|  |  |  |  |
| --- | --- | --- | --- |
| **Team ID** | **City & Name** | **Team ID** | **City & Name** |
| ARI | Arizona Cardinals | LA | Los Angeles Rams |
| ATL | Atlanta Falcons | MIA | Miami Dolphins |
| BAL | Baltimore Ravens | MIN | Minnesota Vikings |
| BUF | Buffalo Bills | NE | New England Patriots |
| CAR | Carolina Panthers | NO | New Orleans Saints |
| CHI | Chicago Bears | NYG | New York Giants |
| CIN | Cincinnati Bengals | NYJ | New York Jets |
| CLE | Cleveland Browns | OAK | Oakland Raiders |
| DAL | Dallas Cowboys | PHI | Philadelphia Eagles |
| DEN | Denver Broncos | PIT | Pittsburgh Steelers |
| DET | Detroit Lions | SD | San Diego Chargers |
| GB | Green Bay Packers | SEA | Seattle Seahawks |
| HOU | Houston Texans | SF | San Francisco 49ers |
| IND | Indianapolis Colts | TB | Tampa Bay Buccaneers |
| JAC | Jacksonville Jaguars | TEN | Tennessee Titans |
| KC | Kansas City Chiefs | WAS | Washington Redskins |

Example to find top 5 running backs

import nflgame

games = nflgame.games(2013, week=1)

players = nflgame.combine\_game\_stats(games)

for p in players.rushing().sort('rushing\_yds').limit(5):

msg = '%s %d carries for %d yards and %d TDs'

print msg % (p, p.rushing\_att, p.rushing\_yds, p.rushing\_tds)

Example to find top 5 passing plays

import nflgame

games = nflgame.games(2013, week=1)

plays = nflgame.combine\_plays(games)

for p in plays.sort('passing\_yds').limit(5):

print p

def find(name, team=None):

*"""*

*Finds player or players with a name matching (case insensitive)*

*name and returns them as a list.*

*If team is not None, it is used as an additional search constraint.*

*"""*

hits = []

for player in players.itervalues():

if player.name.lower() == name.lower():

if team is None or team.lower() == player.team.lower():

hits.append(player)

return hits

def combine(games, plays=False):

*"""*

*Combines a list of games into one big player sequence containing game*

*level statistics.*

*This can be used, for example, to get PlayerStat objects corresponding to*

*statistics across an entire week, some number of weeks or an entire season.*

*If the plays parameter is True, then statistics will be derived from*

*play by play data.*

*"""*

if plays:

return combine\_play\_stats(games)

else:

return combine\_game\_stats(games)

def combine\_max\_stats(games):

*"""*

*Combines a list of games into one big player sequence containing maximum*

*stats based on game and play level stats.*

*This can be used, for example, to get GamePlayerStats objects corresponding*

*to statistics across an entire week, some number of weeks or an entire*

*season.*

*This function should be used of combine\_game\_stats or*

*combine\_play\_stats when the best possible accuracy is desired.*

*"""*

return reduce(lambda a, b: a + b,

[g.max\_player\_stats() for g in games if g is not None])

def combine\_play\_stats(games):

*"""*

*Combines a list of games into one big player sequence containing play*

*level statistics.*

*This can be used, for example, to get PlayPlayerStats objects corresponding*

*to statistics across an entire week, some number of weeks or an entire*

*season.*

*This function should be used in lieu of combine\_game\_stats when more*

*detailed statistics such as receiver targets, yards after the catch and*

*punt/FG blocks are needed.*

*Since this combines all play data, this function may take a while*

*to complete depending on the number of games passed in.*

*"""*

return reduce(lambda p1, p2: p1 + p2,

[g.drives.players() for g in games if g is not None])