CNS Lab Assignment 2: Caeser and Monosubstitution Ciphers

111703013 Akshay Deodhar

Caeser Cipher

Code

```
1
2  from sys import stdin, argv, stderr
3  import time
4
5  def caeserify(c, shift):
6      if c.isalpha():
7          if c.islower():
8              base = ord('a')
9          else:
10             base = ord('A')
11
12         return chr(base + ((ord(c) - base) + shift) % 26)
13
14     else:
15
16         return c
17
18
19 def encode(s, shift):
20
21     def shifter(c):
22         return caeserify(c, shift)
23
24     res = ""
25     for c in s:
26         cdash = caeserify(c, shift)
27         res += cdash
28
29     return res
30
31 if __name__ == "__main__":
32
33     n = len(argv)
34     if n != 3:
35         print("usage: python3 caeser.py <mode> <shift>", file = stderr)
36         exit(1)
37     __, mode, shift = argv
38
39     if mode not in ['e', 'd']:
40         print("<mode> must be in {e, d}", file = stderr)
41         exit(1)
42
43     shift = int(shift)
44
```

```
    if mode == 'd':
        # encoding and decoding are the same, but with reverse rotation
        shift = -shift
        phrase = 'decode'
    else:
        phrase = 'encode'

    iptext = stdin.read()

    t1 = time.time()
    optext = encode(iptext, shift)
    t2 = time.time()

    print(optext, end = "")

    print("Time required to", phrase, ":", t2 - t1, file = stderr)
```

Output

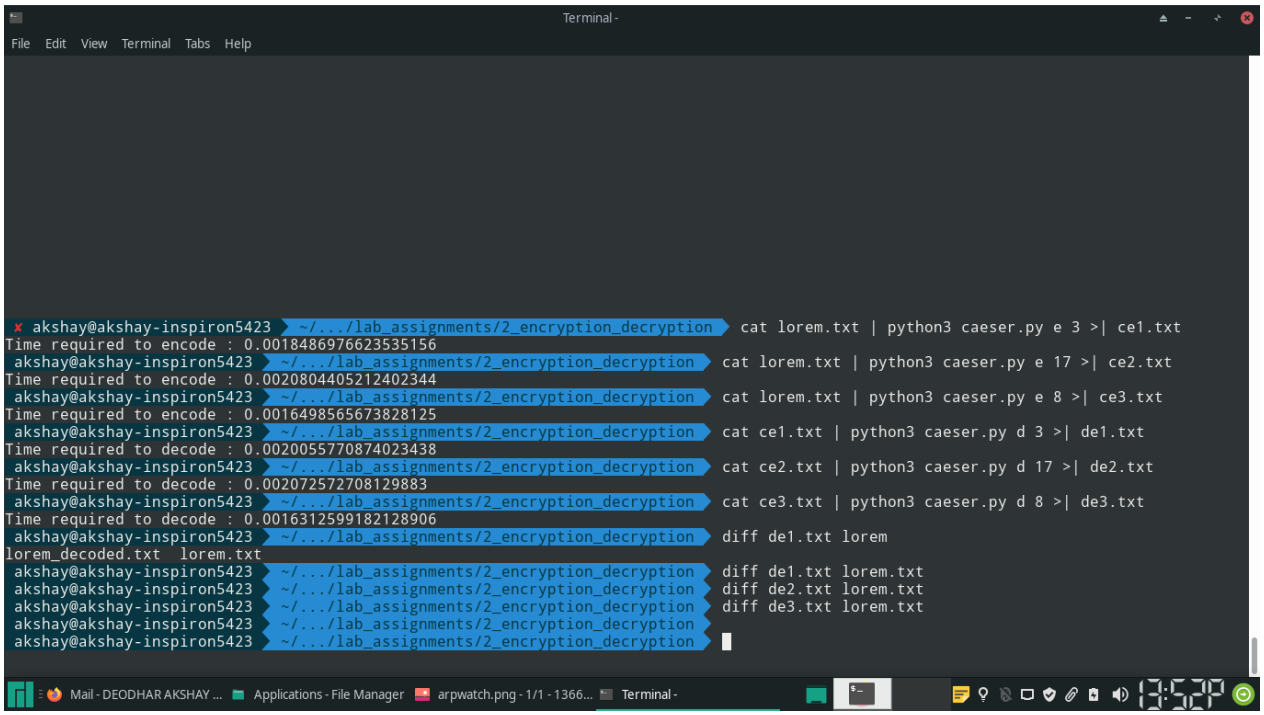

Figure 1: Execution of Caeser Cipher using 3 different shifts

Statistics

- The size of file used for encryption and decryption is **3.4KB**.
- The average time required for encryption was **0.00181s**
- The average time needed for decryption was **0.00186s**

Monosubstitution Cipher

Code

```
1
2  from sys import stdin, argv, stderr
3  import time
4
5  def flipkey(key):
6
7      newkey_list = [None] * 26
8
9      base = ord('a')
10     for i, c in enumerate(key):
11         newkey_list[ord(c) - base] = chr(base + i)
12
13     return "".join(newkey_list)
14
15 def transform(c, key):
16     if c.isalpha():
17         if c.isupper():
18             return key[ord(c.lower()) - ord('a')].upper()
19         else:
20             return key[ord(c) - ord('a')]
21     else:
22         return c
23
24 def encode(s, key):
25
26     def transform_with_key(s):
27         return transform(s, key)
28
29     res = ""
30     for c in s:
31         cdash = transform(c, key)
32         res += cdash
33
34     return res
35
36
37 if __name__ == "__main__":
38
39     n = len(argv)
40     if n != 3:
41         print("usage: python3 monosubstution.py <mode> <shift>", file = stderr)
42         exit(1)
43     __, mode, key = argv
44
45     if mode not in ['e', 'd']:
46         print("<mode> must be in {e, d}", file = stderr)
47         exit(1)
48
49     if mode == 'd':
50         # encoding and decoding are the same, but with reverse rotation
51         key = flipkey(key)
52         phrase = 'decode'
53     else:
54         phrase = 'encode'
55
```

```
56      iptext = stdin.read()
57
58      t1 = time.time()
59      optext = encode(iptext, key)
60      t2 = time.time()
61
62      print(optext, end = "")
63
64      print("Time required to", phrase, ":", t2 - t1, file = stderr)
```

Output

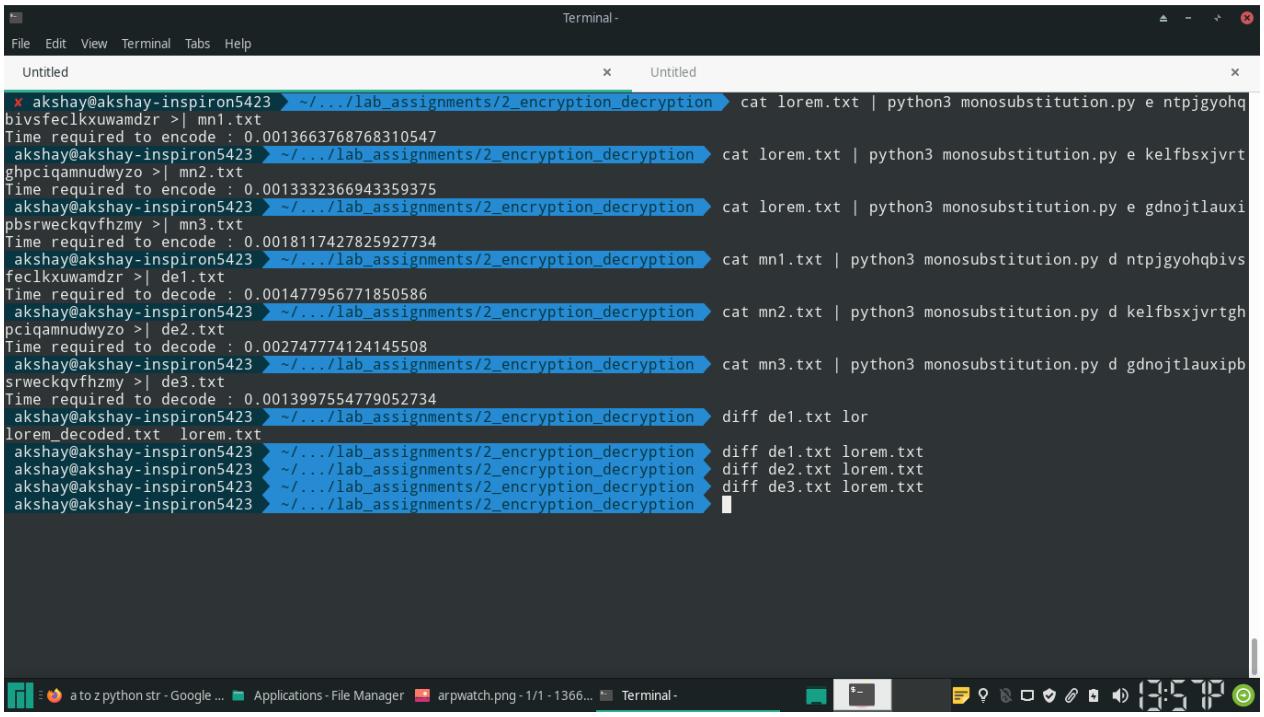

Figure 2: Encoding and decoding using monosubstitution cipher using 3 different subkeys

Statistics

- The size of the file used for encryption was **3.4KB**
- The average time required for encryption, using three different keys was **0.00146s**
- The average time required for decryption, using three different keys was **0.00180s**0.00180s**0.00180s**