python-elections Documentation

Release 0.4

Los Angeles Times Data Desk

October 08, 2014

A Python wrapper for the Associated Press's U.S. election data service

CHAPTER 1

Features

- Easy access to AP election results for races in U.S. states and counties.
- Easy access to presidential election results for the nation, the 50 states and all U.S. counties.
- Easy access to AP's delegate estimates in presidential nomination contests.

Requirements

In order to use this library, you must pay AP for access to the data. More information can be found on the AP's web site or by contacting Anthony Marquez at amarquez@ap.org.

CHAPTER 3

In the wild

- Los Angeles Times, again and again
- Chicago Tribune, again and again
- Washington Times
- Palm Beach Post and again
- Tampa Bay Times
- Spokane Spokesman-Review and again
- Atlanta Journal-Constitution
- WYNC
- DJournal.com

CHAPTER 4

Documentation

4.1 Getting Started

Provided that you have pip installed, you can install the library like so.

```
$ pip install python-elections
```

If you'd rather work on the source code, you can clone it from GitHub using the usual methods.

```
$ git clone https://github.com/datadesk/python-elections.git
```

4.1.1 Creating a client

Before you can interact with AP's data, you first must import the library and initialize a client to talk with the FTP on your behalf.

```
>>> from elections import AP
>>> client = AP(USERNAME, PASSWORD)
```

4.1.2 Some basics

Request all the data available for a particular state by providing its postal code. This will return a state object.

```
>>> ca = client.get_state('CA')
>>> ca
<State: CA>
```

Among other things, the state has a list of races.

```
>>> ca.races
[<Race: District Attorney Los Angeles>, <Race: U.S. House District 49 - Carlsbad>, <Race: U.S. House
>>> prez = ca.filter_races(office_name='President')[0]
>>> prez
<Race: President>
```

The race contains a list of candidates.

```
>>> prez.candidates
[<Candidate: Roseanne Barr>, <Candidate: Thomas Hoefling>, <Candidate: Gary Johnson>, <Candidate: Ba
```

You can find results for the whole state.

```
>>> prez.state.results
[<Result: Barack Obama, California (state), 3698001>, <Result: Mitt Romney, California (state), 34693
```

You can get all counties in the state.

```
>>> prez.counties
[<ReportingUnit: Alameda>, <ReportingUnit: Alpine>, <ReportingUnit: Amador>, <ReportingUnit: Butte>,
```

And, of course, the results in each county.

```
>>> prez.counties[0].results
[<Result: Barack Obama, Alameda, 160048>, <Result: Mitt Romney, Alameda, 152934>, <Result: Roseanne B
```

4.1.3 A working example

Let's say, hypothetically, that the United States is electing a president for the next four years, and your news organization bought access to the AP's FTP service for California results. Your boss wants you to write a simple widget that will sit on the homepage and output live results. All you need are the candidate names, their vote totals and percentages, the number of precincts reporting and whether the AP has called a winner yet. How do you feed it? Here's how.

```
from elections import AP
try:
    import json
except ImportError:
    import simplejson as json

client = AP(uname, pwd)
calif = client.get_state('CA')
# Now the calif variable holds all of the AP result data
prez = iowa.filter_races(office_name='President')[0]
# prez is a Race object containing the results of the presidential race

# Set up the main data dict and set the percent of precincts reporting
data = {
    'precincts_reporting_percent': prez.state.precincts_reporting_percent,
    'candidates': []
}

# Loop through the statewide candidate results, and append them
# in a format we like into the data dict's candidate list.
for result in prez.state.results:
    data['candidates'].append({
        'name': result.candidate.last_name,
        'vote_total': result.vote_total,
        'vote_percent': result.vote_total_percent,
        'is_winner': result.candidate.is_winner,
    })

# Then dump the data dict out as JSON
print json.dumps(data, indent=4)
```

There you have it: a simple JSON dump in about 20 lines of code. From here, you can set this script to upload the JSON file every few minutes to Amazon S3 or a similar file-serving service. Then point your front-end widget to pull from there.

4.2 The Associated Press client

4.2.1 Retrieving data

The AP client is public class you can use to connect to the AP's data feed.

`client.get_state(state_postal_code)`

Takes a single state postal code, returns that state's results.

```
>>> from elections import AP
>>> client = AP(USERNAME, PASSWORD)
>>> client.get_state('IA')
<State: IA>
```

`client.get_states(*state_postal_codes)`

Takes one to many state postal codes as arguments, returns a list of results for the requested states.

```
>>> from elections import AP
>>> client = AP(USERNAME, PASSWORD)
>>> client.get_states('IA', 'NH')
[<State: IA>, <State: NH>]
```

`client.get_topofticket(election_date)`

Top of the ticket is an AP data service that provides limited results on the top races for all 50 states (i.e. President, Governor, US Senate, and US House). It requires a date in any common format, YYYY-MM-DD is preferred, and returns all results for that date.

If you do not provide a date, it defaults to the next major election. Today that is the Nov. 6, 2012 general election.

```
>>> from elections import AP
>>> client = AP(USERNAME, PASSWORD)
>>> client.get_topofticket('2012-02-07')
<TopOfTicket: 20120207>
```

`client.get_presidential_summary(districts=False)`

Returns a summary of presidential election results at three levels: nationwide popular vote and electoral vote; state-level popular vote and electoral vote; county-level popular vote.

If *districts* is provided and set to True the results will include Congressional district-level results in the two states that break out their presidential electors: Maine and Nebraska. This feature only works if the AP has given your account access to the ME and NE data folders. By default, *districts* is set to False.

```
>>> from elections import AP
>>> client = AP(USERNAME, PASSWORD)
>>> client.get_presidential_summary()
<PresidentialSummary: None>
```

`client.get_delegate_summary()`

Return a nationwide summary and state-level totals contain delegate counts for all the candidates in the presidential nomination contest held by the two major parties.

Warning: This method does not currently work because the 2012 primaries are over and the AP has removed the folders it depends on.

```
>>> from elections import AP
>>> client = AP(USERNAME, PASSWORD)
>>> client.documents.get_delegate_summary()
[<Nomination: Dem>, <Nomination: GOP>]
```

4.2.2 Election result collections

Depending on which client method you use to harvest data, results may be returned as *State* or *TopOfTicket* objects. Don't worry about the distinction, because they act pretty much the same. They share the following attributes for you to use.

obj.**counties**

Returns a list of all the counties from the pool of reporting units.

```
>>> obj = client.get_state('IA')
>>> obj.counties
[<ReportingUnit: Guthrie>, <ReportingUnit: Union>, <ReportingUnit: Crawford>, <ReportingUnit: Wr
```

obj.**filter_races**(***kwargs*)

Takes a series of keyword arguments and returns any races that match.

```
>>> obj = client.get_state('IA')
>>> obj.filter_races(office_name='President', party='GOP')
[<Race: GOP Caucus - President>]
```

obj.**races**

Returns a list of all the races reporting results.

```
>>> obj = client.get_state('IA')
>>> obj.races
[<Race: GOP Caucus - President>]
```

obj.**reporting_units**

Returns a list of all reporting units in the result collection.

```
>>> obj = client.get_state("IA")
>>> obj.reporting_units
[<ReportingUnit: Guthrie>, <ReportingUnit: Union>, <ReportingUnit: Crawford>, <ReportingUnit: Wr
```

obj.**states**

Returns a list of all the states from the pool of reporting units. Only available on *TopOfTicket* result collections.

```
>>> obj = client.get_topofticket('2012-02-07')
>>> obj.states
[<ReportingUnit: Missouri (state)>, <ReportingUnit: Minnesota (state)>, <ReportingUnit: Colorado
```

Races

A contest being decided by voters choosing between candidates. This object is the key to everything about it. It is often found in the *races* attribute of a result collection.

obj.**ap_race_number**

AP-assigned race number. Race numbers are guaranteed to be unique only within a state.

```
>>> obj.ap_race_number
'16957'
```

obj.**candidates**

The list of candidates participating in the race.

```
>>> obj.candidates
[<Candidate: Michele Bachmann>, <Candidate: Herman Cain>, <Candidate: Newt Gingrich>, <Candidate
```

`obj.`**`counties`**

Returns all the counties that report results for this race as a list.

```
>>> obj.counties
[<ReportingUnit: Adair>, <ReportingUnit: Adams>, <ReportingUnit: Allamakee>, <ReportingUnit: App
```

`obj.`**`date`**

The date of the election in Python's datetime format.

```
>>> obj.date
datetime.date(2012, 1, 3)
```

`obj.`**`is_primary`**

Returns *True* if the race is a primary.

`obj.`**`is_caucus`**

Returns *True* if the race is a caucus.

`obj.`**`is_general`**

Returns *True* if the race is part of a general election.

`obj.`**`name`**

The name of the race.

```
>>> obj.name
'GOP Caucus - President'
```

`obj.`**`num_winners`**

Integer giving the maximum number of winners.

```
>>> obj.num_winners
1
```

`obj.`**`office_name`**

Character string for office name (e.g., U.S. House, Governor, etc.)

```
>>> obj.office_name
'President'
```

`obj.`**`office_description`**

Character string further describing the office type. May be empty.

`obj.`**`office_id`**

Single character Office Type ID. Only top-of-the-ticket races (President, Governor, US Senate, and US House) are guaranteed to be unique on a national level. All other office types are guaranteed to be unique only within a state. A full list of the office identifiers can be found in AP's documentation.

`obj.`**`party`**

Name of party to which race applies, i.e., GOP if a Republican Primary.

`obj.`**`race_type_name`**

Returns a descriptive name for the race_type.

```
>>> obj.race_type_name
'GOP Caucus'
```

`obj.`**`reporting_units`**

Returns all reporting units that belong to this race as a list.

```
>>> obj.reporting_units
[<ReportingUnit: Guthrie>, <ReportingUnit: Union>, <ReportingUnit: Crawford>, <ReportingUnit: Wr
```

`obj.`**scope**
Office scope – whether the race is a Local (L) or Statewide (S) race

```
>>> obj.scope
'S'
```

`obj.`**state**
Returns the state-level results for this race as a ReportingUnit object.

```
>>> obj.state
<ReportingUnit: Iowa (state)>
```

`obj.`**seat_name**
Character string giving the district or initiative name (e.g., District 46, 1A-Gay Marriage, etc.) This may be empty for a statewide race (e.g., a Governor race).

`obj.`**seat_number**
Integer indicating district number or an initiative number. This may be zero (0) for a statewide race.

`obj.`**state_postal**
Two character state postal string (e.g., IA, LA, etc.).

`self.`**uncontested**
Returns *True* is the race is uncontested.

`self.`**is_referendum**
Returns *True* if this is a race where the people vote to decide about a law, measure, proposition, amendment, etc.

Reporting units

An area or unit that groups votes into a total. For instance, a state, a congressional district, a county.

`obj.`**abbrev**
Short Name of reporting unit

```
>>> obj.abbrev
'Poweshiek'
```

`obj.`**ap_number**
Unique ID within a state for reporting unit.

```
>>> obj.ap_number
'16079'
```

`obj.`**name**
The full name of the reporting unit

```
>>> obj.name
'Poweshiek'
```

`obj.`**fips**
The unique FIPS code for this reporting unit, assigned by the U.S. government.

```
>>> obj.fips
'19157'
```

`obj.`**num_reg_voters**
The number of registered votes who live in this reporting unit.

```
>>> obj.num_reg_voters
3897
```

`obj.`**`votes_cast`**

The number of votes cast in this reporting unit.

```
>>> obj.votes_cast
709
```

`obj.`**`precincts_total`**

The number of voting precincts in this reporting unit.

```
>>> obj.precincts_total
10
```

`obj.`**`precincts_reporting`**

The number of precincts that have already provided results.

```
>>> obj.precincts_reporting
10
```

`obj.`**`precincts_reporting_percent`**

The percentage of precincts that have already provided results.

```
>>> obj.precincts_reporting_percent
100.0
```

`obj.`**`results`**

Returns a list of result objects sorted by total votes (highest first). If no votes are in, it returns the candidates in alphabetical order.

```
>>> obj.results
[<Result: Rick Santorum, Iowa (state), 29839>, <Result: Mitt Romney, Iowa (state), 29805>, <Resu
```

`obj.`**`is_state`**

Returns *True* if the reporting unit is a state, rather than some other unit like a county.

`obj.`**`electoral_votes_total`**

Returns the number of presidential electors this area controls. Typically only found on states.

Candidates

A choice for voters in a race. In the presidential race, a person, like Barack Obama. In a ballot measure, a direction, like Yes or No.

`obj.`**`abbrev_name`**

Candidate's abbreviated name, usually last name with some vowels removed if too long.

```
>>> obj.abbrev_name
'Bchmnn'
```

`obj.`**`ap_natl_number`**

Unique ID to identify this politician across states and races.

```
>>> obj.ap_natl_number
'302'
```

`obj.`**`ap_pol_number`**

Unique ID within a state for this candidate.

```
>>> obj.ap_pol_number
'18538'
```

obj.**ap_polra_number**
Unique ID within a state for this candidate for this race for their party.

```
>>> obj.ap_polra_number
'21304'
```

obj.**ap_race_number**
Unique ID within a state for the race object this candidate object is linked to.

```
>>> obj.ap_race_number
'16957'
```

obj.**delegates**
The number of delegates the candidate has won in this state, according to AP's estimates. Warning: AP has told The Times that it stops updating these totals after they decide a race has "closed" following the election. That means that if you want to track changes to these totals between the vote and the eventual nomination, you should use the nationwide delegate methods detailed below.

```
>>> obj.delegates
0
```

obj.**first_name**
The first name of the candidate.

```
>>> obj.first_name
'Michele'
```

obj.**is_incumbent**
Returns *True* if the candidate is the current officeholder.

obj.**is_winner**
Returns *True* if the candidate has won the race.

obj.**is_runoff**
Returns *True* is the candidate is advancing to a runoff.

obj.**last_name**
The last name of the candidate.

```
>>> obj.last_name
'Bachmann'
```

obj.**middle_name**
The middle name of the candidate. Might not always exist.

```
>>> obj.middle_name
'J.'
```

obj.**name**
The full name of candidate.

```
>>> obj.name
u'Michele Bachmann'
```

obj.**party**
Candidate's party abbreviation.

```
>>> obj.party
'GOP'
```

obj.**suffix**
The suffix to the candidate's name. Might not exist.

```
>>> obj.suffix
'Jr.'
```

obj.**use_suffix**
Returns *True* if you should use the suffix with the name.

Result

The vote count for a candidate in a race in a particular reporting unit.

obj.**candidate**
The candidate this result is for.

```
>>> obj.candidate
<Candidate: Rick Santorum>
```

obj.**reporting_unit**
The reporting unit this result is for.

```
>>> obj.reporting_unit
<ReportingUnit: Iowa (state)>
```

obj.**vote_total**
The number of votes the candidate has collected in this reporting unit.

```
>>> obj.vote_total
29839
```

obj.**vote_total_percent**
The percentage of the tpta; votes the candidate has collected in this reporting unit.

```
>>> obj.vote_total_percent
24.558645607855077
```

obj.**electoral_votes_total**
Returns the number of presidential electors awarded by this result.

4.2.3 Presidential summary collections

Calling presidential methods, like *get_presidential_summary* will return a slightly different, and simpler, result collection.

obj.**nationwide**
Returns only the nationwide reporting unit.

```
>>> obj.nationwide
<ReportingUnit: US>
```

obj.**states**
Returns only the state-level reporting units

```
>>> obj.states
[<ReportingUnit: South Carolina (state)>, <ReportingUnit: North Carolina (state)>, <ReportingUni
```

obj.**counties**
Returns only the county-level reporting units

```
>>> obj.counties
[<ReportingUnit: Abbeville>, <ReportingUnit: Aiken>, <ReportingUnit: Allendale>, <ReportingUnit:
```

obj.**districts**
Returns only Congressional district-level results in the two states that break out their presidential electors: Maine and Nebraska. This feature only works if *districts* is set to True and passed into the *get_presidential_summary* model.

```
>>> prez = client.get_presidential_summary(districts=True)
>>> prez.districts
[<ReportingUnit: ME District 2>, <ReportingUnit: ME District 1>, <ReportingUnit: NE District 2>,
```

4.2.4 Delegate summary collections

Calling delegate related methods, like *get_delegate_summary* will return a slightly different, and simpler, result collection. To start, you should receive a list containing two Nomination objects.

Nominations

A contest to be the presidential nominee of one of the two major parties.

obj.**candidates**
The list of candidates participating in the race.

```
>>> obj.candidates
[<Candidate: Michele Bachmann>, <Candidate: Herman Cain>, <Candidate: Newt Gingrich>, <Candidate
```

obj.**delegates_needed**
The number of delegates needed to capture the nomination.

obj.**delegates_total**
The total number of delegates available.

obj.**delegates_chosen**
The total number of delegates that have been awarded.

obj.**delegates_chosen_percent**
The percentage of the total delegates that have been awarded.

obj.**party**
Candidate’s party abbreviation.

```
>>> obj.party
'GOP'
```

obj.**states**
Returns a list of all the state delegates we have counts for.

```
>>> obj.states
[<StateDelegation: AK>, <StateDelegation: AL>, <StateDelegation: AR>, <StateDelegation: AS>, <St
```

State delegations

A state's delegation and who they choose to be a party's presidential nominee.

`obj.`**`candidates`**
The list of candidates participating in the race.

```
>>> obj.candidates
[<Candidate: Michele Bachmann>, <Candidate: Herman Cain>, <Candidate: Newt Gingrich>, <Candidate
```

`obj.`**`name`**
The name of the state. The AP only provides the postal code.

```
>>> obj.name
'IA'
```

4.3 Changelog

4.3.1 1.0.0

- PEP8 and Pyflakes compliance and testing
- Proper semantic versioning

4.3.2 0.4

- Bug fix for Virgina elections

4.3.3 0.3

- Added delegate summary method thanks to contributions of David Eads.

4.3.4 0.2

- Added *get_topofticket* methods thanks to contributions by Corey Oordt.

4.3.5 beta

- Added all the basic features for the first release

4.4 Authors

- Ken Schwencke
- Ben Welsh
- Corey Oordt
- David Eads

CHAPTER 5

Contributing

- Code repository: https://github.com/datadesk/python-elections
- Issues: https://github.com/datadesk/python-elections/issues
- Packaging: https://pypi.python.org/pypi/python-elections

Index