clickupython

Release 0.0.1

Zach Johnson & Robert Mullis

Aug 04, 2023

CONTENTS:

clickupython is a Python client for the ClickUp API and can be used to interact with the ClickUp API in your projects. ClickUp's API exposes the entire ClickUp infrastructure via a standardized programmatic interface. Using ClickUp's API, you can do just about anything you can do on clickup.com.

CHAPTER

ONE

GETTING STARTED

To start, install clickupython via pip.

```
$ pip install clickupython
```

CHAPTER

TWO

AUTHENTICATION

There are two ways to authenticate with ClickUp API 2.0, with a personal token or creating an application and authenticating with an OAuth2 flow.

Note: IMPORTANT - If you are creating an application for other's to use, it is highly recommended that you use the OAuth2 flow.

2.1 Method 1: API Key (Fastest)

Sign in to ClickUp and navigate to Settings > Apps. There you will see a an API token. Copy this and save it. You will use this to authenticate the clickupython client with ClickUp's API.

```
$ from clickupython import client

  API_KEY = 'YOUR API KEY'
  client = ClickUpClient(API_KEY)

  # Example request | Creating a task in a list
  c = client.ClickUpClient(API_KEY)
  task = c.create_task("list_id", name="Test task", due_date="march 2 2021")

  if task:
     print(task.id)
```

2.1.1 Tasks

class clickupython.models.**Task**(*, *id: str = None*, *custom_id: str = None*, *name: str = None*, *text_content: str = None*, *description: str = None*, *status: Status = None*, *orderindex: str = None*, *date_created: str = None*, *date_updated: str = None*, *date_closed: str = None*, *creator: Creator = None*, *assignees: List[Asssignee] = None*, *checklists: List[Any] = None*, *tags: List[Any] = None*, *parent: str = None*, *priority: Any = None*, *due_date: str = None*, *start_date: str = None*, *time_estimate: str = None*, *time_spent: str = None*, *custom_fields: List[CustomField] = None*, *list: ClickupList = None*, *folder: Folder = None*, *space: Folder = None*, *url: str = ''*)

Bases: `BaseModel`

add_comment(*client_instance*, *comment_text: str*, *assignee: Optional[str] = None*, *notify_all: bool = True*)

assignees: List[Asssignee]

build_task()

creator: Creator

custom_fields: Optional[List[CustomField]]

custom_id: str

date_closed: str

date_created: str

date_updated: str

delete()

description: str

due_date: str

folder: Folder

get_comments(*client_instance*)

id: str

list: ClickupList

name: str

orderindex: str

parent: str

priority: Any

space: Folder

start_date: str

status: Status

task_checklists: List[Any]

task_tags: List[Any]

text_content: str

time_estimate: str

time_spent: Optional[str]

update(*client_instance*, *name: Optional[str] = None*, *description: Optional[str] = None*, *status: Optional[str] = None*, *priority: Optional[Any] = None*, *time_estimate: Optional[int] = None*, *archived: Optional[bool] = None*, *add_assignees: Optional[List[str]] = None*, *remove_assignees: Optional[List[int]] = None*)

upload_attachment(*client_instance*, *file_path: str*)

url: str

Creating a Task

Here's how to create a new ClickupClient instance and validate a personal API key with the ClickUp API and create a new task with a due date. When creating a new task, the only required arguments are list_id and name. Name will be the title of your list on ClickUp.

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
t = c.create_task("LIST_ID", name="Test Task", due_date="march 2 2021")
```

Fetching a Single Task

Example: Lookup via ClickUpClient:

```
c = client.ClickUpClient("YOUR_API_KEY")
task = c.get_task(task_id)

print(task.name)
```

Fetching all Tasks from a List

You can quickly get all tasks for a given list via the list id:

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks("list_id")
```

Filtering Tasks

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks("list_id", date_updated_gt="august 1 2021",
       assignees=["4523","4562","5871"], include_closed=True)
```

This example will return all tasks that have been updated after August 1st, 2021 and are assigned to users with the ids of 4523, 4562, and 5871. This request will also include tasks that have been marked as "closed."

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks("list_id", subtasks=True,
        statuses=["todo", "in progress"])
```

This example will return all tasks and subtasks that are marked as "Todo" and "In Progress". These values can be changed depending on the statuses you have available in your list.

You can extend the calls above by passing any of the arguments to the `get_tasks` method. You can use as many or as few as you would like.

Working With Tasks

Now that you have a list of `Task` objects you can access the attributes of each task in a number of ways:

Example: Loop:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks(list_id)

for task in tasks:
   print(task.name)
```

Example: Direct Access via an Index:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks(list_id)

print(tasks[0].name])
```

Getting Tasks Associated with a List Object

Certain calls can be made directly from a parent object. We can access a single `Task` or all `Tasks` associated with a `List` with the following methods.

Note: IMPORTANT - When calling a method from a parent object you must pass in a reference to the ClickUpClient object as the first argument.

Example: Lookup Tasks via a List Object:

```
c = client.ClickUpClient("YOUR_API_KEY")
list = c.get_list(list_id)
tasks = list.get_tasks(c)
filtered_tasks = list.get_tasks(c, subtasks=True, statuses=["todo", "in progress"])
task = list.get_task(c, task_id)
```

Task Methods

get_tasks()

ClickUpClient.**get_tasks**(*list_id: str*, *archived: bool = False*, *page: int = 0*, *order_by: str = 'created'*, *reverse: bool = False*, *subtasks: bool = False*, *statuses: Optional[List[str]] = None*, *include_closed: bool = False*, *assignees: Optional[List[str]] = None*, *due_date_gt: Optional[str] = None*, *due_date_lt: Optional[str] = None*, *date_created_gt: Optional[str] = None*, *date_created_lt: Optional[str] = None*, *date_updated_gt: Optional[str] = None*, *date_updated_lt: Optional[str] = None*) → Tasks

The maximum number of tasks returned in this response is 100. When you are paging this request, you should check list limit against the length of each response to determine if you are on the last page.

Args:

list_id (str)
The ID of the list to retrieve tasks from.

archived (bool, optional)
Include archived tasks in the retrieved tasks. Defaults to False.

page (int, optional)
Page to fetch (starts at 0). Defaults to 0.

order_by (str, optional)
Order by field, defaults to "created". Options: id, created, updated, due_date.

reverse (bool, optional)
Reverse the order of the returned tasks. Defaults to False.

subtasks (bool, optional)
Include archived tasks in the retrieved tasks. Defaults to False.

statuses (List[str], optional)
Only retrieve tasks with the supplied status. Defaults to None.

include_closed (bool, optional)
Include closed tasks in the query. Defaults to False.

assignees (List[str], optional)
Retrieve tasks for specific assignees only. Defaults to None.

due_date_gt (str, optional)
Retrieve tasks with a due date greater than the supplied date. Defaults to None.

due_date_lt (str, optional)
Retrieve tasks with a due date less than the supplied date. Defaults to None.

date_created_gt (str, optional)
Retrieve tasks with a creation date greater than the supplied date. Defaults to None.

date_created_lt (str, optional)
Retrieve tasks with a creation date less than the supplied date. Defaults to None.

date_updated_gt (str, optional)
Retrieve tasks where the last update date is greater than the supplied date. Defaults to None.

date_updated_lt (str, optional)
Retrieve tasks where the last update date is greater than the supplied date. Defaults to None.

Raises:

exceptions.ClickupClientError
Invalid order_by value

Returns:

models.Tasks
Returns a list of item Task.

get_task()

ClickUpClient.**get_task**(*task_id: str*) → *Task*

Fetches a single ClickUp task item and returns a Task object.

Args:

task_id (str)
The ID of the task to return.

Returns:

Task
Returns an object of type Task.

create_task()

ClickUpClient.**create_task**(*list_id: str*, *name: str*, *description: str = None*, *priority: int = None*, *assignees: [] = None*, *tags: [] = None*, *status: str = None*, *due_date: str = None*, *start_date: str = None*, *notify_all: bool = True*) → *Task*

[summary]

Args:

list_id (str)
[description]

name (str)
[description]

description (str, optional)
[description]. Defaults to None.

priority (int, optional)
[description]. Defaults to None.

assignees ([type], optional)
[description]. Defaults to None.

tags ([type], optional)
[description]. Defaults to None.

status (str, optional)
[description]. Defaults to None.

due_date (str, optional)
[description]. Defaults to None.

start_date (str, optional)
[description]. Defaults to None.

notify_all (bool, optional)
[description]. Defaults to True.

Raises:

exceptions.ClickupClientError
[description]

Returns:

models.Task
[description]

update_task()

ClickUpClient.**update_task**(*task_id*, *name: Optional[str] = None*, *description: Optional[str] = None*, *status: Optional[str] = None*, *priority: Optional[int] = None*, *time_estimate: Optional[int] = None*, *archived: Optional[bool] = None*, *add_assignees: Optional[List[str]] = None*, *remove_assignees: Optional[List[int]] = None*) → *Task*

[summary]

Args:

task_id ([type])
The ID of the ClickUp task to update.

name (str, optional)
Sting value to update the task name to. Defaults to None.

description (str, optional)
Sting value to update the task description to. Defaults to None.

status (str, optional)
String value of the tasks status. Defaults to None.

priority (int, optional)
Priority of the task. Range 1-4. Defaults to None.

time_estimate (int, optional)
Time estimate of the task. Defaults to None.

archived (bool, optional)
Whether the task should be archived or not. Defaults to None.

add_assignees (List[str], optional)
List of assignee IDs to add to the task. Defaults to None.

remove_assignees (List[int], optional)
List of assignee IDs to remove from the task. Defaults to None.

Raises:

exceptions.ClickupClientError
Raises "Priority out of range" exception for invalid priority range.

Returns:

models.Task
Returns an object of type Task.

delete_task()

`ClickUpClient.delete_task(task_id: str) → None`

Deletes a task via a given task ID.

Args:

folder_id (str)
The ID of the ClickUp task to delete.

get_task_comments()

`ClickUpClient.get_task_comments(task_id: str) → Comments`

Get all the comments for a task from a given task id.

Args:

task_id (str)
The id of the ClickUp task to retrieve comments from.

Returns:

models.Comments
Returns an object of type Comments.

2.1.2 Lists

class clickupython.models.**SingleList**(*, *id: str = None*, *name: str = None*, *deleted: bool = None*, *archived: bool = None*, *orderindex: int = None*, *override_statuses: bool = None*, *priority: Priority = None*, *assignee: Asssignee = None*, *due_date: str = None*, *start_date: str = None*, *folder: ListFolder = None*, *space: ListFolder = None*, *statuses: List[StatusElement] = None*, *inbound_address: str = None*, *permission_level: str = None*, *content: str = None*, *status: Status = None*, *task_count: int = None*, *start_date_time: str = None*, *due_date_time: bool = None*)

Bases: BaseModel

archived: bool

assignee: Asssignee

build_list()

content: Optional[str]

deleted: bool

due_date: str

due_date_time: Optional[bool]

folder: ListFolder

id: str

inbound_address: str

name: str

orderindex: int

override_statuses: bool

permission_level: str

priority: Optional[Priority]

space: ListFolder

start_date: str

start_date_time: Optional[str]

status: Optional[Status]

statuses: Optional[List[StatusElement]]

task_count: Optional[int]

Creating a List

Here's how to create a new ClickupClient instance and validate a personal API key with the ClickUp API and create a new task with a due date. When creating a new task, the only required arguments are list_id and name. Name will be the title of your list on ClickUp.

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
t = c.create_task("LIST_ID", name="Test Task", due_date="march 2 2021")
```

Fetching a Single Task

Example: Lookup via ClickUpClient:

```
c = client.ClickUpClient("YOUR_API_KEY")
task = c.get_task(task_id)

print(task.name)
```

Fetching all Tasks from a List

You can quickly get all tasks for a given list via the list id:

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks("list_id")
```

Filtering Tasks

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks("list_id", date_updated_gt="august 1 2021",
       assignees=["4523","4562","5871"], include_closed=True)
```

This example will return all tasks that have been updated after August 1st, 2021 and are assigned to users with the ids of 4523, 4562, and 5871. This request will also include tasks that have been marked as "closed."

Example:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks("list_id", subtasks=True,
        statuses=["todo", "in progress"])
```

This example will return all tasks and subtasks that are marked as "Todo" and "In Progress". These values can be changed depending on the statuses you have available in your list.

You can extend the calls above by passing any of the arguments to the `get_tasks` method. You can use as many or as few as you would like.

Working With Tasks

Now that you have a list of `Task` objects you can access the attributes of each task in a number of ways:

Example: Loop:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks(list_id)

for task in tasks:
   print(task.name)
```

Example: Direct Access via an Index:

```
c = client.ClickUpClient("YOUR_API_KEY")
tasks = c.get_tasks(list_id)

print(tasks[0].name])
```

Getting Tasks Associated with a List Object

Certain calls can be made directly from a parent object. We can access a single `Task` or all `Tasks` associated with a `List` with the following methods.

Note: IMPORTANT - When calling a method from a parent object you must pass in a reference to the ClickUpClient object as the first argument.

Example: Lookup Tasks via a List Object:

```
c = client.ClickUpClient("YOUR_API_KEY")
list = c.get_list(list_id)
tasks = list.get_tasks(c)
filtered_tasks = list.get_tasks(c, subtasks=True, statuses=["todo", "in progress"])
task = list.get_task(c, task_id)
```

List Methods

get_list()

ClickUpClient.**get_list**(*list_id: str*) → *SingleList*

Fetches a single list item from a given list id and returns a List object.

Args:

list_id (str)
The id of the ClickUp list.

Returns:

models.SingleList
Returns an object of type List.

get_lists()

ClickUpClient.**get_lists**(*folder_id: str*) → AllLists

Fetches all lists from a given folder id and returns a list of List objects.

Args:

folder_id (str)
The ID of the ClickUp folder to be returned.

Returns:

list.AllLists
Returns a list of type AllLists.

create_list()

ClickUpClient.**create_list**(*folder_id: str*, *name: str*, *content: str*, *due_date: str*, *priority: int*, *status: str*) → *SingleList*

Creates and returns a List object in a folder from a given folder ID.

Args:

folder_id (str)
The ID of the ClickUp folder.

name (str)
The name of the created list.

content (str)
The description content of the created list.

due_date (str)
The due date of the created list.

priority (int)
An integer 1 : Urgent, 2 : High, 3 : Normal, 4 : Low.

status (str)
Refers to the List color rather than the task Statuses available in the List.

Returns:

list.SingleList
Returns an object of type SingleList.

INDEX

T

U