ソースコード(huffman_coding.py)

```
import json
import math
import argparse


class Node:
    def __init__(self, left, right):
        self.name = '({} + {})'.format(left.name, right.name)
        self.p = left.p + right.p
        self.left = left
        self.right = right
        self.__signal = None

    @property
    def signal(self):
        return self.__signal

    @signal.setter
    def signal(self, signal):
        self.__signal = signal


class Leaf:
    def __init__(self, name, p):
        self.name = name
        self.p = p
        self.__signal = None

    @property
    def signal(self):
        return self.__signal

    @signal.setter
    def signal(self, signal):
        self.__signal = signal


def give_signal(node):
    if isinstance(node, Node):
        node.left.signal = node.signal + str(0)
        node.right.signal = node.signal + str(1)
        give_signal(node.left)
        give_signal(node.right)

def ave_codeword_length(leafs):
    result = 0
    for leaf in leafs:
        result += len(leaf.signal) * leaf.p
    return result
```

```
def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--file', type=str, default=None,help='give
frequency json file')
    args = parser.parse_args()
    if args.file is None:
      raise("give file path \n [usage] python huffman_coding.py --file
<JSON file>")

    with open(args.file, 'r') as f:
        frequency = json.load(f)

    leafs = [Leaf(name, p) for name, p in frequency.items()]

    nodes = {leaf.name:{'p':leaf.p, 'node': leaf} for leaf in leafs}

    while(len(nodes) > 1):
        for i, (k, v) in enumerate(sorted(nodes.items(), key=lambda x:
x[1]['p'])):
            if i == 0:
                left = v['node']
                nodes.pop(k)
            elif i == 1:
                right = v['node']
                nodes.pop(k)
            else:
                break
        new_node = Node(left, right)
        nodes[new_node.name] = {'p' : new_node.p, 'node': new_node}
    root = list(nodes.values())[0]['node']
    root.signal = ''


    give_signal(root)
    print('===  ハフマン符号化  ===')
    for leaf in leafs:
        print('{} : {}'.format(leaf.name, leaf.signal))
    print('========================')

    print('平均符号語長: {}'.format(ave_codeword_length(leafs)))


if __name__ == '__main__':
    main()
```

実行結果

```
$ cat letter_frecency.json
{
  "A" : 0.08167,
  "B" : 0.01492,
  "C" : 0.02782,
  "D" : 0.04253,
  "E" : 0.12702,
  "F" : 0.02228,
  "G" : 0.02015,
  "H" : 0.06094,
  "I" : 0.06966,
  "J" : 0.00153,
  "K" : 0.00772,
  "L" : 0.04025,
  "M" : 0.02406,
  "N" : 0.06749,
  "O" : 0.07507,
  "P" : 0.01929,
  "Q" : 0.00095,
  "R" : 0.05987,
  "S" : 0.06327,
  "T" : 0.09056,
  "U" : 0.02758,
  "V" : 0.00978,
  "W" : 0.02360,
  "X" : 0.00150,
  "Y" : 0.01974,
  "Z" : 0.00074
}

$ python huffman_coding.py --file letter_frecency.json
===  ハフマン符号化  ===
A : 1110
B : 110000
C : 01001
D : 11111
E : 100
F : 00101
G : 110011
H : 0110
I : 1011
J : 001001011
K : 0010011
L : 11110
M : 00111
N : 1010
O : 1101
P : 110001
Q : 001001001
R : 0101
S : 0111
T : 000
U : 01000
V : 001000
W : 00110
X : 001001010
Y : 110010
Z : 001001000
```

```
========================
平均符号語長: 4.20502
```