# Исходный код 5 битового кодирования Python 3.7

# -\*- coding: utf8 -\*-  
  
import sys  
  
values = {'binary' : ["00001", "00010", "00011", "00100", "00101", "00110", "00111",  
 "01000", "01001", "01010", "01011", "01100", "01101", "01110",  
 "01111", "10000", "10001", "10010", "10011", "10100", "10101",  
 "10110", "10111", "11000", "11001", "11010", "11011", "11100"],  
 'managings' : ["00000", "11111"],  
 'firstLine' : lambda x: (1, 'firstLine', "ОЕАИТНСРВЛКМДПУЯЫГЗБЧЙХЪЖЬЮ ".find(x)),  
 'secondLine' : lambda x: (2, 'secondLine', "0123456789.,:/ШЦЩЭФЁ+-?!()%=".find(x))}  
  
  
def add\_binary(line\_name, letter): return values.get("binary")[values.get(line\_name).index(letter)]  
  
  
def calculate(code):  
 return sys.getsizeof(code) / 1024  
  
  
def change\_line(count=1):  
 # if count == 3: count = -1  
 # elif count == -3: count = 1  
 if count < 0:  
 margin\_index = 1 # down  
 else:  
 margin\_index = 0 # up  
 return ''.join([values.get("managings")[margin\_index] for i in range(0, abs(count))])  
  
  
def main():  
 #message = input("Введите сообщение: ").upper()  
 message = "".upper()  
 code, last\_line = "", 0  
 message = list(message)  
 for letter in message:  
 new\_level\_number, new\_level\_name, binary\_index = list(filter(lambda x: x[2] > -1, [values.get(lvl)(letter) for lvl in ('firstLine', 'secondLine')]))[0]  
 code += change\_line(last\_line - new\_level\_number) + values.get("binary")[binary\_index]  
 last\_line = new\_level\_number  
 print(code)  
 print("%.3f" % calculate(code), "КБ")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Статистика и анализ

## Тестирование и анализ времени сжатия

Таблица 1. Свод по 10 тестам по 3 методам кодирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Время** | | | |
|  | 5bit | 4bit | 8 bit |
| Тест 1 | 0,00020448 | 0,00028800 | 0,00010208 |
| Тест 2 | 0,00019072 | 0,00026848 | 0,0000976 |
| Тест 3 | 0,00019104 | 0,000265921 | 0,0000973 |
| Тест 4 | 0,000190081 | 0,00029024 | 0,00014208 |
| Тест 5 | 0,0001888 | 0,00027424 | 0,0000986 |
| Тест 6 | 0,0001952 | 0,000267841 | 0,0000950 |
| Тест 7 | 0,00019968 | 0,00027072 | 0,0000941 |
| Тест 8 | 0,00021248 | 0,00030656 | 0,0000938 |
| Тест 9 | 0,000184641 | 0,00028032 | 0,0000934 |
| Тест 10 | 0,0001824 | 0,00028 | 0,0000934 |
| **среднее** | **0,0001936** | **0,000279** | **0,0001005** |

Рисунок 1 - График стабильности времени тестирования

## Тестирование объема сообщения после предполагаемого сжатия

Таблица 2. Отношение объема символов к методу кодирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Объем** | | | | | |
|  | 5bit (КБ) | 4bit (КБ) | 8 bit (КБ) | % 4bit -> 8 bit | % 5bit -> 8 bit |
| 2525 | 13,16 | 14,32 | 19,56 | 73% | 67% |
| 88673 | 249,59 | 271,81 | 371,12 | 73% | 67% |
| 354692 | 998,27 | 1087,17 | 1484,40 | 73% | 67% |

Рисунок 2 - Отношение объема сообщения к весу после кодирования