```
import urllib
import MySQLdb

PLAY_NULL = -1
PLAY_KICKOFF = 0
PLAY_RUSH = 1
PLAY_PASS = 2
PLAY_LATERAL = 3
PLAY_FIELD_GOAL = 4
PLAY_PUNT = 5
PLAY_SACK = 6
PLAY_RETURN = 7
PLAY_RECOVERY = 8
PLAY_PENALTY = 9

class Play:
    quarter = 1
    homeScore = 0
    awayScore = 0

    homePoss = True
    teamPoss = ''
    teamDef = ''
    down = 1
    distance = 10
    startingYard = 20
    endingYard = 20

    body = ''

    kickoff = False
    changePoss = False
    interception = False
    fumble = False
    block = False

    playType = PLAY_NULL
    laterals = 0
    fumbles = 0
    nextPlay = None

    firstDown = False
    fieldGoal = False
    touchdown = False
    safety = False
    tryDown = False
    touchback = False

    firstDownsThisDrive = 0
    pointsThisPlay = 0
    pointsThisDrive = 0

    qb_kicker = ''
    player_returner = ''
    defense1 = ''
    defense2 = ''
```

```
    def __init__(self, qtr, hmPoss, teamName, defTeamName, dn, dist, yd):
        self.quarter = qtr
        self.homePoss = hmPoss
        self.teamPoss = teamName.replace('State','St')
        self.teamDef = defTeamName.replace('State','St')
        self.down = dn
        self.distance = dist
        self.startingYard = yd

    def points(self):
        pts = self.pointsThisPlay
        if self.nextPlay:
            return pts + self.nextPlay.points()
        else:
            return pts

    def turnoverOnDowns(self):
        if self.changePoss:
            return False
        elif self.firstDown or self.touchdown or self.safety or self.touchback or self.fieldGoal
            return False
        elif self.nextPlay:
            nextPlayType = self.nextPlay.playType
            if nextPlayType != PLAY_NULL and nextPlayType != PLAY_PENALTY:
                return self.nextPlay.turnoverOnDowns()
        self.changePoss = True
        return True

    def didChangePoss(self):
        if self.nextPlay:
            nextPlayChangePoss = self.nextPlay.didChangePoss()
            if self.changePoss:
                return not nextPlayChangePoss
            else:
                return nextPlayChangePoss
        else:
            return self.changePoss

    def nextStartingYard(self):
        if self.nextPlay:
            return self.nextPlay.nextStartingYard()
        elif self.changePoss:
            return 100 - self.endingYard
        else:
            return self.endingYard

    def didFumble(self):
        if self.nextPlay:
            return self.fumble or self.nextPlay.didFumble()
        else:
            return self.fumble

    def wasBlocked(self):
        if self.nextPlay:
            return self.block or self.nextPlay.wasBlocked()
        else:
            return self.block
```

```
    def checkLostFumble(self, homePoss, seenFumble):
        if seenFumble:
            if self.homePoss != homePoss:
                self.homePoss = homePoss
                teamPoss = self.teamPoss
                self.teamPoss = self.teamDef
                self.teamDef = teamPoss
                self.startingYard = 100 - self.startingYard
                self.endingYard = 100 - self.endingYard
        elif self.fumble and not self.changePoss:
            if self.homePoss != homePoss:
                #if not (self.nextPlay and self.nextPlay.touchdown and self.nextPlay.endingYard
                self.changePoss = True
            seenFumble = True
        if self.nextPlay:
            self.nextPlay.checkLostFumble(homePoss, seenFumble)

    def checkBlock(self, homePoss, seenBlock):
        if seenBlock:
            if self.homePoss != homePoss:
                self.homePoss = homePoss
                teamPoss = self.teamPoss
                self.teamPoss = self.teamDef
                self.teamDef = teamPoss
                self.startingYard = 100 - self.startingYard
                self.endingYard = 100 - self.endingYard
        elif self.block and not self.changePoss:
            if self.homePoss != homePoss:
                self.changePoss = True
            seenBlock = True
        if self.nextPlay:
            self.nextPlay.checkBlock(homePoss, seenBlock)

    def playString(self):
        if self.playType == PLAY_KICKOFF:
            return 'Kickoff'
        elif self.playType == PLAY_RUSH:
            return 'Rush'
        elif self.playType == PLAY_PASS:
            return 'Pass'
        elif self.playType == PLAY_LATERAL:
            return 'Lateral'
        elif self.playType == PLAY_FIELD_GOAL:
            return 'Field Goal'
        elif self.playType == PLAY_PUNT:
            return 'Punt'
        elif self.playType == PLAY_SACK:
            return 'Sack'
        elif self.playType == PLAY_RETURN:
            return 'Returned'
        elif self.playType == PLAY_RECOVERY:
            return 'Recovered'
        elif self.playType == PLAY_PENALTY:
            return 'Penalty'
        else:
            return 'NO PLAY'
```

```
    def printSelf(self):
        if self.playType == PLAY_NULL:
            return
        situation = '>> [' + self.teamPoss + ' - Dn: ' + str(self.down) + ', Dist: ' + str(self.
        if self.tryDown:
            if self.playType == PLAY_FIELD_GOAL:
                play = 'Extra point attempt'
                if self.fieldGoal:
                    play += ' GOOD; 1 point'
                elif self.block:
                    play += ' BLOCKED'
                else:
                    play += ' NO GOOD'
            else:
                play = self.playString() + ' for a Two Point Conversion attempt'
                if self.touchdown:
                    play += ' GOOD; 2 points'
                else:
                    play += ' FAILED'
        elif self.playType == PLAY_FIELD_GOAL:
            play = str(117 - self.startingYard) + ' yard field goal attempt'
            if self.fieldGoal:
                play += ' GOOD; 3 points'
            elif self.block:
                play += ' BLOCKED'
            else:
                play += ' NO GOOD'
        else:
            play = self.playString() + ' for ' + str(self.endingYard - self.startingYard) + ' ya
            if self.firstDown:
                play += ' for a FIRST DOWN'
            elif self.touchdown:
                play += ' for a TOUCHDOWN; 6 points'
            elif self.safety:
                play += ' for a SAFETY; 2 points'
            elif self.touchback:
                play += ' for a TOUCHBACK'
            elif self.fumble:
                play += ', fumbled'
            elif self.interception:
                play += ', intercepted'
        print situation + ' ' + play
        if self.nextPlay:
            self.nextPlay.printSelf()

    def outputSelf(self, f, i):
        if self.playType == PLAY_NULL:
            return
        kickingPlay = False
        if self.playType == PLAY_KICKOFF or self.playType == PLAY_FIELD_GOAL or self.playType ==
            kickingPlay = True
        nextPlay = self.nextPlay
        # ID
        # play num
        f.write(str(i) + "\t")
        # quarter
```

```
        f.write(str(self.quarter) + "\t")
        # road team name
        if self.homePoss:
            f.write(self.teamDef + "\t")
        else:
            f.write(self.teamPoss + "\t")
        # road score
        f.write(str(self.awayScore) + "\t")
        # road team off/def
        if self.homePoss:
            f.write("Def\t")
        else:
            f.write("Off\t")
        # home team name
        if self.homePoss:
            f.write(self.teamPoss + "\t")
        else:
            f.write(self.teamDef + "\t")
        # home score
        f.write(str(self.homeScore) + "\t")
        # home team off/def
        if self.homePoss:
            f.write("Off\t")
        else:
            f.write("Def\t")
        # offense team name
        f.write(self.teamPoss + "\t")
        # defense team name
        f.write(self.teamDef + "\t")
        # leading team name
        # trailing team name
        if self.homeScore > self.awayScore:
            if self.homePoss:
                f.write(self.teamPoss + "\t" + self.teamDef + "\t")
            else:
                f.write(self.teamDef + "\t" + self.teamPoss + "\t")
        elif self.homeScore < self.awayScore:
            if self.homePoss:
                f.write(self.teamDef + "\t" + self.teamPoss + "\t")
            else:
                f.write(self.teamPoss + "\t" + self.teamDef + "\t")
        else:
            f.write("Tie\tTie\t")
        # possible points
        f.write(str(self.pointsThisDrive) + "\t")
        # QB name
        if not kickingPlay:
            f.write(self.qb_kicker + "\t")
        else:
            f.write("\t")
        # down
        f.write(str(self.down) + "\t")
        # distance
        f.write(str(self.distance) + "\t")
        # yardline
        f.write(str(self.startingYard) + "\t")
        # run/pass
```

```
        if self.playType == PLAY_KICKOFF:
            f.write("KICKOFF\t")
        elif self.playType == PLAY_RUSH:
            f.write("Run\t")
        elif self.playType == PLAY_PASS:
            f.write("Pass\t")
        elif self.playType == PLAY_FIELD_GOAL:
            if self.tryDown:
                f.write("EP\t")
            else:
                f.write("FG\t")
        elif self.playType == PLAY_PUNT:
            f.write("PUNT\t")
        elif self.playType == PLAY_SACK:
            f.write("Sack\t")
        elif self.playType == PLAY_PENALTY:
            f.write("PENALTY\t")
        # player name
        if not kickingPlay:
            f.write(self.player_returner + "\t")
        else:
            f.write("\t")
        # yards
        if not kickingPlay:
            f.write(str(self.endingYard - self.startingYard) + "\t")
        else:
            f.write("\t")
        # tackler name 1
        # tackler name 2
        if not (kickingPlay or self.fumble or self.interception):
            f.write(self.defense1 + "\t" + self.defense2 + "\t")
        else:
            f.write("\t\t")
        # result
        if self.tryDown:
            if self.playType == PLAY_FIELD_GOAL:
                if self.fieldGoal:
                    f.write("EP\t")
                elif self.block:
                    if nextPlay and nextPlay.playType == PLAY_RETURN and nextPlay.touchdown:
                        f.write("EP BLOCKED RETURN TD\t")
                    else:
                        f.write("EP BLOCKED\t")
                else:
                    f.write("EP MISSED\t")
            elif self.touchdown:
                f.write("2 PT CONVERSION\t")
            else:
                f.write("2 PT CONVERSION FAILED\t")
        if self.interception:
            if nextPlay and nextPlay.playType == PLAY_RETURN and nextPlay.touchdown:
                f.write("INTERCEPTION RETURN TD\t")
            else:
                f.write("INTERCEPTION\t")
        elif self.fumble:
            if nextPlay and nextPlay.playType == PLAY_RETURN and nextPlay.touchdown:
                f.write("FUMBLE RETURN TD\t")
```

```
            else:
                f.write("FUMBLE\t")
        elif self.touchdown:
            f.write("TOUCHDOWN\t")
        elif self.safety:
            f.write("SAFETY\t")
        elif self.touchback:
            f.write("TOUCHBACK\t")
        elif self.playType == PLAY_FIELD_GOAL:
            if self.fieldGoal:
                f.write("FG\t")
            elif self.block:
                if nextPlay and nextPlay.playType == PLAY_RETURN and nextPlay.touchdown:
                    f.write("FG BLOCKED RETURN TD\t")
                else:
                    f.write("FG BLOCKED\t")
            else:
                f.write("FG MISSED\t")
        elif self.playType == PLAY_PUNT:
            if self.block:
                if nextPlay and nextPlay.playType == PLAY_RETURN and nextPlay.touchdown:
                    f.write("PUNT BLOCKED RETURN TD\t")
                else:
                    f.write("PUNT BLOCKED\t")
            else:
                f.write("PUNT\t")
        else:
            f.write("\t")
        # kicker
        if kickingPlay:
            f.write(self.qb_kicker + "\t")
        else:
            f.write("\t")
        # kick yards
        if self.playType == PLAY_KICKOFF or self.playType == PLAY_PUNT:
            f.write(str(self.endingYard - self.startingYard) + "\t")
        else:
            f.write("\t")
        # returner
        if kickingPlay and nextPlay and nextPlay.playType == PLAY_RETURN:
            f.write(nextPlay.player_returner + "\t")
        else:
            f.write("\t")
        # return yards
        if kickingPlay and nextPlay and nextPlay.playType == PLAY_RETURN:
            f.write(str(nextPlay.endingYard - nextPlay.startingYard) + "\t")
        else:
            f.write("\t")
        # net kick yards
        if kickingPlay:
            kickYards = self.endingYard - self.startingYard
            if nextPlay and nextPlay.playType == PLAY_RETURN:
                returnYards = nextPlay.endingYard - nextPlay.startingYard
            elif self.touchback:
                returnYards = 20
            else:
                returnYards = 0
```

```
            if self.playType == PLAY_KICKOFF or self.playType == PLAY_PUNT or returnYards != 0:
                f.write(str(kickYards - returnYards) + "\t")
            else:
                f.write("\t")
        else:
            f.write("\t")
        # fumble forced name
        # fumble return name
        if self.fumble:
            f.write(self.defense1 + "\t" + self.defense2 + "\t")
        else:
            f.write("\t\t")
        # interception name
        if self.interception:
            f.write(self.defense1 + "\t")
        else:
            f.write("\t")
        # game margin
        # close?
        margin = abs(self.homeScore - self.awayScore)
        f.write(str(margin) + "\t")
        if margin < 17:
            f.write("Close\t")
        else:
            f.write("\t")
        # success?
        # line yards
        # pressure?
        # down yardage
        # passing down
        # red zone
        # special teams?
        # special team points
        # turnover?
        # point value
        # conference game?
        # date
        # month

        f.write("\n")

def geturlstring(page):
    sock = urllib.urlopen(page)
    htmlSource = sock.read()
    sock.close()
    return htmlSource

def getsked():
    gameIds = []

    year = 2008
    week = 1
    while week < 18:
        page = 'http://sports.espn.go.com/ncf/schedules'
        page += '?year=' + str(year) + '&week=' + str(week) + '&season=2&groupId=80'
        urlstring = geturlstring(page)
        ids = getgameids(urlstring)
```

```
        gameIds.extend(ids)
        week += 1

    leads = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

    i = 0
    numGameIds = len(gameIds)
    while i < numGameIds:
        gameId = gameIds[i]
        page = 'http://sports.espn.go.com/ncf/playbyplay?gameId=' + gameId + '&period=0'
        urlstring = geturlstring(page)
        readOne(urlstring, leads)
        #parsePlayByPlay(urlstring)
        i += 1

    print leads

def getgameids(urlstring):
    gameIds = []
    loc = urlstring.find('boxscore?gameId=')
    while loc != -1:
        start = loc + 16
        loc = start
        while urlstring[loc].isdigit():
            loc += 1
        gameId = urlstring[start:loc]
        gameIds.append(gameId)
        loc = urlstring.find('boxscore?gameId=', loc)
    return gameIds

def test():
    url = 'http://sports.espn.go.com/ncf/playbyplay?gameId=272440235&period=0'
    urlstring = geturlstring(url)
    #f = file('playbyplay.htm')
    #urlstring = f.read()
    leads = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
    readOne(urlstring, leads)

#def t(url):
def t():
    url = 'http://www.calbears.com/sports/m-footbl/stats/2008-2009/msucal.html#GAME.PLY'
    urlstring = geturlstring(url)
    readTwo(urlstring)

def readOne(urlstring, leads):
    plays = parsePlayByPlay(urlstring)
    if len(plays) <= 1:
        return

    f = open('test.txt', 'w')

    #conn = MySQLdb.connect (host = "localhost", user = "erik", passwd = "erj007", db = "test")
    conn = MySQLdb.connect (host = "localhost", db = "test")
    cursor = conn.cursor ()
    cursor.execute ("SELECT VERSION()")
    row = cursor.fetchone ()
    print "server version:", row[0]
```

```
    cursor.execute ("DROP TABLE IF EXISTS animal")
    cursor.execute ("""
        CREATE TABLE games
        (
            gameID      MEDIUMINT(10),
            homeName    CHAR(40),
            awayName    CHAR(40)
        )
         """)
    cursor.execute ("""
        INSERT INTO animal (name, category)
        VALUES
            ('snake', 'reptile'),
            ('frog', 'amphibian'),
            ('tuna', 'fish'),
            ('racoon', 'mammal')
        """)
    print "Number of rows inserted: %d" % cursor.rowcount

    cursor.close ()
    conn.close ()


    homeScore = 0
    awayScore = 0

    quarter = 1
    homeScore3Q = 0
    awayScore3Q = 0

    homePoss = True
    drive = 0
    pointsThisDrive = 0
    touchdownThisDrive = False
    defensiveScore = False

    i = 1
    while i < len(plays):
        play = plays[i]

        #play.printSelf()

        play.homeScore = homeScore
        play.awayScore = awayScore

        if play.quarter != quarter:
            quarter = play.quarter
            if quarter == 4:
                homeScore3Q = homeScore
                awayScore3Q = awayScore

        if play.homePoss != homePoss:
            homePoss = play.homePoss

        if play.kickoff:
            drive = i
```

```
            pointsThisDrive = 0
            touchdownThisDrive = False
            defensiveScore = False

        if play.didFumble() and i+1 < len(plays):
            nextPlay = plays[i+1]
            play.checkLostFumble(nextPlay.homePoss, False)

        if play.wasBlocked() and i+1 < len(plays):
            nextPlay = plays[i+1]
            play.checkBlock(nextPlay.homePoss, False)

        points = play.points()
        if points > 0:
            #play.printSelf()
            #print 'Points = ' + str(points)
            if play.didChangePoss() or (points == 2 and not play.tryDown):
                defensiveScore = True
                if play.homePoss:
                    awayScore += points
                else:
                    homeScore += points
            else:
                if play.homePoss:
                    homeScore += points
                else:
                    awayScore += points
            pointsThisDrive += points
            #print 'homeScore = ' + str(homeScore)
            #print 'awayScore = ' + str(awayScore)

            if points == 6:
                touchdownThisDrive = True

            if play.tryDown or not touchdownThisDrive:
                while drive <= i:
                    drivePlay = plays[drive]
                    if defensiveScore:
                        drivePlay.pointsThisDrive = -pointsThisDrive
                    else:
                        drivePlay.pointsThisDrive = pointsThisDrive
                    drive += 1

        i += 1

    p = plays[0]
    if p.homeScore != homeScore:
        print p.teamPoss + ' @ ' + p.teamDef
        print "Home Score doesn't add up : " + str(p.homeScore) + " vs. " + str(homeScore)
    if p.awayScore != awayScore:
        print p.teamPoss + ' @ ' + p.teamDef
        print "Away Score doesn't add up : " + str(p.awayScore) + " vs. " + str(awayScore)
    #print "Home Score : " + str(p.homeScore) + " vs. " + str(homeScore)
    #print "Away Score : " + str(p.awayScore) + " vs. " + str(awayScore)

    if p.homeScore == homeScore and p.awayScore == awayScore:
        homeLead3Q = homeScore3Q - awayScore3Q
```

```
        homeMargin = homeScore - awayScore
        leads[0] += 1
        if homeMargin > 0:
            if homeLead3Q > 17:
                leads[1] += 1
            elif homeLead3Q > 9:
                leads[2] += 1
            elif homeLead3Q > 4:
                leads[3] += 1
            elif homeLead3Q > 0:
                leads[4] += 1
            elif homeLead3Q == 0:
                leads[5] += 1
            elif homeLead3Q > -4:
                leads[6] += 1
            elif homeLead3Q > -9:
                leads[7] += 1
            elif homeLead3Q > -17:
                leads[8] += 1
            else:
                leads[9] += 1
            if leads[10] < -homeLead3Q:
                leads[10] = -homeLead3Q
        if homeMargin < 0:
            if homeLead3Q < -17:
                leads[1] += 1
            elif homeLead3Q < -9:
                leads[2] += 1
            elif homeLead3Q < -4:
                leads[3] += 1
            elif homeLead3Q < 0:
                leads[4] += 1
            elif homeLead3Q == 0:
                leads[5] += 1
            elif homeLead3Q < 4:
                leads[6] += 1
            elif homeLead3Q < 9:
                leads[7] += 1
            elif homeLead3Q < 17:
                leads[8] += 1
            else:
                leads[9] += 1
            if leads[10] < homeLead3Q:
                leads[10] = homeLead3Q

    #f.write("Play\tQ\tAway\tAway Score\tAway O/D\tHome\tHome Score\tHome O/D\t")
    #f.write("Offense\tDefense\tLEADER\tTRAILER\tPOSS PTS\tQB\tDown\tDistance\tYdLine\t")
    #f.write("Run/Pass\tPlayer\tYards\tTackler 1\tTackler 2\tRESULT\tKicker\tKickYds\t")
    #f.write("Returner\tKickRet\tNetKick\tFF\tFR\tINT\tGameMargin\tClose?")

    i = 1
    while i < len(plays):
        play = plays[i]


        cursor.execute ("""
            INSERT INTO plays (name, category)
```

```
            VALUES
                ('snake', 'reptile'),
                ('frog', 'amphibian'),
                ('tuna', 'fish'),
                ('racoon', 'mammal')
            """)

        #play.outputSelf(f, i)
        i += 1

        if i < len(plays):
            #print str(play.startingYard)
            #play.printSelf()
            #print play.body
            #print str(play.nextStartingYard())
            #play.printSelf()
            #if play.didChangePoss():
            #   print 'Change of Possession'
            nextPlay = plays[i]
            if play.nextStartingYard() != nextPlay.startingYard:
                if play.quarter == 2 and nextPlay.quarter == 3:
                    continue
                elif play.touchdown or play.fieldGoal or play.tryDown or play.safety:
                    continue
                if abs(play.nextStartingYard() - nextPlay.startingYard) <= 1:
                    continue
                #print 'Play ends at ' + str(play.nextStartingYard())
                #if play.didChangePoss():
                #   print 'Change of Possession'
                #print play.body
                #play.printSelf()
                #print 'Next play starts at ' + str(nextPlay.startingYard)
                #print nextPlay.body
                #nextPlay.printSelf()

def parsePlayByPlay(urlstring):
    # get full names and Ids
    # get visiting teamId
    loc = urlstring.find('teamId=')
    loc2 = urlstring.find('"', loc)
    if loc == -1 or loc2 == -1:
        return []
    teamId1 = int(urlstring[loc+7:loc2])
    # get visiting team full name
    loc = urlstring.find('</a>', loc)
    if urlstring[loc+4:loc+9] == '<span':
        loc = urlstring.find('</span>', loc) + 4
    loc2 = loc
    while not urlstring[loc2].isdigit():
        loc2 += 1
    teamName1 = urlstring[loc+4:loc2-1]
    loc3 = loc2
    while urlstring[loc3].isdigit():
        loc3 += 1
    teamScore1 = int(urlstring[loc2:loc3])
    # get second teamId
    loc = urlstring.find('teamId=', loc)
```

```
    loc2 = urlstring.find('"', loc)
    teamId2 = int(urlstring[loc+7:loc2])
    # get home team full name
    loc = urlstring.find('</a>', loc)
    if urlstring[loc+4:loc+9] == '<span':
        loc = urlstring.find('</span>', loc) + 4
    loc2 = loc
    while not urlstring[loc2].isdigit():
        loc2 += 1
    teamName2 = urlstring[loc+4:loc2-1]
    loc3 = loc2
    while urlstring[loc3].isdigit():
        loc3 += 1
    teamScore2 = int(urlstring[loc2:loc3])
    #print teamName1 + ' @ ' + teamName2

    # find play by play
    loc = urlstring.find('gp-body', loc)
    loc2 = urlstring.find('gp-body', loc+1)
    playByPlay = urlstring[loc:loc2]
    # separate lines
    rows = []
    loc = playByPlay.find('<tr')
    while loc != -1:
        loc2 = playByPlay.find('</tr>', loc)
        row = playByPlay[loc:loc2]
        rowList = parseRow(row)
        rows.append(rowList)
        loc = playByPlay.find('<tr', loc2)

    # placeholders
    shortName1 = teamName1
    shortName2 = teamName2

    # set up play array
    plays = []
    # set up variables
    qtr = 0
    homePoss = False
    teamPoss = teamName1
    teamDef = teamName2

    # add starter play
    p = Play(qtr, homePoss, teamPoss, teamDef, 0, 0, 0)
    p.awayScore = teamScore1
    p.homeScore = teamScore2
    plays.append(p)

    # process rows
    i = 0
    while i < len(rows):
        rowList = rows[i]
        first = rowList[0]
        #print '#first# : ' + first
        #print ' '.join(rowList)
        rest = ' '.join(rowList[1:])
        rest = rest.replace('(','')
```

```
        rest = rest.replace(')','')
        restLower = rest.lower()
        if first.find('Quarter') != -1:
            if first.find('1st') != -1:
                qtr = 1
                shortName1 = rowList[2]
                shortName2 = rowList[3]
            elif first.find('2nd') != -1:
                qtr = 2
            elif first.find('3rd') != -1:
                qtr = 3
            elif first.find('4th') != -1:
                qtr = 4
            elif first.find('5th') != -1:
                qtr = 5
            elif first.find('6th') != -1:
                qtr = 6
            elif first.find('7th') != -1:
                qtr = 7
            elif first.find('8th') != -1:
                qtr = 8
            elif first.find('9th') != -1:
                qtr = 9
            elif first.find('10th') != -1:
                qtr = 10
            elif first.find('11th') != -1:
                qtr = 11
            #print '>> [quarter ' + str(qtr) + '] ' + ' '.join(rowList)
        elif first.find('DRIVE TOTALS') != -1:
            pass
            #print '>> [drive totals] ' + ' '.join(rowList)
        elif first.find(teamName1 + ' at ') != -1:
            homePoss = False
            teamPoss = teamName1
            teamDef = teamName2
            #print '>> [visitor] ' + ' '.join(rowList)
        elif first.find(teamName2 + ' at ') != -1:
            homePoss = True
            teamPoss = teamName2
            teamDef = teamName1
            #print '>> [home] ' + ' '.join(rowList)
        elif first.find(shortName1) != -1:
            down, distance, yardline = ddy(first, shortName1, homePoss)
            p = Play(qtr, homePoss, teamPoss, teamDef, down, distance, yardline)
            touchLoc = restLower.find('touchdown')
            if touchLoc != -1 and (restLower.find('extra point') != -1 or restLower.find('two-po
                touchRest = rest[0:touchLoc+10]
                touchRestLower = restLower[0:touchLoc+10]
                getPlay(p, touchRestLower, touchRest)
                plays.append(p)
                rest = rest[touchLoc+10:]
                restLower = restLower[touchLoc+10:]
                if p.didChangePoss():
                    p = Play(qtr, not homePoss, teamDef, teamPoss, 0, 'Goal', 97)
                else:
                    p = Play(qtr, homePoss, teamPoss, teamDef, 0, 'Goal', 97)
            getPlay(p, restLower, rest)
```

```
            #print ' '.join(rowList)
            #p.printSelf()
            if p.playType != PLAY_NULL:
                plays.append(p)
            #print '>> [' + teamPoss + ' - Dn: ' + str(down) + ', Dist: ' + str(distance) + ', Y
        elif first.find(shortName2) != -1:
            down, distance, yardline = ddy(first, shortName2, not homePoss)
            p = Play(qtr, homePoss, teamPoss, teamDef, down, distance, yardline)
            touchLoc = restLower.find('touchdown')
            if touchLoc != -1 and (restLower.find('extra point') != -1 or restLower.find('two-po
                touchRest = rest[0:touchLoc+10]
                touchRestLower = restLower[0:touchLoc+10]
                getPlay(p, touchRestLower, touchRest)
                plays.append(p)
                rest = rest[touchLoc+10:]
                restLower = restLower[touchLoc+10:]
                if p.didChangePoss():
                    p = Play(qtr, not homePoss, teamDef, teamPoss, 0, 'Goal', 97)
                else:
                    p = Play(qtr, homePoss, teamPoss, teamDef, 0, 'Goal', 97)
            getPlay(p, restLower, rest)
            #print ' '.join(rowList)
            #p.printSelf()
            if p.playType != PLAY_NULL:
                plays.append(p)
            #print '>> [' + teamPoss + ' - Dn: ' + str(down) + ', Dist: ' + str(distance) + ', Y
        elif first.find(' ') != -1:
            yardline = 0
            if restLower.find('kickoff') != -1:
                yardline = 30
            if len(plays) > 0 and (restLower.find('kickoff') != -1 or restLower.find('extra poin
                prevPlay = plays[len(plays)-1]
                if prevPlay.playType == PLAY_PENALTY:
                    yardline = prevPlay.endingYard
            p = Play(qtr, homePoss, teamPoss, teamDef, 0, 0, yardline)
            touchLoc = restLower.find('touchdown')
            if touchLoc != -1 and (restLower.find('extra point') != -1 or restLower.find('two-po
                touchRest = rest[0:touchLoc+10]
                touchRestLower = restLower[0:touchLoc+10]
                getPlay(p, touchRestLower, touchRest)
                plays.append(p)
                rest = rest[touchLoc+10:]
                restLower = restLower[touchLoc+10:]
                if p.didChangePoss():
                    p = Play(qtr, not homePoss, teamDef, teamPoss, 0, 'Goal', 97)
                else:
                    p = Play(qtr, homePoss, teamPoss, teamDef, 0, 'Goal', 97)
            getPlay(p, restLower, rest)
            #print ' '.join(rowList)
            #p.printSelf()
            if p.playType != PLAY_NULL:
                plays.append(p)
            #print '>> [nothing] ' + ' '.join(rowList)
        else:
            pass
            #print '>> [???] ' + ' '.join(rowList)
        i += 1
```

```
    return plays

def parseRow(row):
    loc = row.find('<')
    while loc != -1:
        loc2 = row.find('>', loc)
        row = row[:loc] + '#' + row[loc2+1:]
        loc = row.find('<')
    row.strip('#')
    rowList = row.split('#')
    rowList = filter(notEmpty, rowList)
    return rowList

def notEmpty(item):
    return item != ''

# for a give play, get the down, distance, and yardline
def ddy(item, name, oppSide):
    down = 0
    if item.find('1st') != -1:
        down = 1
    elif item.find('2nd') != -1:
        down = 2
    elif item.find('3rd') != -1:
        down = 3
    elif item.find('4th') != -1:
        down = 4
    andLoc = item.find('and')
    atLoc = item.find('at')
    distance = item[andLoc+4:atLoc-1]
    nameLoc = item.find(name)
    yardline = int(item[nameLoc+len(name)+1:])
    if oppSide:
        yardline = 100 - yardline
    return [down, distance, yardline]

def getPlay(play, item, itemWNames):
    play.body = itemWNames
    gain = 0

    kickoffLoc = item.find(' kickoff')
    onsideLoc = item.find('on-side kick')
    fieldGoalLoc = item.find(' field goal')
    extraPointLoc = item.find(' extra point')
    if extraPointLoc == -1:
        extraPointLoc = item.find('blocked pat')
    puntLoc = item.find(' punt ')
    rushLoc = item.find(' rush ')
    sackLoc = item.find(' sacked')
    scrambleLoc = item.find(' scramble')
    passLoc = item.find(' pass ')
    recoverLoc = item.find(' recover')

    returnLoc = item.find(' return')
    fumbleLoc = item.find(' fumble')
    lateralLoc = item.find(' lateral')
```

```
    penaltyLoc = item.find(' penalty')
    nextPlayLoc = len(item)
    if returnLoc != -1 and returnLoc < nextPlayLoc:
        nextPlayLoc = returnLoc
    if fumbleLoc != -1 and fumbleLoc < nextPlayLoc:
        nextPlayLoc = fumbleLoc
    if lateralLoc != -1 and lateralLoc < nextPlayLoc:
        nextPlayLoc = lateralLoc
    if penaltyLoc != -1 and penaltyLoc < nextPlayLoc:
        nextPlayLoc = penaltyLoc

    # two-point conversion
    if item.find('two-point conversion', 0, nextPlayLoc) != -1:
        #print "2PC - " + item
        play.tryDown = True
        play.down = 0
        play.distance = 'Goal'
        play.startingYard = 97
        attemptLoc = item.find(' attempt,', 0, nextPlayLoc)
        passLoc = item.find(' pass ', attemptLoc, nextPlayLoc)
        rushLoc = item.find(' rush ', attemptLoc, nextPlayLoc)
        if item.find(' failed', 0, nextPlayLoc) != -1:
            if passLoc != -1:
                play.playType = PLAY_PASS
                qbName = itemWNames[attemptLoc+9:passLoc]
                play.qb_kicker = qbName.strip()
            elif rushLoc != -1:
                play.playType = PLAY_RUSH
                rusherName = itemWNames[attemptLoc+9:rushLoc]
                play.player_returner = rusherName.strip()
            play.endingYard = play.startingYard
        elif item.find(' good', 0, nextPlayLoc) != -1:
            play.touchdown = True
            play.pointsThisPlay = 2
            if passLoc != -1:
                play.playType = PLAY_PASS
                qbName = itemWNames[attemptLoc+9:passLoc]
                play.qb_kicker = qbName.strip()
                toLoc = item.find(' to ', attemptLoc, nextPlayLoc)
                goodLoc = item.find(' good', attemptLoc, nextPlayLoc)
                if toLoc != -1:
                    receiverName = itemWNames[toLoc+4:goodLoc]
                    play.player_returner = receiverName.strip()
            elif rushLoc != -1:
                play.playType = PLAY_RUSH
                rusherName = itemWNames[attemptLoc+9:rushLoc]
                play.player_returner = rusherName.strip()
            play.endingYard = 100
    # extra point
    elif extraPointLoc != -1:
        play.tryDown = True
        play.playType = PLAY_FIELD_GOAL
        play.down = 0
        play.distance = 'Goal'
        play.startingYard = 97
        kickerName = itemWNames[0:extraPointLoc]
        play.qb_kicker = kickerName
```

```
        if item.find('missed', extraPointLoc, nextPlayLoc) != -1:
            play.endingYard = play.startingYard
        elif item.find('good', 0, nextPlayLoc) != -1:
            play.fieldGoal = True
            play.pointsThisPlay = 1
            play.endingYard = play.startingYard
        elif item.find('blocked', extraPointLoc, nextPlayLoc) != -1:
            #print "EP BLOCKED - " + item
            play.block = True
            # recovery
            if recoverLoc != -1:
                if returnLoc != -1:
                    recovery = item[recoverLoc+7:returnLoc]
                else:
                    recovery = item[recoverLoc+7:]
                byLoc = recovery.find(' by ', recoverLoc, nextPlayLoc)
                if byLoc != -1:
                    atLoc = recovery.find(' at ', byLoc, nextPlayLoc)
                    commaLoc = recovery.find(',', byLoc, nextPlayLoc)
                    periodLoc = recovery.find('.', byLoc, nextPlayLoc)
                    if periodLoc == byLoc+5:
                        periodLoc = len(recovery)
                        periodLoc = recovery.find('.', byLoc+8, nextPlayLoc)
                    if periodLoc == byLoc+6:
                        periodLoc = len(recovery)
                        periodLoc = recovery.find('.', byLoc+9, nextPlayLoc)
                    loc = len(recovery)
                    if atLoc != -1 and atLoc < loc:
                        loc = atLoc
                    if commaLoc != -1 and commaLoc < loc:
                        loc = commaLoc
                    if periodLoc != -1 and periodLoc < loc:
                        loc = periodLoc
                    recovererName = itemWNames[recoverLoc+byLoc+11:loc]
                    play.defense1 = recovererName.strip()
                kickLength = getGain(play, recovery)
                play.endingYard = play.startingYard + kickLength
            else:
                play.endingYard = play.startingYard
            if item.find('two-point') != -1:
                play.pointsThisPlay = 2
    # field goal
    elif fieldGoalLoc != -1:
        play.playType = PLAY_FIELD_GOAL
        i = 0
        while i < fieldGoalLoc and not item[i].isdigit():
            i += 1
        kickerName = itemWNames[0:i-1]
        play.qb_kicker = kickerName
        if item.find('missed', fieldGoalLoc, nextPlayLoc) != -1:
            play.changePoss = True
            if play.startingYard > 80:
                play.endingYard = 80
            else:
                play.endingYard = play.startingYard
        elif item.find('good', fieldGoalLoc, nextPlayLoc) != -1:
            play.fieldGoal = True
```

```
            play.pointsThisPlay = 3
            play.endingYard = play.startingYard
        elif item.find('blocked', fieldGoalLoc, nextPlayLoc) != -1:
            play.block = True
            play.changePoss = True
            # recovery
            if recoverLoc != -1:
                if returnLoc != -1:
                    recovery = item[recoverLoc+7:returnLoc]
                else:
                    recovery = item[recoverLoc+7:]
                byLoc = recovery.find(' by ', recoverLoc, nextPlayLoc)
                if byLoc != -1:
                    atLoc = recovery.find(' at ', byLoc, nextPlayLoc)
                    commaLoc = recovery.find(',', byLoc, nextPlayLoc)
                    periodLoc = recovery.find('.', byLoc, nextPlayLoc)
                    if periodLoc == byLoc+5:
                        periodLoc = len(recovery)
                        periodLoc = recovery.find('.', byLoc+8)
                    if periodLoc == byLoc+6:
                        periodLoc = len(recovery)
                        periodLoc = recovery.find('.', byLoc+9)
                    loc = len(recovery)
                    if atLoc != -1 and atLoc < loc:
                        loc = atLoc
                    if commaLoc != -1 and commaLoc < loc:
                        loc = commaLoc
                    if periodLoc != -1 and periodLoc < loc:
                        loc = periodLoc
                    recovererName = itemWNames[recoverLoc+byLoc+11:loc]
                    play.defense1 = recovererName.strip()
                kickLength = getGain(play, recovery)
                play.endingYard = play.startingYard + kickLength
            else:
                play.endingYard = play.startingYard
    # kickoff
    elif kickoffLoc != -1:
        play.playType = PLAY_KICKOFF
        play.kickoff = True
        play.changePoss = True
        kickerName = itemWNames[0:kickoffLoc]
        play.qb_kicker = kickerName
        if returnLoc != -1:
            kickLength = getGain(play, item[kickoffLoc+8:returnLoc])
        else:
            kickLength = getGain(play, item[kickoffLoc+8:])
        play.endingYard = play.startingYard + kickLength
    # on-side kick
    elif onsideLoc != -1:
        play.playType = PLAY_KICKOFF
        play.kickoff = True
        kickerName = itemWNames[0:onsideLoc]
        play.qb_kicker = kickerName
        if returnLoc != -1:
            kickLength = getGain(play, item[kickoffLoc+8:returnLoc])
        else:
            kickLength = getGain(play, item[kickoffLoc+8:])
```

```
        play.endingYard = play.startingYard + kickLength
    # punt
    elif puntLoc != -1:
        play.playType = PLAY_PUNT
        play.changePoss = True
        kickerName = itemWNames[0:puntLoc]
        play.qb_kicker = kickerName
        if item.find('blocked', puntLoc, nextPlayLoc) != -1:
            play.block = True
            puntLength = 0
            # recovery
            if recoverLoc != -1:
                if returnLoc != -1:
                    recovery = item[recoverLoc+7:returnLoc]
                else:
                    recovery = item[recoverLoc+7:]
                byLoc = recovery.find(' by ', recoverLoc, nextPlayLoc)
                if byLoc != -1:
                    atLoc = recovery.find(' at ', byLoc, nextPlayLoc)
                    commaLoc = recovery.find(',', byLoc, nextPlayLoc)
                    periodLoc = recovery.find('.', byLoc, nextPlayLoc)
                    if periodLoc == byLoc+5:
                        periodLoc = len(recovery)
                        periodLoc = recovery.find('.', byLoc+8, nextPlayLoc)
                    if periodLoc == byLoc+6:
                        periodLoc = len(recovery)
                        periodLoc = recovery.find('.', byLoc+9, nextPlayLoc)
                    loc = len(recovery)
                    if atLoc != -1 and atLoc < loc:
                        loc = atLoc
                    if commaLoc != -1 and commaLoc < loc:
                        loc = commaLoc
                    if periodLoc != -1 and periodLoc < loc:
                        loc = periodLoc
                    recovererName = itemWNames[recoverLoc+byLoc+11:loc]
                    play.defense1 = recovererName.strip()
                puntLength = getGain(play, recovery)
            else:
                play.endingYard = play.startingYard
        elif returnLoc != -1:
            puntLength = getGain(play, item[puntLoc+5:returnLoc])
        else:
            puntLength = getGain(play, item[puntLoc+5:])
        play.endingYard = play.startingYard + puntLength
    # rush
    elif rushLoc != -1:
        play.playType = PLAY_RUSH
        rusherName = itemWNames[0:rushLoc]
        play.player_returner = rusherName
        if returnLoc != -1:
            gain = getGain(play, item[rushLoc+5:returnLoc])
        else:
            gain = getGain(play, item[rushLoc+5:])
        play.endingYard = play.startingYard + gain
    # sack
    elif sackLoc != -1:
        play.playType = PLAY_SACK
```

```
        qbName = itemWNames[0:sackLoc]
        play.qb_kicker = qbName
        byLoc = item.find(' by ', sackLoc, nextPlayLoc)
        if byLoc != -1:
            atLoc = item.find(' at ', byLoc, nextPlayLoc)
            forLoc = item.find(' for ', byLoc, nextPlayLoc)
            commaLoc = item.find(',', byLoc, nextPlayLoc)
            periodLoc = item.find('.', byLoc, nextPlayLoc)
            if periodLoc == byLoc+5:
                periodLoc = len(itemWNames)
                periodLoc = item.find('.', byLoc+8, nextPlayLoc)
            if periodLoc == byLoc+6:
                periodLoc = len(itemWNames)
                periodLoc = item.find('.', byLoc+9, nextPlayLoc)
            loc = len(itemWNames)
            if atLoc != -1 and atLoc < loc:
                loc = atLoc
            if forLoc != -1 and forLoc < loc:
                loc = forLoc
            if commaLoc != -1 and commaLoc < loc:
                loc = commaLoc
            if periodLoc != -1 and periodLoc < loc:
                loc = periodLoc
            sackerName = itemWNames[byLoc+4:loc]
            andLoc = sackerName.find(' and ', sackLoc, nextPlayLoc)
            if andLoc != -1:
                sackerName1 = sackerName[0:andLoc]
                sackerName2 = sackerName[andLoc+5:]
                play.defense1 = sackerName1.strip()
                play.defense2 = sackerName2.strip()
            else:
                play.defense1 = sackerName.strip()
        if returnLoc != -1:
            gain = getGain(play, item[sackLoc+7:returnLoc])
        else:
            gain = getGain(play, item[sackLoc+7:])
        if gain > 0:
            gain = -gain
        play.endingYard = play.startingYard + gain
    # scramble
    elif scrambleLoc != -1:
        play.playType = PLAY_RUSH
        qbName = itemWNames[0:scrambleLoc]
        play.qb_kicker = qbName
        if returnLoc != -1:
            gain = getGain(play, item[rushLoc+9:returnLoc])
        else:
            gain = getGain(play, item[rushLoc+9:])
        play.endingYard = play.startingYard + gain
    # pass
    elif passLoc != -1 and (penaltyLoc == -1 or passLoc < penaltyLoc):
        play.playType = PLAY_PASS
        qbName = itemWNames[0:passLoc]
        play.qb_kicker = qbName
        toLoc = item.find(' to ', passLoc, nextPlayLoc)
        if toLoc != -1:
            forLoc = item.find(' for ', toLoc, nextPlayLoc)
```

```
            commaLoc = item.find(',', toLoc, nextPlayLoc)
            periodLoc = item.find('.', toLoc, nextPlayLoc)
            if periodLoc == toLoc+5:
                periodLoc = len(itemWNames)
                periodLoc = item.find('.', toLoc+8, nextPlayLoc)
            if periodLoc == toLoc+6:
                periodLoc = len(itemWNames)
                periodLoc = item.find('.', toLoc+9, nextPlayLoc)
            loc = len(itemWNames)
            if forLoc != -1 and forLoc < loc:
                loc = forLoc
            if commaLoc != -1 and commaLoc < loc:
                loc = commaLoc
            if periodLoc != -1 and periodLoc < loc:
                loc = periodLoc
            receiverName = itemWNames[toLoc+4:loc]
            play.player_returner = receiverName.strip()
        if item.find('incomplete', passLoc, nextPlayLoc) != -1:
            gain = 0
        elif returnLoc != -1:
            gain = getGain(play, item[passLoc+5:returnLoc])
        else:
            gain = getGain(play, item[passLoc+5:])
        play.endingYard = play.startingYard + gain
        interceptionLoc = item.find('intercepted', passLoc, nextPlayLoc)
        if interceptionLoc != -1:
            play.interception = True
            play.changePoss = True
            play.player_returner = ''
            byLoc = item.find(' by ', interceptionLoc, nextPlayLoc)
            if byLoc != -1:
                atLoc = item.find(' at ', byLoc, nextPlayLoc)
                commaLoc = item.find(',', byLoc, nextPlayLoc)
                periodLoc = item.find('.', byLoc, nextPlayLoc)
                if periodLoc == byLoc+5:
                    periodLoc = len(itemWNames)
                    periodLoc = item.find('.', byLoc+8, nextPlayLoc)
                if periodLoc == byLoc+6:
                    periodLoc = len(itemWNames)
                    periodLoc = item.find('.', byLoc+9, nextPlayLoc)
                loc = len(itemWNames)
                if atLoc != -1 and atLoc < loc:
                    loc = atLoc
                if commaLoc != -1 and commaLoc < loc:
                    loc = commaLoc
                if periodLoc != -1 and periodLoc < loc:
                    loc = periodLoc
                interceptorName = itemWNames[byLoc+4:loc]
                play.defense1 = interceptorName.strip()
    # timeout
    elif item.find('timeout') != -1:
        play.playType = PLAY_NULL
        play.endingYard = play.startingYard
    # end of quarter
    elif item.find('end of ') != -1 and (item.find(' quarter') != -1 or item.find(' ot') != -1):
        play.playType = PLAY_NULL
        play.endingYard = play.startingYard
```

```
    # nothing listed
    elif len(item) == 0:
        play.playType = PLAY_NULL
        play.endingYard = play.startingYard
    # has a next play
    if nextPlayLoc != len(item):
        # don't worry
        pass
    # unknown play
    #else:
    #   if item != '   ':
    #       print '[unknown] ' + item

    if nextPlayLoc != len(item):
        getNextPlay(play, item, itemWNames, returnLoc, fumbleLoc, lateralLoc, penaltyLoc, nextPl
    if item.find('1st down', 0, nextPlayLoc) != -1:
        play.firstDown = True
    if item.find('touchdown', 0, nextPlayLoc) != -1:
        play.touchdown = True
        play.pointsThisPlay = 6
    if item.find('safety', 0, nextPlayLoc) != -1:
        play.safety = True
        play.pointsThisPlay = 2
    if item.find('touchback', 0, nextPlayLoc) != -1:
        play.touchback = True
        play.changePoss = True
        play.endingYard = 80

    if play.down == 4:
        play.turnoverOnDowns()
    return 0

def getNextPlay(play, item, itemWNames, returnLoc, fumbleLoc, lateralLoc, penaltyLoc, nextPlayLo
    if returnLoc == nextPlayLoc:
        play.changePoss = True
        p = Play(play.quarter, (not play.homePoss), play.teamDef, play.teamPoss, 0, 0, 100 - pla
        p.playType = PLAY_RETURN
        play.nextPlay = p
        getReturn(p, item, itemWNames, returnLoc)
    elif fumbleLoc == nextPlayLoc:
        if play.playType == PLAY_NULL:
            play.playType = PLAY_RUSH
            play.endingYard = play.startingYard
        play.fumble = True
        getFumble(play, item, itemWNames, fumbleLoc)
    elif lateralLoc == nextPlayLoc:
        p = Play(play.quarter, play.homePoss, play.teamPoss, play.teamDef, 0, 0, play.endingYard
        p.playType = PLAY_LATERAL
        play.nextPlay = p
        getLateral(p, item, itemWNames, lateralLoc)
    else:
        if play.playType == PLAY_NULL:
            play.playType = PLAY_PENALTY
            p = play
        else:
            p = Play(play.quarter, play.homePoss, play.teamPoss, play.teamDef, 0, 0, play.ending
            p.playType = PLAY_PENALTY
```

```
            play.nextPlay = p
        getPenalty(p, item, itemWNames, penaltyLoc)

def getReturn(play, item, itemWNames, returnLoc):
    returnLoc2 = item.find(' return', returnLoc+1)
    fumbleLoc = item.find(' fumble', returnLoc)
    lateralLoc = item.find(' lateral', returnLoc)
    penaltyLoc = item.find(' penalty', returnLoc)
    nextPlayLoc = len(item)
    if returnLoc2 != -1 and returnLoc2 < nextPlayLoc:
        nextPlayLoc = returnLoc2
    if fumbleLoc != -1 and fumbleLoc < nextPlayLoc:
        nextPlayLoc = fumbleLoc
    if lateralLoc != -1 and lateralLoc < nextPlayLoc:
        nextPlayLoc = lateralLoc
    if penaltyLoc != -1 and penaltyLoc < nextPlayLoc:
        nextPlayLoc = penaltyLoc

    byLoc = item.find(' by ', returnLoc, nextPlayLoc)
    if byLoc != -1:
        forLoc = item.find(' for ', byLoc, nextPlayLoc)
        commaLoc = item.find(',', byLoc, nextPlayLoc)
        periodLoc = item.find('.', byLoc, nextPlayLoc)
        if periodLoc == byLoc+5:
            periodLoc = len(itemWNames)
            periodLoc = item.find('.', byLoc+8, nextPlayLoc)
        if periodLoc == byLoc+6:
            periodLoc = len(itemWNames)
            periodLoc = item.find('.', byLoc+9, nextPlayLoc)
        loc = len(itemWNames)
        if forLoc != -1 and forLoc < loc:
            loc = forLoc
        if commaLoc != -1 and commaLoc < loc:
            loc = commaLoc
        if periodLoc != -1 and periodLoc < loc:
            loc = periodLoc
        returnerName = itemWNames[byLoc+4:loc]
        play.player_returner = returnerName.strip()
    spaceLoc = item.find(' ', returnLoc+1, nextPlayLoc)
    returnLength = getGain(play, item[spaceLoc:])
    play.endingYard = play.startingYard + returnLength

    if nextPlayLoc != len(item):
        getNextPlay(play, item, itemWNames, returnLoc2, fumbleLoc, lateralLoc, penaltyLoc, nextP
    if item.find('1st down', returnLoc, nextPlayLoc) != -1:
        play.firstDown = True
    if item.find('touchdown', returnLoc, nextPlayLoc) != -1:
        play.touchdown = True
        play.pointsThisPlay = 6
    if item.find('safety', returnLoc, nextPlayLoc) != -1:
        play.safety = True
        play.pointsThisPlay = 2
    if item.find('touchback', returnLoc, nextPlayLoc) != -1:
        play.touchback = True
        play.changePoss = True
        play.endingYard = 80
```

```
def getFumble(play, item, itemWNames, fumbleLoc):
    returnLoc = item.find(' return', fumbleLoc)
    fumbleLoc2 = item.find(' fumble', fumbleLoc+1)
    lateralLoc = item.find(' lateral', fumbleLoc)
    penaltyLoc = item.find(' penalty', fumbleLoc)
    nextPlayLoc = len(item)
    if returnLoc != -1 and returnLoc < nextPlayLoc:
        nextPlayLoc = returnLoc
    if fumbleLoc2 != -1 and fumbleLoc2 < nextPlayLoc:
        nextPlayLoc = fumbleLoc2
    if lateralLoc != -1 and lateralLoc < nextPlayLoc:
        nextPlayLoc = lateralLoc
    if penaltyLoc != -1 and penaltyLoc < nextPlayLoc:
        nextPlayLoc = penaltyLoc

    forcedLoc = item.find(' forced ', fumbleLoc, nextPlayLoc)
    recoverLoc = item.find(' recover', fumbleLoc, nextPlayLoc)
    # forcing
    if forcedLoc != -1:
        if recoverLoc != -1:
            forcing = item[forcedLoc+7:recoverLoc]
        elif returnLoc != -1:
            forcing = item[forcedLoc+7:returnLoc]
        else:
            forcing = item[forcedLoc+7:]
        byLoc = forcing.find(' by ', forcedLoc, nextPlayLoc)
        if byLoc != -1:
            atLoc = forcing.find(' at ', byLoc, nextPlayLoc)
            commaLoc = forcing.find(',', byLoc, nextPlayLoc)
            periodLoc = forcing.find('.', byLoc, nextPlayLoc)
            if periodLoc == byLoc+5:
                periodLoc = len(forcing)
                periodLoc = forcing.find('.', byLoc+8, nextPlayLoc)
            elif periodLoc == byLoc+6:
                periodLoc = len(forcing)
                periodLoc = forcing.find('.', byLoc+9, nextPlayLoc)
            loc = len(forcing)
            if atLoc != -1 and atLoc < loc:
                loc = atLoc
            if commaLoc != -1 and commaLoc < loc:
                loc = commaLoc
            if periodLoc != -1 and periodLoc < loc:
                loc = periodLoc
            forcerName = itemWNames[forcedLoc+byLoc+11:loc]
            play.defense1 = forcerName.strip()
    # recovery
    recovery = ''
    if recoverLoc != -1:
        if returnLoc != -1:
            recovery = item[recoverLoc+7:returnLoc]
        else:
            recovery = item[recoverLoc+7:]
        byLoc = recovery.find(' by ', recoverLoc, nextPlayLoc)
        if byLoc != -1:
            atLoc = recovery.find(' at ', byLoc, nextPlayLoc)
            forLoc = recovery.find(' for ', byLoc, nextPlayLoc)
            andLoc = recovery.find(' and ', byLoc, nextPlayLoc)
```

```
            outOfBoundsLoc = recovery.find(' out-of-bounds', byLoc, nextPlayLoc)
            commaLoc = recovery.find(',', byLoc, nextPlayLoc)
            periodLoc = recovery.find('.', byLoc, nextPlayLoc)
            if periodLoc == byLoc+5:
                periodLoc = len(recovery)
                periodLoc = recovery.find('.', byLoc+8, nextPlayLoc)
            elif periodLoc == byLoc+6:
                periodLoc = len(recovery)
                periodLoc = recovery.find('.', byLoc+9, nextPlayLoc)
            loc = len(recovery)
            if atLoc != -1 and atLoc < loc:
                loc = atLoc
            if forLoc != -1 and forLoc < loc:
                loc = forLoc
            if andLoc != -1 and andLoc < loc:
                loc = andLoc
            if outOfBoundsLoc != -1 and outOfBoundsLoc < loc:
                loc = outOfBoundsLoc
            if commaLoc != -1 and commaLoc < loc:
                loc = commaLoc
            if periodLoc != -1 and periodLoc < loc:
                loc = periodLoc
            recovererName = itemWNames[recoverLoc+byLoc+11:loc]
            play.defense2 = recovererName.strip()

    if returnLoc != -1:
        gain = getGain(play, recovery)
        play.endingYard = play.endingYard + gain
    elif recoverLoc != -1:
        p = Play(play.quarter, play.homePoss, play.teamPoss, play.teamDef, 0, 0, play.endingYard
        p.playType = PLAY_RECOVERY
        gain = getGain(p, recovery)
        p.endingYard = p.startingYard + gain
        play.nextPlay = p
        play = p
    if nextPlayLoc != len(item):
        getNextPlay(play, item, itemWNames, returnLoc, fumbleLoc2, lateralLoc, penaltyLoc, nextP
    if item.find('1st down', fumbleLoc, nextPlayLoc) != -1:
        play.firstDown = True
    if item.find('touchdown', fumbleLoc, nextPlayLoc) != -1:
        play.touchdown = True
        play.pointsThisPlay = 6
    if item.find('safety', fumbleLoc, nextPlayLoc) != -1:
        play.safety = True
        play.pointsThisPlay = 2
    if item.find('touchback', fumbleLoc, nextPlayLoc) != -1:
        play.touchback = True
        play.changePoss = True
        play.endingYard = 80

def getLateral(play, item, itemWNames, lateralLoc):
    returnLoc = item.find(' return', lateralLoc)
    fumbleLoc = item.find(' fumble', lateralLoc)
    lateralLoc2 = item.find(' lateral', lateralLoc+1)
    penaltyLoc = item.find(' penalty', lateralLoc)
    nextPlayLoc = len(item)
    if returnLoc != -1 and returnLoc < nextPlayLoc:
```

```
        nextPlayLoc = returnLoc
    if fumbleLoc != -1 and fumbleLoc < nextPlayLoc:
        nextPlayLoc = fumbleLoc
    if lateralLoc2 != -1 and lateralLoc2 < nextPlayLoc:
        nextPlayLoc = lateralLoc2
    if penaltyLoc != -1 and penaltyLoc < nextPlayLoc:
        nextPlayLoc = penaltyLoc

    toLoc = item.find(' to ', lateralLoc, nextPlayLoc)
    if toLoc != -1:
        forLoc = item.find(' for ', toLoc, nextPlayLoc)
        commaLoc = item.find(',', toLoc, nextPlayLoc)
        periodLoc = item.find('.', toLoc, nextPlayLoc)
        if periodLoc == toLoc+5:
            periodLoc = len(itemWNames)
            periodLoc = item.find('.', toLoc+8, nextPlayLoc)
        if periodLoc == toLoc+6:
            periodLoc = len(itemWNames)
            periodLoc = item.find('.', toLoc+9, nextPlayLoc)
        loc = len(itemWNames)
        if forLoc != -1 and forLoc < loc:
            loc = forLoc
        if commaLoc != -1 and commaLoc < loc:
            loc = commaLoc
        if periodLoc != -1 and periodLoc < loc:
            loc = periodLoc
        receiverName = itemWNames[toLoc+4:loc]
        play.player_returner = receiverName.strip()
    gain = getGain(play, item[lateralLoc+8:])
    play.endingYard = play.startingYard + gain
    if nextPlayLoc != len(item):
        getNextPlay(play, item, itemWNames, returnLoc, fumbleLoc, lateralLoc2, penaltyLoc, nextP
    if item.find('1st down', lateralLoc, nextPlayLoc) != -1:
        play.firstDown = True
    if item.find('touchdown', lateralLoc, nextPlayLoc) != -1:
        play.touchdown = True
        play.pointsThisPlay = 6
    if item.find('safety', lateralLoc, nextPlayLoc) != -1:
        play.safety = True
        play.pointsThisPlay = 2
    if item.find('touchback', lateralLoc, nextPlayLoc) != -1:
        play.touchback = True
        play.changePoss = True
        play.endingYard = 80

def getPenalty(play, item, itemWNames, penaltyLoc):
    penaltyLoc2 = item.find(' penalty', penaltyLoc+1)
    acceptedLoc = item.find(' accepted', penaltyLoc, penaltyLoc2)
    declinedLoc = item.find(' declined', penaltyLoc, penaltyLoc2)
    offsettingLoc = item.find('off-setting', penaltyLoc, penaltyLoc2)
    gain = 0
    if declinedLoc != -1 or  offsettingLoc != -1:
        gain = 0
    else:
        i = penaltyLoc+9
        while i < len(item) and not item[i].isdigit():
            i += 1
```

```
        gain = getGain(play, item[i:])
        if item.find(play.teamPoss.lower(), penaltyLoc, penaltyLoc2) != -1:
            gain = -gain
        elif item.find('1st down', penaltyLoc, penaltyLoc2) != -1:
            play.firstDown = True
    onLoc = item.find(' on ', penaltyLoc, penaltyLoc2)
    if onLoc != -1:
        commaLoc = item.find(',', onLoc, penaltyLoc2)
        periodLoc = item.find('.', onLoc, penaltyLoc2)
        if periodLoc == onLoc+5:
            periodLoc = len(itemWNames)
            periodLoc = item.find('.', onLoc+8)
        if periodLoc == onLoc+6:
            periodLoc = len(itemWNames)
            periodLoc = item.find('.', onLoc+9)
        loc = len(itemWNames)
        if acceptedLoc != -1 and acceptedLoc < loc:
            loc = acceptedLoc
        if declinedLoc != -1 and declinedLoc < loc:
            loc = declinedLoc
        if offsettingLoc != -1 and offsettingLoc < loc:
            loc = offsettingLoc
        if commaLoc != -1 and commaLoc < loc:
            loc = commaLoc
        if periodLoc != -1 and periodLoc < loc:
            loc = periodLoc
        penalizedName = itemWNames[onLoc+4:loc]
        play.player_returner = penalizedName.strip()
    play.endingYard = play.startingYard + gain
    if penaltyLoc2 != -1:
        p = Play(play.quarter, play.homePoss, play.teamPoss, play.teamDef, 0, 0, play.endingYard
        p.playType = PLAY_PENALTY
        play.nextPlay = p
        getPenalty(p, item, itemWNames, penaltyLoc2)

def getGain(play, item):
    gain = 0
    penaltyLoc = item.find(' penalty')
    if penaltyLoc == -1:
        penaltyLoc = len(item)
    forLoc = item.find(' for ')
    if forLoc == -1:
        yardLoc = item.find(' yard', 0, penaltyLoc)
        if yardLoc == -1:
            yardLoc = item.find(' yds ', 0, penaltyLoc)
    elif forLoc == item.find(' for a touch') or forLoc == item.find(' for a safety'):
        forLoc = -1
        yardLoc = item.find(' yard', 0, penaltyLoc)
        if yardLoc == -1:
            yardLoc = item.find(' yds ', 0, penaltyLoc)
    else:
        yardLoc = item.find(' yard', forLoc, penaltyLoc)
        if yardLoc == -1:
            yardLoc = item.find(' yds ', forLoc, penaltyLoc)
    fiftyYardLoc = item.find(' the 50 yard ')
    if fiftyYardLoc != -1 and (fiftyYardLoc + 7 ) == yardLoc:
        yardLoc = -1
```

```
    lossLoc = item.find('loss of', forLoc, yardLoc)
    #print str(forLoc) + ' ' + str(yardLoc) + ' ' + str(lossLoc)
    noGainLoc = item.find('no gain')
    toTheLoc = item.find(' to the ')
    atTheLoc = item.find(' at the ')
    atLoc = item.find(' at ')
    ballOnLoc = item.find(' ball on ')
    if toTheLoc != -1 and atTheLoc != -1:
        if toTheLoc < atTheLoc:
            atTheLoc = -1
        else:
            toTheLoc = -1
    if yardLoc != -1 and (noGainLoc == -1 or yardLoc < noGainLoc):
        if lossLoc != -1:
            gainStr = item[lossLoc+8:yardLoc]
            if gainStr.strip('- ').isdigit():
                gain = -int(gainStr)
            else:
                print '[loss err] ' + item + ' - ' + gainStr
        else:
            if forLoc != -1:
                gainStr = item[forLoc+5:yardLoc]
            else:
                gainStr = item[0:yardLoc]
            parenLoc = gainStr.find(')')
            if parenLoc != -1:
                gainStr = gainStr[parenLoc+1:]
            if gainStr.strip('- ').isdigit():
                gain = int(gainStr)
            else:
                print '[gain err] ' + item + ' - ' + gainStr
    elif noGainLoc != -1:
        gain = 0
    elif toTheLoc != -1:
        yardLine = getYardLine(play, item[toTheLoc+8:])
        play.endingYard = yardLine
        gain = play.endingYard - play.startingYard
    elif atTheLoc != -1:
        yardLine = getYardLine(play, item[atTheLoc+8:])
        play.endingYard = yardLine
        gain = play.endingYard - play.startingYard
    elif atLoc != -1:
        yardLine = getYardLine(play, item[atLoc+4:])
        play.endingYard = yardLine
        gain = play.endingYard - play.startingYard
    elif ballOnLoc != -1:
    #   print 'ball on'
    #   print item[ballOnLoc+9:]
        yardLine = getYardLine(play, item[ballOnLoc+9:])
    #   print 'yardline = ' + str(yardline)
        play.endingYard = yardLine
        gain = play.endingYard - play.startingYard
    #else:
        #print '[lossorgain err] ' + item
    #print 'gain = ' + str(gain) + ', item = ' + str(item)
    return gain
```

```
def getYardLine(play, item):
    # decide team
    ownSide = True
    offense = play.teamPoss.lower()
    defense = play.teamDef.lower()
    if item[0] == offense[0] and item[0] != defense[0]:
        ownSide = True
    elif item[0] != offense[0] and item[0] == defense[0]:
        ownSide = False
    else:
        i = 1
        while i < 5:
            letter = item[i]
            offLoc = offense.find(letter)
            defLoc = defense.find(letter)
            if offLoc != -1 and defLoc == -1:
                ownSide = True
                break
            elif offLoc == -1 and defLoc != -1:
                ownSide = False
                break
            elif offLoc < defLoc:
                ownSide = True
                break
            elif offLoc > defLoc:
                ownSide = False
                break
            else:
                i += 1
    # get yardline
    i = 1
    while i < len(item) and not item[i].isdigit():
        i += 1
    numStart = i
    while i < len(item) and item[i].isdigit():
        i += 1
    numEnd = i
    numerals = item[numStart:numEnd]
    if numerals.isdigit():
        yardLine = int(numerals)
    else:
        print '[yardline err:' + str(numStart) + ':' + str(numEnd) + '] ' + item
        yardLine = 50
    if not ownSide:
        yardLine = 100 - yardLine
    return yardLine

if __name__ == '__main__':
    import sys
    print geturlstring(sys.argv[1])
```