riochr17 / **Jurnal-Sandi**

Branch: master ▾ **Jurnal-Sandi** / task1 / **vigenere.py** Find file Copy path

riochr17 final vigenere + playfair 9fddaf6 3 minutes ago

1 contributor

117 lines (98 sloc) 2.7 KB

```python
from file_handler import FileHandler
import re

class Vigenere:
        # key
        key = ''
        # key by number
        key_num = []
        # key length
        key_len = 0

        def __init__(self, key):
                self.key = key
                self.ckbn()

        # create key by number -> void
        def ckbn(self):
                self.key_len = 0
                for c in self.key:
                        self.key_num.append(ord(c))
                        self.key_len += 1

        # get char value -> return int
        def gcv(self, c, is_extended):
                ch = c if is_extended else c.lower()
                return ord(ch) if is_extended else ord(ch) - 97

        # get value char -> return char
        def gvc(self, i, is_extended):
                return chr(i) if is_extended else chr(97 + i)

        # encrypt -> return number
        def encrypt(self, p, k, mod):
                return (int(p) + int(k)) % int(mod)

        # decrypt -> return number
        def decrypt(self, c, k, mod):
                return (int(c) - int(k)) % int(mod)

        # encrypt file & write encrypted file -> void
        def encFile(self, filename, filename_out):
                # file handler
                fh = FileHandler()
                # bytes data
                bd = fh.readFileReturnBytes(filename)
                # bytes out
                bo = bytearray()
                # key number iterator
                kni = 0
                for b in bd:
                        bo.append(self.encrypt(b, self.key_num[kni], 256))
                        kni += 1
                        kni = kni % self.key_len
                fh.writeFileByBytes(filename_out, bo)

        # decrypt file & write decrypted file -> void
        def decFile(self, filename, filename_out):
                # file handler
                fh = FileHandler()
                # bytes data
                bd = fh.readFileReturnBytes(filename)
                # bytes out
                bo = bytearray()
                # key number iterator
                kni = 0
                for b in bd:
                        bo.append(self.decrypt(b, self.key_num[kni], 256))
                        kni += 1
                        kni = kni % self.key_len
                fh.writeFileByBytes(filename_out, bo)

        # generate key with len -> return string
        def gk(self, len_exp):
                # len key
                lk = len(self.key)
                # out key
                ok = ''
                # iterate char in string self.key
                for i in range(len_exp):
                        ok += self.key[i % lk]

                return ok

        # encrypt vigenere -> return string
        def ev(self, pt, is_extended = False):
                # remove non-w character
                if not is_extended:
                        pt = re.sub(r'\W', '', pt)
                # len plain text
                lpt = len(pt)
                # new key
                nk = self.gk(lpt)
                # chiper text
                ct = ''
                # iterate char in string plain text and new key
                for i in range(lpt):
                        ct += self.gvc(self.encrypt(self.gcv(pt[i], is_extended), self.gcv(nk[i], is_extended), (256 if is_exte

                return ct if is_extended else ct.upper()

        # decrypt vigenere -> return string
        def dv(self, ct, is_extended = False):
                # len chiper text
                lct = len(ct)
                # new key
                nk = self.gk(lct)
                # plain text
                pt = ''
                # iterate char in string plain text and new key
                for i in range(lct):
                        pt += self.gvc(self.decrypt(self.gcv(ct[i], is_extended), self.gcv(nk[i], is_extended), (256 if is_exte

                return pt if is_extended else pt.upper()


```