akinator.py

Release 0.2.4

Tom-the-Bomb

May 01, 2023

CONTENTS:

CHAPTER

ONE

AKINATOR PACKAGE

1.1 Module contents

Python bindings for `akinator-rs`, a wrapper around the undocumented akinator API

designed for easy implementation of an akinator game in code, providing a simple and easy to use API.

class akinator.**AsyncAkinator**(*, *theme=None*, *language=None*, *child_mode=None*)

Bases: `object`

Represents an async akinator game

Note: All attributes and methods are the same as the blocking *Akinator* class but instead all methods should be awaited

---

Parameters are also set as properties which also have a setter to change the values if necessary in the future

Parameters

- **theme** (Optional[*Theme*]) – the theme of the akinator game, would be one of `Characters`, `Animals` or `Objects` pass in using an answer enum, using the `from_str` classmethod if necessary, defaults to `Characters`
- **language** (Optional[*Language*]) – the language for the akinator game, refer to the *Language* enum, defaults to `English`
- **child_mode** (Optional[`bool`]) – when set to `True`, NSFW content will not be provided, defaults to `False`

answer(*answer*)

This function is a *coroutine*.

Answers the akinator's current question with the provided `answer` and returns the next question

Parameters
answer (*Answer*) – the answer to the current question

Return type
Optional[`str`]

Raises

- `RuntimeError` – Something internal went wrong, this could be in this case: - missing required data to continue - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)

- **ValueError** – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

back()

This function is a *coroutine*.

Goes back a question and returns said (current) question

Return type
Optional[str]

Raises

- ***CantGoBackAnyFurther*** – Could not go back anymore, likely that we are already on the first question
- **RuntimeError** – Something internal went wrong, this could be in this case: - missing required data to continue - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- **ValueError** – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

child_mode

whether child_mode is on or off for the akinator game

Type
bool

first_guess

the akinator's best guess

Type
Optional[*Guess*]

guesses

a list of all the akinator's potential guesses, ordered by likeliness

Type
List[*Guess*]

language

the language of the akinator game

Type
Language

progression

the progression of the akinator

Type
float

question

the current question of the akinator game

Type
Optional[str]

`start_game()`

This function is a *coroutine*.

Starts the akinator game and returns the first question

Return type
Optional[str]

Raises

- **RuntimeError** – Something internal went wrong, this could be in this case: - getting the starting timestamp failed - the data required to start the game such as the server url, frontaddr or game UID could not be found - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- **ValueError** – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

`step`

a counter for the question # the akinator is on currently

Type
int

`theme`

the theme of the akinator game

Type
Theme

`win()`

This function is a *coroutine*.

Tells the akinator to end the game and make its guess should be called once when the `progression` is high enough such as >=`80.0` and returns its best guess

Return type
Optional[*Guess*]

Raises

- **RuntimeError** – Something internal went wrong, this could be in this case: - missing required data to continue - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- **ValueError** – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

`class akinator.Akinator(*, theme=None, language=None, child_mode=None)`

Bases: object

Represents an akinator game

Parameters are also set as properties which also have a setter to change the values if necessary in the future

Parameters

- **theme** (Optional[*Theme*]) – the theme of the akinator game, would be one of `Characters`, `Animals` or `Objects` pass in using an answer enum, using the `from_str` classmethod if necessary, defaults to `Characters`
- **language** (Optional[*Language*]) – the language for the akinator game, refer to the `Language` enum, defaults to `English`
- **child_mode** (Optional[`bool`]) – when set to `True`, NSFW content will not be provided, defaults to `False`

answer(*answer*)

Answers the akinator's current question with the provided `answer` and returns the next question

Parameters
answer (*Answer*) – the answer to the current question

Return type
Optional[`str`]

Raises

- `RuntimeError` – Something internal went wrong, this could be in this case: - missing required data to continue - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- `ValueError` – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

back()

Goes back a question and returns said (current) question

Return type
Optional[`str`]

Raises

- *CantGoBackAnyFurther* – Could not go back anymore, likely that we are already on the first question
- `RuntimeError` – Something internal went wrong, this could be in this case: - missing required data to continue - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- `ValueError` – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

child_mode

whether `child_mode` is on or off for the akinator game

Type
`bool`

first_guess

the akinator's best guess

Type
Optional[*Guess*]

guesses

a list of all the akinator's potential guesses, ordered

Type
List[*Guess*]

language

the language of the akinator game

Type
Language

progression

the progression of the akinator

Type
float

question

the current question of the akinator game

Type
Optional[str]

start_game()

Starts the akinator game and returns the first question

Return type
Optional[str]

Raises

- **RuntimeError** – Something internal went wrong, this could be in this case: - getting the starting timestamp failed - the data required to start the game such as the server url, frontaddr or game UID could not be found - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- **ValueError** – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

step

a counter for the question # the akinator is on currently

Type
int

theme

the theme of the akinator game

Type
Theme

win()

Tells the akinator to end the game and make its guess should be called once when the progression is high enough such as >=80.0 and returns its best guess

Return type
Optional[*Guess*]

Raises

- **RuntimeError** – Something internal went wrong, this could be in this case: - missing required data to continue - request error: any sort of error when making the HTTP requests - updating the internal data fields errored (either a field was missing or was of the wrong type)
- **ValueError** – Could not parse the API returned JSON properly (invalid, missing fields etc.)
- **Other api errors** – Refer to the exceptions at the bottom of the page

class akinator.**Guess**

Bases: object

a model class representing an akinator's guess not meant for the user to construct, but is returned in various properties and methods in the *Akinator* class

absolute_picture_path

an absolute url to the picture of the guess's entity

Type
str

award_id

award id

Type
str

confidence

the accuracy / confidence of the akinator that this guess is correct

Type
float

description

a brief description of the specific guess's entity

Type
str

flag_photo

flag photo

Type
int

id

the unique ID of the specific guess's entity

Type
str

name

the common name of the specific guess's entity

Type
str

picture_path

a relative path to a picture of the guess's entity

Type
str

ranking

the rank of the specific guess's entity

Type
str

class akinator.**Theme**

Bases: object

An enum class representing the theme of an akinator game

This is meant for the user to use to pass into the Akinator constructor, or to set the theme property

Animals = <Theme theme="Animals">

Characters = <Theme theme="Characters">

Objects = <Theme theme="Objects">

from_str(*theme*)

a classmethod to return a *Theme* enum variant parsing from a str useful when you have external user input

Parameters

- **theme** (str) – the string representation of the theme to parse from
- **::** (*.. note*) – if an invalid string for the theme is given, no error will be raised instead it will just fallback to Theme.Characters as the default

class akinator.**Answer**

Bases: object

An enum class representing an answer given to the akinator

This is meant for the user to use to pass into methods such as *Akinator.answer*

Idk = <Answer answer="Idk">

No = <Answer answer="No">

Probably = <Answer answer="Probably">

ProbablyNot = <Answer answer="ProbablyNot">

Yes = <Answer answer="Yes">

from_str(*answer*)

a classmethod to return an *Answer* enum variant parsing from a str useful when you have external user input

aliases for answer variants are also accepted (trims ws & case-insensitive):

- yes | y | 0 -> Answer.Yes
- no | n | 1 -> Answer.No
- i don(')?t know | idk | 2 -> Answer.Idk

- `probably | p | 3` -> `Answer.Probably`
- `probably not | pn | 4` -> `Answer.ProbablyNot`

Parameters
answer (`str`) – the string representation of the answer to parse from

Raises
InvalidAnswer – raised if the provided answer cannot match one of the above (is invalid)

class akinator.**Language**

Bases: `object`

An enum class representing the language of the akinator game

This is meant for the user to use to pass into the Akinator constructor, or to set the language property

Arabic = <Language lang="Arabic">

Chinese = <Language lang="Chinese">

Dutch = <Language lang="Dutch">

English = <Language lang="English">

French = <Language lang="French">

German = <Language lang="German">

Hebrew = <Language lang="Hebrew">

Indonesian = <Language lang="Indonesian">

Italian = <Language lang="Italian">

Japanese = <Language lang="Japanese">

Korean = <Language lang="Korean">

Polish = <Language lang="Polish">

Portugese = <Language lang="Portugese">

Russian = <Language lang="Russian">

Spanish = <Language lang="Spanish">

Turkish = <Language lang="Turkish">

from_str(*language*)

a classmethod to return a *Language* enum variant parsing from a `str` useful when you have external user input

Short forms such as `en` or `fr` are also accepted along with the full name

Parameters
language (`str`) – the string representation of the language to parse from

Raises
InvalidLanguage – Raised if the given string is of an invalid language

exception akinator.**CantGoBackAnyFurther**

Bases: Exception

Raised when the akinator is already on the 1st question / there are no more questions to go back on

exception akinator.**InvalidAnswer**

Bases: Exception

Raised when an invalid answer string is used when instantiating a Language enum from str

exception akinator.**InvalidLanguage**

Bases: Exception

Raised when an invalid language string is used when instantiating a Language enum from str

exception akinator.**ConnectionError**

Bases: Exception

Raised when we fail the connect to the akinator servers for whatever reason

exception akinator.**NoMoreQuestions**

Bases: Exception

Raised when there are no more questions the akinator can offer

exception akinator.**TimeoutError**

Bases: Exception

Raised when the akinator session timed out waiting for a response

exception akinator.**TechnicalError**

Bases: Exception

Raised when there is a technical internal error with the akinator servers

exception akinator.**ServersDown**

Bases: Exception

Raised when the akinator servers in the requested region are down

CHAPTER

TWO

EXAMPLES

Installation

wheels are prebuilt and uploaded to pypi, so if your platform is supported, simply doing

```
$ py -m pip install akinator.py
```

will do it

You can also manually build from source, but that requires rust to be installed.

importing

```
import akinator
```

Full sync example

Here is a full working example of using the sync akinator class

```
from akinator import (
    CantGoBackAnyFurther,
    InvalidAnswer,
    Akinator,
    Answer,
    Theme,
)

def test() -> None:
    # create akinator instance
    aki = Akinator(
        child_mode=True,
        theme=Theme.from_str('characters'),
    )

    # start the game, and get the first question
    first_question = aki.start_game()
    # recieve console input for first question
    answer = input(f'{first_question}: ')

    # keep asking and recieving answers while akinator's progression is <=80
    while aki.progression <= 80:
        if answer == 'back':
            # go back a question if response is "back"
            try:
```

(continues on next page)

(continued from previous page)

```
                aki.back()
                print('went back 1 question')
            except CantGoBackAnyFurther:
                print('cannot go back any further!')
        else:
            try:
                # parse to an answer enum variant
                answer = Answer.from_str(answer)
            except InvalidAnswer:
                print('Invalid answer')
            else:
                # answer current question
                aki.answer(answer)

        # recieving console input for next question
        answer = input(f'{aki.question}: ')

    # tell akinator to end the game and make its guess
    first_guess = aki.win()

    if first_guess:
        # print out its first guess's details
        print('name:', first_guess.name)
        print('desc:', first_guess.description)
        print('image:', first_guess.absolute_picture_path)

if __name__ == '__main__':
    test()
```

Full async example

Here is a full working example of using the async akinator class with `asyncio`

(pretty much the same except all methods are awaited)

```
import asyncio

from akinator import (
    CantGoBackAnyFurther,
    InvalidAnswer,
    AsyncAkinator,
    Answer,
    Theme,
)

async def test() -> None:
    # create akinator instance
    aki = AsyncAkinator(
        child_mode=True,
        theme=Theme.from_str('characters'),
    )

    # start the game, and get the first question
```

(continues on next page)

(continued from previous page)

```
    first_question = await aki.start_game()
    # recieve console input for first question
    answer = input(f'{first_question}: ')

    # keep asking and recieving answers while akinator's progression is <=80
    while aki.progression <= 80:
        if answer == 'back':
            # go back a question if response is "back"
            try:
                await aki.back()
                print('went back 1 question')
            except CantGoBackAnyFurther:
                print('cannot go back any further!')
        else:
            try:
                # parse to an answer enum variant
                answer = Answer.from_str(answer)
            except InvalidAnswer:
                print('Invalid answer')
            else:
                # answer current question
                await aki.answer(answer)

        # recieving console input for next question
        answer = input(f'{aki.question}: ')

    # tell akinator to end the game and make its guess
    first_guess = await aki.win()

    if first_guess:
        # print out its first guess's details
        print('name:', first_guess.name)
        print('desc:', first_guess.description)
        print('image:', first_guess.absolute_picture_path)

if __name__ == '__main__':
    asyncio.run(test())
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- search

PYTHON MODULE INDEX

INDEX

P

Q

R

S

T

W

Y