Μεταγλωττιστές 2017
Προγραμματιστική Εργασία #1

Ονοματεπώνυμο: Παπαχρήστος Γεώργιος **Α.Μ.:** Π2010035

1. Ανάλυση και περιγραφή του Κώδικα βήμα βήμα

```
import sys
import re
import argparse

# Χρήσεις από Command line
# python srt-fixer.py test-input.srt 3.24 > test-output.srt
# python srt-fixer.py test-input.srt -3.24 > test-output.srt

# Συνάρτηση που λαμβάνει το timecode σε μορφή 00:02:07,208
# και το μετατρέπει σε milliseconds
def tc2ms(tc):

    TIMECODE_RE = re.compile('(?:(?:(?:(\d?\d):)?(\d?\d):)?(\d?\d))?(?:[,.](\d?\d?\d))?')

#Μια χρήσιμη σημείωση
#[0-9] is not always equivalent to \d.
#In python3, [0-9] matches only 0123456789 characters,
#while \d matches [0-9] and other #digit characters,
#for example Eastern Arabic numerals
#Βέβαια μας εξυπηρετεί προς το παρόν
#TIMECODE_RE = re.compile('(?:(?:(?:([0-9]?[0-9]):)?([0-9]?[0-9]):)?([0-9]?[0-9]))?(?:[,.]([0-9]?[0-9]?[0-9]))?')

    # Με το παραπάνω Regex εξυπηρετούμε τις παρακάτω περιπτώσεις
    # 12:34:56,789
    # 01:02:03,004
    # 1:2:3,4   => 01:02:03,004
    # ,4        => 00:00:00,004
    # 3         => 00:00:03,000
    # 3,4       => 00:00:03,004
    # 1:2       => 00:01:02,000
    # 1:2,3     => 00:01:03,003
    # 1:2:3     => 01:02:03
    # δέχεται "." και "," ως διαχωριστής για τα ms

#επιλέγουμε search για να το βρίσκει οπούδηποτε στο κείμενο
```

```
    match   = TIMECODE_RE.search(tc)

# Λαμβάνουμε όλα τα groups  που θέλουμε (όχι αυτά που ξεκινάνε από ?:)
# και τα μετατρέπουμε σε int (τις ώρες, τα λεπτά κτλ)
  # χρησιμοποιώντας την Συνάρτηση map για να τα παίρνει ένα-ένα
  # και να τα βγάζει σε ωραία λίστα ένα-ένα
    hh,mm,ss,ms = map(lambda x: 0 if x==None else int(x), match.groups())
    return ((hh*3600 + mm*60 + ss) * 1000 + ms)

# Συνάρτηση που λαμβάνει σε milliseconds
# και το μετατρέπει σε timecode σε μορφή 00:02:07,208
def ms2tc(ms):
  #Παίρνει το συνολικό χρόνο σε ms και παράγει
  # πόσες ώρες είνια, πόσα λεπτά κτλ
    ms      = abs(ms)
    ss, ms  = divmod(ms, 1000)
    hh, ss  = divmod(ss, 3600)
    mm, ss  = divmod(ss, 60)
  # και το βάζουμε σε ειδικό Format Timecode
    TIMECODE_FORMAT = '%02d:%02d:%02d,%03d'
    return TIMECODE_FORMAT % (hh, mm, ss, ms)

# Συνάρτηση που λαμβάνει ένα string (την κάθε γραμμή από το αρχείο)
# και εντοπίζει την γραμμή που περιέχει τα timecodes, όπως 00:02:07,208 --> 00:02:09,278
# Σε περίπτωση που βρίσκει την ενδιαφερόμενη
# επιστρέφει 3 blocks - τμήματα
# group(0): το συνολικό κείμενο 00:02:07,208 --> 00:02:09,278
# group(1): το πρώτο κείμενο-timecode (.*) δηλαδή το 00:02:07,208
# group(2): το δεύτερο κείμενο-timecode (.*) δηλαδή το 00:02:09,278
def parse_block(line):
            TIMECODE_SEP    = re.compile(r'(.*)--> (.*)')
            return TIMECODE_SEP.search(line)

parser = argparse.ArgumentParser()
# add mandatory (positional) arguments
parser.add_argument("fname",help="input srt file name")
parser.add_argument("offset",type=float,help="subtitle offset in seconds to apply (can be fractional)")

# parse arguments
args = parser.parse_args()

# Λαμβάνουμε από το command line το offset χρόνου σε seconds
# και το μετατρέπουμε σε milliseconds
offsetInMs = args.offset * 1000
```

```
# Λαμβάνουμε από το αρχείο εισόδου μια-μια γραμμή
with open(args.fname,newline='') as ifp:
        for line in ifp:

#Αφαιρούμε από κάθε γραμμή τον τελικό χαρακτήρα newline,
#διότι κατα την εκτύπωση στην έξοδο τοποθετείται ένα νέο newline
                line = line.rstrip("\n")

                #Εντοπίζουμε την γραμμή που περιέχει τα Timecodes
                theLineWithTimes = parse_block(line)

                #Αν μας επέστρεψε την γραμμή που μας ενδιαφέρει και όχι NoneType
                #τότε συνεχίσουμε στην επεξεργασία της γραμμής
                if theLineWithTimes is not None:
                        #print ("theLineWithTimes=",theLineWithTimes.group(0))

                        # Εξάγουμε το πρώτο Timecode και το μετατρέπουμε σε ms
                        getTheFirstTimeInMs = tc2ms(theLineWithTimes.group(1))
                        #print ("getTheFirstTimeInMs=",theLineWithTimes.group(1))

                        # Εξάγουμε το δεύτερο Timecode και το μετατρέπουμε σε ms
                        getTheSecondTimeInMs = tc2ms(theLineWithTimes.group(2))
                        #print ("theLineWithTimes.group(2)=",theLineWithTimes.group(2))
                        #print ("getTheSecondTimeInMs=",getTheSecondTimeInMs)

                        #Προσθέτουμε τον Offset χρόνο σε ms (ισχύει και για αρνητικές τιμές offset)
                        #και για τα δύο timecodes τα οποία είναι σε μορφή ms
                        #Στην συνέχεια τα μετατρέπουμε πάλι σε Timecode Format
                        getTheFirstTimeInMs += offsetInMs;
                        getTheFirstTimeAfterOffset = ms2tc(getTheFirstTimeInMs)

                        getTheSecondTimeInMs += offsetInMs;
                        getTheSecondTimeAfterOffset = ms2tc(getTheSecondTimeInMs)

                        #print ("getTheFirstTimeAfterOffset=",getTheFirstTimeAfterOffset)
                        #print ("getTheSecondTimeAfterOffset=",getTheSecondTimeAfterOffset)

                        #Επανασυνθέτουμε το string-line, όπως το 00:02:07,208 --> 00:02:09,278
                        #για να εκτυπωθεί στο αρχείο ή στην έξοδο
                        line = getTheFirstTimeAfterOffset + " --> " + getTheSecondTimeAfterOffset + "\n"
                        #print ("line=",line)

                #Έξοδος στο αρχείο
                sys.stdout.write(line)
                # όσο υπάρχει το > test-output.srt στο command line δεν θα προβάλεται
                # τίποτε στην οθόνη με το print
```

2. Ανάλυση της κανονικής έκφρασης

2.1

TIMECODE_RE = re.compile('(?:(?:(?:(?:(\d?\d):)?(\d?\d):)?(\d?\d))?(?:[,.](\d?\d?\d))?')

Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί και η
TIMECODE_RE = re.compile('(?:(?:(?:(?:([0-9]?[0-9]):)?([0-9]?[0-9]):)?([0-9]?[0-9]))?(?:[,.]([0-9]?[0-9]?[0-9]))?')

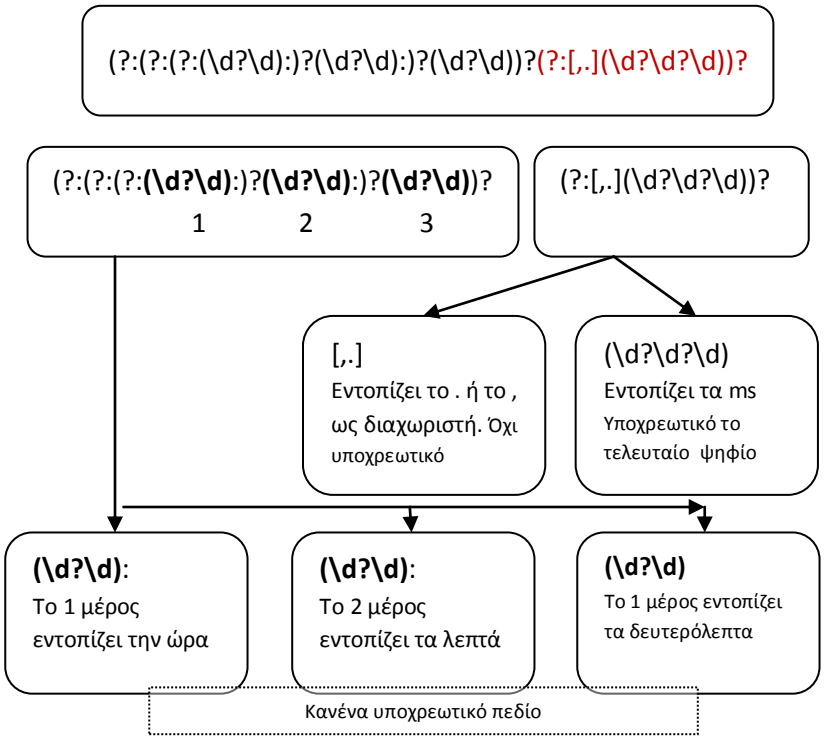
Το [0-9] δεν είναι πάντα το ίδιο με το \d, το οποίο δηλώνει digit.
Στην python3, το [0-9] αποτελείται από τους χαρακτήρες 0123456789
Ενώ το \d από [0-9] και άλλους #digit χαρακτήρες όπως Αραβικούς.
Λόγω ευκολίας αλλά και εκπαιδευτικούς λόγους χρησιμοποιήσαμε το δεύτερο.

Με το παραπάνω Regex εξυπηρετούμε τις παρακάτω περιπτώσεις

```
# 12:34:56,789
# 01:02:03,004
# 1:2:3,4   => 01:02:03,004
# ,4        => 00:00:00,004
# 3         => 00:00:03,000
# 3,4       => 00:00:03,004
# 1:2       => 00:01:02,000
# 1:2,3     => 00:01:03,003
# 1:2:3     => 01:02:03
# δέχεται "." και "," ως διαχωριστής για τα ms
```

Ανάλυση

Το ?: δηλώνει ότι δεν χρειάζεται το περιεχόμενο της ηγούμενης παρένθεσης να γίνει block.



2.2

TIMECODE_SEP = re.compile(r'(.*)--> (.*)')

Εντοπίζει και ξεχωρίζει την κάθε γραμμή που περιέχει το σύμβολο -->
Τα (.*) αριστερά και δεξιά δηλώνουν ότι θα πρέπει να πάρουμε δύο groups – δύο strings που θα περιέχουν τους χρόνους (timecodes).

3. Πηγές

https://github.com/riobard/srt.py/blob/master/srt.py

Ονοματεπώνυμο: Παπαχρήστος Γεώργιος
Α.Μ.: Π2010035