wakapy Documentation

Release 0.1

surister

Mar 13, 2019

Contents

Welcome to the wakapy documentation, a Python 3.6+ library that facilitates your WakaTime data processing by bundling the data into a convenient OOP scheme.

CHAPTER 1

Documentation Contents

1.1 Introduction

wakapy is a python library aiming for Python 3.6+ versions whose purpose is to provide easy Wakatime data manipulation to the developer.

Note:

1. This project is not related to WakaTime or its developer team in any way.
2. This project revolves around the json file that you can *download* from your **Wakatime account**

Wakapy basically loads the **big** json file containing all of your data provided for free by WakaTime and group every piece of data in convenient classes. The Wakatime json file is big, a 161 days file, where only 121 days actually contain relevant data is roughly 75k long (in my case).

1.1.1 Wakapy features:

1. **Extensive data class containerization** (Every bit of data from the json file is accessible with the library).

2. **Extra functionalities** added to ease the data manipulation.

3. **Date slicing**, in other words, you can get the data from a chosen range, similar to the Wakatime *paid features*

4. Some nice **charts** out of the box for the people who just want to get a quick insight of the data without putting too much effort on it

1.2 Installation and Set up

1.2.1 Downloading the library:

Using bare git:

```
$ git clone https://github.com/surister/wakapy.git
```

Using pip:

```
$ pip install wakapy
```

Using pipenv:

```
$ pipenv install wakapy
```

1.2.2 Downloading the Wakatime data:

1. Log in Wakatime
2. Settings
3. Export my coding activities

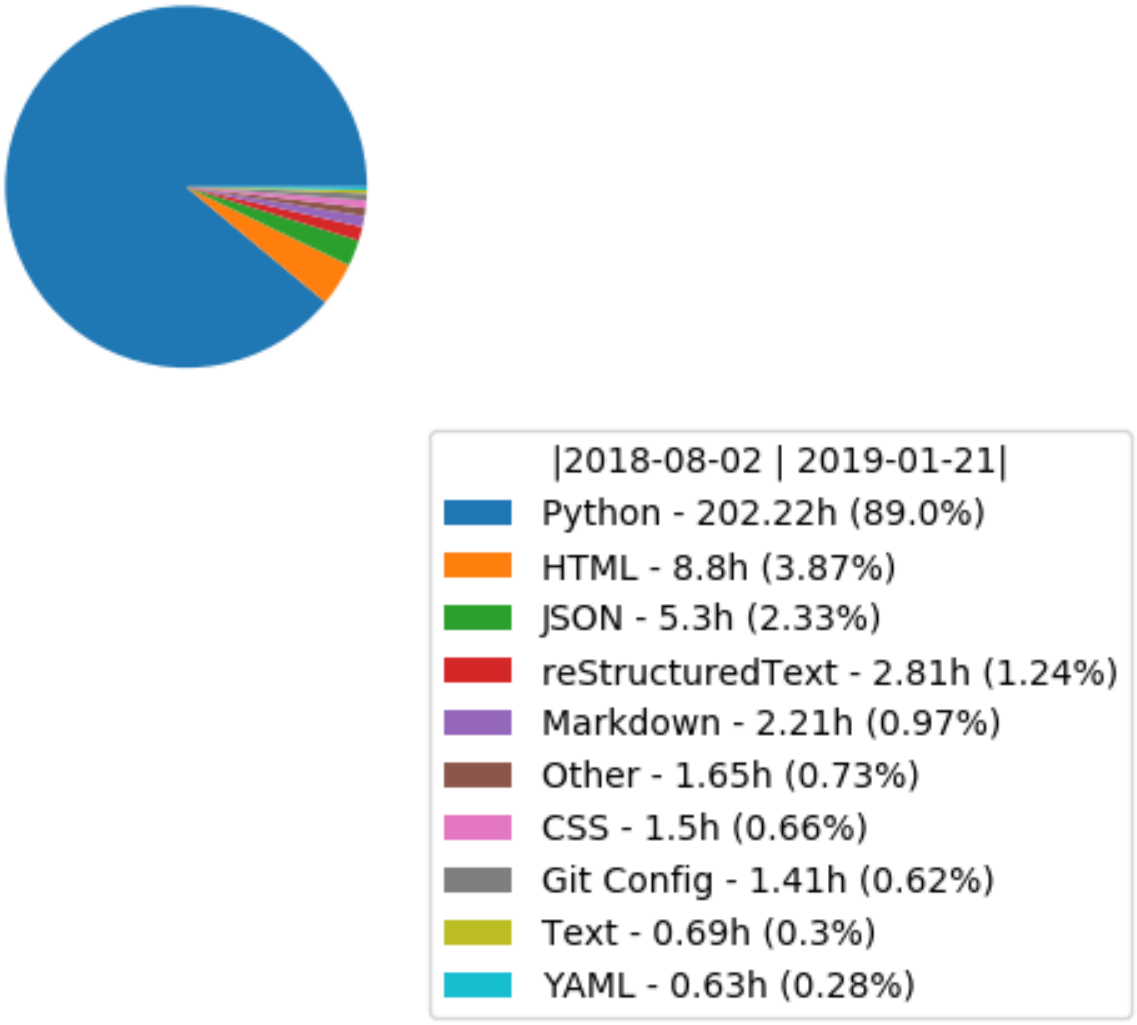
Note: Keep the data in the original JSON format.

1.3 Basic examples

1.3.1 Example 1: Basic charting usage

```
from wakapy import User

user = User('/home/surister/data.json')
chart = user.pie_chart('lan')
# lan = languages. See the different options
# in Day.raw_containers
chart.show()

# Shows the chart.
chart.save('/home/surister/mychart.png')
# Saves the chart to the desired filepath.
```

output:



1.3.2 Example 2: Date slices in chart

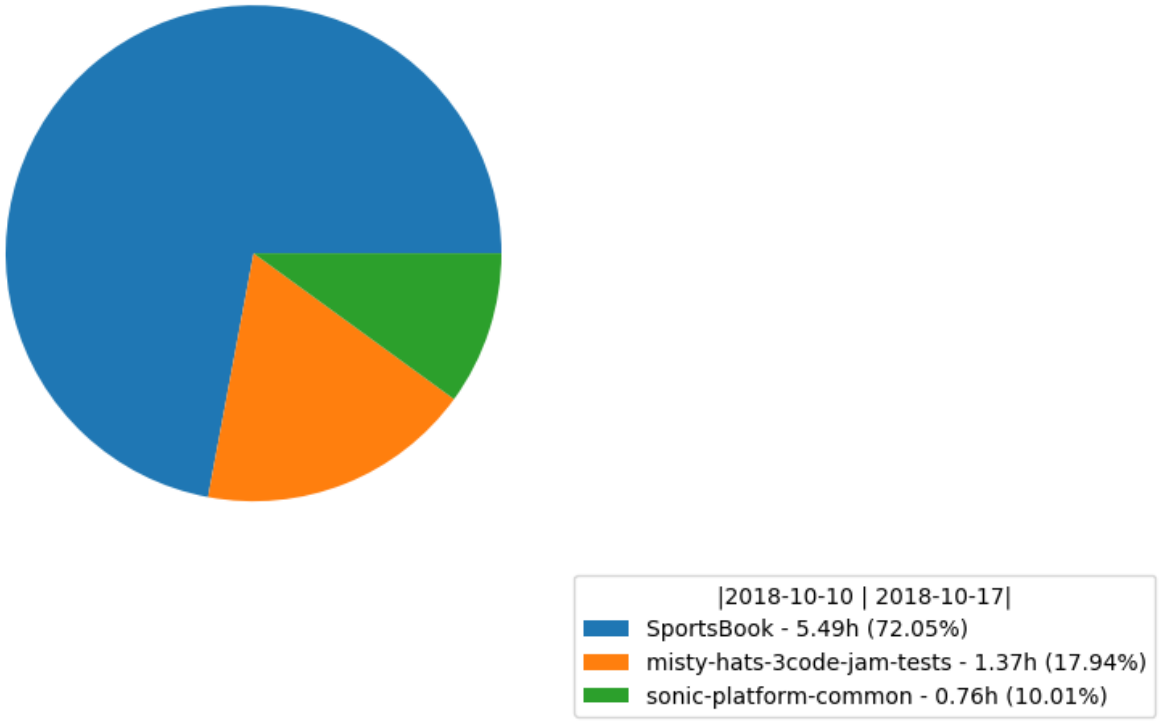
```
from datetime import date

from wakapy import User

date1 = '2018-10-10'
date2 = date(year=2018, month=10, day=17)
# Dates can either be a str or a datetime.date object.

a = User('/home/surister/info.json')

a_slice = a.get_slice(date1, date2)
```

(continues on next page)

(continued from previous page)

```
# Returns a Slice object containing
# the Days object between the two given dates

chart = a.pie_chart('proj', num=3, sliced=True)
# Num is the number of projets that will be displayed
# Sliced is set to True, so the chart will be created with
# sliced object created before.

chart.save('/home/surister/mychart.png')
```

output:



1.3.3 Example 3: Creating your own chart

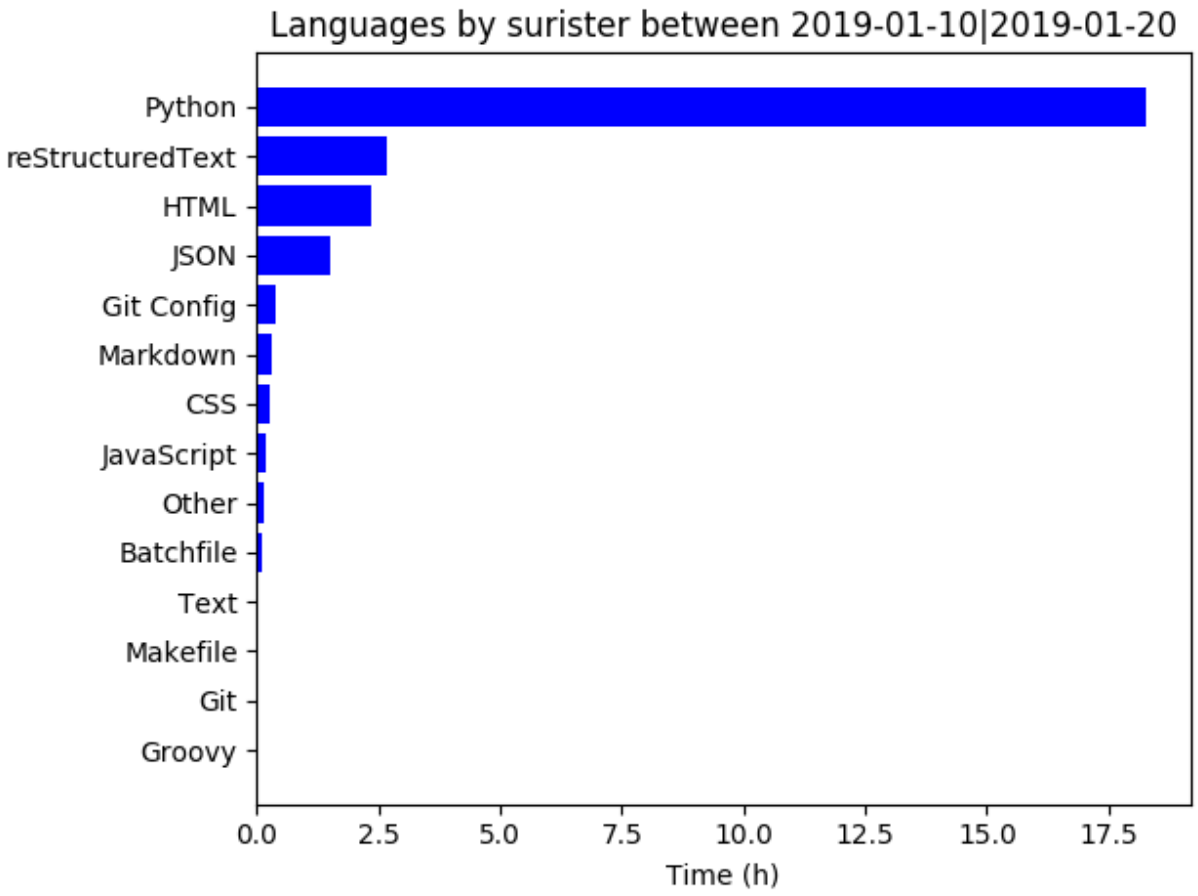
```
from datetime import date

from wakapy import User

import matplotlib.pyplot as plt
import numpy as np

date1 = date(year=2019, month=1, day=10)
date2 = date(year=2019, month=1, day=20)

user = User('/home/surister/info.json')
user_slice = user.get_slice(date1, date2)
```

(continues on next page)

(continued from previous page)

```
data = user.fetch_data('lan', True)

# You could use the fetch_data function
# or iterate yourself:

data2 = {}
for day in user_slice:
    if not day.is_empty:
        for container in day.container_dict['lan']:
            if container.name not in data2.keys():
                data2[container.name] = container.total_time
            else:
                data2[container.name] += container.total_time

# In this case the same data1 and data2 have the same values.
# Note that fetch_data returns an ordered dict
# while data2 would not be ordered.


fig, ax = plt.subplots()

keys = data.keys()

y_pos = np.arange(len(keys))

hours = list(map(lambda x: round(x / 3600, 2), data.values()))
# We convert the seconds into hours


ax.barh(y_pos, hours, align='center',
        color='blue')
ax.set_yticks(y_pos)
ax.set_yticklabels(keys)
ax.invert_yaxis()  # labels read top-to-bottom
ax.set_xlabel(f'Time (h)')
ax.set_title(f'Languages by {user.username} between {date1}|{date2}')

plt.show()
```

output:



Note: These examples and images can also be found in the examples folder

1.4 Api reference

Note:

The data containers are divided in two sections:

1. Containers: They contain data and have some functionalities and processing properties.
2. Containers-bare: They do not have any special functionality other than being datacontainers.

1.4.1 Containers

Main classes, data containers with functionalities. |

JsonDict

class wakapy.parser.**JsonDict**(*fp: str*)
Class that extracts the data from the json file.

Parameters **fp** (str) – file path, received from *User*

fp
str

file
dict The whole data.

days
property – :return: List[*Day*] The list of days that will be stored in *User*

user_data
property – :return: dict User's data, doesn't containt any *Day*

User

class wakapy.**User**(*fp: str*)
Represents a user containing all the data from source file

Parameters **fp** (str) – The file path to the Wakatime json file

fp
str - File path

file
The *JsonDict* containing all the data.

display_name
str - User's wakatime displayed name, usually @ + **username**.

created_at
str - User's creation date, eg: *2018-08-02T15:40:06Z*.

email
str - User's email

is_email_public
bool - True if your email is set to public in wakatime settings.

full_name
str - Full name if available in wakatime settings.

has_premium_features
bool - True if user has premium features.

human_readable_website
str - User's website

id
str - User's id

is_email_confirmed
bool - True if user's email is confirmed

is_hireable
bool - True if user is set to hireable in wakatime

languages_used_public
bool

last_heartbeat
str - Last time the user connected to wakatime

last_plugin
str - Last plugin used by the user - Verbose

last_plugin_name
str - Last plugin used by the user

last_project
str - Last project the user worked on

location
str - user's location if given

logged_time_public
bool

modified_at
Optional[str, NoneType] - Last time user was modified

photo
str - user's wakatime gravatar profile picture

photo_public
bool - True if user's profile picture is public

plan
str - user's wakatime paid plan, most common: basic.

timeout
int

timezone
str - user's timezone

username
str - user's username

website
str - user's website if given

writes_only
bool

slice
default NoneType contains a list of days between 2 dates.

days
property – :return: List[*Day*] - list containing **every** Day class

fetch_data(*to_fetch: str*, *sliced: bool*) → dict

Parameters

- **to_fetch** – str container to search options are *Day*.container_dict.
- **sliced** – bool True if you want to use the data from the sliced object you created beforehand.

Returns dict containing the data.

Note: The fetched data is the total_time of each *Day* container.(s).

This is an example of it:

```
# 'lan' - language
data = {'python': 253, 'bash': 623}
```

get_slice(*date1=None*, *date2=None*) → wakapy.user.Slice

Dates Default NoneType, can be either str with the following format: YYYY-MM-DD or a datetime.date object

Returns *Slice*

pie_chart(*to_fetch: str*, *num: int = 10*, *sliced=False*) → wakapy.plot.PieChart
Creates a pie chart with the given parameters.

Parameters

- **to_fetch** – str container to search options are *Day*.container_dict.
- **num** – int number of items
- **sliced** – bool True if you want to use the data from the sliced object you created beforehand.

Returns *PieChart*

total_worked_days
property – :return: int - The total amount of days that have activity

Day

class wakapy.day.**Day**(*_dict: dict*)
Represents a day. It contains all the data regarding a specific day. Every date is unique.

Parameters **_dict** (dict) – A dictionary containing all the data within a one day range.

date
str The day's date, main identifier of each instance, Unique.

is_empty
bool True if the day doesn't contain any data.

year
str date's year.

month
str date's month.

day
str date's day.

container_dict
dict shortcut reference to the class containers.

- os: operative_systems
- lan: languages
- ent: entities

- edit: editors
- cat: categories
- proj: projects

operative_systems
List[*Os*] List containing every Os object, representing every s that was used this **date**.

languages
List[*Language*] List containing every Language object, representing every programming language that was used this **date**.

entities
List[*Dependency*] List containing every Dependency object, representing every dependency that was used this **date**.

editors
List[*Editor*] List containing every Editor object, representing every editor that was used this **date**.

categories
List[*Category*] List containing every Category object, representing every category that was used this **date**.

projects
List[*Project*] List containing every Project object, representing every project that was worked on this **date**.

split(*splitwith: str*) → str
Allows the direct **date** split.

Parameters **splitwith** – str str.split(splitwith)

Note: This is the same as: day.date.split(‘’)

Returns str

Slice

class wakapy.user.**Slice**(*days*, *date_1=None*, *date_2=None*)
Given two valid dates it’ll slice and save them. It always includes the given dates.

Parameters

- **days** (List[*Day*]) – The list of Day objects to slice by dates.
- **dates** (Union[str, datetime.date]) – date_1 has to be lower than date_2 and both have to follow the YYYY-MM-DD format.

Note: str given dates will be converted to datetime.date, so both dates can be given in any of those 2 formats independently. The slicing will be done automatically upon creating the class with two valid dates. This class can be iterated, yields *Day* from *days()*

So this twos are equivalent:

```
myslice = Slice(dates_list, date1, date2)

for day in myslice:
```

(continues on next page)

(continued from previous page)

```
    print(day.date)

for day in myslice.days:
    print(day.date)
```

---

date_1 see dates.

date_2 see dates.

days
property – :return: List[*Day*] sliced.

1.4.2 Containers-bare

The following classes are the static data containers

Parent

class wakapy.containers.**Parent**(*_dict*)
Parent for all bare containers.

Parameters **_dict** (dict) – Dictionary containing every specific container data.

digital
str Time in digital format

name
str Name, example: 'Python' if os, 'Pycharm' if editor...

hours
int hours

minutes
int minutes

seconds
int seconds

percent
float Percentage

text
str Time written in text format, ex: 5 minutes.

total_time
int total time in seconds

type
Usually NoneType

Project

class wakapy.containers.**Project**(*_dict*)
Represents a Project. It contains every container-bare plus three more.

This class adds overall more accuracy in the timings within a *Day*

Parameters **_dict** (dict) – Dictionary that contains all the data.

name
str Name of the project

num
int If a project is untitled, they are counted as <untitled-num>

branches
List[*Branch*] list of worked branches within the Project

operative_systems
List[*Os*] list containing every Os object, representing every s that was used this **date**.

languages
List[*Language*] list containing every Language object, representing every programming language that was used this **date**.

entities
List[*Dependency*] list containing every Dependency object, representing every dependency that was used this **date**.

editors
List[*Editor*] list containing every Editor object, representing every editor that was used this **date**.

categories
List[*Category*] list containing every Category object, representing every category that was used this **date**.

dependencies
List[*Dependency*] list containing every Dependency object, representing every dependency that was used this **date**.

grand_total
dict Contains the total time data

Child Containers

The following classes are all child's of *Parent*

class wakapy.containers.**Os**(*_dict*)
Class representing a wakatime Operative System data within a one day range.

class wakapy.containers.**Language**(*_dict*)
Class representing a wakatime Programming language data within a one day range.

class wakapy.containers.**Entity**(*_dict*)
Class representing a wakatime Entity data within a one day range. A entity can be a module, dependency.. it may vary from plugin to plugin. It's only present in *Project*

class wakapy.containers.**Dependency**(*_dict*)
Class representing a wakatime Entity data within a one day range. A entity can be a module, dependency.. it may vary from plugin to plugin.

class wakapy.containers.**Category**(*_dict*)
Class representing a wakatime Category data within a one day range, usually 'Coding'

class wakapy.containers.**Editor**(*_dict*)
Class representing a wakatime Editor data within a one day range, it does not distinguish between editors and IDEs

class wakapy.containers.**Branch**(*_dict*)
Class representing a wakatime Branch data within a one day range. It's only present in *Project*

class wakapy.containers.**GrandTotal**(*_dict*)

Class representing a wakatime Grand total data within a one day range. It's only present in *Project*

1.4.3 Extra

PieChart

class wakapy.plot.**PieChart**(*_dict*, *num: int*, *dates=None*)
Matplotlib pie chart that comes with the library for the people that just want a quick insight of their data.

Parameters

- **_dict** (dict) – The dictionary containing all the data that will be showed in the piechart.
- **num** (int) – Number of max elements that will be shown in the chart
- **dates** (datetime.date) – The dates if the data comes from a slice. Just for tagging purposes.

Note: You are suppose to access this by *pie_chart()*

---

save(*fp: str*) → None
Saves the figure in the given path. matplotlib.pyplot.save(fp)

Parameters **fp** – str

show() → None
Shows the figure. matplotlib.pyplot.show

Utils

utils.**order_dict**(*reverse: bool = True*) → dict
Takes a dictionary and returns it ordered.

Parameters

- **_dict** – dict
- **reverse** – bool

Returns dict

Index