl
zlclufhykzlhzavmaolahynla
dohaullkzavilwyvaljalk
tbzauvailjvuxblylkifaollultf
ricard@HP-ProBook:~$ python3 freq_anal.py ciphertext
'a'  13.79%
'b'   2.30%
'c'   1.15%
'd'   1.15%
'f'   3.45%
'h'   6.90%
'i'   3.45%
'j'   3.45%
'k'   4.60%
'l'  20.69%
'm'   1.15%
'n'   1.15%
'o'   4.60%
'p'   1.15%
's'   1.15%
't'   2.30%
'u'   5.75%
'v'   5.75%
'w'   2.30%
'x'   1.15%
'y'   5.75%
'z'   6.90%

letter 'e' must be letter 'l'
ricard@HP-ProBook:~$
```

05. Create a program in Python language, using the imperative procedural programming paradigm, that decrypts through brute force the following message:

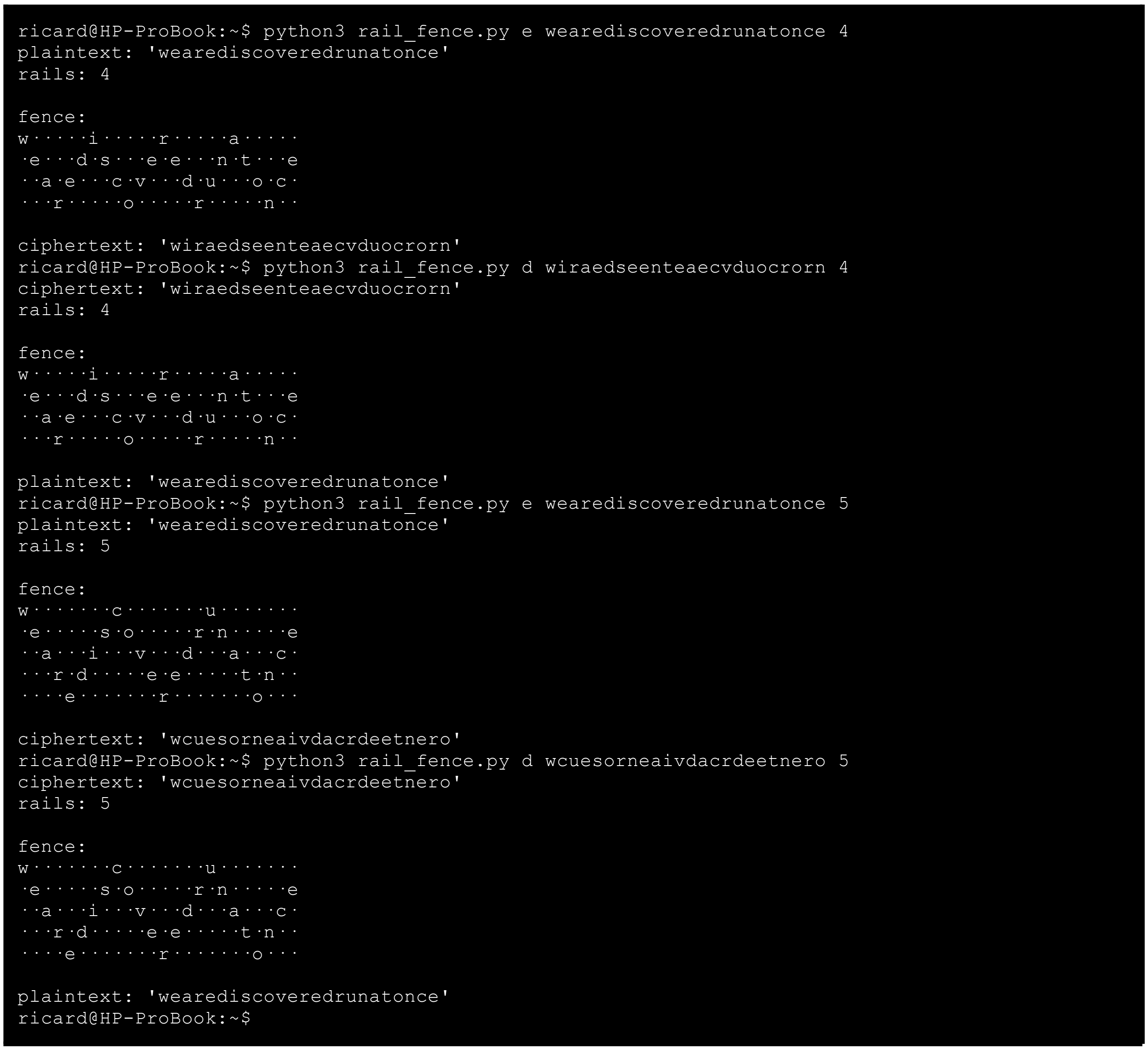
drobosckczisxpsvdbkdonkwyxqdrovsoedoxkxdc

For this, the following must be taken into account:

- The original message has been encrypted with Caesar cipher.
- The alphabet used in the original message is the English alphabet.
- The shift is unknown.
- The number of vowels with respect to the total number of letters in the original message exceeds 37.00%.

06. Create a program in Python language, using the imperative procedural programming paradigm, that encrypts and decrypts messages using the rail fence cipher with the English alphabet. It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

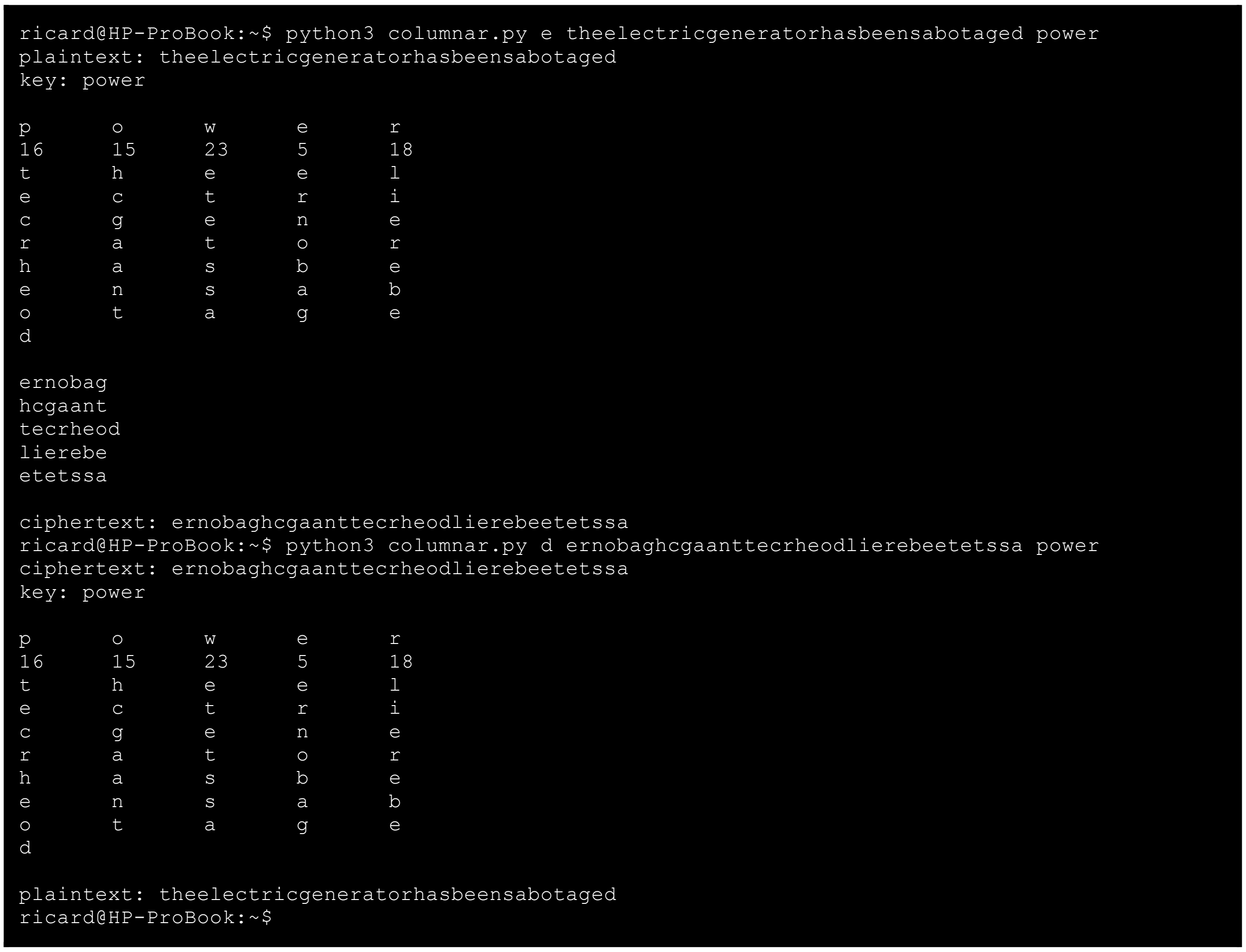


```
ricard@HP-ProBook:~$ python3 rail_fence.py e wearediscoveredrunatonce 4
plaintext: 'wearediscoveredrunatonce'
rails: 4

fence:
w·····i·····r·····a·····
·e···d·s···e·e···n·t···e
··a·e···c·v···d·u···o·c·
···r·····o·····r·····n··

ciphertext: 'wiraedseenteaecvduocrorn'
ricard@HP-ProBook:~$ python3 rail_fence.py d wiraedseenteaecvduocrorn 4
ciphertext: 'wiraedseenteaecvduocrorn'
rails: 4

fence:
w·····i·····r·····a·····
·e···d·s···e·e···n·t···e
··a·e···c·v···d·u···o·c·
···r·····o·····r·····n··

plaintext: 'wearediscoveredrunatonce'
ricard@HP-ProBook:~$ python3 rail_fence.py e wearediscoveredrunatonce 5
plaintext: 'wearediscoveredrunatonce'
rails: 5

fence:
w·······c·······u·······
·e·····s·o·····r·n·····e
··a···i···v···d···a···c·
···r·d·····e·e·····t·n··
····e·······r·······o···

ciphertext: 'wcuesorneaivdacrdeetnero'
ricard@HP-ProBook:~$ python3 rail_fence.py d wcuesorneaivdacrdeetnero 5
ciphertext: 'wcuesorneaivdacrdeetnero'
rails: 5

fence:
w·······c·······u·······
·e·····s·o·····r·n·····e
··a···i···v···d···a···c·
···r·d·····e·e·····t·n··
····e·······r·······o···

plaintext: 'wearediscoveredrunatonce'
ricard@HP-ProBook:~$
```

07. Create a program in Python language, using the imperative procedural programming paradigm, that encrypts and decrypts messages using the columnar cipher. It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

```
ricard@HP-ProBook:~$ python3 columnar.py e theskyisblue cat
plaintext: theskyisblue
key: cat

c       a       t
3       1       20
t       h       e
s       k       y
i       s       b
l       u       e

hksu
tsil
eybe

ciphertext: hksutsileybe
ricard@HP-ProBook:~$ python3 columnar.py d hksutsileybe cat
ciphertext: hksutsileybe
key: cat

c       a       t
3       1       20
t       h       e
s       k       y
i       s       b
l       u       e

plaintext: theskyisblue
ricard@HP-ProBook:~$
```



```
ricard@HP-ProBook:~$ python3 columnar.py e theelectricgeneratorhasbeensabotaged power
plaintext: theelectricgeneratorhasbeensabotaged
key: power

p       o       w       e       r
16      15      23      5       18
t       h       e       e       l
e       c       t       r       i
c       g       e       n       e
r       a       t       o       r
h       a       s       b       e
e       n       s       a       b
o       t       a       g       e
d

ernobag
hcgaant
tecrheod
lierebe
etetssa

ciphertext: ernobaghcgaanttecrheodlierebeetetssa
ricard@HP-ProBook:~$ python3 columnar.py d ernobaghcgaanttecrheodlierebeetetssa power
ciphertext: ernobaghcgaanttecrheodlierebeetetssa
key: power

p       o       w       e       r
16      15      23      5       18
t       h       e       e       l
e       c       t       r       i
c       g       e       n       e
r       a       t       o       r
h       a       s       b       e
e       n       s       a       b
o       t       a       g       e
d

plaintext: theelectricgeneratorhasbeensabotaged
ricard@HP-ProBook:~$
```

08. Create a program in Python language, using the object oriented programming paradigm, that encrypts and decrypts files using the AES algorithm.

It is recommended to use the module PyCryptodome (https://pypi.org/project/pycryptodome/):

```
ricard@HP-ProBook:~$ python3 --version
Python 3.12.3
ricard@HP-ProBook:~$ sudo apt-get update

...

ricard@HP-ProBook:~$ sudo apt-get install python3-pycryptodome

...

ricard@HP-ProBook:~$ python3 -c "import Cryptodome; print(Cryptodome.__version__)"
3.20.0
ricard@HP-ProBook:~$
```

The resulting file must be able to be used by another like a module, the main part of the program has to be the body of `if __name__ == '__main__':` so that it is not executed when it is used by another file as a module.

It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

```
ricard@HP-ProBook:~$ ls *quote*
quote
ricard@HP-ProBook:~$ file quote
quote: ASCII text
ricard@HP-ProBook:~$ cat quote
"If you can't explain it to a six year old, you don't understand it yourself."

Albert Einstein
ricard@HP-ProBook:~$ python3 aes_file.py
Provide the required arguments: operation, input file, password and output file.
ricard@HP-ProBook:~$ python3 aes_file.py e quote 12345678 e_quote
ricard@HP-ProBook:~$ ls *quote*
e_quote  quote
ricard@HP-ProBook:~$ cat e_quote; echo
�qpWF.#I �a�bYĄ�_���E��ĸ�WhS�k73�U���xRU�`���a�_<h
s<�����:zN�3s�]���f�J�O�+;_�?F���5�9#�4(�J�ENn������tx��
ricard@HP-ProBook:~$ python3 aes_file.py d e_quote 12345678 d_quote
ricard@HP-ProBook:~$ ls *quote*
d_quote  e_quote  quote
ricard@HP-ProBook:~$ cat d_quote
"If you can't explain it to a six year old, you don't understand it yourself."

Albert Einstein
ricard@HP-ProBook:~$
```

09. Create a program in Python language, using the object oriented programming paradigm, that encrypts and decrypts messages using the RSA algorithm.

It is recommended to use the module RSA (https://pypi.org/project/rsa/):

```
ricard@HP-ProBook:~$ python3 --version
Python 3.12.3
ricard@HP-ProBook:~$ sudo apt-get update

...

ricard@HP-ProBook:~$ sudo apt-get install python3-rsa

...

ricard@HP-ProBook:~$ python3 -c "import rsa; print(rsa.__version__)"
4.9
ricard@HP-ProBook:~$
```

The resulting file must be able to be used by another like a module, the main part of the program has to be the body of `if __name__ == '__main__':` so that it is not executed when it is used by another file as a module.

It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

```
ricard@HP-ProBook:~$ ls *rsa*
do_rsa.py
ricard@HP-ProBook:~$ python3 do_rsa.py
Provide the required arguments:
           Generate keys: g path_of_keys
           Encrypt: e path_of_keys plain_text
           Decrypt: d path_of_keys cipher_text
ricard@HP-ProBook:~$ python3 do_rsa.py g .
generated keys in path /home/ricard/
ricard@HP-ProBook:~$ ls *rsa*
do_rsa.py  private-rsa-key.pem  public-rsa-key.pem
ricard@HP-ProBook:~$ python3 do_rsa.py e . 'attack on sunday afternoon'
ktWy3TVWBhys3wz7z9Nmy/1tJbTj4X34Q1hhQWvnyJD538xs61K5Ndkn3W26fo7ufjxCXqkIT9L8KlGriWjqnA==
ricard@HP-ProBook:~$ python3 do_rsa.py d . ktWy3TVWBhys3wz7z9Nmy/1tJbTj4X34Q1hhQWvnyJD538xs61K5Ndkn3W26fo7ufjxCXqkIT9L8KlGriWjqnA==
attack on sunday afternoon
ricard@HP-ProBook:~$
```

10. Create a program in Python language, using the object oriented programming paradigm, that hash a message with MD5, SHA-1, SHA-256 and SHA-512.

The resulting file must be able to be used by another like a module, the main part of the program has to be the body of `if __name__ == '__main__':` so that it is not executed when it is used by another file as a module.

It must be verified that the arguments of the program execution are acceptable.

Examples of execution:

```
ricard@HP-ProBook:~$ python3 do_hash.py
Provide the required arguments: hash function and input.
ricard@HP-ProBook:~$ python3 do_hash.py rja "Hello World!"
Error, function rja is unknown!
ricard@HP-ProBook:~$ python3 do_hash.py md5 'Hello World!'
ed076287532e86365e841e92bfc50d8c
ricard@HP-ProBook:~$ python3 do_hash.py sha1 "Hello World!"
2ef7bde608ce5404e97d5f042f95f89f1c232871
ricard@HP-ProBook:~$ python3 do_hash.py sha256 "Hello World!"
7f83b1657ff1fc53b92dc18148a1d65dfc2d4b1fa3d677284addd200126d9069
ricard@HP-ProBook:~$ python3 do_hash.py sha512 "Hello World!"
861844d6704e8573fec34d967e20bcfef3d424cf48be04e6dc08f2bd58c729743371015ead891cc3cf1c9d34b49264b510751b1ff9e537937bc46b5d6ff4ecc8
ricard@HP-ProBook:~$
```