```
# encoding: utf-8

class Ord:
    """
    Klass för att skapa objekt för orden som kommer in som indata
    """
    def __init__(self, o):
        self.ordet = o #strängen med själva ordet
        self.stavelsekomponenter = [] #'O', 'N' och 'C' för varje bokstav i samma ordobjekts ord
        self.stavelsegraenser = []
        """
        self.stavelsegraenser är en lista med intar som anger index för vilken som är
        den sista bokstaven i varje stavelse (förutom den sista stavelsen vars sista bokstav första
också är ordets sista).
        Dessa index anger därmed stavelsegränserna som, om utritade som streck, skulle
placeras precis efter de angivna indexen.
        """

    def visa_ord_med_stavelser(self):
        """
        Ordobjektets ord skrivs ut med stavelseindelning (och inget returneras)
        """
        ordet_stavelseindelat = ""
        x = 0
        for indx in self.stavelsegraenser:
            ordet_stavelseindelat = ordet_stavelseindelat + self.ordet[x:indx+1] + "-"
            x = indx+1

        ordet_stavelseindelat = ordet_stavelseindelat + self.ordet[x:]
        print (ordet_stavelseindelat)
        print ("\n")

    def laegga_till_stavelsegraens(self, indx):
        self.stavelsegraenser.append(indx)

class Stavelseindelare:

    def __init__(self):
        #Det portugisiska språkets alla vokaler, plus tecknet för den neutrala vokalen
        self.vokaler = ['a', 'e', 'i', 'o', 'u', 'à', 'á', 'é', 'í', 'ó', 'ú', 'â', 'ê', 'î', 'ô', 'û', 'ã', 'õ']

    def stavelseindela(self, ordobj):
```

```
        """
        Indela ordobjektets ord i stavelser
        :param ordobj: ordobjektet
        :return: först när while-lopen är slut och ordet stavelseindelats färdigt returnerar metoden,
utan returneringsvärde
        """

        x = self.finna_naesta_vokal(ordobj, 0) #hitta första vokalen

        while True:

            aktuell_vokalgrps_storlek = self.hitta_vokalgrps_storlek(ordobj, x + 1)

            stavelsegraens_mellan_vokaler = self.hantera_vokalgrupp(ordobj, x,
aktuell_vokalgrps_storlek)
            if stavelsegraens_mellan_vokaler != -1:
                # En stavelsegräns finns mitt i vokalgruppen
                ordobj.laegga_till_stavelsegraens(x + stavelsegraens_mellan_vokaler)

            y = self.finna_naesta_vokal(ordobj, x + aktuell_vokalgrps_storlek) #hitta nästa
vokalgrupp

            if y == None:
                #Inga fler vokalgrupper att analysera, och därmed inga fler stavelsegränser att hitta
                self.bestaemma_stavelsekomponenter(ordobj)
                self.laegga_till_neutral_vokal(ordobj)
                ordobj.visa_ord_med_stavelser()
                return

            ordobj.laegga_till_stavelsegraens(x + (aktuell_vokalgrps_storlek-1) +
self.hantera_konsonantgrupp(ordobj.ordet[x+aktuell_vokalgrps_storlek:y]) )
            # lägger till stavelsegräns utifrån konsonantgruppshantering

            x = y #nästa vända börjar från där den tidigare slutade

    def hitta_vokalgrps_storlek(self, ordobj, startindx):
        """
        bestäm hur många vokaler som finns i vokalgruppen
        :param ordobj: ordobjektet
        :param startindx: index från vilket vi ska börja räkna i ordet, dvs var vokalgruppen börjar +1
        :return: hur många vokaler som finns i vokalgruppen
        """
```

```
        vokalgrupps_storlek = 1 # Vi börjar vid vokalgruppens andra vokal. Den inledande
storleken, 1, är minimum
        raeknare = startindx

        while raeknare < len(ordobj.ordet):
            if ordobj.ordet[raeknare] in self.vokaler:
                vokalgrupps_storlek += 1
            else:
                return vokalgrupps_storlek
            raeknare += 1
        return vokalgrupps_storlek

    def finna_naesta_vokal(self, ordobj, startindx):
        """
        hitta nästa vokal
        :param ordobj: ordobjekt
        :param startindx: index från vilket vi ska börja leta efter en vokal
        :return: indexet där nästa vokal finns
        """
        raeknare = startindx
        while raeknare < len(ordobj.ordet):
            if ordobj.ordet[raeknare] in self.vokaler:
                return raeknare # returnera så fort en vokal har hittats
            raeknare += 1

        return None #ingen vokal hittad

    def hantera_2_vokaler(self, ordobj, startindx):
        """
        hantera vokalgrupper med två vokaler
        :param ordobj: ett ordobjekt
        :param startindx: det index ifrån vilket vi ska kolla
        :return: index för var stavelsegräns mitt i vokalgruppen ska gå
        """
        vokalgrp = ordobj.ordet[startindx:startindx+2]
        vokalgrupps_storlek = 2

        if vokalgrp[0] == 'u' and startindx > 0:
            sekv = ordobj.ordet[startindx - 1:startindx + 2]
            if sekv == 'gue' or sekv == 'gui' or sekv == 'que' or sekv == 'qui':
                return -1
        if vokalgrp == 'ãe' or vokalgrp == 'õe' or vokalgrp == 'ão' or vokalgrp == 'iu' or vokalgrp ==
'ui':
```

```
            # speciella vokalgrupper som aldrig ska åtskiljas i olika stavelser

            return -1
        elif vokalgrp[0] != 'i' and vokalgrp[1] != 'i' and vokalgrp[0] != 'u' and vokalgrp[1] != 'u':
            # ifall ingen av vokalerna aer i eller u, tillhör de olika stavelser
            return 0
        else:

            # endast vokalgrupp med två vokaler (varav en är i eller u) är nu kvar
            if self.har_accent(ordobj) == True:

                # om det finns en vokal med accent i detta ord, ska dessa två vokaler inte åtskiljas till
olika stavelser
                return -1

            if vokalgrp[0] == 'i' or vokalgrp[0] == 'u':
                # vokalerna som är kvar är i/u + en vokal till
                stigande_diftong = True
            else:
                # vokalerna som är kvar är en vokal (annan än i och u) + i/u
                stigande_diftong = False

            if startindx + vokalgrupps_storlek == len(ordobj.ordet):
                # om vi kommer in hit, är vokalerna i denna vokalgrupp de sista bokstäverna i detta
ord
                if stigande_diftong:
                    return 0  # tia = ti-a
                else:
                    return -1  # matei = ma-tei
            elif startindx + 3 == len(ordobj.ordet):
                slutbokstav = ordobj.ordet[len(ordobj.ordet) - 1]
                if stigande_diftong:
                    if slutbokstav == 's' or slutbokstav == 'm':
                        return 0
                    else:
                        return -1
                else:
                    if slutbokstav == "s":
                        return -1
                    else:
                        return 0
            elif ordobj.ordet[startindx+2] == 'n' and stigande_diftong == False:
                return 0
```

```
            else:
                # övriga kombinationer med u/i + annan vokal (eller annan vokal + u/i) åtskiljs inte då de inte är diftonger
                return -1

    def hantera_vokalgrupp(self, ordobj, startindx, vokalgrupps_storlek):
        """
        Bestäm om det inuti vokalgruppen ska gå en stavelsgräns och i så fall var
        :param ordobj: ordobjektet
        :param startindx: index från vilket vi ska börja i ordobjektets ord
        :param vokalgrupps_storlek: hur många vokaler vi har att göra med
        :return:
        -1 om ingen stavelsegräns ska gå inuti den här vokalgruppen,
        annars 0 om vokalgruppens första vokal är sista bokstav i en stavelse, 1 för den andra eller 2 för den tredje.
        """

        vokalgrp = ordobj.ordet[startindx:startindx+vokalgrupps_storlek]
        if vokalgrupps_storlek == 1:
            # har vokalgruppen endast en vokal, kan man förstås inte sätta en stavelsegräns mitt i vokalgruppen
            return -1
        elif vokalgrupps_storlek == 2:
            # vokalgruppen i fråga har TVÅ vokaler
            return (self.hantera_2_vokaler(ordobj, startindx))

        elif vokalgrupps_storlek == 3:
            # vokalgruppen i fråga har TRE vokaler
            halvvokaler = ['i', 'u'] #vokaler som kan vara kaernvokaler i en stavelse men också vara halvvokaler i en vokalgrupp
            if vokalgrp[0] == 'u' and startindx > 0:
                sekv = ordobj.ordet[startindx - 1:startindx + 2]
                if sekv == 'gue' or sekv == 'gui' or sekv == 'que' or sekv == 'qui':
                    # Om vokalgruppen börjar med ui eller ue och föregås av eller g, så är u:et plötsligt stumt.
                    # I de fallen hanteras detta därför som en två vokaler stor vokalgrupp
                    graenspos = self.hantera_2_vokaler(ordobj, startindx+1)
                    if graenspos != -1:
                        return graenspos+1
                    else:
                        return -1
            if vokalgrp[0] in halvvokaler:
                if vokalgrp[2] in halvvokaler or vokalgrp[1] == 'ã' or vokalgrp[1] == 'õ':
```

```
                return -1
            else:
                return 1
        elif vokalgrp[1] == 'ã' or vokalgrp[1] == 'õ':
            return 0
        elif vokalgrp[1:] == 'iu':
            if (self.har_accent(ordobj) == False) and (self.finna_naesta_vokal(ordobj, startindx+3)
== None):
                return 0
            else:
                return 1
        elif vokalgrp[1] in halvvokaler:
            return 1;
        elif vokalgrp[2] in halvvokaler:
            return 0
        else:
            ordobj.laegga_till_stavelsegraens(startindx)
            return 1
    else:
        return 2

  def hantera_konsonantgrupp(self, konsonantgrupp):
    """
    fastställer om en stavelsegräns ska gå i, före eller efter konsonantgruppen
    :param konsonantgrupp:
    :return: hur många konsonanter som tillhör föregående stavelses coda
    """
    konsgrp_storlek = len(konsonantgrupp)
    if konsgrp_storlek == 1:
        return 0;
    elif konsgrp_storlek == 2:
        if konsonantgrupp[0] == 'r' or konsonantgrupp[0] == 'x':
            return 1
        elif konsonantgrupp[0] == 'n' or konsonantgrupp[0] == 'l':
            if konsonantgrupp[1] != 'h':
                return 1
            else:
                return 0
        elif konsonantgrupp[0] == 'm':
            if konsonantgrupp[1] == 'p' or konsonantgrupp[1] == 'b':
                return 1
            else:
                return 0
```

```
        elif konsonantgrupp[0] == 's':
            return 1
        else:
            return 0
    elif konsgrp_storlek == 3:
        if konsonantgrupp[1] == 's':
            return 2;
        else:
            return 1;
    else:
        return 2;

def har_accent(self, ordobj):
    for bokstav in ordobj.ordet:
        if bokstav in self.vokaler[5:]: #kolla vokaler med accent
            return True
    return False

def bestaemma_stavelsekomponenter(self, ordobj):

    nucleo = 'N' #kärna
    onset = 'O' #ansats
    coda = 'C' # svans / koda
    stavelsens_1_bokstav = 0
    for stavelsegraens in ordobj.stavelsegraenser:
        vokal_funnen = False
        stavelseslut = stavelsegraens+1
        for bokstav in ordobj.ordet[stavelsens_1_bokstav:stavelseslut]:
            if bokstav in self.vokaler:
                vokal_funnen = True
                ordobj.stavelsekomponenter.append(nucleo)
            elif vokal_funnen == False:
                ordobj.stavelsekomponenter.append(onset)
            else:
                ordobj.stavelsekomponenter.append(coda)
        stavelsens_1_bokstav = stavelseslut

    vokal_funnen = False
    for b in ordobj.ordet[stavelsens_1_bokstav:]:
        if b in self.vokaler:
            vokal_funnen = True
            ordobj.stavelsekomponenter.append(nucleo)
        elif vokal_funnen == True:
```

```
            if b == 'm':
                ordobj.stavelsekomponenter.append(nucleo)
            else:
                ordobj.stavelsekomponenter.append(coda)
        else:
            ordobj.stavelsekomponenter.append(onset)

    for w in range(len(ordobj.stavelsekomponenter)):
        print (ordobj.stavelsekomponenter[w] + "\t" + ordobj.ordet[w])

def laegga_till_neutral_vokal(self, ordobj):

    comp = 1

    while comp < len(ordobj.stavelsekomponenter):

        if ordobj.stavelsekomponenter[comp] == 'O' and ordobj.stavelsekomponenter[comp-1]
== 'O':
            ansatsens_2_bokstav = ordobj.ordet[comp]
            if ansatsens_2_bokstav != 'r' and ansatsens_2_bokstav != 'l' and
ansatsens_2_bokstav != 'h':

                ordobj.ordet = ordobj.ordet[:comp] + 'ø' + ordobj.ordet[comp:]
                ordobj.stavelsekomponenter.insert(comp, 'N')

                novo_limite = comp

                ordobj.stavelsegraenser.append(novo_limite)
                ordobj.stavelsegraenser.sort()

                for x in range(len(ordobj.stavelsegraenser)):
                    if ordobj.stavelsegraenser[x] > novo_limite:
                        ordobj.stavelsegraenser[x] += 1

                comp += 1
        comp += 1

def utvaerdera(textfil, silab):

    rad = 0
    totalt = 0
    korrekt_indelade = 0
    blev_fel = []
```

```
    while rad < len(textfil):

        korrekta_stavelsegraenser = textfil[rad + 1].strip('\n').split(' ')
        korrekta_komponenter = textfil[rad+2].strip('\n').split(' ')
        if korrekta_stavelsegraenser[0] == '':
            del korrekta_stavelsegraenser [:]

        p = Ord(textfil[rad].strip('\n').lower())
        silab.stavelseindela(p)

        bra = True
        totalt += 1
        if len(korrekta_komponenter) != len(p.stavelsekomponenter):
            bra = False
            print ("Fel, i antalet komponenter:\n")
            print ("Ord: " + textfil[rad].strip('\n'))
            print ("från stavelseindelaren: " + str(p.stavelsekomponenter))
            print ("korrekta komponenter: " + str(korrekta_komponenter) + "\n")
        elif len(korrekta_stavelsegraenser) != len(p.stavelsegraenser):
            bra = False
            print ("Fel, i antalet stavelsegraenser:")
            print ("Ord: " + textfil[rad].strip('\n'))
            print ("från stavelseindelaren: " + str(p.stavelsegraenser))
            print ("korrektastavelsegraenser: " + str(korrekta_stavelsegraenser) + "\n")
        else:
            for componente in range(len(korrekta_komponenter)):
                if korrekta_komponenter[componente] != p.stavelsekomponenter[componente]:
                    bra = False
                    print ("Fel, i en av komponenterna")
                    print ("Ord: " + textfil[rad].strip('\n'))
                    print ("Stavelseindelarens komponenter: " +
str(p.stavelsekomponenter)[componente])
                    print ("Korrekta komponenter: " + str(korrekta_komponenter)[componente] + "\n")

            for limite in range(len(korrekta_stavelsegraenser)):
                if korrekta_stavelsegraenser[limite] != str(p.stavelsegraenser[limite]):
                    bra = False
                    print ("Fel, vid stavelsegraens nmr " + str(limite))
                    print ("Ord: " + textfil[rad].strip('\n'))
                    print ("Stavelseindelarens stavelsegraenser: " + str(p.stavelsegraenser[limite]))
                    print ("Korrekta stavelsegraenser: " + str(korrekta_stavelsegraenser[limite]) + "\n")

        if bra:
```

```
            korrekt_indelade += 1
        else:
            blev_fel.append(textfil[rad].strip('\n'))


        rad += 4
        del korrekta_stavelsegraenser[:]
        del korrekta_komponenter[:]
    print("Totalt: " + str(totalt))
    print("Rätt: " + str(korrekt_indelade))
    if len(blev_fel) != 0:
        print("Följande blev fel:\n")
        for f in blev_fel:
            print (f)
    else:
        print("ALLA RÄTT!")

if __name__ == "__main__":
    s = Stavelseindelare()

    indataord = input("Escreva uma palavra para silabificar, ou 1 para silabificar todas as palavras do arquivo txt:\nSkriv ett portugisiskt ord för stavelseindelning, eller 1 för att gå igenom alla ord i txt-filen:\n")

    txt = open("plvrpt.txt", 'r', encoding='utf-8')
    orden_fr_fil = txt.readlines()

    while (indataord != '00'):

        if indataord == '1':
            utvaerdera(orden_fr_fil, s)
        else:
            p = Ord(indataord.lower())
            s.stavelseindela(p)

        indataord = input("\nEscreva uma palavra para silabificar, ou 1 para silabificar todas as palavras do arquivo txt (00 para terminar):\nSkriv ett portugisiskt ord för stavelseindelning, eller 1 för att gå igenom alla ord i txt-filen (00 för att avsluta):\n")

    txt.close()
```