Documentation

The `Game` class

class **yahoo_fantasy_api.game.Game**(*sc*, *code*)

Abstraction for a Yahoo! fantasy game

Parameters:
- **sc** (`yahoo_oauth.OAuth2`) – Fully constructed session context
- **code** (*str*) – Sport code (mlb, nhl, etc)

game_id()

Return the Yahoo! Game ID :return: Game ID :rtype: str

league_ids(*year=None*, *is_available=False*, *game_types=None*, *game_codes=None*, *seasons=None*)

Return the Yahoo! league IDs that the current user played in

Parameters:
- **year** – Optional year, when provide _league_ids_deprecated() will be used to fetch the league ids
- **is_available** (*bool*) – Optional flag to filter out leagues that are not available
- **game_types** (*list[str]*) – Optional list of game types to filter league IDs returned. Valid values are full|pickem-team|pickem-group|pickem-team-list
- **game_codes** (*list[str]*) – Optional list of game codes(i.e. nfl, mlb, nhl, nba) to filter league IDs returned.
- **seasons** (*list[str]*) – Optional list of seasons to filter league IDs returned.

Returns: List of league ids

to_league(*league_id*)

Construct a League object from a Game

Parameters: **league_id** (*str*) – League ID of the new League to construct
Returns: Fully constructed object
Return type: League

The `League` class

class **yahoo_fantasy_api.league.League**(*sc*, *league_id*, *handler=None*)

An abstraction for all of the league-level APIs in Yahoo! fantasy

Parameters:
- **sc** (`yahoo_oauth.OAuth2`) – Fully constructed session context
- **league_id** (*str*) – League ID to setup this class for. All API requests will be for this league.

`current_week()`

Return the current week number of the league

Returns: Week number
Return type: int

```
>>> lg.current_week()
12
```

`draft_results()`

Get the results of the league's draft

This will return details about each pick made in the draft. For auction style drafts it includes the auction price paid for the player.

The players are returned as player IDs. Use the player_details() API to find more specifics on the player.

If this is called for a league that has not yet done a draft then it will return an empty list.

If this is called during the draft this includes the players that have been drafted thus far. For auction style drafts, it does not include the player currently being nominated.

Returns: Details about all of the players drafted.
Return type: list

```
>>> draft_res = lg.draft_results()
>>> len(draft_res)
210
>>> draft_res[0]
{'pick': 1,
 'round': 1,
 'cost': '4',
 'team_key': '388.l.27081.t.4',
 'player_id': 9490}
```

`edit_date()`

Return the next day that you can edit the lineups.

Returns: edit date
Return type: **class:** datetime.date

`end_week()`

Return the ending week number of the league.

Returns: Week number
Return type: int

```
>>> lg.end_week()
24
```

`free_agents(position)`

Return the free agents for the given position

Parameters: **position** (*str*) – All free agents must be able to play this position. Use the short code of the position (e.g. 2B, C, etc.). You can also specify the position type (e.g. 'B' for all batters and 'P' for all pitchers).
Returns: Free agents found. Particulars about each free agent will be returned.
Return type: List(Dict)

```
>>> fa_CF = lg.free_agents('CF')
>>> len(fa_CF)
60
>>> fa_CF[0]
{'player_id': 8370,
 'name': 'Dexter Fowler',
 'position_type': 'B',
 'eligible_positions': ['CF', 'RF', 'Util']}
```

`get_team(team_name)`

Construct a Team object from a League

Parameters: **team_name** (*str*) – Team name of the Team object to construct
Returns: A dictionary with the team name as the key and team object as the value
Return type: dict

matchups(*week=None*)

Retrieve matchups data for a given week. Defaults to current week.

Parameters:	**week** (*int*, *optional*) – Week to request, defaults to None
Returns:	Matchup details as key/value pairs
Return type:	dict

ownership(*player_ids*)

Retrieve the owner of a player

Parameters:	**player_ids** (*list(int)*) – Yahoo! Player IDs to retrieve owned for
Returns:	Ownership status of player
Return type:	dict

```
>>> lg.ownership([3737])
{"3737" : {"ownership_tpye" : "team", "owner_team_name": "team name"}}
```

percent_owned(*player_ids*)

Retrieve ownership percentage of a list of players

Parameters:	**player_ids** (*list(int)*) – Yahoo! Player IDs to retrieve % owned for
Returns:	Ownership percentage of players requested
Return type:	dict

```
>>> lg.percent_owned(1, [3737, 6381, 4003, 3705])
[{'player_id': 3737, 'name': 'Sidney Crosby', 'percent_owned': 100},
 {'player_id': 6381, 'name': 'Dylan Larkin', 'percent_owned': 89},
 {'player_id': 4003, 'name': 'Semyon Varlamov', 'percent_owned': 79},
 {'player_id': 3705, 'name': 'Dustin Byfuglien', 'percent_owned': 82}]
```

player_details(*player*)

Retrieve details about a number of players

Parm player:	If a str, this is a search string that will return all matches of the name (to a maximum of 25 players). It it is a int or list(int), then these are player IDs to lookup.
Returns:	Details of all of the players found. If given a player ID that does not exist, then a RuntimeError exception is thrown. If searching for players by name and none are found an empty list is returned.
Return type:	list(dict)

```
>>> lg.player_details('Phil Kessel')
[{'player_key': '396.p.3983',
  'player_id': '3983',
  'name': {'full': 'Phil Kessel',
           'first': 'Phil',
           'last': 'Kessel',
           'ascii_first': 'Phil',
           'ascii_last': 'Kessel'},
  'editorial_player_key': 'nhl.p.3983',
  'editorial_team_key': 'nhl.t.24',
  'editorial_team_full_name': 'Arizona Coyotes',
  'editorial_team_abbr': 'Ari',
  'uniform_number': '81',
  'display_position': 'RW',
  'headshot': {...},
  'image_url': '...',
  'is_undroppable': '0',
  'position_type': 'P',
  'primary_position': 'RW',
  'eligible_positions': [{'position': 'RW'}],
  ...
}]
>>> plyrs = lg.player_details([3983, 5085, 5387])
>>> len(plyrs)
3
>>> [p['name']['full'] for p in plyrs]
['Phil Kessel', 'Philipp Grubauer', 'Phillip Danault']
>>> plyrs = lg.player_details('Phil')
>>> len(plyrs)
14
```

player_stats(*player_ids*, *req_type*, *date=None*, *week=None*, *season=None*)

Return stats for a list of players

Parameters:

- **player_ids** (*list(int)*) – Yahoo! player IDs of the players to get stats for
- **req_type** (*str*) – Defines the date range for the stats. Valid values are: 'season', 'average_season', 'lastweek', 'lastmonth', 'date', 'week'. 'season' returns stats for a given season, specified by the season parameter. 'date' returns stats for a single date, specified by the date parameter. 'week' returns stats for a single week, specified by the week parameter. The 'last*' types return stats for a given time frame relative to the current.
- **date** (*datetime.date*) – When requesting stats for a single date, this identifies what date to request the stats for. If left as None, and range is for a date this returns stats for the current date.
- **week** (*int*) – NFL ONLY: When requesting stats for a week, this identifies the week. If None and requesting stats for a season, this will return stats for the current season.
- **season** (*int*) – When requesting stats for a season, this identifies the season. If None and requesting stats for a season, this will return stats for the current season.

Returns: Return the stats requested. Each entry in the list are stats for a single player. The list will one entry for each player ID requested.

Return type: list(dict)

```
>>> lg.player_stats([6743], 'season')
 [{'player_id': 6743,
   'name': 'Connor McDavid',
   'position_type': 'P',
   'GP': 32.0,
   'G': 19.0,
   'A': 33.0,
   'PTS': 52.0,
   '+/-': 1.0,
   'PIM': 18.0,
   'PPG': 8.0,
   'PPA': 15.0,
   'PPP': 23.0,
   'GWG': 2.0,
   'SOG': 106.0,
   'S%': 0.179,
   'PPT': 7429.0,
   'Avg-PPT': 232.0,
   'SHT': 277.0,
   'Avg-SHT': 9.0,
   'COR': -64.0,
   'FEN': -51.0,
   'Off-ZS': 310.0,
   'Def-ZS': 167.0,
   'ZS-Pct': 64.99,
   'GStr': 7.0,
   'Shifts': 684.0}]
```

positions()

Return the positions that are used in the league.

Returns: Dictionary of positions. Each key is a position, with a count and position type as the values.

Return type: dict(dict(position_type, count))

```
>>> lg.positions()
{'C': {'position_type': 'P', 'count': 2},
 'LW': {'position_type': 'P', 'count': 2},
 'RW': {'position_type': 'P', 'count': 2},
 'D': {'position_type': 'P', 'count': 4},
 'G': {'position_type': 'G', 'count': 2},
 'BN': {'count': 2},
 'IR': {'count': '3'}}
```

settings()

Return the league settings

Returns: League settings as key/value pairs

Return type: Dict

```
>>> lg.setings()
{'league_key': '398.l.10372', 'league_id': '10372', 'name': "Buck you're next!",
 'url': 'https://baseball.fantasysports.yahoo.com/b1/10372', 'logo_url': False,
 'draft_status': 'predraft', 'num_teams': 9, 'edit_key': '2020-02-03',
 'weekly_deadline': '1', 'league_update_timestamp': None, 'scoring_type': 'head',
 'league_type': 'private', 'renew': '388_27081', 'renewed': '',
 'iris_group_chat_id': 'ZP2QUJTUB5CPXMXWAVSYZRJI3Y', 'allow_add_to_dl_extra_pos': 0,
 'is_pro_league': '0', 'is_cash_league': '0', 'current_week': '1', 'start_week': '1',
 'start_date': '2020-03-26', 'end_week': '24', 'end_date': '2020-09-20',
 'game_code': 'mlb', 'season': '2020', 'draft_type': 'self', 'is_auction_draft': '0',
 'uses_playoff': '1', 'has_playoff_consolation_games': True, 'playoff_start_week': '22',
 'uses_playoff_reseeding': 1, 'uses_lock_eliminated_teams': 0, 'num_playoff_teams': '6',
 'num_playoff_consolation_teams': 6, 'has_multiweek_championship': 0,
 'uses_roster_import': '1', 'roster_import_deadline': '2020-03-25', 'waiver_type': 'R',
 'waiver_rule': 'all', 'uses_faab': '0', 'draft_pick_time': '60',
 'post_draft_players': 'FA', 'max_teams': '14', 'waiver_time': '2',
 'trade_end_date': '2020-08-09', 'trade_ratify_type': 'vote', 'trade_reject_time': '2',
 'player_pool': 'ALL', 'cant_cut_list': 'none', 'can_trade_draft_picks': '1'}
```

standings()

Return the standings of the league id

For each team in the standings it returns info about their place in the stan
record, number of games back).

Returns: An ordered list of the teams in the standings. First entry is the first place team.

Return type: List

```
>>> lg.standings()[0]
{'team_key': '388.l.27081.t.5',
 'name': 'Lumber Kings',
 'rank': 1,
 'playoff_seed': '5',
 'outcome_totals': {'wins': '121',
  'losses': '116',
  'ties': '15',
  'percentage': '.510'},
 'games_back': '19'}
```

stat_categories()

Return the stat categories for a league

Returns: Each dict entry will have the stat name along with the position type ('B' for batter or 'P' for pitcher).

Return type: list(dict)

```
>>> lg.stat_categories('370.l.56877')
[{'display_name': 'R', 'position_type': 'B'}, {'display_name': 'HR',
'position_type': 'B'}, {'display_name': 'W', 'position_type': 'P'}]
```

taken_players()

Return the players taken by teams.

Returns: Players taken by teams.

Return type: List(dict)

```
>>> tp = lg.taken_players()
>>> len(tp)
88
>>> tp[0]
{'player_id': 3341,
 'name': 'Marc-Andre Fleury',
 'position_type': 'G',
 'eligible_positions': ['G'],
 'percent_owned': 99,
 'status': ''}
```

team_key()

latest

Return the team_key for logged in users team in this league

Returns: The team key
Return type: str

```
>>> lg.team_key()
388.l.27081.t.5
```

`teams()`

Return details of all of the teams in the league.

Returns: A dictionary of teams, each entry is for a team. The team key is the key to the dict, where the values are all of the particulars for that team.
Return type: dict

```
>>> tms = lg.teams()
>>> tms.keys()
dict_keys(['388.l.27081.t.5', '388.l.27081.t.1', '388.l.27081.t.3',
           '388.l.27081.t.7', '388.l.27081.t.8', '388.l.27081.t.4',
           '388.l.27081.t.2', '388.l.27081.t.9', '388.l.27081.t.6',
           '388.l.27081.t.10'])
>>> tms['388.l.27081.t.5'].keys()
dict_keys(['team_key', 'team_id', 'name', 'is_owned_by_current_login',
           'url', 'team_logos', 'waiver_priority', 'number_of_moves',
           'number_of_trades', 'roster_adds', 'clinched_playoffs',
           'league_scoring_type', 'has_draft_grade',
           'auction_budget_total', 'auction_budget_spent', 'managers'])
```

`to_team(team_key)`

Construct a Team object from a League

Parameters: **team_key** (*str*) – Team key of the new Team object to construct
Returns: Fully constructed object
Return type: Team

`transactions(tran_types, count)`

Fetch transactions of a given type for the league.

Parameters:
- **tran_types** (*str*) – The comman seperated types of transactions retrieve. Valid values are: add,drop,commish,trade
- **count** (*str*) – The number of transactions to retrieve. Leave blank to return all transactions

Returns: Details about all the transactions from the league of a given type
Return type: list

```
>>> transactions('trade', '1')
[
    {'players': {...}, 'status': 'successful', 'timestamp': '1605168906',
'tradee_team_key': '399.l.710921.t.3', 'tradee_team_name': 'Red Skins Matter',
'trader_team_key': '399.l.710921.t.9', 'trader_team_name': 'Too Many Cooks',
'transaction_id': '319', 'transaction_key': '399.l.710921.tr.319', ...},
    {'players': {...}, 'status': 'successful', 'timestamp': '1604650727',
'tradee_team_key': '399.l.710921.t.5', 'tradee_team_name': 'Nuklear JuJu Charks',
'trader_team_key': '399.l.710921.t.2', 'trader_team_name': 'JuJus Golden Johnson',
'transaction_id': '295', 'transaction_key': '399.l.710921.tr.295', ...},
    {'players': {...}, 'status': 'successful', 'timestamp': '1601773444',
'tradee_team_key': '399.l.710921.t.4', 'tradee_team_name': 'DJ chark juju juju',
'trader_team_key': '399.l.710921.t.9', 'trader_team_name': 'Too Many Cooks',
'transaction_id': '133', 'transaction_key': '399.l.710921.tr.133', ...}
]
```

`waivers()`

Return the players currently on waivers.

Returns: Players on waiver.
Return type: List(dict)

```
>>> lg.waivers()
[{'player_id': 5986,
  'name': 'Darnell Nurse',
  'position_type': 'P',
  'eligible_positions': ['D'],
  'percent_owned': 65,
  'status': ''},
 {'player_id': 5999,
  'name': 'Anthony Mantha',
  'status': 'IR',
  'position_type': 'P',
  'eligible_positions': ['LW', 'RW', 'IR'],
  'percent_owned': 84},
 {'player_id': 7899,
  'name': 'Rasmus Dahlin',
  'status': 'IR',
  'position_type': 'P',
  'eligible_positions': ['D', 'IR'],
  'percent_owned': 87}]
```

`week_date_range(`*`week`*`)`

Return the start and end date of a given week.

Can only request the date range at most one week in the future. This restriction exists because Yahoo! only provides the week range when the matchups are known. And during the playoffs, the matchup is only known for the current week. A `RuntimeEr` returned if a request is for a week too far in the future.

latest

Returns: Start and end date of the given week

Return type: Tuple of two `datetime.date` objects

```
>>> lg.week_date_range(12)
(datetime.date(2019, 6, 17), datetime.date(2019, 6, 23))
```

The `Team` class

class **yahoo_fantasy_api.team.Team**(*sc*, *team_key*)

An abstraction for all of the team-level APIs in Yahoo! fantasy

Parameters:
- **sc** (`yahoo_oauth.OAuth2`) – Fully constructed session context
- **team_key** (*str*) – Team key identifier for the team we are constructing this object for.

accept_trade(*transaction_key*, *trade_note=''*)

Accept a proposed trade

Parameters: **transaction_key** (*str*) – Transction to accept. This key is taken from the output of the proposed_trades() API.

add_and_drop_players(*add_player_id*, *drop_player_id*)

Add one player and drop another in the same transaction

Parameters:
- **add_player_id** (*int*) – Yahoo! player ID of the player to add
- **drop_player_id** (*int*) – Yahoo! player ID of the player to drop

```
>>> tm.add_and_drop_players(6770, 6767)
```

add_player(*player_id*)

Add a single player by their player ID

Parameters: **player_id** (*int*) – Yahoo! player ID of the player to add

```
>>> tm.add_player(6767)
```

change_positions(*time_frame*, *modified_lineup*)

Change the starting position of a subset of players in your lineup

This raises a RuntimeError if any error occurs when communicating with Yahoo!

Parameters:
- **time_frame** (`datetime.date` | int) – The time frame that the new positions take affect. This should be the starting day of the week (MLB, NBA, or NHL) or the week number (NFL).
- **modified_lineup** (*list(dict)*) – List of players to modify. Each entry should have a dict with the following keys: player_id - player ID of the player to change; selected_position - new position of the player.

```
>>>
import datetime
cd = datetime.date(2019, 10, 7)
plyrs = [{'player_id': 5981, 'selected_position': 'BN'},
         {'player_id': 4558, 'selected_position': 'BN'}]
tm.change_positions(cd, plyrs)
```

`claim_and_drop_players(add_player_id, drop_player_id, faab=None)`

Submit a waiver claim for one player and drop another in the same transaction

Parameters:
- **add_player_id** (*int*) – Yahoo! player ID of the player to add
- **drop_player_id** (*int*) – Yahoo! player ID of the player to drop
- **faab** (*int*) – Number of faab dollars to bid on the claim

```
>>> tm.claim_and_drop_players(6770, 6767, faab=22)
```

`claim_player(player_id, faab=None)`

Submit a waiver claim for a single player by their player ID

Parameters:
- **player_id** (*int*) – Yahoo! player ID of the player to add
- **faab** (*int*) – Number of faab dollars to bid on the claim

```
>>> tm.add_player(6767, faab=7)
```

`details()`

Return the details of the team

Returns: Dictionary of the team details

```
>>> tm.details()
{'team_key': '388.l.27081.t.9', 'team_id': '9', 'name': 'Team Name',
 'url': 'http://baseball.fantasysports.yahoo.com/archive/mlb/2013/27081/9',
 'team_logos': [{'team_logo': {'size': 'large', 'url': 'http://l.yimg
```

drop_player(*player_id*)

Drop a single player by their player ID

Parameters: **player_id** (*int*) – Yahoo! player ID of the player to drop

```
>>> tm.drop_player(6770)
```

matchup(*week*)

Return the team of the matchup my team is playing in a given week

Parameters: **week** (*int*) – Week number to find the matchup for
Returns: Team key of the opponent

```
>>> tm.matchup(3)
388.l.27081.t.9
```

propose_trade(*tradee_team_key: str*, *players: list[dict[str, str]]*, *trade_note: str = ''*) → None

Propose a trade

Parameters:
- **tradee_team_key** (*str*) – Team key of the team receiving the trade
- **players** (*list(dict)*) – List of players to trade. Each entry should have a dict with the following keys: player_key - player key of the player to trade; source_team_key - team key of the team that currently owns the player; destination_team_key - team key of the team that will receive the player.
- **trade_note** (*str*) – Optional note to include with the trade

proposed_trades()

Retrieve information for any proposed trades that include your team

Returns: List of proposed trade transactions that you have offered and have been offered to you.

```
>>> tm.proposed_trades()
[{'transaction_key': '396.l.49770.pt.1',
  'status': 'proposed',
  'trader_team_key': '396.l.49770.t.4',
  'tradee_team_key': '396.l.49770.t.5',
  'trader_players': [{'player_id': '4472',
    'name': 'Drew Doughty',
    'position_type': 'P'}],
  'tradee_players': [{'player_id': '5689',
    'name': 'Jacob Trouba',
    'position_type': 'P'}]},
 {'transaction_key': '396.l.49770.pt.2',
  'status': 'proposed',
  'trader_team_key': '396.l.49770.t.4',
  'tradee_team_key': '396.l.49770.t.3',
  'trader_players': [{'player_id': '4002',
    'name': 'Claude Giroux',
    'position_type': 'P'},
 {'player_id': '3798', 'name': 'Tuukka Rask', 'position_type': 'G'}],
  'tradee_players': [{'player_id': '5981',
  'name': 'Aleksander Barkov',
  'position_type': 'P'},
 {'player_id': '4685', 'name': 'Brayden Schenn',
   'position_type': 'P'}]}],
 {'transaction_key': '396.l.49770.pt.3',
  'status': 'proposed',
  'trader_team_key': '396.l.49770.t.2',
  'tradee_team_key': '396.l.49770.t.4',
  'trader_players': [{'player_id': '5987',
    'name': 'Rasmus Ristolainen', 'position_type': 'P'}],
  'tradee_players': [{'player_id': '4064',
    'name': 'Kris Letang', 'position_type': 'P'}]}]
```

reject_trade(*transaction_key*, *trade_note=''*)

Reject a proposed trade

Parameters: **transaction_key** (*str*) – Transction to reject. This key is taken from the output of the proposed_trades() API.

roster(*week=None*, *day=None*)

Return the team roster for a given week or date

If neither week or day is specified it will return today's roster.

Parameters:
- **week** (*int*) – Week number of the roster to get
- **day** – Day to get the roster

Returns: Array of players. Each entry is a dict with the following fields: player_id, name, position_type, eligible_positions, selected_position

```
>>> tm.roster(3)
[{'player_id': 8578, 'name': 'John Doe', 'position_type': 'B',
 'eligible_positions': ['C','1B'], 'selected_position': 'C',
 'status': ''},
 {'player_id': 8967, 'name': 'Joe Baseball', 'position_type': 'B',
 'eligible_positions': ['SS'], 'selected_position': 'SS',
 'status': 'DTD'},
 {'player_id': 9961, 'name': 'Ed Reliever', 'position_type': 'P',
 'eligible_positions': ['RP'], 'status': ''}]
```