nhl

Matt Hostetter

Sep 17, 2020

CONTENTS

CHAPTER

ONE

API

1.1 nhl

Classes

Conference(*args, **kwargs)	NHL conference object.
Division(*args, **kwargs)	NHL division object.
Event(*args, **kwargs)	NHL event object.
Franchise(*args, **kwargs)	NHL franchise object.
Game(*args, **kwargs)	NHL game object.
GameInfo(id, season_id, type, start, end, ...)	NHL game info.
Gametime(*args, **kwargs)	NHL gametime object.
List([iterable])	Searchable, sortable, and filter-able `list` subclass
Location(*args, **kwargs)	NHL location object.
Official(*args, **kwargs)	NHL official object.
Player(*args, **kwargs)	NHL player object.
PlayerStats(player, shifts, events)	NHL player statistics.
Shift(*args, **kwargs)	NHL shift object.
Team(*args, **kwargs)	NHL team object.
Venue(*args, **kwargs)	NHL venue object.

class nhl.**Conference**(**args*, ***kwargs*)

Bases: nhl.flyweight.Flyweight

NHL conference object.

This is the detailed docstring.

classmethod from_key(*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str
Conference abbreviation

Type str

id: int
The NHL statsapi universal conference ID

Type int

name: str
Conference name

Type str

name_short: str
Conference name shortened

Type str

class nhl.**Division**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL division object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str
Division abbreviated name

Type str

id: int
The NHL statsapi universal division ID

Type int

name: str
Division name

Type str

name_short: str
Division short name

Type int

class nhl.**Event**(**args*, ***kwargs*)

Bases: nhl.flyweight.Flyweight

NHL event object.

This is the detailed docstring.

classmethod from_key(*game_id*, *id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*game_id*, *id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

by_player: nhl.player.Player

by_players_on_ice: nhl.list.List

property by_strength

by_team: nhl.team.Team

game_id: int

gametime: nhl.gametime.Gametime

id: int

property lead

location: nhl.location.Location

property name

on_player: nhl.player.Player

on_players_on_ice: nhl.list.List

property on_strength

on_team: nhl.team.Team

score: tuple

property strength

property subname

subtype: str

type: str

value: float

property valuename

class nhl.**Franchise**(**args*, ***kwargs*)

Bases: nhl.flyweight.Flyweight

NHL franchise object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int
The NHL statsapi universal franchise ID

Type int

name: str
Franchise name

Type str

class nhl.**Game**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL game object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

away: nhl.team.Team
Game away

Type *Team*

property defensemen

events: nhl.list.List
List:

property forwards

property goalies

home: nhl.team.Team
Game home

Type *Team*

info: nhl.gameinfo.GameInfo
Game info

Type *GameInfo*

players: nhl.list.List
List:

property skaters

class nhl.**GameInfo**(*id: int*, *season_id: int*, *type: str*, *start: datetime.datetime*, *end: datetime.datetime*, *venue: nhl.venue.Venue*, *home_team: nhl.team.Team*, *away_team: nhl.team.Team*, *score: tuple*, *end_type: str*, *end_time: nhl.gametime.Gametime*, *referees: nhl.list.List*, *linesmen: nhl.list.List*)

Bases: object

NHL game info.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type *nhl.Game* or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property away_score

away_team: nhl.team.Team

property date

property description

end: datetime.datetime

end_time: nhl.gametime.Gametime

end_type: str

property home_score

home_team: nhl.team.Team

id: int

linesmen: nhl.list.List

property playoff_game

property playoff_round

property playoff_series

referees: nhl.list.List

score: tuple

property season

season_id: int

property season_type

start: datetime.datetime

type: str

venue: nhl.venue.Venue

class nhl.**Gametime**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL gametime object.

This object represents a unique time of the game. There are convenience properties to convert the gametime into convenient formats.

Parameters

- **period** (*int*) – The period of the game.
- **period_sec** (*int*) – The number of elapsed seconds in the period.

classmethod from_key(*period*, *period_sec*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*period*, *period_sec*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property min_sec
Elapsed minutes and seconds of the game.

Type (int, int)

period: int
Game period. 1-3 for regulation. 4+ for overtime.

Type int

property period_min_sec
Elapsed minutes and seconds of the period.

Type (int, int)

period_sec: int
Elapsed seconds of the period.

Type int

property period_str
Period number as string (i.e. "2nd")

Type str

property sec
Elapsed seconds of the game.

Type int

class nhl.**List**(*iterable=()*, */*)
Bases: list

Searchable, sortable, and filter-able list subclass

filter(*attr*, *value*, *compare='=='*)
Filter list by a comparison of a given attribute (or chain or attributes).

Parameters

- **attr** (*str*) – attributes to be compared (with initial . omitted)
- **value** (*any*) – value to compare attr against
- **compare** (*str*) – comparison type "==", "=", "eq": *attr == value* "!=", "ne": *attr != value* ">", "gt": *attr > value* ">=", "ge": *attr >= value* "<", "lt": *attr < value* "<=", "le": *attr <= value* "in", "contains": *value in attr*

Returns reduced list with items that satisfy filter criterion

Return type *nhl.List*

select(*attr*, *default=None*)
Select a given attribute (or chain or attributes) from the objects within the list.

Parameters

- **attr** (*str*) – attributes to be selected (with initial . omitted)
- **default** (*any*) – value to return if given element in list doesn't contain desired attribute

Returns list of selected attribute values

Return type *nhl.List*

sort(*key*, *reverse=False*)
Sort the list by *key* either ascending or descending.

Parameters

- **key** (*str*) – atrribute to sort by (with initial . omitted)
- **reverse** (*bool, optional*) – reverse the direction of sort to descending, (default False, ascending)

Returns sorted list

Return type *nhl.List*

unique()
Reduce the list to unique elements.

Returns reduced list

Return type *nhl.List*

property len
Helper property to return length of list

Returns length of list *len(self)*

Return type int

class nhl.**Location**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL location object.

This is the detailed docstring.

distance(*other*)
Measure distance between current location and another on-ice location.

Parameters **other** (Location) – location to measure distance from

Returns distance (ft)

Return type float

classmethod from_key(*x*, *y*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*x*, *y*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

x: int
Rink position in ft along x-axis (length) [-100, 100]

Type int

y: int
Rink position in ft along y-axis (breadth) [-42, 42]

Type int

class nhl.**Official**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL official object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type *nhl.Official* or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property first_name
Official's first name

Type str

id: int
The NHL statsapi universal official ID

Type int

property last_name
Official's last name

Type str

name: str
Official's full name

Type str

class nhl.**Player**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL player object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property age
Current age in years

Type int

birth_city: str
Player's birth city

Type str

birth_country: str
Player's birth country

Type str

birth_date: datetime.date
Player's birth date

Type datetime.date

property first_name
Player's first name

Type str

height: int
Player's height in total inches

Type int

property height_ft_in
Height in feet and inches (*height* // 12, *height* % 12)

Type int

id: int
The NHL statsapi universal player ID

Type int

property last_name
Player's last name

Type str

name: str
Player's full name

Type str

number: int
Player's primary number

Type int

position: str
Player's primary position ("LW", "C", "RW", "D", "G")

Type str

shoots_catches: str
Indication of whether the player shoots (skater)/catches (goalie) "L" or "R"

Type str

weight: int
Player's weight in lbs

Type int

class nhl.**PlayerStats**(*player: nhl.player.Player*, *shifts: nhl.list.List*, *events: nhl.list.List*)
Bases: object

NHL player statistics.

This is the detailed docstring.

events: nhl.list.List
Player's events

Type *List*[*Event*]

player: nhl.player.Player
Player object

Type *Player*

shifts: nhl.list.List
Player's shifts

Type *List*[*Shift*]

class nhl.**Shift**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL shift object.

This is the detailed docstring.

classmethod from_key(*game_id*, *player_id*, *shift_id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*game_id*, *player_id*, *shift_id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

game_id: int
NHL statsapi unique game ID

Type int

property length

off: int
Shift end gametime

Type *Gametime*

on: int
Shift start gametime

Type *Gametime*

property player

player_id: int
NHL statsapi unique player ID

Type int

shift_id: int
Shift number for specified game

Type int

class nhl.**Team**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL team object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str
Team's name abbreviated

Type int

conference: nhl.conference.Conference
Conference:

division: nhl.division.Division
Division:

first_year: int
First year of play

Type int

franchise: nhl.franchise.Franchise
Franchise:

property full_name
Team's full name

Type str

id: int
The NHL statsapi universal team ID

Type int

location: str
Team's location

Type str

name: str
Team's name

Type str

class nhl.**Venue**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL venue object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int
The NHL statsapi universal venue ID

Type int

name: str
Venue name

Type str

1.2 nhl.rink

Module Attributes

LENGTH	Rink length (ft)
WIDTH	Rink width (ft)
DX	Delta length from center ice (ft)
DY	Delta width from center ice (ft)
BLUE_LINE_X	Blue line x-position from center ice (ft)
GOAL_LINE_X	Goal line x-position from center ice (ft)
NZ_FACEOFF_DOT_X	Neutral zone faceoff dot x-position from center ice (ft)
OZ_FACEOFF_DOT_X	Offensive zone faceoff dot x-position from center ice (ft)
FACEOFF_DOT_Y	All faceoff dot y-position from center ice (ft)
FACEOFF_DOTS	Tuple of faceoff dots (x, y)

Classes

Location(*args, **kwargs)	NHL location object.

class nhl.rink.**Location**(**args*, ***kwargs*)

Bases: nhl.flyweight.Flyweight

NHL location object.

This is the detailed docstring.

distance(*other*)

Measure distance between current location and another on-ice location.

Parameters **other** (Location) – location to measure distance from

Returns distance (ft)

Return type float

classmethod from_key(*x*, *y*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*x*, *y*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

x: int

Rink position in ft along x-axis (length) [-100, 100]

Type int

y: int

Rink position in ft along y-axis (breadth) [-42, 42]

Type int

nhl.rink.**BLUE_LINE_X = 25**
Blue line x-position from center ice (ft)

nhl.rink.**DX = 100.0**
Delta length from center ice (ft)

nhl.rink.**DY = 42.5**
Delta width from center ice (ft)

nhl.rink.**FACEOFF_DOTS = ((0, 0), (20, 22), (20, -22), (-20, 22), (-20, -22), (69, 22), (69,**
Tuple of faceoff dots (x, y)

nhl.rink.**FACEOFF_DOT_Y = 22**
All faceoff dot y-position from center ice (ft)

nhl.rink.**GOAL_LINE_X = 89**
Goal line x-position from center ice (ft)

nhl.rink.**LENGTH = 200**
Rink length (ft)

nhl.rink.**NZ_FACEOFF_DOT_X = 20**
Neutral zone faceoff dot x-position from center ice (ft)

nhl.rink.**OZ_FACEOFF_DOT_X = 69**
Offensive zone faceoff dot x-position from center ice (ft)

nhl.rink.**WIDTH = 85**
Rink width (ft)

1.3 nhl.statsapi

Functions

conference(id)
conferences()
division(id)
divisions()
franchise(id)
franchises()
game(id)
parse_conference(json)
parse_conferences(json)
parse_division(json)
parse_events(json, info, home_score, . . .)
parse_franchise(json)
parse_game(json)
parse_location(json, gametime, flip_sides)
parse_official(json)
parse_player(json)
parse_shifts(game_id, team_id, player_id, . . .)
parse_team(json)
parse_teams(json)

continues on next page

Table 5 – continued from previous page

parse_venue(json)	
player(id)	
players(ids)	
team(id)	
teams([ids])	
venue(id)	
venues()	

Classes

BeautifulSoup([markup, features, builder, ...])	A data structure representing a parsed HTML or XML document.
Conference(*args, **kwargs)	NHL conference object.
Division(*args, **kwargs)	NHL division object.
Event(*args, **kwargs)	NHL event object.
Franchise(*args, **kwargs)	NHL franchise object.
Game(*args, **kwargs)	NHL game object.
GameInfo(id, season_id, type, start, end, ...)	NHL game info.
Gametime(*args, **kwargs)	NHL gametime object.
List([iterable])	Searchable, sortable, and filter-able `list` subclass
Location(*args, **kwargs)	NHL location object.
Official(*args, **kwargs)	NHL official object.
Player(*args, **kwargs)	NHL player object.
PlayerStats(player, shifts, events)	NHL player statistics.
Shift(*args, **kwargs)	NHL shift object.
Team(*args, **kwargs)	NHL team object.
Venue(*args, **kwargs)	NHL venue object.

class nhl.statsapi.**BeautifulSoup**(*markup=''*, *features=None*, *builder=None*, *parse_only=None*, *from_encoding=None*, *exclude_encodings=None*, *element_classes=None*, ***kwargs*)

Bases: bs4.element.Tag

A data structure representing a parsed HTML or XML document.

Most of the methods you'll call on a BeautifulSoup object are inherited from PageElement or Tag.

Internally, this class defines the basic interface called by the tree builders when converting an HTML/XML document into a data structure. The interface abstracts away the differences between parsers. To write a new tree builder, you'll need to understand these methods as a whole.

These methods will be called by the BeautifulSoup constructor:

- reset()
- feed(markup)

The tree builder may call these methods from its feed() implementation:

- handle_starttag(name, attrs) # See note about return value
- handle_endtag(name)
- handle_data(data) # Appends to the current data node
- endData(containerClass) # Ends the current data node

No matter how complicated the underlying parser is, you should be able to build a tree using 'start tag' events, 'end tag' events, 'data' events, and "done with data" events.

If you encounter an empty-element tag (aka a self-closing tag, like HTML's
 tag), call handle_starttag and then handle_endtag.

decode(*pretty_print=False*, *eventual_encoding='utf-8'*, *formatter='minimal'*)

Returns a string or Unicode representation of the parse tree as an HTML or XML document.

Parameters

- **pretty_print** – If this is True, indentation will be used to make the document more readable.
- **eventual_encoding** – The encoding of the final document. If this is None, the document will be a Unicode string.

endData(*containerClass=None*)
Method called by the TreeBuilder when the end of a data segment occurs.

handle_data(*data*)
Called by the tree builder when a chunk of textual data is encountered.

handle_endtag(*name*, *nsprefix=None*)
Called by the tree builder when an ending tag is encountered.

Parameters

- **name** – Name of the tag.
- **nsprefix** – Namespace prefix for the tag.

handle_starttag(*name*, *namespace*, *nsprefix*, *attrs*, *sourceline=None*, *sourcepos=None*)
Called by the tree builder when a new tag is encountered.

Parameters

- **name** – Name of the tag.
- **nsprefix** – Namespace prefix for the tag.
- **attrs** – A dictionary of attribute values.
- **sourceline** – The line number where this tag was found in its source document.
- **sourcepos** – The character position within *sourceline* where this tag was found.

If this method returns None, the tag was rejected by an active SoupStrainer. You should proceed as if the tag had not occurred in the document. For instance, if this was a self-closing tag, don't call handle_endtag.

insert_after(*successor*)
This method is part of the PageElement API, but *BeautifulSoup* doesn't implement it because there is nothing before or after it in the parse tree.

insert_before(*successor*)
This method is part of the PageElement API, but *BeautifulSoup* doesn't implement it because there is nothing before or after it in the parse tree.

new_string(*s*, *subclass=None*)
Create a new NavigableString associated with this BeautifulSoup object.

new_tag(*name*, *namespace=None*, *nsprefix=None*, *attrs={}*, *sourceline=None*, *sourcepos=None*, ***kwattrs*)
Create a new Tag associated with this BeautifulSoup object.

Parameters

- **name** – The name of the new Tag.
- **namespace** – The URI of the new Tag's XML namespace, if any.
- **prefix** – The prefix for the new Tag's XML namespace, if any.
- **attrs** – A dictionary of this Tag's attribute values; can be used instead of *kwattrs* for attributes like 'class' that are reserved words in Python.
- **sourceline** – The line number where this tag was (purportedly) found in its source document.
- **sourcepos** – The character position within *sourceline* where this tag was (purportedly) found.
- **kwattrs** – Keyword arguments for the new Tag's attribute values.

object_was_parsed(*o*, *parent=None*, *most_recent_element=None*)
Method called by the TreeBuilder to integrate an object into the parse tree.

popTag()
Internal method called by _popToTag when a tag is closed.

pushTag(*tag*)
Internal method called by handle_starttag when a tag is opened.

reset()
Reset this object to a state as though it had never parsed any markup.

string_container(*base_class=None*)

ASCII_SPACES = ' \n\t\x0c\r'

DEFAULT_BUILDER_FEATURES = ['html', 'fast']

NO_PARSER_SPECIFIED_WARNING = 'No parser was explicitly specified, so I\'m using the b

ROOT_TAG_NAME = '[document]'

class nhl.statsapi.**Conference**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL conference object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str
Conference abbreviation

Type str

id: int
The NHL statsapi universal conference ID

Type int

name: str
Conference name

Type str

name_short: str
Conference name shortened

Type str

class nhl.statsapi.**Division**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL division object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str
Division abbreviated name

Type str

id: int
The NHL statsapi universal division ID

Type int

name: str
Division name

Type str

name_short: str
Division short name

Type int

class nhl.statsapi.**Event**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL event object.

This is the detailed docstring.

classmethod from_key(*game_id*, *id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*game_id*, *id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

by_player: nhl.player.Player

by_players_on_ice: nhl.list.List

property by_strength

by_team: nhl.team.Team

game_id: int

gametime: nhl.gametime.Gametime

id: int

property lead

location: nhl.location.Location

property name

on_player: nhl.player.Player

on_players_on_ice: nhl.list.List

property on_strength

on_team: nhl.team.Team

score: tuple

property strength

property subname

subtype: str

type: str

value: float

property valuename

class nhl.statsapi.**Franchise**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL franchise object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int
The NHL statsapi universal franchise ID

Type int

name: str
Franchise name

Type str

class nhl.statsapi.**Game**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL game object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

away: nhl.team.Team
Game away

Type *Team*

property defensemen

events: nhl.list.List
List:

property forwards

property goalies

home: nhl.team.Team
Game home

Type *Team*

info: nhl.gameinfo.GameInfo
Game info

Type *GameInfo*

players: nhl.list.List
List:

property skaters

class nhl.statsapi.**GameInfo**(*id: int*, *season_id: int*, *type: str*, *start: datetime.datetime*, *end: datetime.datetime*, *venue: nhl.venue.Venue*, *home_team: nhl.team.Team*, *away_team: nhl.team.Team*, *score: tuple*, *end_type: str*, *end_time: nhl.gametime.Gametime*, *referees: nhl.list.List*, *linesmen: nhl.list.List*)

Bases: object

NHL game info.

This is the detailed docstring.

classmethod from_key(*id*)

Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type *nhl.Game* or None

classmethod has_key(*id*)

Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property away_score

away_team: nhl.team.Team

property date

property description

end: datetime.datetime

end_time: nhl.gametime.Gametime

end_type: str

property home_score

home_team: nhl.team.Team

id: int

linesmen: nhl.list.List

property playoff_game

property playoff_round

property playoff_series

referees: nhl.list.List

score: tuple

property season

season_id: int

property season_type

start: datetime.datetime

type: str

venue: nhl.venue.Venue

class nhl.statsapi.**Gametime**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL gametime object.

This object represents a unique time of the game. There are convenience properties to convert the gametime into convenient formats.

Parameters

- **period** (*int*) – The period of the game.
- **period_sec** (*int*) – The number of elapsed seconds in the period.

classmethod from_key(*period*, *period_sec*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*period*, *period_sec*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property min_sec
Elapsed minutes and seconds of the game.

Type (int, int)

period: int
Game period. 1-3 for regulation. 4+ for overtime.

Type int

property period_min_sec
Elapsed minutes and seconds of the period.

Type (int, int)

period_sec: int
Elapsed seconds of the period.

Type int

property period_str
Period number as string (i.e. "2nd")

Type str

property sec
Elapsed seconds of the game.

Type int

class nhl.statsapi.**List**(*iterable=()*, */*)
Bases: list

Searchable, sortable, and filter-able list subclass

filter(*attr*, *value*, *compare='=='*)
Filter list by a comparison of a given attribute (or chain or attributes).

Parameters

- **attr** (*str*) – attributes to be compared (with initial . omitted)
- **value** (*any*) – value to compare attr against
- **compare** (*str*) – comparison type "==", "=", "eq": *attr == value* "!=", "ne": *attr != value* ">", "gt": *attr > value* ">=", "ge": *attr >= value* "<", "lt": *attr < value* "<=", "le": *attr <= value* "in", "contains": *value in attr*

Returns reduced list with items that satisfy filter criterion

Return type *nhl.List*

select (*attr*, *default=None*)
Select a given attribute (or chain or attributes) from the objects within the list.

Parameters

- **attr** (*str*) – attributes to be selected (with initial . omitted)
- **default** (*any*) – value to return if given element in list doesn't contain desired attribute

Returns list of selected attribute values

Return type *nhl.List*

sort (*key*, *reverse=False*)
Sort the list by *key* either ascending or descending.

Parameters

- **key** (*str*) – atrribute to sort by (with initial . omitted)
- **reverse** (*bool, optional*) – reverse the direction of sort to descending, (default False, ascending)

Returns sorted list

Return type *nhl.List*

unique ()
Reduce the list to unique elements.

Returns reduced list

Return type *nhl.List*

property len
Helper property to return length of list

Returns length of list *len(self)*

Return type int

class nhl.statsapi.**Location** (**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL location object.

This is the detailed docstring.

distance (*other*)
Measure distance between current location and another on-ice location.

Parameters **other** (Location) – location to measure distance from

Returns distance (ft)

Return type float

classmethod from_key(*x*, *y*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*x*, *y*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

x: int
Rink position in ft along x-axis (length) [-100, 100]

Type int

y: int
Rink position in ft along y-axis (breadth) [-42, 42]

Type int

class nhl.statsapi.**Official**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL official object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or *None* if key not found

Return type *nhl.Official* or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property first_name
Official's first name

Type str

id: int
The NHL statsapi universal official ID

Type int

property last_name
Official's last name

Type str

name: str
Official's full name

Type str

class nhl.statsapi.**Player**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL player object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

property age
Current age in years

Type int

birth_city: str
Player's birth city

Type str

birth_country: str
Player's birth country

Type str

birth_date: datetime.date
Player's birth date

Type datetime.date

property first_name
Player's first name

Type str

height: int
Player's height in total inches

Type int

property height_ft_in
Height in feet and inches (*height* // 12, *height* % 12)

Type int

id: int
The NHL statsapi universal player ID

Type int

property last_name
Player's last name

Type str

name: str
Player's full name

Type str

number: int
Player's primary number

Type int

position: str
Player's primary position ("LW", "C", "RW", "D", "G")

Type str

shoots_catches: str
Indication of whether the player shoots (skater)/catches (goalie) "L" or "R"

Type str

weight: int
Player's weight in lbs

Type int

class nhl.statsapi.**PlayerStats**(*player: nhl.player.Player*, *shifts: nhl.list.List*, *events: nhl.list.List*)

Bases: object

NHL player statistics.

This is the detailed docstring.

events: nhl.list.List
Player's events

Type *List*[*Event*]

player: nhl.player.Player
Player object

Type *Player*

shifts: nhl.list.List
Player's shifts

Type *List*[*Shift*]

class nhl.statsapi.**Shift**(**args*, ***kwargs*)

Bases: nhl.flyweight.Flyweight

NHL shift object.

This is the detailed docstring.

classmethod from_key(*game_id*, *player_id*, *shift_id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*game_id*, *player_id*, *shift_id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

game_id: int
NHL statsapi unique game ID

Type int

property length

off: int
Shift end gametime

Type *Gametime*

on: int
Shift start gametime

Type *Gametime*

property player

player_id: int
NHL statsapi unique player ID

Type int

shift_id: int
Shift number for specified game

Type int

class nhl.statsapi.**Team**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL team object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

abbreviation: str
Team's name abbreviated

Type int

conference: nhl.conference.Conference
Conference:

division: nhl.division.Division
Division:

first_year: int
First year of play

Type int

franchise: nhl.franchise.Franchise
Franchise:

property full_name
Team's full name

Type str

id: int
The NHL statsapi universal team ID

Type int

location: str
Team's location

Type str

name: str
Team's name

Type str

class nhl.statsapi.**Venue**(**args*, ***kwargs*)
Bases: nhl.flyweight.Flyweight

NHL venue object.

This is the detailed docstring.

classmethod from_key(*id*)
Return flyweight object with specified key, if it has already been created.

Returns Previously constructed flyweight object with given key or None if key not found

Return type cls or None

classmethod has_key(*id*)
Check whether flyweight object with specified key has already been created.

Returns True if already created, False if not

Return type bool

id: int
The NHL statsapi universal venue ID

Type int

name: str
Venue name

Type str

nhl.statsapi.**conference**(*id*)

nhl.statsapi.**conferences**()

nhl.statsapi.**division**(*id*)

nhl.statsapi.**divisions**()

nhl.statsapi.**franchise**(*id*)

nhl.statsapi.**franchises**()

nhl.statsapi.**game**(*id*)

nhl.statsapi.**parse_conference**(*json*)

nhl.statsapi.**parse_conferences**(*json*)

nhl.statsapi.**parse_division**(*json*)

nhl.statsapi.**parse_events**(*json*, *info*, *home_score*, *away_score*, *home_shifts*, *away_shifts*, *flip_sides*)

nhl.statsapi.**parse_franchise**(*json*)

nhl.statsapi.**parse_game**(*json*)

nhl.statsapi.**parse_location**(*json*, *gametime*, *flip_sides*)

nhl.statsapi.**parse_official**(*json*)

nhl.statsapi.**parse_player**(*json*)

nhl.statsapi.**parse_shifts**(*game_id*, *team_id*, *player_id*, *player_number*, *html*)

nhl.statsapi.**parse_team**(*json*)

nhl.statsapi.**parse_teams**(*json*)

nhl.statsapi.**parse_venue**(*json*)

nhl.statsapi.**player**(*id*)

nhl.statsapi.**players**(*ids*)

nhl.statsapi.**team**(*id*)

nhl.statsapi.**teams**(*ids=None*)

nhl.statsapi.**venue**(*id*)

nhl.statsapi.**venues**()

CHAPTER

TWO

EXAMPLES

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Y