hockey_scraper Documentation

Release 1.35

Harry Shomer

Jan 19, 2020

Contents

CHAPTER 1

Contents

1.1 NHL Scraping Functions

1.1.1 Scraping

There are three ways to scrape games:

1. *Scrape by Season*:

Scrape games on a season by season level (Note: A given season is referred to by the first of the two years it spans. So you would refer to the 2016-2017 season as 2016).

```
import hockey_scraper

 # Scrapes the 2015 & 2016 season with shifts and stores the data in a Csv file (both␣
↪are equivalent!!!)
 hockey_scraper.scrape_seasons([2015, 2016], True)
 hockey_scraper.scrape_seasons([2015, 2016], True, data_format='Csv')

 # Scrapes the 2008 season without shifts and returns a dictionary with the DataFrame
 scraped_data = hockey_scraper.scrape_seasons([2008], False, data_format='Pandas')

 # Scrapes 2014 season without shifts including preseason games
 hockey_scraper.scrape_seasons([2014], False, preseason=True)
```

2. *Scrape by Game*:

Scrape a list of games provided. All game ID's can be found using this link (you need to play around with the dates in the url).

```
import hockey_scraper

# Scrapes the first game of 2014, 2015, and 2016 seasons with shifts and stores the␣
↪data in a Csv file
```

(continues on next page)

(continued from previous page)

```
hockey_scraper.scrape_games([2014020001, 2015020001, 2016020001], True)

# Scrapes the first game of 2007, 2008, and 2009 seasons with shifts and returns a a␣
↪dictionary with the DataFrames
scraped_data = hockey_scraper.scrape_games([2007020001, 2008020001, 2009020001], True,
↪ data_format='Pandas')
```

3. *Scrape by Date Range*:

Scrape all games between a specified date range. All dates must be written in a "yyyy-mm-dd" format.

```
import hockey_scraper

# Scrapes all games between date range without shifts and stores the data in a Csv␣
↪file (both are equivalent!!!)
hockey_scraper.scrape_date_range('2016-10-10', '2016-10-20', False)
hockey_scraper.scrape_date_range('2016-10-10', '2016-10-20', False, preseason=False)

# Scrapes all games between 2015-1-1 and 2015-1-15 without shifts and returns a a␣
↪dictionary with the DataFrame
scraped_data = hockey_scraper.scrape_date_range('2015-1-1', '2015-1-15', False, data_
↪format='Pandas')

# Scrapes all games from 2014-09-15 to 2014-11-01 with shifts including preseason␣
↪games
hockey_scraper.scrape_date_range('2014-09-15', '2014-11-01', True, preseason=True)
```

4. *Scrape Schedule*

Scrape the schedule between any given date range for past and future games. All dates must be written in a "yyyy-mm-dd" format. The default data_format is equal to 'Pandas'. This returns a DataFrame and not a dictionary like others. The columns returned are: ['game_id', 'date', 'venue', 'home_team', 'away_team', 'start_time', 'home_score', 'away_score', 'status']

```
import hockey_scraper

sched_df = hockey_scraper.scrape_schedule("2019-10-01", "2020-07-01")
```

Saving Files

The option also exists to save the scraped files in another directory. This would speed up re-scraping any games since we already have the docs needed for it. It would also be useful if you want to grab any extra information from them as some of them contain a lot more information. In order to do this you can use the 'docs_dir' keyword. One can specify the boolean value True to either create or refer (to an already created) directory in the home directory called hockey_scraper data. Or you can specify the directory with the string of the path. If this is a valid directory, when scraping each page it would first check if it was already scraped (therefore saving us the time of scraping it). If it hasn't been scraped yet, it will then grab it from the source and save it in the given directory.

Sometimes you may have already scraped and saved a file but you want to re-scrape it from the source and save it again (this may seem strange but the NHL frequently fixes mistakes so you may want to update what you have). This can be done by setting the keyword argument rescrape equal to True.

```
import hockey_scraper

# Path to the given directory
# Can also be True if you want the scraper to take care of it
USER_PATH = "/...."
```

(continues on next page)

(continued from previous page)

```
# Scrapes the 2015 & 2016 season with shifts and stores the data in a Csv file
# Also includes a path for an existing directory for the scraped files to be placed␣
↪in or retrieved from.
hockey_scraper.scrape_seasons([2015, 2016], True, docs_dir=USER_PATH)

# Once could chose to re-scrape previously saved files by making the keyword argument␣
↪rescrape=True
hockey_scraper.scrape_seasons([2015, 2016], True, docs_dir=USER_PATH, rescrape=True)
```

Additional Notes:

1. For all three functions you must specify if you want to also scrape shifts (TOI tables) with a boolean. The Play by Play is automatically scraped.

2. When scraping by date range or by season, preseason games aren't scraped unless otherwise specified. Also preseason games are scraped at your own risk. There is no guarantee it will work or that the files are even there!!!

3. For all three functions the scraped data is deposited into a Csv file unless it's specified to return the DataFrames

4. The Dictionary with the DataFrames (and scraping errors) returned by setting data_format='Pandas' is structured like:

```
{
   # Both of these are always included
   'pbp': pbp_df,
   'errors': scraping_errors,

   # This is only included when the argument 'if_scrape_shifts' is set equal to True
   'shifts': shifts_df
 }
```

5. When including a directory, it must be a valid directory. It will not create it for you. You'll get an error message but otherwise it will scrape as if no directory was provided.

Scrape Functions

Functions to scrape by season, games, and date range

hockey_scraper.nhl.scrape_functions.**print_errors**()

Print errors with scraping.

Also puts errors in the "error" string (would just print the string but it would look like shit on one line. I could store it as I "should" print it but that isn't how I want it).

Returns None

hockey_scraper.nhl.scrape_functions.**scrape_date_range**(*from_date*, *to_date*, *if_scrape_shifts*, *data_format='csv'*, *preseason=False*, *rescrape=False*, *docs_dir=False*)

Scrape games in given date range

Parameters

- **from_date** – date you want to scrape from
- **to_date** – date you want to scrape to

- **if_scrape_shifts** – Boolean indicating whether to also scrape shifts
- **data_format** – format you want data in - csv or pandas (csv is default)
- **preseason** – Boolean indicating whether to include preseason games (default if False) This is may or may not work!!! I don't give a shit.
- **rescrape** – If you want to rescrape pages already scraped. Only applies if you supply a docs dir. (def. = None)
- **docs_dir** – Directory that either contains previously scraped docs or one that you want them to be deposited in after scraping. When True it'll refer to (or if needed create) such a repository in the home directory. When provided a string it'll try to use that. Here it must be a valid directory otheriwse it won't work (I won't make it for you). When False the files won't be saved.

Returns Dictionary with DataFrames and errors or None

hockey_scraper.nhl.scrape_functions.**scrape_games**(*games*, *if_scrape_shifts*, *data_format='csv'*, *rescrape=False*, *docs_dir=False*)

Scrape a list of games

Parameters

- **games** – list of game_ids
- **if_scrape_shifts** – Boolean indicating whether to also scrape shifts
- **data_format** – format you want data in - csv or pandas (csv is default)
- **rescrape** – If you want to rescrape pages already scraped. Only applies if you supply a docs dir.
- **docs_dir** – Directory that either contains previously scraped docs or one that you want them to be deposited in after scraping. When True it'll refer to (or if needed create) such a repository in the home directory. When provided a string it'll try to use that. Here it must be a valid directory otheriwse it won't work (I won't make it for you). When False the files won't be saved.

Returns Dictionary with DataFrames and errors or None

hockey_scraper.nhl.scrape_functions.**scrape_list_of_games**(*games*, *if_scrape_shifts*)

Given a list of game_id's (and a date for each game) it scrapes them

Parameters

- **games** – list of [game_id, date]
- **if_scrape_shifts** – Boolean indicating whether to also scrape shifts

Returns DataFrame of pbp info, also shifts if specified

hockey_scraper.nhl.scrape_functions.**scrape_schedule**(*from_date*, *to_date*, *data_format='pandas'*, *rescrape=False*, *docs_dir=False*)

Scrape the games schedule in a given range.

Parameters

- **from_date** – date you want to scrape from
- **to_date** – date you want to scrape to
- **data_format** – format you want data in - csv or pandas (pandas is default)

- **rescrape** – If you want to rescrape pages already scraped. Only applies if you supply a docs dir. (def. = None)
- **docs_dir** – Directory that either contains previously scraped docs or one that you want them to be deposited in after scraping. When True it'll refer to (or if needed create) such a repository in the home directory. When provided a string it'll try to use that. Here it must be a valid directory otheriwse it won't work (I won't make it for you). When False the files won't be saved.

Returns DataFrame of None

hockey_scraper.nhl.scrape_functions.**scrape_seasons**(*seasons*, *if_scrape_shifts*, *data_format='csv'*, *preseason=False*, *rescrape=False*, *docs_dir=False*)

Given list of seasons it scrapes all the seasons

Parameters

- **seasons** – list of seasons
- **if_scrape_shifts** – Boolean indicating whether to also scrape shifts
- **data_format** – format you want data in - csv or pandas (csv is default)
- **preseason** – Boolean indicating whether to include preseason games (default if False) This is may or may not work!!! I don't give a shit.
- **rescrape** – If you want to rescrape pages already scraped. Only applies if you supply a docs dir.
- **docs_dir** – Directory that either contains previously scraped docs or one that you want them to be deposited in after scraping. When True it'll refer to (or if needed create) such a repository in the home directory. When provided a string it'll try to use that. Here it must be a valid directory otheriwse it won't work (I won't make it for you). When False the files won't be saved.

Returns Dictionary with DataFrames and errors or None

Game Scraper

This module contains code to scrape data for a single game

hockey_scraper.nhl.game_scraper.**check_goalie**(*row*)

Checks for bad goalie names (you can tell by them having no player id)

Parameters **row** – df row

Returns None

hockey_scraper.nhl.game_scraper.**combine_espn_html_pbp**(*html_df*, *espn_df*, *game_id*, *date*, *away_team*, *home_team*)

Merge the coordinate from the espn feed into the html DataFrame

Can't join here on event_id because the plays are often out of order and pre-2009 are often missing events.

Parameters

- **html_df** – DataFrame with info from html pbp
- **espn_df** – DataFrame with info from espn pbp
- **game_id** – json game id

- **date** – ex: 2016-10-24
- **away_team** – away team
- **home_team** – home team

Returns merged DataFrame

hockey_scraper.nhl.game_scraper.**combine_html_json_pbp**(*json_df*, *html_df*, *game_id*, *date*)

Join both data sources. First try merging on event id (which is the DataFrame index) if both DataFrames have the same number of rows. If they don't have the same number of rows, merge on: Period', Event, Seconds_Elapsed, p1_ID.

Parameters

- **json_df** – json pbp DataFrame
- **html_df** – html pbp DataFrame
- **game_id** – id of game
- **date** – date of game

Returns finished pbp

hockey_scraper.nhl.game_scraper.**combine_players_lists**(*json_players*, *roster_players*, *game_id*)

Combine the json list of players (which contains id's) with the list in the roster html

Parameters

- **json_players** – dict of all players with id's
- **roster_players** – dict with home and and away keys for players
- **game_id** – id of game

Returns dict containing home and away keys -> which contains list of info on each player

hockey_scraper.nhl.game_scraper.**get_players_json**(*players_json*)

Return dict of players for that game

Parameters **players_json** – players section of json

Returns dict of players->keys are the name (in uppercase)

hockey_scraper.nhl.game_scraper.**get_sebastian_aho**(*player*)

This checks which Sebastian Aho it is based on the position. I have the player id's hardcoded here.

This function is needed because "get_players_json" doesn't control for when there are two Sebastian Aho's (it just writes over the first one).

Parameters **player** – player info

Returns Player ID for specific Aho

hockey_scraper.nhl.game_scraper.**get_teams_and_players**(*game_json*, *roster*, *game_id*)

Get list of players and teams for game

Parameters

- **game_json** – json pbp for game
- **roster** – players from roster html
- **game_id** – id for game

Returns dict for both - players and teams

hockey_scraper.nhl.game_scraper.**scrape_game**(*game_id*, *date*, *if_scrape_shifts*)

This scrapes the info for the game. The pbp is automatically scraped, and the whether or not to scrape the shifts is left up to the user.

Parameters

- **game_id** – game to scrap
- **date** – ex: 2016-10-24
- **if_scrape_shifts** – Boolean indicating whether to also scrape shifts

Returns DataFrame of pbp info (optional) DataFrame with shift info otherwise just None

hockey_scraper.nhl.game_scraper.**scrape_pbp**(*game_id*, *date*, *roster*, *game_json*, *players*, *teams*, *espn_id=None*, *html_df=None*)

Automatically scrapes the json and html, if the json is empty the html picks up some of the slack and the espn xml is also scraped for coordinates.

Parameters

- **game_id** – json game id
- **date** – date of game
- **roster** – list of players in pre game roster
- **game_json** – json pbp for game
- **players** – dict of players
- **teams** – dict of teams
- **espn_id** – Game Id for the espn game. Only provided when live scraping
- **html_df** – Can provide DataFrame for html. Only done for live-scraping

Returns DataFrame with info or None if it fails

hockey_scraper.nhl.game_scraper.**scrape_pbp_live**(*game_id*, *date*, *roster*, *game_json*, *players*, *teams*, *espn_id=None*)

Scrape the live pbp

Parameters

- **game_id** – json game id
- **date** – date of game
- **roster** – list of players in pre game roster
- **game_json** – json pbp for game
- **players** – dict of players
- **teams** – dict of teams
- **espn_id** – Game Id for the espn game. Only provided when live scraping

Returns Tuple - pbp & status

hockey_scraper.nhl.game_scraper.**scrape_shifts**(*game_id*, *players*, *date*)

Scrape the Shift charts (or TOI tables)

Parameters

- **game_id** – json game id
- **players** – dict of players with numbers and id's

- **date** – date of game

Returns DataFrame with info or None if it fails

Html PBP

This module contains functions to scrape the Html Play by Play for any given game

hockey_scraper.nhl.pbp.html_pbp.**add_event_players**(*event_dict*, *event*, *players*, *home_team*)

Add players involved in the event to event_dict

Parameters

- **event_dict** – dict of parsed event stuff
- **event** – fixed up html
- **players** – dict of players and id's
- **home_team** – home team

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_event_team**(*event_dict*, *event*)

Add event team for event

Parameters

- **event_dict** – dict of event info
- **event** – list with parsed event info

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_home_zone**(*event_dict*, *home_team*)

Determines the zone relative to the home team and add it to event.

Keep in mind that the 'ev_zone' recorded is the zone relative to the event team. And for blocks the NHL counts the ev_team as the blocking team (I like counting the shooting team for blocks). Therefore, when it's the home team the zone only gets flipped when it's a block. For away teams it's the opposite. Think about it...

Parameters

- **event_dict** – dict of event info
- **home_team** – home team

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_period**(*event_dict*, *event*)

Add period for event

Parameters

- **event_dict** – dict of event info
- **event** – list with parsed event info

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_score**(*event_dict*, *event*, *current_score*, *home_team*)

Change if someone scored... also change current score

Parameters

- **event_dict** – dict of parsed event stuff

- **event** – event info from pbp
- **current_score** – current score in game
- **home_team** – home team for game

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_strength**(*event_dict*, *home_players*, *away_players*)
Get strength for event -> It's home then away

Parameters

- **event_dict** – dict of event info
- **home_players** – list of players for home team
- **away_players** – list of players for away team

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_time**(*event_dict*, *event*)
Fill in time and seconds elapsed

Parameters

- **event_dict** – dict of parsed event stuff
- **event** – event info from pbp

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_type**(*event_dict*, *event*, *players*, *home_team*)
Add "type" for event -> either a penalty or a shot type

Parameters

- **event_dict** – dict of event info
- **event** – list with parsed event info
- **players** – dict of home and away players in game
- **home_team** – home team for game

Returns None

hockey_scraper.nhl.pbp.html_pbp.**add_zone**(*event_dict*, *play_description*)
Determine which zone the play occurred in (unless one isn't listed) and add it to dict

Parameters

- **event_dict** – dict of event info
- **play_description** – the zone would be included here

Returns Off, Def, Neu, or NA

hockey_scraper.nhl.pbp.html_pbp.**clean_html_pbp**(*html*)
Get rid of html and format the data

Parameters **html** – the requested url

Returns a list with all the info

hockey_scraper.nhl.pbp.html_pbp.**cur_game_status**(*doc*)
Return the game status

Parameters **doc** – Html text

Returns String -> one of [‘Final’, ‘Intermission’, ‘Progress’]

hockey_scraper.nhl.pbp.html_pbp.**get_pbp**(*game_id*)
Given a game_id it returns the raw html Ex: http://www.nhl.com/scores/htmlreports/20162017/PL020475.HTM

Parameters **game_id** – the game

Returns raw html of game

hockey_scraper.nhl.pbp.html_pbp.**get_penalty**(*play_description*, *players*, *home_team*)
Get the penalty info

Parameters

- **play_description** – description of play field
- **players** – all players with info
- **home_team** – home team for game

Returns penalty info

hockey_scraper.nhl.pbp.html_pbp.**get_player_name**(*number*, *players*, *team*, *home_team*)
This function is used for the description field in the html. Given a last name and a number it return the player’s full name and id.

Parameters

- **number** – player’s number
- **players** – all players with info
- **team** – team of player
- **home_team** – home team

Returns dict with full and and id

hockey_scraper.nhl.pbp.html_pbp.**get_soup**(*game_html*)
Uses Beautiful soup to parses the html document. Some parsers work for some pages but don’t work for others….I’m not sure why so I just try them all here in order

Parameters **game_html** – html doc

Returns “soupified” html

hockey_scraper.nhl.pbp.html_pbp.**if_valid_event**(*event*)
Checks if it’s a valid event (‘#’ is meaningless and I don’t like those other one’s) to parse

Don’t remember what ‘GOFF’ is but ‘EGT’ is for emergency goaltender. The reason I get rid of it is because it’s not in the json and there’s another ‘EGPID’ that can be found in both (not sure why ‘EGT’ exists then).

Events ‘PGSTR’, ‘PGEND’, and ‘ANTHEM’ have been included at the start of each game for the 2017 season…I have no idea why.

Parameters **event** – list of stuff in pbp

Returns boolean

hockey_scraper.nhl.pbp.html_pbp.**parse_block**(*description*, *players*, *home_team*)
Parse the description field for a BLOCK

MTL #76 SUBBAN BLOCKED BY TOR #2 SCHENN, Wrist, Def. Zone

Parameters

- **description** – Play Description

- **players** – players in game
- **home_team** – Home Team for game

Returns Dict with info

hockey_scraper.nhl.pbp.html_pbp.**parse_event**(*event*, *players*, *home_team*, *current_score*)
Receives an event and parses it

Parameters

- **event** – event type
- **players** – players in game
- **home_team** – home team
- **current_score** – current score for both teams

Returns dict with info

hockey_scraper.nhl.pbp.html_pbp.**parse_fac**(*description*, *players*, *ev_team*, *home_team*)
Parse the description field for a face-off MTL won Neu. Zone - MTL #11 GOMEZ vs TOR #37 BRENT

Parameters

- **description** – Play Description
- **players** – players in game
- **ev_team** – Event Team
- **home_team** – Home Team for game

Returns Dict with info

hockey_scraper.nhl.pbp.html_pbp.**parse_goal**(*description*, *players*, *ev_team*, *home_team*)
Parse the description field for a GOAL

TOR #81 KESSEL(1), Wrist, Off. Zone, 14 ft. Assists: #42 BOZAK(1); #8 KOMISAREK(1)

Parameters

- **description** – Play Description
- **players** – players in game
- **ev_team** – Event Team
- **home_team** – Home Team for game

Returns Dict with info

hockey_scraper.nhl.pbp.html_pbp.**parse_hit**(*description*, *players*, *home_team*)
Parse the description field for a HIT

MTL #20 O'BYRNE HIT TOR #18 BROWN, Def. Zone

Parameters

- **description** – Play Description
- **players** – players in game
- **home_team** – Home Team for game

Returns Dict with info

hockey_scraper.nhl.pbp.html_pbp.**parse_html**(*html*, *players*, *teams*)
Parse html game pbp

Parameters

- **html** – raw html
- **players** – players in the game (from json pbp)
- **teams** – dict with home and away teams

Returns DataFrame with info

hockey_scraper.nhl.pbp.html_pbp.**parse_penalty**(*description*, *players*, *home_team*)

Parse the description field for a Penalty

MTL #81 ELLER Hooking(2 min), Def. Zone Drawn By: TOR #11 SJOSTROM

Parameters

- **description** – Play Description
- **players** – players in game
- **home_team** – Home Team for game

Returns Dict with info

hockey_scraper.nhl.pbp.html_pbp.**parse_shot_miss_take_give**(*description*, *players*, *ev_team*, *home_team*)

Parse the description field for a: SHOT, MISS, TAKE, GIVE

MTL ONGOAL - #81 ELLER, Wrist, Off. Zone, 11 ft. ANA #23 BEAUCHEMIN, Slap, Wide of Net, Off. Zone, 42 ft. TOR GIVEAWAY - #35 GIGUERE, Def. Zone TOR TAKEAWAY - #9 ARMSTRONG, Off. Zone

Parameters

- **description** – Play Description
- **players** – players in game
- **ev_team** – Event Team
- **home_team** – Home Team for game

Returns Dict with info

hockey_scraper.nhl.pbp.html_pbp.**populate_players**(*event_dict*, *players*, *away_players*, *home_players*)

Populate away and home player info (and num skaters on each side)

Parameters

- **event_dict** – dict with event info
- **players** – all players in game and info
- **away_players** – players for away team
- **home_players** – players for home team

Returns None

hockey_scraper.nhl.pbp.html_pbp.**return_name_html**(*info*)

In the PBP html the name is in a format like: 'Center - MIKE RICHARDS' Some also have a hyphen in their last name so can't just split by '-'

Parameters **info** – position and name

Returns name

hockey_scraper.nhl.pbp.html_pbp.**scrape_game**(*game_id*, *players*, *teams*)
Scrape the data for the game when not live

Parameters

- **game_id** – game to scrape
- **players** – dict with player info
- **teams** – dict with home and away teams

Returns DataFrame of game info or None if it fails

hockey_scraper.nhl.pbp.html_pbp.**scrape_game_live**(*game_id*, *players*, *teams*)
Scrape the data for the game when it's live

Parameters

- **game_id** – game to scrape
- **players** – dict with player info
- **teams** – dict with home and away teams

Returns Tuple - get_pbp(), cur_game_status()

hockey_scraper.nhl.pbp.html_pbp.**scrape_pbp**(*game_html*, *game_id*, *players*, *teams*)
Scrape the data for the pbp

Parameters

- **game_html** – Html doc for the game
- **game_id** – game to scrape
- **players** – dict with player info
- **teams** – dict with home and away teams

Returns DataFrame of game info or None if it fails

hockey_scraper.nhl.pbp.html_pbp.**shot_type**(*play_description*)
Determine which zone the play occurred in (unless one isn't listed)

Parameters **play_description** – the type would be in here

Returns the type if it's there (otherwise just NA)

hockey_scraper.nhl.pbp.html_pbp.**strip_html_pbp**(*td*)
Strip html tags and such

Parameters **td** – pbp

Returns list of plays (which contain a list of info) stripped of html

Json PBP

This module contains functions to scrape the Json Play by Play for any given game

hockey_scraper.nhl.pbp.json_pbp.**change_event_name**(*event*)
Change event names from json style to html (ex: BLOCKED_SHOT to BLOCK).

Parameters **event** – event type

Returns fixed event type

hockey_scraper.nhl.pbp.json_pbp.**get_pbp**(*game_id*)
Given a game_id it returns the raw json Ex: http://statsapi.web.nhl.com/api/v1/game/2016020475/feed/live

Parameters **game_id** – string - the game

Returns raw json of game or None if couldn't get game

hockey_scraper.nhl.pbp.json_pbp.**get_teams**(*pbp_json*)
Get teams

Parameters **pbp_json** – raw play by play json

Returns dict with home and away

hockey_scraper.nhl.pbp.json_pbp.**parse_event**(*event*)
Parses a single event when the info is in a json format

Parameters **event** – json of event

Returns dictionary with the info

hockey_scraper.nhl.pbp.json_pbp.**parse_json**(*game_json*, *game_id*)
Scrape the json for a game

Parameters

- **game_json** – raw json
- **game_id** – game id for game

Returns Either a DataFrame with info for the game

hockey_scraper.nhl.pbp.json_pbp.**scrape_game**(*game_id*)
Used for debugging. HTML depends on json so can't follow this structure

Parameters **game_id** – game to scrape

Returns DataFrame of game info

Espn PBP

This module contains code to scrape coordinates for games off of espn for any given game

hockey_scraper.nhl.pbp.espn_pbp.**event_type**(*play_description*)
Returns the event type (ex: a SHOT or a GOAL. . . etc) given the event description

Parameters **play_description** – description of play

Returns event

hockey_scraper.nhl.pbp.espn_pbp.**get_espn_date**(*date*)
Get the page that contains all the games for that day

Parameters **date** – YYYY-MM-DD

Returns response

hockey_scraper.nhl.pbp.espn_pbp.**get_espn_game**(*date*, *home_team*, *away_team*, *game_id=None*)
Gets the ESPN pbp feed Ex: http://www.espn.com/nhl/gamecast/data/masterFeed?lang=en&isAll=true&gameId=400885300

Parameters

- **date** – date of the game

- **home_team** – home team
- **away_team** – away team
- **game_id** – Game id of we already have it - for live scraping. None if not there

Returns raw xml

hockey_scraper.nhl.pbp.espn_pbp.**get_espn_game_id**(*date*, *home_team*, *away_team*)
Scrapes the day's schedule and gets the id for the given game Ex: http://www.espn.com/nhl/scoreboard/_/date/20161024

Parameters

- **date** – format-> YearMonthDay-> 20161024
- **home_team** – home team
- **away_team** – away team

Returns 9 digit game id as a string

hockey_scraper.nhl.pbp.espn_pbp.**get_game_ids**(*response*)
Get game_ids for date from doc

Parameters **response** – doc

Returns list of game_ids

hockey_scraper.nhl.pbp.espn_pbp.**get_teams**(*response*)
Extract Teams for date from doc

ul-> class = ScoreCell__Competitors

div -> class = ScoreCell__TeamName ScoreCell__TeamName–shortDisplayName truncate db

Parameters **response** – doc

Returns list of teams

hockey_scraper.nhl.pbp.espn_pbp.**parse_espn**(*espn_xml*)
Parse feed

Parameters **espn_xml** – raw xml of feed

Returns DataFrame with info

hockey_scraper.nhl.pbp.espn_pbp.**parse_event**(*event*)
Parse each event. In the string each field is separated by a '~'. Relevant for here: The first two are the x and y coordinates. And the 4th and 5th are the time elapsed and period.

Parameters **event** – string with info

Returns return dict with relevant info

hockey_scraper.nhl.pbp.espn_pbp.**scrape_game**(*date*, *home_team*, *away_team*, *game_id=None*)
Scrape the game

Parameters

- **date** – ex: 2016-20-24
- **home_team** – tricode
- **away_team** – tricode
- **game_id** – Only provided for live games.

Returns DataFrame with info

Json Shifts

This module contains functions to scrape the Json toi/shifts for any given game

hockey_scraper.nhl.shifts.json_shifts.**fix_team_tricode**(*tricode*)
Some of the tricodes are different than how I want them

Parameters **tricode** – 3 letter team name - ex: NYR

Returns fixed tricode

hockey_scraper.nhl.shifts.json_shifts.**get_shifts**(*game_id*)
Given a game_id it returns the raw json Ex: https://api.nhle.com/stats/rest/en/shiftcharts?cayenneExp=gameId=2019020001

Parameters **game_id** – the game

Returns json or None

hockey_scraper.nhl.shifts.json_shifts.**parse_json**(*shift_json*, *game_id*)
Parse the json

Parameters

- **shift_json** – raw json
- **game_id** – if of game

Returns DataFrame with info

hockey_scraper.nhl.shifts.json_shifts.**parse_shift**(*shift*)
Parse shift for json

Parameters **shift** – json for shift

Returns dict with shift info

hockey_scraper.nhl.shifts.json_shifts.**scrape_game**(*game_id*)
Scrape the game.

Parameters **game_id** – game

Returns DataFrame with info for the game

Html Shifts

This module contains functions to scrape the Html Toi Tables (or shifts) for any given game

hockey_scraper.nhl.shifts.html_shifts.**analyze_shifts**(*shift*, *name*, *team*, *home_team*, *player_ids*)
Analyze shifts for each player when using. Prior to this each player (in a dictionary) has a list with each entry being a shift.

Parameters

- **shift** – info on shift
- **name** – player name
- **team** – given team
- **home_team** – home team for given game

- **player_ids** – dict with info on players

Returns dict with info for shift

hockey_scraper.nhl.shifts.html_shifts.**get_shifts**(*game_id*)
Given a game_id it returns a the shifts for both teams Ex: http://www.nhl.com/scores/htmlreports/20162017/TV020971.HTM

Parameters **game_id** – the game

Returns Shifts or None

hockey_scraper.nhl.shifts.html_shifts.**get_soup**(*shifts_html*)
Uses Beautiful soup to parses the html document. Some parsers work for some pages but don't work for others....I'm not sure why so I just try them all here in order

Parameters **shifts_html** – html doc

Returns "soupified" html and player_shifts portion of html (it's a bunch of td tags)

hockey_scraper.nhl.shifts.html_shifts.**get_teams**(*soup*)
Return the team for the TOI tables and the home team

Parameters **soup** – souped up html

Returns list with team and home team

hockey_scraper.nhl.shifts.html_shifts.**parse_html**(*html*, *player_ids*, *game_id*)
Parse the html

Note: Don't fuck with this!!! I'm not exactly sure how or why but it works.

Parameters

- **html** – cleaned up html
- **player_ids** – dict of home and away players
- **game_id** – id for game

Returns DataFrame with info

hockey_scraper.nhl.shifts.html_shifts.**scrape_game**(*game_id*, *players*)
Scrape the game.

Parameters

- **game_id** – id for game
- **players** – list of players

Returns DataFrame with info for the game

Schedule

This module contains functions to scrape the json schedule for any games or date range

hockey_scraper.nhl.json_schedule.**chunk_schedule_calls**(*from_date*, *to_date*)
The schedule endpoint sucks when handling a big date range. So instead I call in increments of n days.

Parameters

- **date_from** – scrape from this date
- **date_to** – scrape until this date

Returns raw json of schedule of date range

hockey_scraper.nhl.json_schedule.**get_dates**(*games*)
Given a list game_ids it returns the dates for each game.

We go from the beginning of the earliest season in the sample to the end of the most recent

Parameters **games** – list with game_id's ex: 2016020001

Returns list with game_id and corresponding date for all games

hockey_scraper.nhl.json_schedule.**get_schedule**(*date_from*, *date_to*)
Scrapes games in date range Ex: https://statsapi.web.nhl.com/api/v1/schedule?startDate=2010-10-03&endDate=2011-06-20

Parameters

- **date_from** – scrape from this date
- **date_to** – scrape until this date

Returns raw json of schedule of date range

hockey_scraper.nhl.json_schedule.**scrape_schedule**(*date_from*, *date_to*, *preseason=False*, *not_over=False*)
Calls getSchedule and scrapes the raw schedule Json

Parameters

- **date_from** – scrape from this date
- **date_to** – scrape until this date
- **preseason** – Boolean indicating whether include preseason games (default if False)
- **not_over** – Boolean indicating whether we scrape games not finished. Means we relax the requirement of checking if the game is over.

Returns list with all the game id's

Playing Roster

This module contains functions to scrape the Html game roster for any given game

hockey_scraper.nhl.playing_roster.**fix_name**(*player*)
Get rid of (A) or (C) when a player has it attached to their name

Parameters **player** – list of player info -> [number, position, name]

Returns fixed list

hockey_scraper.nhl.playing_roster.**get_coaches**(*soup*)
scrape head coaches

Parameters **soup** – html

Returns dict of coaches for game

hockey_scraper.nhl.playing_roster.**get_content**(*roster*)
Uses Beautiful soup to parses the html document. Some parsers work for some pages but don't work for others....I'm not sure why so I just try them all here in order

Parameters **roster** – doc

Returns players and coaches

hockey_scraper.nhl.playing_roster.**get_players**(*soup*)
scrape roster for players

Parameters **soup** – html

Returns dict for home and away players

hockey_scraper.nhl.playing_roster.**get_roster**(*game_id*)
Given a game_id it returns the raw html Ex: http://www.nhl.com/scores/htmlreports/20162017/RO020475.HTM

Parameters **game_id** – the game

Returns raw html of game

hockey_scraper.nhl.playing_roster.**scrape_roster**(*game_id*)
For a given game scrapes the roster

Parameters **game_id** – id for game

Returns dict of players (home and away) an dict for both head coaches

Save Pages

Saves the scraped docs so you don't have to re-scrape them every time you want to parse the docs.

**** Don't mess with this unless you know what you're doing ****

hockey_scraper.utils.save_pages.**check_file_exists**(*file_info*)
Checks if the file exists. Also check if structure for holding scraped file exists to. If not, it creates it.

Parameters **file_info** – Dictionary containing the info on the file. Includes the name, season, file type, and the dir we want to deposit any data in.

Returns Boolean - True if it exists

hockey_scraper.utils.save_pages.**create_file_path**(*file_info*)
Creates the file path for a given file

Parameters **file_info** – Dictionary containing the info on the file. Includes the name, season, file type, and the dir we want to deposit any data in.

Returns path

hockey_scraper.utils.save_pages.**create_season_dirs**(*file_info*)
Creates the infrastructure to hold all the scraped docs for a season

Parameters **file_info** – Dictionary containing the info on the file. Includes the name, season, file type, and the dir we want to deposit any data in.

Returns None

hockey_scraper.utils.save_pages.**get_page**(*file_info*)
Get the file so we don't need to re-scrape

Parameters **file_info** – Dictionary containing the info on the file. Includes the name, season, file type, and the dir we want to deposit any data in.

Returns Response or None

hockey_scraper.utils.save_pages.**save_page**(*page*, *file_info*)
Save the page we just scraped.

Note: It'll only get saved if the directory already exists!!!!!!. I'm not dealing with any fuck ups. That would involve checking if it's even a valid path and creating it. Make sure you get it right.

Parameters

- **page** – File scraped

- **file_info** – Dictionary containing the info on the file. Includes the name, season, file type, and the dir we want to deposit any data in.

Returns None

Shared Functions

This file is a bunch of the shared functions or just general stuff used by the different scrapers in the package.

hockey_scraper.utils.shared.**add_dir**(*user_dir*)
Add directory to store scraped docs if valid. Or create in the home dir

NOTE: After this functions docs_dir is either None or a valid directory

Parameters **user_dir** – If bool=True create in the home dire or if user provided directory on their machine

Returns None

hockey_scraper.utils.shared.**check_data_format**(*data_format*)
Checks if data_format specified (if it is at all) is either None, 'Csv', or 'pandas'. It exits program with error message if input isn't good.

Parameters **data_format** – data_format provided

Returns Boolean - True if good

hockey_scraper.utils.shared.**check_valid_dates**(*from_date*, *to_date*)
Check if it's a valid date range

Parameters

- **from_date** – date should scrape from
- **to_date** – date should scrape to

Returns None

hockey_scraper.utils.shared.**convert_to_seconds**(*minutes*)
Return minutes elapsed in time format to seconds elapsed

Parameters **minutes** – time elapsed

Returns time elapsed in seconds

hockey_scraper.utils.shared.**fix_name**(*name*)
Check if a name falls under those that need fixing. If it does... fix it.

Parameters **name** – name in pbp

Returns Either the given parameter or the fixed name

hockey_scraper.utils.shared.**get_file**(*file_info*)
Get the specified file.

If a docs_dir is provided we check if it exists. If it does we see if it contains that page (and saves if it doesn't). If the docs_dir doesn't exist we just scrape from the source and not save.

Parameters **file_info** – Dictionary containing the info for the file. Contains the url, name, type, and season

Returns page

hockey_scraper.utils.shared.**get_season**(*date*)
Get Season based on from_date

Parameters **date** – date

Returns season -> ex: 2016 for 2016-2017 season

hockey_scraper.utils.shared.**get_team**(*team*)
Get the fucking team

hockey_scraper.utils.shared.**if_rescrape**(*user_rescrape*)
If you want to re_scrape. If someone is a dumbass and feeds it a non-boolean it terminates the program

Note: Only matters when you have a directory specified

Parameters **user_rescrape** – Boolean

Returns None

hockey_scraper.utils.shared.**print_warning**(*msg*)
Print the Warning

hockey_scraper.utils.shared.**scrape_page**(*url*)
Scrape a given url

Parameters **url** – url for page

Returns response object

hockey_scraper.utils.shared.**to_csv**(*base_file_name*, *df*, *league*, *file_type*)
Write DataFrame to csv file

Parameters

- **base_file_name** – name of file
- **df** – DataFrame
- **league** – nhl or nwhl
- **file_type** – type of file despoiting

Returns None

1.2 Live Scraping

1.2.1 Standard Usage

To get all the info for every game on a specific day we create a ScrapeLiveGames object.

```
import hockey_scraper as hs

todays_games = hs.ScrapeLiveGames("2018-11-15", if_scrape_shifts=True, pause=15)
```

Once created this object will contain an attribute called 'live_games' that holds a list of LiveGame objects for that day. LiveGame objects hold all the pertinent game information for each game. This includes the most recent pbp and shift data for that game. Here are all the attributes for the LiveGame class:

```
class LiveGame:
 """
 This is a class holds all the information for a given game

 :param int game_id: The NHL game id (ex: 2018020001)
 :param datetime start_time: The UTC time of when the game begins
```

(continues on next page)

(continued from previous page)

```
 :param str home_team: Tricode for the home team (ex: NYR)
 :param str away_team: Tricode for the home team (ex: MTL)
 :param int espn_id: The ESPN game id for their feed
 :param str date: Date of the game (ex: 2018-10-30)
 :param bool if_scrape_shifts: Whether or not you want to scrape shifts
 :param str api_game_status: Current Status of the game - ["Final", "Live",
↪"Intermission]
 :param str html_game_status: Current Status of the game - ["Final", "Live",
↪"Intermission"]
 :param int intermission_time_remaining: Time remaining in the intermission. 0 if not␣
↪in intermission
 :param dict players: Player info for both teams
 :param dict head_coaches: Head coaches for both teams
 :param DataFrame _pbp_df: Holds most recent pbp data
 :param DataFrame _shifts_df: Holds most recent shift data
 :param DataFrame _prev_pbp_df: Holds the previous pbp data (for just in case)
 :param DataFrame _prev_shifts_df: Holds the previous shift data (for just in case)
 """
```

Here's a simple example of scraping the games continuously for a single date. This will run until every game is finished:

```
import hockey_scraper as hs


def to_csv(game):
    """
    Store each game DataFrame in a file

    :param game: LiveGame object

    :return: None
    """

    # If the game:
    # 1. Started - We recorded at least one event
    # 2. Not in Intermission
    # 3. Not Over
    if game.is_ongoing():
        # Print the description of the last event
        print(game.game_id, "->", game.pbp_df.iloc[-1]['Description'])

        # Store in CSV files
        game.pbp_df.to_csv(f"../hockey_scraper_data/{game.game_id}_pbp.csv", sep=',')
        game.shifts_df.to_csv(f"../hockey_scraper_data/{game.game_id}_shifts.csv",␣
↪sep=',')

if __name__ == "__main__":
    # B4 we start set the directory to store the files
    hs.live_scrape.set_docs_dir("../hockey_scraper_data")

    # Scrape the info for all the games on 2018-11-15
    games = hs.ScrapeLiveGames("2018-11-15", if_scrape_shifts=True, pause=20)

    # While all the games aren't finished
    while not games.finished():
```

(continues on next page)

(continued from previous page)

```
        # Update for all the games currently being played
        games.update_live_games(sleep_next=True)

        # Go through every LiveGame object and apply some function
        # You can of course do whatever you want here.
        for game in games.live_games:
            to_csv(game)
```

In the above example, we set a directory to store the most recent version of every scraped file. We then grab the initial game info for each game for that day. We decide we want to include shifts and to pause 15 seconds after updating all the games. We then enter a loop that will be terminated once every game is finished. Once in the loop we first scrape the new info for every game and then pause for the specified time (default is 15).

Once we process the new data we then, presumably, want to do something with it. Here, I decided to merely print the last event in the game and store the newer data in files. We do this by iterating through each LiveGame object in the 'live_games' attribute and calling the function 'to_csv'. In 'to_csv', before doing anything we check if the game is 'ongoing'. This checks whether the game is either over or in intermission. If it is there isn't a whole lot to update. If it's neither we print the last event and then store the data for both the pbp & shifts.

Another option we have is for the program to sleep until the first game starts. Unless you want to start this yourself everyday, you'll probably be scheduling it to start at some time every day. This means from when you start the program to when the first game starts may be a significant amount of time (fwiw, it will just loop and not scrape anything). But you can set it to sleep until the first game is scheduled to start. This can be done by setting the keyword 'sleep_next' to True. This check to see if the only games left are scheduled games yet to start. If so it sleeps until the next earliest game starts.

```
# Causes the program to sleep until the first game starts
games.update_live_games(sleep_next=True)
```

You can also specify which games you want to scrape for that day (maybe you only care about one game), by setting the keyword 'game_ids' equal to a list of NHL Game ID's of the games you want when instantiating a ScrapeLiveGame object. You can of course to choose to filter it however you want as the list of LiveGame objects is a attribute of the object. Either way I strongly suggest creating a ScrapeLiveGames object and then either extracting the game you want or filtering it rather than instantiating a LiveGame object (you will be on the hook for a lot of information)

```
# Only want those those two games.
games = hs.ScrapeLiveGames("2018-11-15", if_scrape_shifts=True, pause=15, game_
↪ids=[2018020280, 2018020281])
```

1.2.2 Further Usage

If you would like more control over what you are doing then you should be dealing directly with LiveGame objects. As mentioned previously, still use ScrapeLiveGames to get the game info but you can then just extract the list of games and do as you please.

Using the previous example here we are scraping each game individually:

```
# Scrape the info for all the games on 2018-11-15
games = hs.ScrapeLiveGames("2018-11-15", if_scrape_shifts=True, pause=15)

while not games.finished():
    # Go through every LiveGame object
    for game in games.live_games:
        # Scrape each game individually
```

(continues on next page)

(continued from previous page)

```
        game.scrape()

        # Apply some function to every game
        to_csv(game)

    # Pause after each scraping chunk
    time.sleep(15)
```

If you don't trust when I choose to not scrape (when the game is over or in intermission), you can make the keyword 'force' equal to True. This will re-scrape it as long as the game already started.

```
game.scrape(force=True)
```

This will override everything and will attempt to scrape the game no matter what. This means you are have to be on top of when to stop scraping the game. You are also on the hook for any potential errors.

You may also want to handle things like the start time of games yourself. As mentioned using ScrapeLiveGames we can set 'sleep_next' equal to True to sleep until the next game starts if no game is going on. You can also use the keyword 'start_time' for a LiveGame object that will give you a datetime object with the scheduled starting time for a given game in UTC time. Lastly, you can also use the function 'time_until_game()' that will return how many seconds until the game starts.

```
>>> games = hs.ScrapeLiveGames("2018-11-09", if_scrape_shifts=True, pause=15)
>>> games.live_games[0].start_time
datetime.datetime(2018, 11, 10, 0, 0)
>>> games.live_games[0].time_until_game()
64599
```

You can use this how you please. For example, you may want to create a separate thread for each game and have it sleep until the game starts. Or maybe you want to use it another way. Either way it's there.

There are also a few methods that return the give information about the current status of the game. The first two return whether the game is in intermission or whether it's over.

```
def is_game_over(self, prev=False):
    """
    Check if the game is over for both the html and json pbp. If prev=True check for␣
↪the previous event

    :param prev: Check the game status for the previous event

    :return: Boolean - True if over
    """
    if not prev:
        return self.html_game_status == self.api_game_status == "Final"
    else:
        return self.prev_html_game_status == self.prev_api_game_status == "Final"

def is_intermission(self, prev=False):
    """
    Check if in intermission for both the html and json pbp. If prev=True check for␣
↪the previous event

    :param prev: Check the game status for the previous event

    :return: Boolean - True if yes
```

(continues on next page)

(continued from previous page)

```
    """
    if not prev:
        return self.html_game_status == self.api_game_status == "Intermission"
    else:
        return self.prev_html_game_status == self.prev_api_game_status ==
↪"Intermission"
```

Two things probably stand out is the option to check the status for the previous event (why do we care what it was earlier?) and the fact that two statuses exist.

First let's talk about the two status. There are currently two pages that always need to be scraped for for data for the Play-By-Play. One is an html file and one is the json api. The issue is that the api updates faster than the html. So the api may say the game is over but the html version is still missing a few events. For this reason we need to check that both are aligned.

The 'prev' keyword for both comes into play when we consider the last method 'is_ongoing'. This checks whether the game is currently being played. Which means the game: Started, is not in intermission, and is not over. Here's the method:

```
def is_ongoing(self):
 """
 Check if the game is currently being played.

 The logic here is that we run into an issue with intermission and the end of game.␣
↪If the game is just changed
 to Final or Intermission the end user will assume the game isn't ongoing and will␣
↪not update with the most
 recent events. They'll be delayed for intermission and won't place it at all for␣
↪Final games. So we use the
 previous event as a guide. If it's currently in intermission or Final - we check the␣
↪previous status. If it's
 the same the user already has the data. Otherwise we 'lie' and say the game is still␣
↪ongoing.

 :return: Boolean
 """
 # The game is currently being played
 if self.time_until_game() == 0 and not self.is_game_over() and not self.is_
↪intermission() and self.pbp_df.shape[0] > 0:
     return True
 # Since it's not being played check if game is over and if it wasn't for the previous
 elif self.is_game_over() and not self.is_game_over(prev=True):
     return True
 # Check if it's in intermission and the if it was for the previous event
 elif self.is_intermission() and not self.is_intermission(prev=True):
     return True
 else:
     return False
```

I recommend looking at the function definition written above. Basically checking the previous event makes sure we got the most recent event if the game is over or in intermission. So if the last status was intermission and this one is too we know we don't need to scrape. But if the last status wasn't the means we are missing some information (presumably something happened between the last event and the end of the period).

Live Scrape

Module to scrape live game info

class hockey_scraper.nhl.live_scrape.**LiveGame**(*game_id*, *start_time*, *home_team*, *away_team*, *status*, *espn_id*, *date*, *if_scrape_shifts*)

This is a class holds all the information for a given game

Parameters

- **game_id** (*int*) – The NHL game id (ex: 2018020001)
- **start_time** (*datetime*) – The UTC time of when the game begins
- **home_team** (*str*) – Tricode for the home team (ex: NYR)
- **away_team** (*str*) – Tricode for the home team (ex: MTL)
- **espn_id** (*int*) – The ESPN game id for their feed
- **date** (*str*) – Date of the game (ex: 2018-10-30)
- **if_scrape_shifts** (*bool*) – Whether or not you want to scrape shifts
- **api_game_status** (*str*) – Current Status of the game - ["Final", "Live", "Intermission]
- **html_game_status** (*str*) – Current Status of the game - ["Final", "Live", "Intermission"]
- **intermission_time_remaining** (*int*) – Time remaining in the intermission. 0 if not in intermission
- **players** (*dict*) – Player info for both teams
- **head_coaches** (*dict*) – Head coaches for both teams
- **_pbp_df** (*DataFrame*) – Holds most recent pbp data
- **_shifts_df** (*DataFrame*) – Holds most recent shift data
- **_prev_pbp_df** (*DataFrame*) – Holds the previous pbp data (for just in case)
- **_prev_shifts_df** (*DataFrame*) – Holds the previous shift data (for just in case)

is_game_over(*prev=False*)

Check if the game is over for both the html and json pbp. If prev=True check for the previous event

Parameters **prev** – Check the game status for the previous event

Returns Boolean - True if over

is_intermission(*prev=False*)

Check if in intermission for both the html and json pbp. If prev=True check for the previous event

Parameters **prev** – Check the game status for the previous event

Returns Boolean - True if yes

is_ongoing()

Check if the game is currently being played.

The logic here is that we run into an issue with intermission and the end of game. If the game is just changed to Final or Intermission the end user will assume the game isn't ongoing and will not update with the most recent events. They'll be delayed for intermission and won't place it at all for Final games. So we use the previous event as a guide. If it's currently in intermission or Final - we check the previous status. If it's the same the user already has the data. Otherwise we 'lie' and say the game is still ongoing.

Returns Boolean

scrape(*force=False*)
Scrape the given game. Check if currently ongoing or started

Parameters **force** (*bool*) – Whether or not to force it to scrape even if it's over

Returns None

scrape_live_game(*force=False*)
Scrape the live info for a given game

Parameters **force** – Whether to scrape no matter what (used for intermission here)

Returns None

time_until_game()
Return the seconds until the game starts

Returns seconds until game

class hockey_scraper.nhl.live_scrape.**ScrapeLiveGames**(*date*, *preseason=False*, *if_scrape_shifts=False*, *pause=15*, *game_ids=[]*)
Class than contains the info for all the games on a specific day

Parameters

- **date** (*str*) – Date of games (ex: 2018-10-30)
- **preseason** (*bool*) – If you want to scrape preseason games
- **if_scrape_shifts** (*bool*) – Whether or not you want to scrape shifts
- **live_games** (*list*) – List of LiveGame objects for
- **pause** (*int*) – Amount to pause after each scraping call

finished()
Check if done with all games

Returns Boolean

get_espn_ids(*games*)
Get espn game ids for all games that day

Parameters **games** (*list*) – games today

Returns Games with corresponding espn game ids

get_games()
Get initial game info -> Called with object creation. Includes: players, espn_ids, standard game info

Returns Dict - LiveGame objects for all games today

sleep_next_game()
Sleep until the next game starts. Otherwise just looping and doing nothing

Returns None

update_live_games(*force=False*, *sleep_next=False*)
Scrape the pbp & shifts of ongoing games

Parameters

- **force** (*bool*) – Whether or not to force it to scrape even if it's in intermission
- **sleep_next** (*bool*) – Sleep until the next game starts

Returns None

hockey_scraper.nhl.live_scrape.**check_date_format**(*date*)
Verify the date format. If wrong raises a ValueError

Parameters **date** – User supplied date

Returns None

hockey_scraper.nhl.live_scrape.**set_docs_dir**(*user_dir*)
Set the docs directory

Parameters **user_dir** – User specified directory for storing saves scraped files

Returns None

1.3 License

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for

software and other kinds of works.

The licenses for most software and other practical works are designed

to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether

gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains

that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run

modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and

modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of

works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this

License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work

in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based

on the Program.

To "propagate" a work means to do anything with it that, without

permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other

parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices"

to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work

for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official

standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other

than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all

the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users

can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that

same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not

convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under

the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological

measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid

circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the

work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you

receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,

and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to

produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
>
> c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent

works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms

of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

> a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
>
> b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software

in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded

from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any

tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods,

procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or

specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a

requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided,

in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this

License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option

remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you

add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

> a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
>
> b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
>
> c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
>
> d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
>
> e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
>
> f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further

restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you

must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the

form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly

provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your

license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is

reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the

licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or

run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically

receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the

rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this

License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims

owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this

definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free

patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express

agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license,

and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or

arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within

the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting

any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have

permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue

to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of

the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the

Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future

versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different

permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY

APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING

WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided

above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest

possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest

to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short

notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,

if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program

into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

Notes:

- Coordinates are only scraped from ESPN for versions 1.33+
- NWHL usage has been deprecated due to the removal of the pbp information for each game.

CHAPTER 2

Purpose

This package is designed to allow people to scrape both NHL and NWHL data. For the NHL, one can scrape the Play by Play and Shift data off of the National Hockey League (NHL) API and website for all preseason, regular season, and playoff games since the 2007-2008 season. For the NWHL, one is able to scrape the Play by Play data off of their API and website for all preseason, regular season, and playoff games since the 2015-2016 season.

CHAPTER 3

Prerequisites

You are going to need to have python installed for this. This should work for both python 2.7 and 3 (I recommend having from at least version 3.6.0 but earlier versions should be fine).

If you don't have python installed on your machine, I'd recommend installing it through the anaconda distribution. Anaconda comes with a bunch of libraries pre-installed so it's easier to start off.

CHAPTER 4

Installation

To install all you need to do is open up your terminal and type in:

```
pip install hockey_scraper
```

CHAPTER 5

NHL Usage

5.1 Standard Scrape Functions

Scrape data on a season by season level:

```
import hockey_scraper

# Scrapes the 2015 & 2016 season with shifts and stores the data in a Csv file
hockey_scraper.scrape_seasons([2015, 2016], True)

# Scrapes the 2008 season without shifts and returns a dictionary containing the pbp␣
↪Pandas DataFrame
scraped_data = hockey_scraper.scrape_seasons([2008], False, data_format='Pandas')
```

Scrape a list of games:

```
import hockey_scraper

# Scrapes the first game of 2014, 2015, and 2016 seasons with shifts and stores the␣
↪data in a Csv file
hockey_scraper.scrape_games([2014020001, 2015020001, 2016020001], True)

# Scrapes the first game of 2007, 2008, and 2009 seasons with shifts and returns a␣
↪Dictionary with the Pandas DataFrames
scraped_data = hockey_scraper.scrape_games([2007020001, 2008020001, 2009020001], True,
↪ data_format='Pandas')
```

Scrape all games in a given date range:

```
import hockey_scraper

# Scrapes all games between 2016-10-10 and 2016-10-20 without shifts and stores the␣
↪data in a Csv file
hockey_scraper.scrape_date_range('2016-10-10', '2016-10-20', False)
```

(continues on next page)

(continued from previous page)

```
# Scrapes all games between 2015-1-1 and 2015-1-15 without shifts and returns a␣
↪Dictionary with the pbp Pandas DataFrame
scraped_data = hockey_scraper.scrape_date_range('2015-1-1', '2015-1-15', False, data_
↪format='Pandas')
```

The dictionary returned by setting the default argument "data_format" equal to "Pandas" is structured like:

```
{
  # Both of these are always included
  'pbp': pbp_df,
  'errors': scraping_errors,

  # This is only included when the argument 'if_scrape_shifts' is set equal to True
  'shifts': shifts_df
}
```

Scraped files can also be saved in a separate directory if wanted. This allows one to re-scrape games quicker as we don't need to retrieve them. This is done by specifying the keyword argument 'docs_dir' equal to True to automatically create, store, and look in the home directory. Or you can provide your own directory where you want everything to be stored (it must exist beforehand).

```
import hockey_scraper

# Create or try to refer to a directory in the home repository
# Will create a directory called 'hockey_scraper_data' in the home directory (if it␣
↪doesn't exist)
hockey_scraper.scrape_seasons([2015, 2016], True, docs_dir=True)

# Path to the given directory
USER_PATH = "/...."

# Scrapes the 2015 & 2016 season with shifts and stores the data in a Csv file
# Also includes a path for an existing directory for the scraped files to be placed␣
↪in or retrieved from.
hockey_scraper.scrape_seasons([2015, 2016], True, docs_dir=USER_PATH)

# Once could chose to re-scrape previously saved files by making the keyword argument␣
↪rescrape=True
hockey_scraper.scrape_seasons([2015, 2016], True, docs_dir=USER_PATH, rescrape=True)
```

5.2 Schedule

The schedule for any past or future games can be scraped as follows:

```
import hockey_scraper

# As oppossed to the other calls the default format is 'Pandas' which returns a␣
↪DataFrame
sched_df = hockey_scraper.scrape_schedule("2019-10-01", "2020-07-01")
```

The columns returned are: ['game_id', 'date', 'venue', 'home_team', 'away_team', 'start_time', 'home_score', 'away_score', 'status']

5.3 Live Scraping

Here is a simple example of a way to setup live scraping. I strongly suggest checking out this section of the docs if you plan on using this.

```
import hockey_scraper as hs


def to_csv(game):
    """
    Store each game DataFrame in a file

    :param game: LiveGame object

    :return: None
    """

    # If the game:
    # 1. Started - We recorded at least one event
    # 2. Not in Intermission
    # 3. Not Over
    if game.is_ongoing():
        # Print the description of the last event
        print(game.game_id, "->", game.pbp_df.iloc[-1]['Description'])

        # Store in CSV files
        game.pbp_df.to_csv(f"../hockey_scraper_data/{game.game_id}_pbp.csv", sep=',')
        game.shifts_df.to_csv(f"../hockey_scraper_data/{game.game_id}_shifts.csv",␣
↪sep=',')

if __name__ == "__main__":
    # B4 we start set the directory to store the files
    # You don't have to do this but I recommend it
    hs.live_scrape.set_docs_dir("../hockey_scraper_data")

    # Scrape the info for all the games on 2018-11-15
    games = hs.ScrapeLiveGames("2018-11-15", if_scrape_shifts=True, pause=20)

    # While all the games aren't finished
    while not games.finished():
        # Update for all the games currently being played
        games.update_live_games(sleep_next=True)

        # Go through every LiveGame object and apply some function
        # You can of course do whatever you want here.
        for game in games.live_games:
            to_csv(game)
```

The full documentation can be found here.

CHAPTER 6

Contact

Please contact me for any issues or suggestions. For any bugs or anything related to the code please open an issue. Otherwise you can email me at Harryshomer@gmail.com.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

H

I

L

P

R

S

T

U