**实验二 凯撒密码及单表置换**

**一、实验目的**

1. 掌握凯撒密码的基本原理，了解其加密和解密的具体过程。

2. 学习实现凯撒密码的移位置换操作，输出不同形式的密文，包括大小写混合、全大写和全小写形式。

3. 理解单表置换的基本概念和实现方法，掌握其在加密中的应用。

**二、实验内容**

1. 凯撒密码的移位置换

 输入明文字符串和偏移量k。

 实现凯撒密码的加密算法，输出以下四种格式的密文：大小写不变、大小写互换、全大写、全小写。

 实现凯撒密码的解密算法，验证加密和解密的正确性。

2. 单表置换

实现基于置换表的加密和解密算法。输入明文字符串，通过置换表加密生成密文，随后解密以恢复原始明文。

**三、实验代码**

1. 凯撒密码的移位置换

|  |
| --- |
| def caesar\_cipher\_encrypt(M, k):  *'''  使用凯撒密码对明文进行加密  :param M:明文  :param k:偏移量  :return:加密完成的明文C分别为大小写不变、大小写互换、全大写、全小写  '''* C = ""  shift = k % 26  for i in M:  if i.islower():  newchar = chr((ord(i) - ord("a") + shift) % 26 + ord("a"))  elif i.isupper():  newchar = chr((ord(i) - ord("A") + shift) % 26 + ord("A"))  else:  newchar = i  C += newchar  return C def swap\_letter(C):  *'''  将字符串中的小写变成大写，大写变成小写  :param C:字符串  :return:转换后的字符串  '''* result = ""  for char in C:  if char.islower():  result += char.upper()  elif char.isupper():  result += char.lower()  else:  result += char  return result  def ceasar\_cipher\_decrypt(C, k):  *'''  对凯撒加密进行解密  :param C:加密后的密文  :param k: 偏移量  :return: 解密后的密文  '''* M = ""  shift = k % 26  for i in C:  if i.islower():  newchar = chr((ord(i) - ord("a") - shift) % 26 + ord("a"))  elif i.isupper():  newchar = chr((ord(i) - ord("A") - shift) % 26 + ord("A"))  else:  newchar = i  M += newchar  return M |

上文是方法定义，共定义了三个方法有两个是有关加密解密的函数，一个为实现输出加密后大小写切换的密码，其中加密部分函数主要核心公式为：



通过读取ASCII码，减去基准‘a’的值后，移动shift个位置实现凯撒加密，同时mod26，实现了在新的位移上不超过26。在解密函数中则是将往后位移改成往前位移，其核心代码为：



而大小写置换函数则直接使用了python中的upper()、lower()函数。

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  M = input("请输入明文字符串:")  k = int(input("请输入偏移量K:"))  C = caesar\_cipher\_encrypt(M, k)  M1 = ceasar\_cipher\_decrypt(C,k)  print("加密完成的密码(大小写不变)是:" + C)  print("加密完成的密码(大小写互换)是:" + swap\_letter(C))  print("加密完成的密码(全大写)是:" + C.upper())  print("加密完成的密码(大小写不变)是:" + C.lower())   print("针对大小写不变的凯撒加密的解密后的明文为:"+M1)  if M == M1:  print("加密解密正确") |

2. 单表置换的实现

|  |
| --- |
| class SubstitutionCipher:  *'''  置换表类型定义  '''* def \_\_init\_\_(self):  # 初始化原始字母表  self.alphabet = 'abcdefghijklmnopqrstuvwxyz'  #构建自己的初始变量  def generate\_substitution\_table(self, keyword):  *'''  生成加密后的置换表  :param keyword:密钥  :return: 置换后的置换表  '''* # 处理密钥词组，转换为大写并移除重复字符  unique\_chars = []#中继存储密钥词组  for char in keyword.upper():  if char not in unique\_chars and char in self.alphabet.upper():  unique\_chars.append(char)  # 按字母表顺序添加剩余字符  for char in self.alphabet.upper():  if char not in unique\_chars:  unique\_chars.append(char)  # 生成置换表  substitution\_table = {}  for i, original\_char in enumerate(self.alphabet):#生成：[(0, 'a'),..]即(i,'X')  substitution\_table[original\_char] = unique\_chars[i]#变成{'a':'X'}新的映射表  return substitution\_table   def generate\_decryption\_table(self, keyword):  *'''  生成解密用的逆置换表  :param keyword: 密钥  :return: 解密置换表  '''* # 生成加密置换表  encrypt\_table = self.generate\_substitution\_table(keyword)#字典操作  # 创建逆置换表  decrypt\_table = {}  for original, substituted in encrypt\_table.items():  decrypt\_table[substituted] = original  return decrypt\_table#新的字典   def encrypt(self, M, keyword):  *"""使用置换表加密明文"""* # 生成置换表  table = self.generate\_substitution\_table(keyword)   # 加密过程  ciphertext = []  for char in M:  if char in table:  ciphertext.append(table[char])  else:  ciphertext.append(char)  return ''.join(ciphertext)   def decrypt(self, ciphertext, keyword):  *"""使用逆置换表解密密文"""* # 生成逆置换表  table = self.generate\_decryption\_table(keyword)   # 解密过程  plaintext = []  for char in ciphertext:  if char in table:  plaintext.append(table[char])  else:  plaintext.append(char)   return ''.join(plaintext) |

上文是Class定义以及方法定义，核心方法有生成加密置换表和解密置换表，加密解密则是python字典简单的映射关系。其中加密置换表的核心代码为:





通过enumerate函数实现生成：[(0, 'a'),..]即(i,'X')，后续以字典的形式生成新的映射表。

|  |
| --- |
| if \_\_name\_\_ == "\_\_main\_\_":  cipher = SubstitutionCipher()#新建一个类   keyword = input("请输入加密的密钥:")  M = input("请输入加密的明文:")   # 生成置换表并打印  encrypt\_table = cipher.generate\_substitution\_table(keyword)  print("加密置换表:")   count = 0  for original, substituted in encrypt\_table.items():  print(f"{original} -> {substituted}", end="\t")  count += 1  if count % 5 == 0: # 每5个元素换行  print() # 仅换行  if count % 5 != 0:  print() # 确保最后有换行符   # 加密并打印结果  C = cipher.encrypt(M, keyword)  print(f"\n明文: {M}")  print(f"密文: {C}")   # 解密并打印结果  decrypted\_text = cipher.decrypt(C, keyword)  print(f"解密后: {decrypted\_text}") |

**四、实验结果**

1. 凯撒密码的移位置换

文本

AI 生成的内容可能不正确。经验证可以生成实验的要求的四种密码输出，并验证了加密的正确性。

2. 单表置换的实现

图片包含 箭头

AI 生成的内容可能不正确。经实验验证可以对输入的密钥进行只要不重复的密码，然后输出的加密置换表也实现了开头为BAN三个字母后续按顺序填充。也实现了加密解密。

**四、实验总结与心得**

通过本次实验简单的了解了最简单的两种加密方式，其本质都是重映射来实现对原本信息的混杂，从而实现了对明文的加密，在本次实验中，了解了针对ASCII码来处理的方式以及，如何建立一张映射表。先建立索引再建立字典能够很好的实现需求。