pyArtifact Documentation

David Jetelina

Sep 17, 2019

Contents:

Pythonic wrapper around Valve's Artifact API

CHAPTER 1

Library overview

There are 2 main functions of pyArtifact

- Wrap Valve's card API with easier to work with pythonic objects
- Wrap deck code API to enable encoding a decoding

1.1 Card API usage

First step will be loading all the cards:

```
from pyartifact import Cards
cards = Cards()
cards.load_all_sets()
```

This enables you to use 3 methods to search and filter cards.

First of all there's filter, for example if you want to find blue spells that cost less than 3 mana:

```
filtered = cards.filter.type('Spell').color('Blue').mana_cost(lt=3)
# To see how many cards we found
len(filtered)
# To see the names of the found cards
for card in filtered.cards:
    print(card.name)
```

If you know what you're looking for, you can simply get it:

```
# Get the card instance by the cards name
storm_spirit = cards.get('Storm Spirit')
# Play around with it!
print(f'Storm spirit has {storm_spirit.attack} attack and {storm_spirit.hit_points}␣
↪health.')
print(f"When you put him into your deck, he brings the spell '{storm_spirit.
↪includes[0].name}' with him.")
```

1.2 Deck code API

pyArtifact offers two approaches to deck encoding and decoding. If you want to use the card objects showcased above, you can use the *Deck* object:

```
from pyartifact import Deck
deck = Deck.from_code(
↪"ADCJQUQI30zuwEYg2ABeF1Bu94BmWIBTEkLtAKlAZakAYmHh0JsdWUvUmVkIEV4YW1wbGU_")
# or alternatively
deck = Deck.loads(
↪"ADCJQUQI30zuwEYg2ABeF1Bu94BmWIBTEkLtAKlAZakAYmHh0JsdWUvUmVkIEV4YW1wbGU_")
# And done!
print(len(deck.overview.items()))  # It's 9. The deck has 9 items in it
# You can now edit it
deck.name = 'Renamed deck'
# And turn back into a deck code
print(deck.deck_code)  # Or str(deck), or deck.dumps(), so you have your options open.
```

To use all this, you need to have all the existing sets loaded with the Card API, as it's enriching the data with the instances of the cards, for easier manipulation. If that is something you don't want and you'd just like to use the encode and decode functions, pyArtifact has your back:

```
from pyartifact import decode_deck_string, encode_deck
deck_data = decode_deck_string(
↪"ADCJQUQI30zuwEYg2ABeF1Bu94BmWIBTEkLtAKlAZakAYmHh0JsdWUvUmVkIEV4YW1wbGU_")
deck_string = encode_deck(deck_data)
```

CHAPTER 2

Changelog

2.1 Unreleased

2.1.1 New Features

- `Cards` can now act as an iterable
- `api_sync.Cards.find()` was implemented

2.1.2 Other Notes

- Future proofed getting localized text. Language -> default -> unknown

2.2 0.3.2

2.2.1 New Features

- With Valve publishing the translations, pyArtifact isn't far behind. While the support for card text was already implemented, large images were always in english. That is now not the case.
- Localize parameter in `Cards` will now turn the language into lowercase for you.

2.2.2 Bug Fixes

- Previously it was impossible to get some cards by name, because they shared it with an ability. This is now fixed, for more read the updated documentation of `get()` The affected cards were:
  - Sister of the Veil
  - Rebel Decoy

- Mercenary Exiles
- Emissary of the Quorum
- Ravenhook
- Ravenous Mass
- Assassin's Apprentice
- Satyr Magician
- Unsupervised Artillery
- Revtel Investments
- Aghanim's Sanctum
- Escape Route
- Messenger Rookery
- Keenfolk Turret
- Steam Cannon
- Assassin's Veil
- Phase Boots
- Blink Dagger
- Demagicking Maul
- Rumusque Vestments
- Keenfolk Musket
- Bracers of Sacrifice
- Helm of the Dominator
- Wingfall Hammer
- Book of the Dead
- Shiva's Guard
- Horn of the Alpha
- Nyctasha's Guard
- Apotheosis Blade

2.3 0.3.0

2.3.1 Prelude

A simple Deck wrapper has been designed and is ready to be tested :)

2.3.2 New Features

- *Deck* has been 'done' (still prototype version). To use it, all sets have to be loaded.
- *deck.Overview* has been added as a property of the deck. It offers a quick glance over the contents of the deck
- *filtering.CardFilter* now has a *sub_type* filter (no *sub_type_in* and *sub_type_not_in* variants just yet)
- *filtering.CardFilter* now works internally with list of cards instead of a set, to be able to filter through deck contents etc.

2.3.3 Bug Fixes

- *decode_deck_string()* and *encode_deck()* have been refactored and very well documented and commented. It should now be fairly easy to understand!

2.3.4 Other Notes

- Mypy is an a** and we won't be working with him again. Killer of productivity that only complicated code. His paws might still be seen here and there, if you see a weird piece of code that could be simplified if it didn't require mypy to pass, feel free to point it out. (Typing is still useful and this library should stay pycharm compatible, no warnings raised there!)
- Tons of docstrings were added in preparation for sphinx autodoc

2.4 0.2.0

2.4.1 Prelude

Deck encoding is now supported

2.4.2 New Features

- *decode_deck_string()* and *encode_deck()* are now available, to encode and decode deck strings. It's wrapper

2.4.3 Known Issues

- Encoding and decoding isn't very readable at the moment and could use a ton of pythonization ;)
- *Deck* - a wrapper for encoding and decoding still needs tons of design work. The question is whether this library should be a deck building tool, or just a lightweight wrapper, more focused on reporting the decks and more 'low level' edits through dictionaries that are used internally.

CHAPTER 3

The main Cards object

class pyartifact.api_sync.**Cards**(*limit_sets=None*, *localize=None*)

Synchronous API around Artifact API sets of cards

Parameters

- **limit_sets** (Optional[List[str]]) – Whether to only fetch some sets, by default all ar used ('00', and '01')
- **localize** (Optional[str]) – Which language to fetch strings for. Will be turned into lowercase automatically.

filter

Creates a new filter instance with all the cards. :rtype: *CardFilter* :return:

find(*name_approx*, *threshold=75*)

Finds a card by name, doesn't have to be exact name, thanks to highly sophisticated AI - a.k.a. simple algorithm, that will try and guess what was meant.

This algorithm can change over time so don't expect the same results across different versions.

Parameters

- **name_approx** (str) – Name to look up
- **threshold** (int) – How strict to be, higher number -> less likely to find a result if the name is off, higher chance the result will be correct

Return type Union[*Item*, *Hero*, *Ability*, *PassiveAbility*, *Improvement*, *Creep*, *Spell*, None]

get(*name*, *ignore_shared_name=True*)

Gets a card instance by name.

This is is a bit problematic, because some cards can have the same names as their ability. By default, this library will ignore that fact and return the not ability card. If it fails to find a card that's not an ability, it'll return the first one it registered.

You can override this behavior in which case this method will return a list of cards, instead of the card directly.

Parameters

- **name** (str) – Name of the card (case insensitive)
- **ignore_shared_name** (bool) – If there are more cards with the same name, get just the first hit.

Return type Union[*Item*, *Hero*, *Ability*, *PassiveAbility*, *Improvement*, *Creep*, *Spell*, List[Union[*Item*, *Hero*, *Ability*, *PassiveAbility*, *Improvement*, *Creep*, *Spell*]]]

load_all_sets()

Loads all the sets it should load from the api.

Return type None

CHAPTER 4

Deck

Deck wrapper with an overview class.

class pyartifact.deck.**Deck**(*deck_contents*)

Class for holding information about a deck. To use this, all sets must be loaded.

As this library isn't supposed to be a fully fledged deck constructor, accessing deck_contents directly is recommended.

Compared to the out-of-the-box data encoded in the deck string, this object enriches them with an additional key *instance* that holds an instance of the deck from `pyartifact.Cards`

As of now, the deck object does no validations, the rules to follow are:

- Some cards have *includes*, that are automatically added to the deck and they shouldn't be in the cards portion of deck contents.
- Abilities and Passive abilities aren't able to be included in a deck, as they come with another cards and are more of a traits than cards.
- Heroes have their own part of deck contents, don't put them into the cards section

Deck code versions

Deck codes currently have two versions. We are able to load both, but only dump to version 2.

Version	Heroes	Cards	Deck name
1	yes	yes	no
2	yes	yes	63 characters

When loading version 1, this library will still provide a name, which will be an empty string.

Parameters **deck_contents** (`DeckContents`) – dict of deck contents

cards

List of dictionaries with all the cards and their information

Return type `List[CardDeckType]`

deck_code
Returns the latest version of the deck code.

Return type str

dumps()
Returns the latest version of the deck code, same as deck_code property. For people who are used to json/yaml api :).

Return type str

expand_cards(*hero_includes=True*)
Expands all the cards in the deck into a list of their instances, once for each count.

Parameters **hero_includes** (bool) – Whether also expand auto-includes coming with the heroes

Return type List[Union[*Item*, *Hero*, *Ability*, *PassiveAbility*, *Improvement*, *Creep*, *Spell*]]

classmethod from_code(*deck_code*)
Constructs the deck object from a deck code string.

Deck.from_code(deck_code) does the exact same thing as *Deck.loads(deck_code)*

Parameters **deck_code** (str) – Version 1 or 2 deck code

Return type *Deck*

heroes
List of dictionaries with all the heroes and their information

Return type List[HeroDeckType]

classmethod loads(*deck_code*)
Constructs the deck object from a deck code string.

Deck.from_code(deck_code) does the exact same thing as *Deck.loads(deck_code)*

Parameters **deck_code** (str) – Version 1 or 2 deck code

Return type *Deck*

name
Name of the deck or an empty string of there is no name.

Return type str

classmethod new(*name*, *heroes*, *cards*)
Constructs the deck object from the insides of deck contents, for when you can't be bothered to make a dict.

Parameters

- **name** (str) – Name of the deck
- **heroes** (List[HeroDeckType]) – List of dictionaries holding information about the heroes
- **cards** (List[CardDeckType]) – List of dictionaries holding information about the cards

static new_card_dict(*card*, *count*)
Construction of a card dict compatible with the encoding process and Deck object internals

Parameters

- **card** (Union[*Item*, *Hero*, *Ability*, *PassiveAbility*, *Improvement*, *Creep*, *Spell*]) – Instance of a card
- **count** (int) – How many copies are in the deck

Return type CardDeckType

static new_hero_dict(*hero*, *turn*)
Construction of a hero dict compatible with the encoding process and Deck object internals

Parameters

- **hero** (*Hero*) – Instance of a hero card
- **turn** (int) – Which turn the hero will be deployed

Return type HeroDeckType

overview
Returns an overview of the deck, used for quick glances at it's contents. The overview holds an instance of the deck and will change with any changes made to the deck.

Return type *Overview*

class pyartifact.deck.**Overview**(*deck*)
A helper object for quick glances over the deck contents.

Parameters **deck** (*Deck*) – An instantiated deck object

creeps(*return_filter=False*)
All the creeps in the deck.

Parameters **return_filter** (bool) – Whether to return a pyartifact.CardFilter object or just a List of cards

Return type Union[List[*Creep*], *CardFilter*]

heroes(*return_filter=False*)
All the heroes in the deck, sorted by their turns of deployment.

Parameters **return_filter** (bool) – Whether to return a pyartifact.CardFilter object or just a List of cards

Return type Union[List[*Hero*], *CardFilter*]

improvements(*return_filter=False*)
All the improvements in the deck.

Parameters **return_filter** (bool) – Whether to return a pyartifact.CardFilter object or just a List of cards

Return type Union[List[*Improvement*], *CardFilter*]

items(*sub_type=None*, *return_filter=False*)
All the items in the deck.

Parameters

- **sub_type** (Optional[str]) – Whether to list only certain sub type of the items
- **return_filter** (bool) – Whether to return a pyartifact.CardFilter object or just a List of cards

Return type Union[List[*Item*], *CardFilter*]

items_per_subtype()
A more detailed overview of items on the deck in a form of defaultdict(list) where each key is a sub type and it's value is a list of those items.

Return type Dict[str, List[*Item*]]

spells(*return_filter=False*)
All the spells in the deck.

Parameters **return_filter** (bool) – Whether to return a pyartifact.CardFilter object or just a List of cards

Return type Union[List[*Spell*], *CardFilter*]

CHAPTER 5

Card objects

In order to easily work with cards (and abilities), pyartifact wraps them in easier to use objects. The objects are made by inheriting multiple bases, where each base indicates the presence of a certain attribute. For example all objects inheriting the *Unit* class will have attack, armor and hit_points attributes.

5.1 Card types

Some cards from Valve's API are not included in this library, because they are more of a core mechanics, these cards would have the type of Stronghold and Pathing.

5.1.1 Cards

class pyartifact.sets_and_cards.**Hero**(***kwargs*)
Inherts from *CardBase*, *ColoredCard*, *Unit* and *NotAbility*.

includes
List of all the cards this card includes automatically in a deck.

Return type List[Union[*Spell*, *Creep*, *Improvement*]]

passive_abilities
List of the cards passive abilities

Return type List[*PassiveAbility*]

class pyartifact.sets_and_cards.**Creep**(***kwargs*)
Inherits from *CardBase*, *ColoredCard*, *Unit*, *NotAbility*, *Castable*.

class pyartifact.sets_and_cards.**Spell**(***kwargs*)
Inherits from *CardBase*, *ColoredCard*, *NotAbility* and *Castable*.

class pyartifact.sets_and_cards.**Improvement**(***kwargs*)
Inherits from CardBase, ColoredCard, NotAbility, Castable.

class pyartifact.sets_and_cards.**Item**(***kwargs*)
Inherits from *CardBase*, *NotAbility*. Also has two attributes unique to this type.

Attribute	Type	Contents
gold_cost	int	How much gold does it take to purchase from the shop.
sub_type	str	Subtype of the item - Weapon, Accessory, Armor, Consumable or Deed

5.1.2 Abilities

class pyartifact.sets_and_cards.**Ability**(***kwargs*)
Inherits from *CardBase*.

class pyartifact.sets_and_cards.**PassiveAbility**(***kwargs*)
Inherits from *CardBase*.

5.2 Card Base classes

5.2.1 Base

class pyartifact.sets_and_cards.**CardBase**(***kwargs*)
All cards (and abilities) inherit the base.

Attribute	Type	Contents
id	int	Id of the card
base_id	int	Id of a card this card is based on (in future sets)
name	str	Name of the card
type	str	Type of the card, also indicated by the actual class holding the card
text	str	Text on the card, includes html
mini_image	Optional[str]	Url to mini image
large_image	Optional[str]	Url to large image
ingame_image	Optional[str]	Url to ingame image

references
List of cards that this card references

Return type List[Union[*Item*, *Hero*, *Ability*, *PassiveAbility*, *Improvement*, *Creep*, *Spell*]]

5.2.2 Colored card

class pyartifact.sets_and_cards.**ColoredCard**(***kwargs*)
Cards that belong under a certain color.

Attribute	Type	Contents
color	str	blue, black, red, green or unknown. There are no multicolor cards yet

5.2.3 Unit

class pyartifact.sets_and_cards.**Unit**(***kwargs*)
Cards that can be deployed to a battlefield and fight

Attribute	Type	Contents
attack	int	Attack of the unit
armor	int	Armor of the unit
hit_points	int	Hit points (health) of the unit

5.2.4 NotAbility

class pyartifact.sets_and_cards.**NotAbility**(***kwargs*)
Cards that are not abilities. Card API provides abilities and passive abilities alongside cards, so in the context of this library they are treated as cards.

Attribute	Type	Contents
rarity	Optional[str]	Rarity of the card, if it has one (base set cards don't have a rarity
item_def	Optional[int]	Unknown integer, only present when rarity is present
illustrator	str	Name of the illustrator that drew the card art

active_abilities
List of the cards active abilities

Return type List[*Ability*]

5.2.5 Castable

class pyartifact.sets_and_cards.**Castable**(***kwargs*)
Cards that can be casted for mana.

Attribute	Type	Contents
mana_cost	int	Mana cost to cast the card

CHAPTER 6

Deck encoding and decoding

6.1 Encoding

Logic for encoding deck into a deck code string.

Encoding is done by writing a few things into a bytearray thanks to the magic of bitwise operations. That is then encoded to base64 and sanitized for url usage.

class pyartifact.deck_encoding.encode.**Encoder**(*deck_contents*, *version=2*)

Main purpose of this class is to hold shared data across the encoding process.

There shouldn't be a need to use this part of the library, It offers a more low level access to the encoding process, but doesn't offer anything more practical than `pyartifact.encode_deck()` does.

Parameters

- **deck_contents** (DeckContents) – The deck contents.
- **version** (int) – Version under which to encode, by default the newest version is used. Must be on of the supported versions for encoding (atm only V2).

deck_code

Returns the deck code for the deck contents provided.

Return type str

pyartifact.deck_encoding.encode.**encode_deck**(*deck_contents*, *version=2*)

Encodes deck content into a deck code string.

Parameters

- **deck_contents** (DeckContents) – A dictionary with name, heroes and cards (without those included automatically)
- **version** (int) – Deck code version, atm only 2 and higher is supported

Return type str

Returns Deck code

6.2 Decoding

pyartifact.deck_encoding.decode.**decode_deck_string**(*deck_code*)

Takes in deck code, e.g. *ADCJWkTZX05uwGDCRV4XQGy3QGLmqUBg4GQJgGLGgO7AaABR3JlZW4vQmxhY2sgRXhhbXBsZ*
and decodes it into a dict of name, heroes and cards.

Parameters **deck_code** (str) – Deck code

Return type DeckContents

Returns Deck contents

Raises

- **InvalidDeckString** – When an invalid deck string is provided, e.g. unknown version, bad checksum etc.
- **DeckDecodeException** – When something odd happens while decoding

CHAPTER 7

Filtering

class pyartifact.filtering.**CardFilter**(*sets=None*, *cards=None*)

Class that allows you to use predefined filters to get the cards you are looking for. The cards are then available in the *cards* attribute.

All filter methods return a new filter instance. That means you can both nest filtering methods and save certain filter to a variables and access them later if you want to go a different path.

color(*color*)

Filters for a single color.

Parameters **color** (str) – Color

Return type *CardFilter*

color_in(*colors*)

Filters for multiple colors.

Parameters **colors** (Iterable[str]) – Colors

Return type *CardFilter*

color_not_in(*colors*)

For filtering out colors

Parameters **colors** (Iterable[str]) – Colors

Return type *CardFilter*

gold_cost(*gt=None*, *gte=None*, *lt=None*, *lte=None*, *eq=None*)

Filters out cards by their gold cost, if they have one. This will always filter out cards without gold cost. If multiple arguments are passed, every card that fits at least one will pass the filter.

Parameters

- **gt** (Optional[int]) – Filters out cards that have higher gold cost than the number provided
- **gte** (Optional[int]) – Filters out cards that have higher or equal gold cost than the number provided

- **lt** (Optional[int]) – Filters out cards that have lower gold cost than the number provided
- **lte** (Optional[int]) – Filters out cards that have lower or equal gold cost than the number provided
- **eq** (Optional[int]) – Filters out cards that have gold cost equal to the number provided

Return type *CardFilter*

mana_cost(*gt=None*, *gte=None*, *lt=None*, *lte=None*, *eq=None*)
Filters out cards by their mana cost, if they have one. This will always filter out cards without mana cost. If multiple arguments are passed, every card that fits at least one will pass the filter.

Parameters

- **gt** (Optional[int]) – Filters out cards that have higher mana cost than the number provided
- **gte** (Optional[int]) – Filters out cards that have higher or equal mana cost than the number provided
- **lt** (Optional[int]) – Filters out cards that have lower mana cost than the number provided
- **lte** (Optional[int]) – Filters out cards that have lower or equal mana cost than the number provided
- **eq** (Optional[int]) – Filters out cards that have mana cost equal to the number provided

Return type *CardFilter*

rarity(*rarity*)
Filters for a rarity

Parameters **rarity** (str) – Rarity

Return type *CardFilter*

rarity_in(*rarities*)
Filters for multiple rarities

Parameters **rarities** (List[str]) – Rarities

Return type *CardFilter*

rarity_not_in(*rarities*)
Filters out cards of specified rarities.

Parameters **rarities** (List[str]) – Rarities

Return type *CardFilter*

sub_type(*sub_type*)
Filters out everything but items and leaves just items with a subtype equal to the provided string

Parameters **sub_type** (str) – Sub type of an item

Return type *CardFilter*

type(*type_*, *filter_out=False*)
Filters for a single type (or anything but a single type)

Parameters

- **type** – Type of the card

- **filter_out** – Whether to filter that type out

Return type *CardFilter*

types_in(*card_types*)
Filters out cards that were not passed to this filter

Parameters **card_types** (Iterable[Union[str, Type[*Item*], Type[*Hero*], Type[*Ability*], Type[*PassiveAbility*], Type[*Improvement*], Type[*Creep*], Type[*Spell*]]]) – Either strings of card types, or this library's classes of card types

Return type *CardFilter*

types_not_in(*card_types*)
Filters out cards that were passed into this filter

Parameters **card_types** (Iterable[Union[str, Type[*Item*], Type[*Hero*], Type[*Ability*], Type[*PassiveAbility*], Type[*Improvement*], Type[*Creep*], Type[*Spell*]]]) – Either strings of card types, or this library's classes of card types

Return type *CardFilter*

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

O

P

R

S

T

U