

---

# **PlayByPlayMiner**

***Release 1.0***

**Jan L. Moffett**

**Jul 29, 2019**



## CONTENTS:

<b>1</b>	<b>Creating an instance of BeautifulSoup</b>	<b>1</b>
1.1	SoupChef – Functions to turn html files into ‘soup’ . . . . .	1
<b>2</b>	<b>Creating a Game Object</b>	<b>3</b>
2.1	Game – Game object class and functions . . . . .	3
<b>3</b>	<b>Gathering and organizing lineups</b>	<b>5</b>
3.1	Lineup – Lineup object class and functions . . . . .	5
<b>4</b>	<b>Sorting play-by-play descriptions by type</b>	<b>7</b>
4.1	PlateAppearance – PlateAppearance object class and functions . . . . .	7
<b>5</b>	<b>Turning play-by-play descriptions into data</b>	<b>9</b>
5.1	ActionUnit – BatUnit, BRUnit, and SubUnit classes . . . . .	9
5.1.1	Mining play-by-play batting descriptions . . . . .	9
5.1.1.1	BatUnit – BatUnit class . . . . .	9
5.1.2	Mining play-by-play baserunning descriptions . . . . .	13
5.1.2.1	BRUnit – BRUnit class . . . . .	13
5.1.3	Tracking substitutions . . . . .	16
5.1.3.1	SubUnit – SubUnit class . . . . .	16
<b>6</b>	<b>Standardizing Player Name Formats</b>	<b>19</b>
6.1	StdzNames – Name Format Standardization . . . . .	19
<b>7</b>	<b>Assigning codes to base-out states</b>	<b>21</b>
7.1	BaseOut – Base-out state coding functions . . . . .	21
<b>8</b>	<b>Storing Miner’s Output in CSV</b>	<b>23</b>
8.1	CsvStuff – CSV string and file functions . . . . .	23
<b>9</b>	<b>Indices and tables</b>	<b>25</b>
	<b>Python Module Index</b>	<b>27</b>



## CREATING AN INSTANCE OF BEAUTIFULSOUP

### 1.1 SoupChef – Functions to turn html files into ‘soup’

`SoupChef.open_soup(htm_string)`

This function requires the filename for an html file, a string ending in ‘.htm’ or ‘html’, as a parameter and returns an instance of the BeautifulSoup class.

`SoupChef.get_file_info(file_end_string)`

This function assumes that the input html files are named with the following format: ‘AWAY\_HOME\_0101201901.html’. This function takes such a filename as a parameter and returns a dict containing a Game ID string, the away team’s ID code, and the home team’s ID code.



## CREATING A GAME OBJECT

### 2.1 Game – Game object class and functions

**class** `Game.Game (BeautifulSoup_Object)`

This class requires an instance of the BeautifulSoup class as a parameter (see <https://www.crummy.com/software/BeautifulSoup/>). BeautifulSoup is imported and the instance created in the SoupChef module.

**month**

The two-digit month in which the game occurred.

**day**

The two-digit day on which the game occurred.

**year**

The four-digit year in which the game occurred.

**starttime**

Start time of game, in *1:00 pm* format.

**duration**

Duration of the game, in minutes.

**attendance**

Attendance count for game.

**city**

City where the game occurred.

**state**

State where the game occurred.

**park**

Park where game occurred.

**weather**

Temperature and conditions at the start of the game.

**umpires**

Names of the umpires who officiated the game.

**daynight**

A string, either 'day' or 'night', indicating when game occurred.

The following list attributes iterate together:

**awaypositions**

A list of player names and positions for the away team.

**homepositions**

A list of player names and positions for the home team.

**awaynames**

A list of player names for the away team.

**homenames**

A list of player names for the home team.

**awaystartsub**

A list of strings, either 'starter' or 'sub', indicating player status for the away team.

**homestartsub**

A list of strings, either 'starter' or 'sub', indicating player status for the home team.

The following list attributes iterate together:

**paunits**

A list of play-by-play strings, scraped from the html file, each describing the events of a plate appearance, a baserunning event, or a substitution.

**paside**

A list of strings, either 'home' or 'away', corresponding to paunits.

**paunitids**

A list of unique alphanumeric identifiers for each element of paunits.

**get\_game\_info()**

Scrapes game data from html file and sets Game attributes.

**get\_month\_day\_year()**

Sets month, day and year attributes based on date.

**get\_game\_site()**

Sets city, state, and park attributes based on site.

**get\_game\_duration()**

Converts game duration to minutes.

**get\_day\_night()**

Assigns 'day' or 'night' value to daynight attribute based on start time.

**get\_lineup\_positions()**

Scrapes strings containing player names and positions from html file and fills *awaypositions* and *homepositions* lists.

**get\_lineup\_names()**

Extracts player names from strings in *awaypositions* and *homepositions* and fills *awaynames* and *homenames* lists, as well as the *awaystartsub* and *homestartsub* lists.

**get\_action\_units()**

Scrapes and analyzes strings of play by play data from html file and fills the *paunits*, *paside*, and *paunitids* lists.

**do\_stuff()**

Runs all class functions in correct order.

**print\_stuff()**

Prints all class attributes for testing purposes.



## GATHERING AND ORGANIZING LINEUPS

### 3.1 Lineup – Lineup object class and functions

This module requires import of StdzNames module.

```
class Lineup.Lineup (away_name_list,      away_pos_list,      away_startsub_list,  
                    home_name_list, home_pos_list, home_startsub_list, gameid)
```

Initializing the Lineup class requires the following parameters:

- *away\_name\_list* - A list of names for away team output by Game class.
- *away\_pos\_list* - A list of positions corresponding to *away\_name\_list*.
- *away\_startsub\_list* - A list, 'starter' or 'sub', corresponding to *away\_name\_list*.
- *home\_name\_list* - A list of names for home team output by Game class.
- *home\_pos\_list* - A list of positions corresponding to *home\_name\_list*.
- *home\_startsub\_list* - A list, 'starter' or 'sub', corresponding to *home\_name\_list*.
- *gameid* - A unique alpha-numeric identifier for the game being analyzed.

**awaystarters**

A list of starters' last names in proper case, in batting order, for the away team.

**homestarters**

A list of starters' last names in proper case, in batting order, for the home team.

**awaystarterpos**

A list of positions ('dh','p','c',...) corresponding to *awaystarters*.

**homestarterpos**

A list of positions ('dh','p','c',...) corresponding to *homestarters*.

**awaysubs**

A list of subs' last names in proper case, in order of appearance, for the away team.

**homesubs**

A list of subs' last names in proper case, in order of appearance, for the home team.

**awaysubpos**

A list of positions ('p','ph',...) corresponding to *awaysubs*.

**homesubpos**

A list of positions ('p','ph',...) corresponding to *homesubs*.

**awayrelievers**

A list of relievers' last names in proper case, in order of appearance, for the away team.

**homerelievers**

A list of relievers' last names in proper case, in order of appearance, for the home team.

**awayoffsubs**

A list of substitute position players' last names in proper case, in order of appearance, for the away team.

**homeoffsubs**

A list of substitute position players' last names in proper case, in order of appearance, for the home team.

**awayorder**

A list of hitters' last names in proper case, in batting order, for the away team.

**homeorder**

A list of hitters' last names in proper case, in batting order, for the home team.

**get\_starters\_subs()**

Separates starting players from substitutes and sets *awaystarters*, *awaystarterpos*, *awaysubs*, *awaysubpos*, *homestarters*, *homestarterpos*, *homesubs*, and *homesubpos*.

**get\_subs()**

Separates relievers from offensive subs and sets *awayrelievers*, *awayoffsubs*, *homerelievers*, *homeoffsubs*.

**get\_batting\_orders()**

Sets *awayorder* and *homeorder*.

**do\_stuff()**

Runs all SubUnit functions in the correct order.

**print\_stuff()**

Prints all SubUnit attributes for testing purposes.

**get\_data\_string()**

Returns csv string of all variables generated by SubUnit class.

## SORTING PLAY-BY-PLAY DESCRIPTIONS BY TYPE

### 4.1 PlateAppearance – PlateAppearance object class and functions

**class** PlateAppearance.**PlateAppearance** (*pa\_unit\_list*, *pa\_side\_list*, *pa\_id\_list*)

Initialization of the PlateAppearance class requires the following parameters:

- *pa\_unit\_list* - A list of 'plate appearance unit' strings output by the Game class.
- *pa\_side\_list* - A list of the values 'away' or 'home' corresponding to *pa\_unit\_list*.
- *pa\_id\_list* - A list of unique identifiers corresponding to *pa\_unit\_list*.

**actionunits**

A list of play-by-play strings, broken into player-specific units of action.

**autypes**

A list of the values "bat", "br", "batbr", "sub", or 'other' corresponding to *actionunits*.

**ausides**

A list of the values "away" or "home" corresponding to *actionunits*.

**aupaidids**

A list of 'plate appearance unit id's' corresponding to *actionunits*. Up to four 'action units' can share the same 'plate appearance unit id'.

**auhalfinnings**

A list of integers denoting the half-inning in which the 'action unit' occurs, corresponding to *actionunits*.

**get\_action\_units()**

Divides 'plate appearance unit' strings of play-by-play descriptions into player-specific units of action and sets *actionunits*, *aupaidids*, and *ausides*.

**get\_au\_types()**

Assigns a type value: 'bat', 'batbr', 'br', 'sub', or 'other' to each unit in *actionunits*.

**get\_au\_innings()**

Sets *auhalfinnings*.

**do\_stuff()**

Runs all PlateAppearance class functions in the correct order.

**print\_stuff()**

Prints all PlateAppearance class attributes for testing purposes.



## TURNING PLAY-BY-PLAY DESCRIPTIONS INTO DATA

### 5.1 ActionUnit – BatUnit, BRUnit, and SubUnit classes

Import of StdzNames and BaseOut modules required for this module.

#### 5.1.1 Mining play-by-play batting descriptions

##### 5.1.1.1 BatUnit – BatUnit class

**class BatUnit** (*bat\_unit\_str, team, batting\_order, start\_bsot*)

Initializing the BatUnit class requires the following parameters:

- *bat\_unit\_str* - A string of play-by-play batting information.
- *team* - A string, either 'away' or 'home'.
- *batting\_order* - A list of names in the batting order of the specified team.
- *start\_bsot* - A dict containing the base-out information for the start of the plate appearance.

**hit\_verbs**

A list of verbs associated with base hits.

**fo\_verbs**

A list of verbs associated with field outs.

**fo\_locations**

A list of field out locations.

**bip\_verbs**

A list of verbs associated with balls in play.

**unit\_string**

Original string of play-by-play batting information.

**team**

A string, either 'away' or 'home'.

**bo**

A list containing names in batting order for the specified team.

**bsotstart**

A dict containing the base-out information for the start of the plate appearance.

**bsotend**

A dict containing the base-out information for the end of the plate appearance.

**batter\_name**

A string, the last name of the batter in proper case.

**hole**

An integer representing the batter's place in batting order, the index of *batter\_name* in *bo* + 1.

**count\_balls**

A number as a string, the final ball count of the plate appearance.

**count\_strikes**

A number as a string, the final strike count of the plate appearance.

**pitch\_string**

A string of letters denoting balls, strikes, fouls, etc, such as 'BFKS'.

**balls**

An integer, the number of 'B' characters in *pitch\_string* representing the number of balls thrown in plate appearance.

**fouls**

An integer, the number of 'F' characters in *pitch\_string* representing the number of fouls hit in plate appearance.

**swingingstrikes**

An integer, the number of 'S' characters in *pitch\_string* representing the number of swinging strikes in plate appearance.

**calledstrikes**

An integer, the number of 'K' characters in *pitch\_string* representing the number of called strikes in plate appearance.

**BIPs**

An integer, the number of pitches that put a ball in play in plate appearance, either 0 or 1.

**pitches**

An integer, the total number of pitches based on *pitch\_string* and BIP.

**BIP**

An integer, 1 if ball in play, 0 otherwise.

**H**

An integer, 1 if hit, 0 otherwise.

**bases**

An integer, the number of bases gained in hit (0, 1 for single, 2 for double,...).

**singles**

An integer, 1 if single, 0 otherwise.

**doubles**

An integer, 1 if double, 0 otherwise.

**triples**

An integer, 1 if triple, 0 otherwise.

**homers**

An integer, 1 if home run, 0 otherwise.

**hitloc**

A string description of hit location, if applicable. (*"right center"*, *"down the lf line"*,...)

**hitqual**

A string description of hit quality, if applicable. (*"soft"*, *"medium"*, *"hard"*)

**hitttype**

A string description of hit type, if applicable. (*“ground ball”, “fly ball”,...*)

**infh**

An integer, 1 if infield hit, 0 otherwise.

**fieldouts**

An integer, 1 if field out, 0 otherwise.

**foloc**

A string description of field out location, if applicable. (*“rf”, “lb”,...*)

**foqual**

A string description of field out quality, if applicable. (*“soft”, “medium”, “hard”*)

**fotype**

A string description of field out type, if applicable. (*“ground ball”, “fly ball”,...*)

**DP**

An integer, 1 if double play, 0 otherwise.

**doubleplaytype**

A string of numbers representing double play type, if applicable. (*“543”, “63”,...*)

**is\_rch**

A boolean, True if ‘reached’ appears in play-by-play description, False otherwise.

**RBOE**

An integer, 1 if batter reached base on error, 0 otherwise.

**FC**

An integer, 1 if batter reached on fielder’s choice, 0 otherwise.

**SAC**

An integer, 1 if sacrifice bunt, 0 otherwise.

**SF**

An integer, 1 if sacrifice fly, 0 otherwise.

**BB**

An integer, 1 if walk, 0 otherwise.

**IBB**

An integer, 1 if intentional walk, 0 otherwise.

**UBB**

An integer, 1 if unintentional walk, 0 otherwise.

**SO**

An integer, 1 if strike out, 0 otherwise.

**kswing**

An integer, 1 if batter struck out swinging, 0 otherwise.

**klook**

An integer, 1 if batter struck out looking, 0 otherwise.

**UCTS**

An integer, 1 if uncaught third strike, 0 otherwise.

**Soreach**

An integer, 1 if batter reached on uncaught third strike, 0 otherwise.

**trueoutcome**

An integer, 1 if walk, strike out, or home run, 0 otherwise.

**HBP**

An integer, 1 if hit by pitch, 0 otherwise.

**basesadded**

An integer, the number of bases gained by batter during plate appearance. (*1 for single, walk, hit by pitch, etc, 2 for double,...*)

**runsadded**

An integer, the number of runs scored by the batter, not including rbi, either 0 or 1.

**outsadded**

An integer, the number of outs made by batter, not including other baserunners, either 0 or 1.

**get\_batter\_name ()**

Finds batter name in *unit\_string* and sets *batter\_name*.

**get\_count ()**

Finds parentheses in unit string and sets *count\_balls*, *count\_strikes*, and *pitch\_string*.

**get\_pitches ()**

Processes *pitch\_string* and sets *balls*, *fouls*, *calledstrikes*, ..., *pitches*.

**is\_bip ()**

Determines if there is a ball in play and sets *BIP*.

**is\_hit ()**

Determines if there is a hit and sets *H*.

**get\_hit ()**

Determines type of base hit and sets *bases*, *basesadded*, *singles*,...

**get\_hit\_info ()**

Reads unit string and sets *infh* and *hitloc*, *hitqual*, and *hitttype*, if applicable.

**is\_field\_out ()**

Determines if there is a field out and sets *fieldouts*, *doubleplays*, and *outsadded*.

**get\_field\_out ()**

Determines type of field out and sets *foqual*, *fotype*, and *foloc*, if applicable.

**get\_double\_play ()**

Determines type of double play and sets *doubleplaytype*, if applicable.

**is\_reach ()**

Determines if 'reached' appears in unit string and sets *is\_rch*.

**get\_reach ()**

Determines what type of play allowed the batter to reach base, if applicable, and sets *RBOE* and *FC*.

**is\_sac ()**

Determines if there is a sacrifice bunt and sets *SAC*.

**is\_sf ()**

Determines if there is a sacrifice fly and sets *SF*.

**is\_bunt ()**

Determines if there is a bunt and sets *bunt*.

**is\_walk ()**

Determines if there is a walk and sets *BB*, *bases*, and *basesadded*.



**get\_walk\_type()**  
Determines if a walk is intentional or unintentional and sets *IBB* and *UBB*, if applicable.

**is\_strikeout()**  
Determines if there is a strike out and sets *SO*.

**get\_so\_type()**  
Determines type of strike out and sets *kswing*, *klook*, *UCTS*, and *SOreach*.

**is\_true\_outcome()**  
Determines if there is a true outcome (BB, SO, or HR) and sets *trueoutcome*.

**is\_hbp()**  
Determines if batter is hit by pitch and sets *hbp*.

**update\_bod()**  
Updates information in base-out dict.

**do\_stuff()**  
Runs all BatUnit functions in correct order.

**print\_stuff()**  
Prints all BatUnit attributes for testing purposes.

**get\_baseout\_end()**  
Returns updated dict of base-out information.

**get\_data\_string()**  
Returns csv string of all variables generated by BatUnit class.

## 5.1.2 Mining play-by-play baserunning descriptions

### 5.1.2.1 BRUnit – BRUnit class

**class BRUnit** (*unit\_str*, *team*, *batting\_order*, *start\_bsot*, *end\_bsot*)

Initializing the BatUnit class requires the following parameters:

- *unit\_str* - A string of play-by-play baserunning information.
- *team* - A string, either 'away' or 'home'.
- *batting\_order* - A list of names in the batting order of the specified team.
- *start\_bsot* - A dict containing the base-out information for the start of the plate appearance.
- *end\_bsot* - A dict containing the base-out information for the end of the plate appearance.

**unitstring**

Original string of play-by-play baserunning information.

**side**

A string, either 'home' or 'away'.

**bo**

A list containing names in batting order for the specified team.

**bsotstart**

A dict containing the base-out information for the start of the plate appearance.

**bsotend**

A dict containing the base-out information for the end of the plate appearance.

**action**

A string description of action performed by baserunner (“advanced”, “stole”, “scored”, ...).

**runnername**

A string, the last name of the baserunner in proper case.

**hole**

An integer representing the baserunner’s place in batting order, index of *runnername* in *bo* + 1.

**advanced**

A boolean, True if runner advanced, False otherwise.

**stole**

A boolean, True if runner stole a base, False otherwise.

**scored**

A boolean, True if runner scored, False otherwise.

**out**

A boolean, True if runner was out, False otherwise.

**fielder1**

A string denoting the position of first fielder involved in runner out (“1b”, “ss”, ...).

**fielder2**

A string denoting the position of second fielder involved in runner out (“2b”, “3b”, ...).

**startbase**

An integer representing the base occupied by runner at start of plate appearance (0, 1, ..., 4).

**endbase**

An integer representing the base occupied by runner at end of plate appearance (0, 1, ..., 4).

**basesadded**

An integer representing the number of bases gained by runner, *endbase* - *startbase*.

**outsadded**

An integer, 1 if runner out, 0 otherwise.

**runsadded**

An integer, 1 if runner scored, 0 otherwise.

**sba**

An integer, 1 if stolen base attempt, 0 otherwise.

**sb**

An integer, 1 if stolen base, 0 otherwise.

**poa**

An integer, 1 if pickoff attempt, 0 otherwise.

**fpo**

An integer, 1 if failed pickoff attempt, 0 otherwise.

**po**

An integer, 1 if pickoff, 0 otherwise.

**cs**

An integer, 1 if caught stealing, 0 otherwise.

**wp**

An integer, 1 if wild pitch, 0 otherwise.

**pb**  
An integer, 1 if passed ball, 0 otherwise.

**aoe**  
An integer, 1 if runner advanced on error, 0 otherwise.

**outatbase**  
An integer representing the base where runner was called out, if applicable.

**get\_runner\_name(self) :**  
Finds runner name in unit string and sets runnername and hole.

**get\_start\_base ()**  
Determines base occupied by runner at start of plate appearance.

**get\_start\_base\_batbr ()**  
Determines base occupied by batter-runner after reaching initially.

**get\_action ()**  
Determines action performed by baserunner and sets action, advanced, stole, scored, and out.

**is\_pickoff ()**  
Determines if there is a pickoff and sets poa and po.

**get\_advanced ()**  
Determines where and why runner advanced, and sets *endbase*, *wp*, *pb*, and *aoe*, if applicable.

**get\_stole ()**  
Determines what base runner stole and sets *sba*, *sb*, and *endbase*, if applicable.

**get\_scored ()**  
Sets *endbase* and *runsadded* and sets *wp*, *pb*, and *aoe* if applicable.

**def get\_out ()**  
Determines where runner was called out and what fielders made the play.

**update\_bsot ()**  
Updates information in base-out dict.

**get\_bases\_added ()**  
Determines bases added by baserunner during plate appearance.

**do\_stuff ()**  
Runs all BRUnit functions in correct order.

**do\_stuff\_batbr ()**  
Runs all BRUnit functions for batter-runner case in correct order.

**print\_stuff ()**  
Prints all BRUnit information for testing purposes.

**get\_baseout\_end ()**  
Returns updated dict of base-out information.

**get\_data\_string ()**  
Returns csv string of all variables generated by BRUnit class.

### 5.1.3 Tracking substitutions

#### 5.1.3.1 SubUnit – SubUnit class

**class SubUnit** (*sub\_unit\_str*, *team*, *away\_batting\_order*, *away\_off\_subs*, *away\_reliefers*,  
*home\_batting\_order*, *home\_off\_subs*, *home\_reliefers*)

Initializing the SubUnit class requires the following parameters:

- *sub\_unit\_str* - The original play-by-play string of substitution information.
- *team* - A string, either 'away' or 'home'.
- *away\_batting\_order* - A list of names in batting order for the away team.
- *away\_off\_subs* - A list offensive subs for the away team.
- *away\_reliefers* - A list of relief pitchers for away team.
- *home\_batting\_order* - A list of names in batting order for the home team.
- *home\_off\_subs* - A list of offensive subs for the home team.
- *home\_reliefers* - A list of relief pitchers for the home team.

**off\_pos**

A list of positions in format used in play-by-play descriptions ("dh","p",...).

**ogawaybattingorder**

The original batting order for the away team, a list of last names in proper case.

**oghomebattingorder**

The original batting order for home team, a list of last names in proper case.

**awaybattingorder**

The updated batting order for the away team after substitution.

**homebattingorder**

The updated batting order for the home team after substitution.

**awayoffsubs**

Offensive subs for the away team, a list of last names in proper case.

**homeoffsubs**

Offensive subs for the home team, a list of last names in proper case.

**awayreliefers**

Relief pitchers for the away team, a list of last names in proper case.

**homereliefers**

Relief pitchers for the home team, a list of last names in proper case.

**offense**

A boolean, True if substitution affects a batting order for one of the teams, False otherwise.

**playerin**

The last name of the player subbing in, in proper case.

**playerout**

The last name of the player coming out, in proper case, or 'NA' if none.

**subpos**

The position being subbed ("rf","p",...).

**subhole**

An integer, the subbed player's place in their team's batting order (list index + 1).

**get\_players()**  
Finds player names in unit string and sets playerout and playerin.

**get\_pos()**  
Determines position that is being substituted.

**get\_offense()**  
Determines if substitution requires change to a batting order.

**get\_hole()**  
Determines subbed player's place in batting order.

**mod\_batting\_order()**  
Modifies batting order based on substitution.

**do\_stuff()**  
Runs all SubUnit functions in the correct order.

**get\_updated\_bo()**  
Returns post-substitution batting order lists for both teams.

**print\_stuff()**  
Prints all SubUnit information for testing purposes.

**get\_data\_string()**  
Returns csv string of all variables generated by SubUnit class.

**ActionUnit.get\_bat\_data\_header()**  
Returns a comma-separated header string with variable names for batting dataset.

**ActionUnit.get\_br\_data\_header()**  
Returns a comma-separated header string with variable names for baserunning dataset.

**ActionUnit.get\_sub\_data\_header()**  
Returns a comma-separated header string with variable names for substitution dataset.



## STANDARDIZING PLAYER NAME FORMATS

### 6.1 StdzNames – Name Format Standardization

`StdzNames.last_first (name_string)`

Processes raw name string and returns names *Last, First* in proper case.

`StdzNames.last_only (name_string)`

Processes raw name string and returns last name in proper case.

`StdzNames.first_only (name_string)`

Processes raw name string and returns first name in proper case.

`StdzNames.lasts_firsts (name_list)`

Takes list of raw name strings and returns *LastList, FirstList*, lists of last names and first names in proper case.

`StdzNames.lasts_only (name_list)`

Takes list of raw name strings and returns list of last names in proper case.

`StdzNames.firsts_only (name_list)`

Takes list of raw name strings and returns list of first names in proper case.





## ASSIGNING CODES TO BASE-OUT STATES

### 7.1 BaseOut – Base-out state coding functions

BaseOut.**get\_base\_state** (*on\_first*, *on\_second*, *on\_third*)

Returns alpha code for base state in the range { 'A','B','C',...,'H' }.

- *on\_first*, *on\_second*, *on\_third* - String indicating runner name or NA if base is empty.

Code	Base Configuration
A	___
B	__1
C	_2_
D	3__
E	_21
F	3_1
G	32_
H	321

BaseOut.**get\_base\_out\_state** (*outs*, *on\_first*, *on\_second*, *on\_third*)

Returns alpha-numeric code for base-out state in the range { 'A0',...,'A3',...,'H0',...,'H3' }.

- *outs* - An int or string indicating number of outs.
- *on\_first*, *on\_second*, *on\_third* - String indicating runner name or NA if base is empty.



## STORING MINER'S OUTPUT IN CSV

### 8.1 `CsvStuff` – CSV string and file functions

`CsvStuff.csv_print` (*data\_list*)

Prints a list of any type as a string of comma-separated values.

`CsvStuff.make_csv_header` (*var\_list*)

Takes a list of variable names and returns a comma-separated header string.

`CsvStuff.make_csv_row` (*data\_list*)

Takes a list of any type and returns a comma-separated string of values.

`CsvStuff.make_csv_file` (*file\_name*, *header\_string*)

Creates a new CSV file with a header of variable names.

`CsvStuff.add_row_csv_file` (*file\_name*, *csv\_row\_string*)

Adds a row of data to an existing csv file.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### **a**

ActionUnit, 9

### **b**

BaseOut, 21

### **c**

CsvStuff, 23

### **g**

Game, 3

### **l**

Lineup, 5

### **p**

PlateAppearance, 7

### **s**

SoupChef, 1

StdzNames, 19

- actionBRUnit attribute, 13
- ActionUnitmodule, 9
- actionunitsPlateAppearance.PlateAppearance attribute, 7
- add\_row\_csv\_file()in module CsvStuff, 23
- advancedBRUnit attribute, 14
- aoeBRUnit attribute, 15
- attendanceGame.Game attribute, 3
- auhalfinningsPlateAppearance.PlateAppearance attribute, 7
- aupaidPlateAppearance.PlateAppearance attribute, 7
- ausidesPlateAppearance.PlateAppearance attribute, 7
- autypesPlateAppearance.PlateAppearance attribute, 7
- awaybattingorderSubUnit attribute, 16
- awaynamesGame.Game attribute, 4
- awayoffsubsLineup.Lineup attribute, 6
- awayoffsubsSubUnit attribute, 16
- awayorderLineup.Lineup attribute, 6
- awaypositionsGame.Game attribute, 3
- awayrelieversLineup.Lineup attribute, 5
- awayrelieversSubUnit attribute, 16
- awaystarterposLineup.Lineup attribute, 5
- awaystartersLineup.Lineup attribute, 5
- awaystartsubGame.Game attribute, 4
- awaysubposLineup.Lineup attribute, 5
- awaysubsLineup.Lineup attribute, 5
- ballsBatUnit attribute, 10
- BaseOutmodule, 21
- basesBatUnit attribute, 10
- basesaddedBatUnit attribute, 12
- basesaddedBRUnit attribute, 14
- batter\_nameBatUnit attribute, 9
- BatUnitbuilt-in class, 9
- BatUnit.do\_stuff()built-in function, 13
- BatUnit.get\_baseout\_end()built-in function, 13
- BatUnit.get\_batter\_name()built-in function, 12
- BatUnit.get\_count()built-in function, 12
- BatUnit.get\_data\_string()built-in function, 13
- BatUnit.get\_double\_play()built-in function, 12
- BatUnit.get\_field\_out()built-in function, 12
- BatUnit.get\_hit()built-in function, 12
- BatUnit.get\_hit\_info()built-in function, 12
- BatUnit.get\_pitches()built-in function, 12
- BatUnit.get\_reach()built-in function, 12
- BatUnit.get\_so\_type()built-in function, 13
- BatUnit.get\_walk\_type()built-in function, 12
- BatUnit.is\_bip()built-in function, 12
- BatUnit.is\_bunt()built-in function, 12
- BatUnit.is\_field\_out()built-in function, 12
- BatUnit.is\_hbp()built-in function, 13
- BatUnit.is\_hit()built-in function, 12
- BatUnit.is\_reach()built-in function, 12
- BatUnit.is\_sac()built-in function, 12
- BatUnit.is\_sf()built-in function, 12
- BatUnit.is\_strikeout()built-in function, 13
- BatUnit.is\_true\_outcome()built-in function, 13
- BatUnit.is\_walk()built-in function, 12
- BatUnit.print\_stuff()built-in function, 13
- BatUnit.update\_bod()built-in function, 13
- BBBatUnit attribute, 11
- BIPBatUnit attribute, 10
- bip\_verbsBatUnit attribute, 9
- BIPsBatUnit attribute, 10
- boBatUnit attribute, 9
- boBRUnit attribute, 13
- BRUnitbuilt-in class, 13
- BRUnit.do\_stuff()built-in function, 15
- BRUnit.do\_stuff\_batbr()built-in function, 15
- BRUnit.get\_action()built-in function, 15
- BRUnit.get\_advanced()built-in function, 15
- BRUnit.get\_baseout\_end()built-in function, 15
- BRUnit.get\_bases\_added()built-in function, 15
- BRUnit.get\_data\_string()built-in function, 15
- BRUnit.get\_scored()built-in function, 15
- BRUnit.get\_start\_base()built-in function, 15
- BRUnit.get\_start\_base\_batbr()built-in function, 15
- BRUnit.get\_stole()built-in function, 15
- BRUnit.is\_pickoff()built-in function, 15
- BRUnit.print\_stuff()built-in function, 15
- BRUnit.update\_bsot()built-in function, 15
- bsotendBatUnit attribute, 9
- bsotendBRUnit attribute, 13
- bsotstartBatUnit attribute, 9
- bsotstartBRUnit attribute, 13
- calledstrikesBatUnit attribute, 10



cityGame.Game attribute, 3  
 count\_ballsBatUnit attribute, 10  
 count\_strikesBatUnit attribute, 10  
 csBRUnit attribute, 14  
 csv\_print()in module CsvStuff, 23  
 CsvStuffmodule, 23  
  
 dayGame.Game attribute, 3  
 daynightGame.Game attribute, 3  
 doubleplaytypeBatUnit attribute, 11  
 doublesBatUnit attribute, 10  
 DPBatUnit attribute, 11  
 durationGame.Game attribute, 3  
  
 endbaseBRUnit attribute, 14  
  
 FCBatUnit attribute, 11  
 fielder1BRUnit attribute, 14  
 fielder2BRUnit attribute, 14  
 fieldoutsBatUnit attribute, 11  
 first\_only()in module StdzNames, 19  
 firsts\_only()in module StdzNames, 19  
 fo\_locationsBatUnit attribute, 9  
 fo\_verbsBatUnit attribute, 9  
 folocBatUnit attribute, 11  
 foqualBatUnit attribute, 11  
 fotypeBatUnit attribute, 11  
 foulsBatUnit attribute, 10  
 fpoBRUnit attribute, 14  
  
 Gameclass in Game, 3  
 Gamemodule, 3  
 Game.do\_stuff()in module Game, 4  
 Game.get\_action\_units()in module Game, 4  
 Game.get\_day\_night()in module Game, 4  
 Game.get\_game\_duration()in module Game, 4  
 Game.get\_game\_info()in module Game, 4  
 Game.get\_game\_site()in module Game, 4  
 Game.get\_lineup\_names()in module Game, 4  
 Game.get\_lineup\_positions()in module Game, 4  
 Game.get\_month\_day\_year()in module Game, 4  
 Game.print\_stuff()in module Game, 4  
 get\_base\_out\_state()in module BaseOut, 21  
 get\_base\_state()in module BaseOut, 21  
 get\_bat\_data\_header()in module ActionUnit, 17  
 get\_br\_data\_header()in module ActionUnit, 17  
 get\_file\_info()in module SoupChef, 1  
 get\_sub\_data\_header()in module ActionUnit, 17  
  
 HBatUnit attribute, 10  
 HBPBatUnit attribute, 12  
 hit\_verbsBatUnit attribute, 9  
 hitlocBatUnit attribute, 10  
 hitqualBatUnit attribute, 10  
 hittypeBatUnit attribute, 10  
  
 holeBatUnit attribute, 10  
 holeBRUnit attribute, 14  
 homebattingorderSubUnit attribute, 16  
 homenamesGame.Game attribute, 4  
 homeoffsubsLineup.Lineup attribute, 6  
 homeoffsubsSubUnit attribute, 16  
 homeorderLineup.Lineup attribute, 6  
 homepositionsGame.Game attribute, 3  
 homerelieversLineup.Lineup attribute, 5  
 homerelieversSubUnit attribute, 16  
 homersBatUnit attribute, 10  
 homestarterposLineup.Lineup attribute, 5  
 homestartersLineup.Lineup attribute, 5  
 homestartsubGame.Game attribute, 4  
 homesubposLineup.Lineup attribute, 5  
 homesubsLineup.Lineup attribute, 5  
  
 IBBBatUnit attribute, 11  
 infhBatUnit attribute, 11  
 is\_rchBatUnit attribute, 11  
  
 klookBatUnit attribute, 11  
 kswingBatUnit attribute, 11  
  
 last\_first()in module StdzNames, 19  
 last\_only()in module StdzNames, 19  
 lasts\_firsts()in module StdzNames, 19  
 lasts\_only()in module StdzNames, 19  
 Lineupclass in Lineup, 5  
 Lineupmodule, 5  
 Lineup.do\_stuff()in module Lineup, 6  
 Lineup.get\_batting\_orders()in module Lineup, 6  
 Lineup.get\_data\_string()in module Lineup, 6  
 Lineup.get\_starters\_subs()in module Lineup, 6  
 Lineup.get\_subs()in module Lineup, 6  
 Lineup.print\_stuff()in module Lineup, 6  
  
 make\_csv\_file()in module CsvStuff, 23  
 make\_csv\_header()in module CsvStuff, 23  
 make\_csv\_row()in module CsvStuff, 23  
 monthGame.Game attribute, 3  
  
 off\_posSubUnit attribute, 16  
 offenseSubUnit attribute, 16  
 ogawaybattingorderSubUnit attribute, 16  
 oghomebattingorderSubUnit attribute, 16  
 open\_soup()in module SoupChef, 1  
 outBRUnit attribute, 14  
 outatbaseBRUnit attribute, 15  
 outsaddedBatUnit attribute, 12  
 outsaddedBRUnit attribute, 14  
  
 parkGame.Game attribute, 3  
 pasideGame.Game attribute, 4  
 paunitidsGame.Game attribute, 4

paunitsGame.Game attribute, 4  
pbBRUnit attribute, 14  
pitch\_stringBatUnit attribute, 10  
pitchesBatUnit attribute, 10  
PlateAppearanceclass in PlateAppearance, 7  
PlateAppearancemodule, 7  
PlateAppearance.do\_stuff()in module PlateAppearance, 7  
PlateAppearance.get\_action\_units()in module PlateAppearance, 7  
PlateAppearance.get\_au\_innings()in module PlateAppearance, 7  
PlateAppearance.get\_au\_types()in module PlateAppearance, 7  
PlateAppearance.print\_stuff()in module PlateAppearance, 7  
playerinSubUnit attribute, 16  
playeroutSubUnit attribute, 16  
poBRUnit attribute, 14  
poaBRUnit attribute, 14  
  
RBOEBatUnit attribute, 11  
runnenameBRUnit attribute, 14  
runsaddedBatUnit attribute, 12  
runsaddedBRUnit attribute, 14  
  
SACBatUnit attribute, 11  
sbBRUnit attribute, 14  
sbaBRUnit attribute, 14  
scoredBRunit attribute, 14  
SFBatUnit attribute, 11  
sideBRUnit attribute, 13  
singlesBatUnit attribute, 10  
SOBatUnit attribute, 11  
SOreachBatUnit attribute, 11  
SoupChefmodule, 1  
startbaseBRUnit attribute, 14  
starttimeGame.Game attribute, 3  
stateGame.Game attribute, 3  
StdzNamesmodule, 19  
stoleBRUnit attribute, 14  
subholeSubUnit attribute, 16  
subposSubUnit attribute, 16  
SubUnitbuilt-in class, 16  
SubUnit.do\_stuff()built-in function, 17  
SubUnit.get\_data\_string()built-in function, 17  
SubUnit.get\_hole()built-in function, 17  
SubUnit.get\_offense()built-in function, 17  
SubUnit.get\_players()built-in function, 16  
SubUnit.get\_pos()built-in function, 17  
SubUnit.get\_updated\_bo()built-in function, 17  
SubUnit.mod\_batting\_order()built-in function, 17  
SubUnit.print\_stuff()built-in function, 17  
swingingstrikesBatUnit attribute, 10  
  
teamBatUnit attribute, 9  
  
triplesBatUnit attribute, 10  
trueoutcomeBatUnit attribute, 11  
  
UBBBatUnit attribute, 11  
UCTSBatUnit attribute, 11  
umpiresGame.Game attribute, 3  
unit\_stringBatUnit attribute, 9  
unitstringBRUnit attribute, 13  
  
weatherGame.Game attribute, 3  
wpBRUnit attribute, 14  
  
yearGame.Game attribute, 3