lectio.py

Release 0.2.1

dnorhoj

Dec 19, 2022

CONTENTS

Lectio.py is a Python library for reading and interacting with lectio.dk, a Danish school management system.

Note: This library is not affiliated with lectio.dk or macom in any way. Nor is it endorsed by them. It scrapes the website as there is no official API.

Features:

- Pythonic API
- Easy to use

CHAPTER

ONE

INFORMATION

Useful links: *Installation* | *Examples*

Reference: *API Reference*

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Installation

To install lectio.py and its dependencies, you can simply install it from PyPi:

```
pip install lectio.py
```

or clone the repository and install it manually:

```
git clone https://github.com/dnorhoj/Lectio.py.git
cd lectio.py
python setup.py install
```

You can check if the installation was successful by running:

```
python

>>> import lectio
>>> lectio.__version__
```

2.2 Examples

On this page you can find some examples on how to use the lectio.py library.

2.2.1 Authentication

To authenticate you need three things:

- Your institution id
- Your username
- Your password

First you need your institution id. To get your institution id, you can simply go to your school's lectio login page, and look at the url. It should look something like this:

> https://www.lectio.dk/lectio/123/login.aspx

The number after lectio is your institution id. (In this case `123`)

We can now create a new *`lectio.Lectio`* object by creating a new file and adding the following code:

```
from lectio import Lectio

lec = Lectio(123, '<username>', '<password>')
```

Now we have a *lectio.Lectio* object, which we can use to get information from lectio.

2.2.2 Get my user object

Your user object contains information about you, such as your name and your id. It can also be used to fetch data such as your schedule, absences and more.

You can get your user object by calling the `lectio.Lectio.get_user()` method:

```
from lectio import Lectio

lec = Lectio(123, '<username>', '<password>')

me = lec.get_user()
```

The `me` variable now contains your user object. We can for example print your name:

```
print(me.name)
```

2.2.3 Get my schedule

You can get your schedule by calling the `lectio.User.get_schedule()` method with a start and end date:

```
from lectio import Lectio
from datetime import datetime

lec = Lectio(123, '<username>', '<password>')

me = lec.get_user()

schedule = me.get_schedule(datetime.now(), datetime.now())
```

The `schedule` variable now contains your schedule, which is a list of *Module* objects.

You might be wondering what the `datetime.now()` is. It is simply a date, and you can replace it with any date you want. In this example we are getting the schedule for today.

We can now loop through the schedule and print the name of each module:

```
for module in schedule:
    print(module.name)
```

You can read more about the *Module* object by going to the reference.

More to come...!

2.3 API Reference

class lectio.**Lectio**(*inst_id: int*, *username: str*, *password: str*, ***kwargs*)

The main Lectio class.

A Lectio object is your gateway to manipulating and getting data from Lectio.

Parameters

- **inst_id** (*int*) – Your Lectio institution id.

  You can find this by going to your institution's Lectio login page and it should be in the URL like this:

  ```
  https://www.lectio.dk/lectio/123/login.aspx
  ```

  Here, the `123` would be my institution id.

- **username** (*str*) – Lectio username for the given institution id.
- **password** (*str*) – Lectio password for the given institution id.

me() → *Me*

Gets the authenticated user

Returns
Own user object

Return type
Me

get_school() → *School*

Gets the school object for the authenticated user

This loads the school object, which gets cached for future use.

Returns
School object

Return type
School

2.3.1 Models

class lectio.models.school.**School**(*lectio:* Lectio)

A school object.

Represents a school.

Note: This class should not be instantiated directly, but rather through the *lectio.Lectio.get_school()* method.

Example

```
import lectio

lectio = lectio.Lectio(123, "username", "password")

school = lectio.get_school()

print(school.name)
```

students: List[*User*]
List of students in the school

teachers: List[*User*]
List of teachers in the school

groups: List[None]
List of groups in the school (TODO

rooms: List[*Room*]
List of rooms in the school

name: str
Name of the school

get_user_by_id(*user_id: int*, *user_type: Optional[*UserType*] = None*, *lazy=False*) → *User*
Gets a user by their id

Parameters
- **user_id** (*int*) – The id of the user
- **user_type** (*UserType*) – The type of the user (student or teacher)
- **lazy** (*bool*) – Whether to return a lazy user object or not (default: False)

Returns
User object

Return type
User

Raises
exceptions.UserDoesNotExistError – When the user does not exist

search_for_teachers_by_name(*query: str*) → Generator[*User*, None, None]
Search for teachers by name or initials

Parameters
query (*str*) – Name to search for

Yields
lectio.models.user.User – Teacher object

search_for_teachers_by_initials(*query: str*) → Generator[*User*, None, None]
Search for teachers by initials

Parameters
query (*str*) – Initials to search for

Yields
User – Teacher user object

search_for_students(*query: str*) → Generator[*User*, None, None]

Search for user

Parameters
query (*str*) – Name to search for

Yields
User – Student user object

search_for_users(*query: str*) → Generator[*User*, None, None]

Search for user

Parameters
query (*str*) – Name to search for

Yields
User – User object

get_room_by_id(*room_id: int*) → *Room*

Gets a room by its id

Parameters
room_id (*int*) – The id of the room

Returns
Room object

Return type
Room

Raises
exceptions.RoomDoesNotExistError – When the room does not exist

search_for_rooms(*query: str*) → Generator[*Room*, None, None]

Search for room

Parameters
query (*str*) – Name to search for

Yields
Room – Room object

class lectio.models.user.**User**(*lectio: Lectio*, *user_id: int*, *user_type: UserType = UserType.STUDENT*, *name: str = None*, *, *lazy=True*, **kwargs*)

Lectio user object

Represents a lectio user

Note: This class should not be instantiated directly, but rather through the *lectio.Lectio.me* or *lectio.models.school.School.search_for_users()* methods or similar.

Note: A *lectio.models.user.User* object is a lazy object by default, which means that no data is fetched from lectio on instantiation.

When you access any non-lazy attribute, the user object will be populated with data from lectio.

Parameters

- **lectio** (*lectio.Lectio*) – Lectio object
- **user_id** (*int*) – User id
- **user_type** (*lectio.models.user.UserType*) – User type (UserType.STUDENT or UserType.TEACHER)
- **lazy** (*bool*) – Whether to not populate user object on instantiation (default: True)

id: int

User id

type: *UserType*

User type

populated: bool

Whether user object has been populated with data from lectio

If this is False, the user object is a lazy object and only contains available data (often just name and id)

populate() → None

Populate user object

Populates the user object with data from lectio, such as name, class name, etc.

get_schedule(*start_date: datetime*, *end_date: datetime*, *strip_time: bool = True*) → List[*Module*]

Get schedule for user

Note: As lectio is weird, you can only get a schedule for a range that is less than one month. If you specify a range greater than one month, you will get an empty return list.

TODO: Make this work for ranges greater than one month

Parameters

- **start_date** (datetime.datetime) – Start date
- **end_date** (datetime.datetime) – End date
- **strip_time** (*bool*) – Whether to remove hours, minutes and seconds from date info, also adds 1 day to end time. Basically just allows you to put in a random time of two days, and still get all modules from all the days including start and end date.

Returns
List of modules

Return type
List[*Module*]

property url: str

User's lectio url

Type
str

get_image_url() → str

Get user's image url

Returns
User's image url

Return type
str

`get_class_name`() → Optional[str]

Get user's class name

Only for *UserType.STUDENT*.

Returns
Class name or None if user is *UserType.TEACHER*.

Return type
str|None

class lectio.models.user.**Me**(*lectio:* Lectio, *user_type:* UserType = *UserType.STUDENT*)

Class that represents the logged in user

This method extends the *lectio.models.user.User* class, and therefore has all the same methods and attributes.

Note: This method can be created with the *lectio.Lectio.me()* method.

`populate`() → None

Populate user object

Populates the user object with data from lectio, such as name, class name, etc.

`get_absences`() → *Absence*

Get absences

Returns
Absences for logged in user

Return type
Absence

class lectio.models.module.**Module**(*lectio:* Lectio, ***kwargs*)

Lectio module object

Represents a lectio module

`title: Optional[str]`

Description of module

`subject: Optional[str]`

"Hold" from lectio, bascially which subject.

`teacher: Optional[str]`

Initials of teacher.

`room: Optional[str]`

Room name of module.

`extra_info: Optional[str]`

Extra info from module, includes homework and other info.

start_time: datetime.datetime

Start time of module

end_time: datetime.datetime

End time of module

status: *ModuleStatus*

Module status

url: Optional[str]

Url for more info for the module

get_homework() → str

get_extra_info() → str

get_teachers() → List[*User*]

Get teachers

Returns
List of teacher user objects

Return type
List[*User*]

get_rooms() → List[*Room*]

Get rooms

Returns
Room object

Return type
List[*Room*]

get_team() → None

Get team

Returns
Team name

Return type
TODO

class lectio.models.room.**Room**(*lectio:* Lectio, *room_id: int*, *name: str*)

A room object.

Represents a room.

Note: This class should not be instantiated directly, but rather through the `lectio.school.School.get_rooms()` method.

Parameters

- **lectio** (Lectio) – Lectio object
- **id** (*int*) – Room id
- **name** (*str*) – Room name

get_schedule(*start_date: datetime*, *end_date: datetime*, *strip_time: bool*) → List[*Module*]

Get schedule for room

Note: As lectio is weird, you can only get a schedule for a range that is less than one month. If you specify a range greater than one month, you will get an empty return list.

Parameters

- **start_date** (datetime.datetime) – Start date
- **end_date** (datetime.datetime) – End date
- **strip_time** (*bool*) – Whether to remove hours, minutes and seconds from date info, also adds 1 day to end time. Basically just allows you to put in a random time of two days, and still get all modules from all the days including start and end date.

Returns

List of modules

Return type

List[*Module*]

is_available(*date: Optional[datetime] = None*) → bool

Check if room is available at a given time

Parameters

datetime (datetime.datetime) – Datetime to check (defaults to now)

class lectio.models.absence.**Absence**(*lectio:* Lectio)

Class for representing a user's absences

Note: This class should not be instantiated directly, but rather through the *lectio.models.user.Me.get_absences()* method

---

subjects: List[*SubjectAbsenceData*]

List of absence data for each subject

total_absences: *AbsenceData*

Total absence data (all subjects combined

toJSON() → str

Return a JSON representation of all the absence data

Returns

JSON string of all the absence data in the following format:

```
{
    "subjects": [...]
    "total": {...}
}
```

Return type

str

2.3.2 Misc

class lectio.models.user.**UserType**(*value*)

User types enum

Example

```
>>> from lectio import Lectio
>>> from lectio.models.user import UserType
>>> lec = Lectio(123, "username", "password")
>>> me = lec.me()
>>> print(me.type)
<UserType.STUDENT: 0>
>>> print(me.type == UserType.STUDENT)
True
```

STUDENT = 0

Student user type

TEACHER = 1

Teacher user type

get_str() → str

Get string representation of user type for lectio interface in english

Returns
String representation of user type

Return type
str

class lectio.helpers.schedule.**ModuleStatus**(*value*)

Module status enum

NORMAL = 0

Module has not been changed

CHANGED = 1

Module has been changed (is green in lectio)

CANCELLED = 2

Module has been cancelled (is red in lectio)

class lectio.models.absence.**AbsenceData**(*physical_total: int*, *physical_absent: int*, *physical_percentage: float*, *physical_calculated_total: int*, *physical_calculated_absent: int*, *physical_calculated_percentage: float*, *assignment_total: int*, *assignment_absent: int*, *assignment_percentage: float*, *assignment_calculated_total: int*, *assignment_calculated_absent: int*, *assignment_calculated_percentage: float*)

Class for representing a subject absence

physical_total: int

Total number of modules for the entire year

physical_absent: int

Number of modules absent for the entire year

physical_percentage: float

Percentage of modules absent for the entire year

physical_calculated_total: int

Total number of modules until the current date

physical_calculated_absent: int

Total number of modules absent until the current date

physical_calculated_percentage: float

Percentage of modules absent until the current date

assignment_total: int

Total number of student hours (elevtimer) for the entire year

assignment_absent: int

Total number of student hours absent for the entire year

assignment_percentage: float

Percentage of student hours absent for the entire year

assignment_calculated_total: int

Total number of student hours until the current date

assignment_calculated_absent: int

Total number of student hours absent until the current date

assignment_calculated_percentage: float

Percentage of student hours absent until the current date

class lectio.models.absence.**SubjectAbsenceData**(*subject: str*, *group_id: int*, *absence_data:* AbsenceData)

Class for representing a subject absence

subject: str

Subject name

group_id: int

Group id

absence_data: *AbsenceData*

Absence data

property group: None

Return group object

Type

TODO

2.4 Exceptions

Here are all the custom Lectio.py exceptions as well as their explanation

exception lectio.exceptions.**LectioError**

Base lectio.py exception

exception lectio.exceptions.**UnauthenticatedError**

Throws when trying to get data while unauthenticated or session expired

exception lectio.exceptions.**IncorrectCredentialsError**

Incorrect credentials error, mostly thrown in auto-login on session expired

exception lectio.exceptions.**InstitutionDoesNotExistError**

The institution with the id you provided does not exist.

exception lectio.exceptions.**UserDoesNotExistError**

The user does not exist.

exception lectio.exceptions.**RoomDoesNotExistError**

The room does not exist.

PYTHON MODULE INDEX

INDEX