
Python Myfitnesspal

Contributors like you

Jun 16, 2023

CONTENTS

1	Getting Started	1
1.1	Installation	1
1.2	Authentication	1
2	Using the Command-Line API	3
2.1	day [\$DATE]	3
3	How-to Guides	5
3.1	Accessing your Diary	5
3.2	Accessing your Exercises	7
3.3	Accessing your Measurements	8
3.4	Searching for Foods	10
3.5	Accessing Reports	10
3.6	Use on Windows Subsystem for Linux	12
4	Contributing	13
4.1	I think I've found a bug	13
4.2	I want to add a new feature	13
4.3	I have a feature idea	14
5	API	15
5.1	Client	15
5.2	Day	17
5.3	Meal	17
5.4	Entry	18
5.5	Exercise	18
5.6	Food Item	18
5.7	Serving	20
5.8	Note	20
5.9	Exceptions	21
5.10	Types	21
6	Upgrading from 1.x to 2.x	27
6.1	Version 1.x (Obsolete)	27
6.2	Version 2.x (Current)	27
7	Indices and tables	29
	Python Module Index	31
	Index	33

GETTING STARTED

1.1 Installation

You can either install from pip:

```
pip install myfitnesspal
```

or checkout and install the source from the [github repository](#):

```
git clone https://github.com/coddingtonbear/python-myfitnesspal.git
cd python-myfitnesspal
python setup.py install
```

1.2 Authentication

This library uses your local browser's MyFitnessPal cookies for interacting with MyFitnessPal via the [browser_cookie3](#) library. To control which user account this library uses for interacting with MyFitnessPal, just log in to the appropriate account in your browser and, with a bit of luck, python-myfitnesspal should be able to find the authentication credentials needed.

By default, this library will look for cookies set for the [www.myfitnesspal.com](#) and [myfitnesspal.com](#) domains in all browsers supported by [browser_cookie3](#). You can control which cookiejar is used by passing a [http.cookiejar.CookieJar](#) object via the constructor's *cookiejar* keyword parameter. See [browser_cookie3's readme](#) for details around how you might select a cookiejar from a particular browser.

Note: Starting on August 25th, 2022, MyFitnessPal added a hidden captcha to their login flow. That change unfortunately prevents this library from logging-in directly, and as of version 2.0 of python-myfitnesspal, this library now relies on reading browser cookies directly for gathering login credentials.

See [Issue #144](#) for details and context.

USING THE COMMAND-LINE API

Although most people will probably be using Python-MyFitnessPal as a way of integrating their MyFitnessPal data into another application, Python-MyFitnessPal does provide a command-line API with a single command described below.

2.1 day [\$DATE]

Display meals and totals for a given date. If no date is specified, totals will be printed for today.

HOW-TO GUIDES

3.1 Accessing your Diary

To access a single day's information:

```
import myfitnesspal

client = myfitnesspal.Client()

day = client.get_date(2013, 3, 2)
day
# >> <03/02/13 {'sodium': 3326, 'carbohydrates': 369, 'calories': 2001, 'fat': 22, 'sugar': 103,
↳ 'protein': 110}>
```

To see all meals you can use the Day object's meals property:

```
day.meals
# >> [<Breakfast {}>,
#     <Lunch {'sodium': 712, 'carbohydrates': 106, 'calories': 485, 'fat': 3, 'sugar': 0,
↳ 'protein': 17}>,
#     <Dinner {'sodium': 2190, 'carbohydrates': 170, 'calories': 945, 'fat': 11, 'sugar': 17,
↳ 'protein': 53}>,
#     <Snacks {'sodium': 424, 'carbohydrates': 93, 'calories': 571, 'fat': 8, 'sugar': 86,
↳ 'protein': 40}>]
```

To access dinner, you can access it by its index in day.meals:

```
dinner = day.meals[2]
dinner
# >> <Dinner {'sodium': 2190, 'carbohydrates': 170, 'calories': 945, 'fat': 11, 'sugar': 17,
↳ 'protein': 53}>
```

To get a list of things you ate for dinner, I can use the dinner Meal object's entries property:

```
dinner.entries
# >> [<Montebello - Spaghetti noodles, 6 oz. {'sodium': 0, 'carbohydrates': 132, 'calories': 630,
↳ 'fat': 3, 'sugar': 3, 'protein': 21}>,
#     <Fresh Market - Arrabiatta Organic Pasta Sauce, 0.5 container (3 cups ea.) {'sodium': 1410,
↳ 'carbohydrates': 24, 'calories': 135, 'fat': 5, 'sugar': 12, 'protein': 6}>,
#     <Quorn - Meatless and Soy-Free Meatballs, 6 -4 pieces (68g) {'sodium': 780,
↳ 'carbohydrates': 14, 'calories': 180, 'fat': 3, 'sugar': 2, 'protein': 26}>]
```

To access one of the items, use the entries property as a list:

```
spaghetti = dinner.entries[0]
spaghetti.name
# >> Montebello - Spaghetti noodles, 6 oz.
```

For a daily summary of your nutrition information, you can use a Day object's totals property:

```
day.totals
# >> {'calories': 2001,
#      'carbohydrates': 369,
#      'fat': 22,
#      'protein': 110,
#      'sodium': 3326,
#      'sugar': 103}
```

Or, if you just want to see how many milliliters of water you've recorded, or the notes you've entered for a day:

```
day.water
# >> 1
day.notes
# >> "This is the note I entered for this day"
```

For just one meal:

```
dinner.totals
# >> {'calories': 945,
#      'carbohydrates': 170,
#      'fat': 11,
#      'protein': 53,
#      'sodium': 2190,
#      'sugar': 17}
```

For just one entry:

```
spaghetti.totals
# >> {'calories': 630,
#      'carbohydrates': 132,
#      'fat': 3,
#      'protein': 21,
#      'sodium': 0,
#      'sugar': 3}
```

3.1.1 Hints

Day objects act as dictionaries:

```
day.keys()
# >> ['Breakfast', 'Lunch', 'Dinner', 'Snack']
lunch = day['Lunch']
print lunch
# >> [<Generic - Ethiopian - Miser Wat (Red Lentils), 2 cup {'sodium': 508, 'carbohydrates
↳ ': 76, 'calories': 346, 'fat': 2, 'sugar': 0, 'protein': 12}>,
```

(continues on next page)

(continued from previous page)

```
#      <Injera - Ethiopian Flatbread, 18 " diameter {'sodium': 204, 'carbohydrates': 30,
↳ 'calories': 139, 'fat': 1, 'sugar': 0, 'protein': 5}>]
```

Meal objects act as lists:

```
len(lunch)
# >> 2
miser_wat = lunch[0]
print miser_wat
# >> <Generic - Ethiopian - Miser Wat (Red Lentils), 2 cup {'sodium': 508, 'carbohydrates': 76,
↳ 'calories': 346, 'fat': 2, 'sugar': 0, 'protein': 12}>
```

and Entry objects act as dictionaries:

```
print miser_wat['calories']
# >> 346
```

and, since the measurement units returned are not necessarily very intuitive, you can enable or disable unit awareness using the `unit_aware` keyword argument.

```
client = myfitnesspal.Client(unit_aware=True)
day = client.get_date(2013, 3, 2)
lunch = day['lunch']
print lunch
# >> [<Generic - Ethiopian - Miser Wat (Red Lentils), 2 cup {'sodium': Weight(mg=508),
↳ 'carbohydrates': Weight(g=76), 'calories': Energy(Calorie=346), 'fat': Weight(g=2), 'sugar'
↳ ': Weight(g=0), 'protein': Weight(g=12)}>,
miser_wat = lunch[0]
print miser_wat['calories']
# >> Energy(Calorie=346)
```

3.2 Accessing your Exercises

Exercises are accessed through the `day.exercises` command - giving an 2-item array of [`<Cardiovascular>`, `<Strength>`], which can be explored using `get_as_list()`

To get a list of cardiovascular exercises

```
import myfitnesspal

client = myfitnesspal.Client()

day = client.get_date(2019, 3, 12)

day.exercises[0].get_as_list()
# >> [{'name': 'Walking, 12.5 mins per km, mod. pace, walking dog', 'nutrition_information':
↳ {'minutes': 60, 'calories burned': 209}}, {'name': 'Running (jogging), 8 kph (7.5 min per
↳ km)', 'nutrition_information': {'minutes': 25, 'calories burned': 211}}]
```

And then access individual properties

```
day.exercises[0].get_as_list()[0]['name']

# >> 'Walking, 12.5 mins per km, mod. pace, walking dog'

day.exercises[0].get_as_list()[0]['nutrition_information']['minutes']
# >> 60

day.exercises[0].get_as_list()[0]['nutrition_information']['calories burned']
# >> 209
```

To get a list of strength exercises

```
import myfitnesspal

client = myfitnesspal.Client()

day = client.get_date(2019, 3, 12)

day.exercises[1].get_as_list()
# >> [{'name': 'Leg Press', 'nutrition_information': {'sets': 3, 'reps/set': 12, 'weight/set': 20}},
      {'name': 'Seated Row, Floor, Machine', 'nutrition_information': {'sets': 3, 'reps/set': 12, 'weight/set': 20}}]
```

And then access individual properties

```
day.exercises[1].get_as_list()[0]['name']
# >> 'Leg Press'

day.exercises[1].get_as_list()[0]['nutrition_information']['sets']
# >> 3

day.exercises[1].get_as_list()[0]['nutrition_information']['reps/set']
# >> 12

day.exercises[1].get_as_list()[0]['nutrition_information']['weight/set']
# >> 20
```

3.3 Accessing your Measurements

To access measurements from the past 30 days:

```
import myfitnesspal

client = myfitnesspal.Client()

weight = client.get_measurements('Weight')
weight
# >> OrderedDict([(datetime.date(2015, 5, 14), 171.0), (datetime.date(2015, 5, 13), 173.8),
                  (datetime.date(2015, 5, 12), 171.8),
                  (datetime.date(2015, 5, 11), 171.6), (datetime.date(2015, 5, 10), 172.4),
                  (datetime.date(2015, 5, 9), 170.2),
```

(continues on next page)

(continued from previous page)

```
#             (datetime.date(2015, 5, 8), 171.0), (datetime.date(2015, 5, 7), 171.
↪2), (datetime.date(2015, 5, 6), 170.8),
#             (datetime.date(2015, 5, 5), 171.8), (datetime.date(2015, 5, 4), 174.
↪2), (datetime.date(2015, 5, 3), 172.2),
#             (datetime.date(2015, 5, 2), 171.0), (datetime.date(2015, 5, 1), 171.
↪2), (datetime.date(2015, 4, 30), 171.6),
#             (datetime.date(2015, 4, 29), 172.4), (datetime.date(2015, 4, 28), 172.
↪2), (datetime.date(2015, 4, 27), 173.2),
#             (datetime.date(2015, 4, 26), 171.8), (datetime.date(2015, 4, 25), 170.
↪8), (datetime.date(2015, 4, 24), 171.2),
#             (datetime.date(2015, 4, 23), 171.6), (datetime.date(2015, 4, 22), 173.
↪2), (datetime.date(2015, 4, 21), 174.2),
#             (datetime.date(2015, 4, 20), 173.6), (datetime.date(2015, 4, 19), 171.
↪8), (datetime.date(2015, 4, 18), 170.4),
#             (datetime.date(2015, 4, 17), 169.8), (datetime.date(2015, 4, 16), 170.
↪4), (datetime.date(2015, 4, 15), 170.8),
#             (datetime.date(2015, 4, 14), 171.6)])
```

To access measurements since a given date:

```
import datetime

may = datetime.date(2015, 5, 1)

body_fat = client.get_measurements('Body Fat', may)
body_fat
# >> OrderedDict([(datetime.date(2015, 5, 14), 12.8), (datetime.date(2015, 5, 13), 13.1),
↪ (datetime.date(2015, 5, 12), 12.7),
#             (datetime.date(2015, 5, 11), 12.7), (datetime.date(2015, 5, 10), 12.8),
↪ (datetime.date(2015, 5, 9), 12.4),
#             (datetime.date(2015, 5, 8), 12.6), (datetime.date(2015, 5, 7), 12.7),
↪ (datetime.date(2015, 5, 6), 12.6),
#             (datetime.date(2015, 5, 5), 12.9), (datetime.date(2015, 5, 4), 13.0),
↪ (datetime.date(2015, 5, 3), 12.6),
#             (datetime.date(2015, 5, 2), 12.6), (datetime.date(2015, 5, 1), 12.7)])
```

To access measurements within a date range:

```
thisweek = datetime.date(2015, 5, 11)
lastweek = datetime.date(2015, 5, 4)

weight = client.get_measurements('Weight', thisweek, lastweek)
weight
# >> OrderedDict([(datetime.date(2015, 5, 11), 171.6), (datetime.date(2015, 5, 10), 172.
↪4), (datetime.date(2015, 5, 9), 170.2),
#             (datetime.date(2015, 5, 8), 171.0), (datetime.date(2015, 5, 7), 171.
↪2), (datetime.date(2015, 5, 6), 170.8),
#             (datetime.date(2015, 5, 5), 171.8), (datetime.date(2015, 5, 4), 174.
↪2)])
```

Measurements are returned as ordered dictionaries. The first argument specifies the measurement name, which can be any name listed in the MyFitnessPal [Check-In](#) page. When specifying a date range, the order of the date arguments does not matter.

3.4 Searching for Foods

To search for items:

```
import myfitnesspal

client = myfitnesspal.Client()

food_items = client.get_food_search_results("bacon cheeseburger")
food_items
# >> [<Bacon Cheeseburger -- Sodexo Campus>,
# <Junior Bacon Cheeseburger -- Wendy's>,
# <Bacon Cheeseburger -- Continental Café>,
# <Bacon Cheddar Cheeseburger -- Applebees>,
# <Bacon Cheeseburger - Plain -- Homemade>,
# <Jr. Bacon Cheeseburger -- Wendys>,
# ...

print("{} ({}), {}, cals={}, mfp_id={}".format(
    food_items[0].name,
    food_items[0].brand,
    food_items[0].serving,
    food_items[0].calories,
    food_items[0].mfp_id
))
# > Bacon Cheeseburger (Sodexo Campus), 1 Sandwich, cals = 420.0
```

To get details for a particular food:

```
import myfitnesspal

client = myfitnesspal.Client()

item = client.get_food_item_details("89755756637885")
item.servings
# > [<1.00 x Sandwich>]
item.saturated_fat
# > 10.0
```

3.5 Accessing Reports

To access report data from the past 30 days:

```
import myfitnesspal

client = myfitnesspal.Client()

client.get_report(report_name="Net Calories", report_category="Nutrition")
# >> OrderedDict([(datetime.date(2015, 5, 14), 1701.0), (datetime.date(2015, 5, 13),
↳ 1732.8), (datetime.date(2015, 5, 12), 1721.8),
# (datetime.date(2015, 5, 11), 1701.6), (datetime.date(2015, 5, 10),
```

(continues on next page)

(continued from previous page)

```

↪1272.4), (datetime.date(2015, 5, 9), 1720.2),
#             (datetime.date(2015, 5, 8), 1071.0), (datetime.date(2015, 5, 7), 1721.
↪2), (datetime.date(2015, 5, 6), 1270.8),
#             (datetime.date(2015, 5, 5), 1701.8), (datetime.date(2015, 5, 4), 1724.
↪2), (datetime.date(2015, 5, 3), 1722.2),
#             (datetime.date(2015, 5, 2), 1701.0), (datetime.date(2015, 5, 1), 1721.
↪2), (datetime.date(2015, 4, 30), 1721.6),
#             (datetime.date(2015, 4, 29), 1072.4), (datetime.date(2015, 4, 28),
↪1272.2), (datetime.date(2015, 4, 27), 1723.2),
#             (datetime.date(2015, 4, 26), 1791.8), (datetime.date(2015, 4, 25),
↪1720.8), (datetime.date(2015, 4, 24), 1721.2),
#             (datetime.date(2015, 4, 23), 1721.6), (datetime.date(2015, 4, 22),
↪1723.2), (datetime.date(2015, 4, 21), 1724.2),
#             (datetime.date(2015, 4, 20), 1273.6), (datetime.date(2015, 4, 19),
↪1721.8), (datetime.date(2015, 4, 18), 1720.4),
#             (datetime.date(2015, 4, 17), 1629.8), (datetime.date(2015, 4, 16),
↪1270.4), (datetime.date(2015, 4, 15), 1270.8),
#             (datetime.date(2015, 4, 14), 1721.6)]]

```

```
import datetime
```

```
may = datetime.date(2015, 5, 1)
```

```
client.get_report("Net Calories", "Nutrition", may)
```

```

# >> OrderedDict([(datetime.date(2015, 5, 14), 172.8), (datetime.date(2015, 5, 13), 173.
↪1), (datetime.date(2015, 5, 12), 127.7),
#             (datetime.date(2015, 5, 11), 172.7), (datetime.date(2015, 5, 10), 172.
↪8), (datetime.date(2015, 5, 9), 172.4),
#             (datetime.date(2015, 5, 8), 172.6), (datetime.date(2015, 5, 7), 172.
↪7), (datetime.date(2015, 5, 6), 172.6),
#             (datetime.date(2015, 5, 5), 172.9), (datetime.date(2015, 5, 4), 173.
↪0), (datetime.date(2015, 5, 3), 172.6),
#             (datetime.date(2015, 5, 2), 172.6), (datetime.date(2015, 5, 1), 172.
↪7)])

```

To access report data within a date range:

```
thisweek = datetime.date(2015, 5, 11)
```

```
lastweek = datetime.date(2015, 5, 4)
```

```
client.get_report("Net Calories", "Nutrition", thisweek, lastweek)
```

```

# >> OrderedDict([(datetime.date(2015, 5, 11), 1721.6), (datetime.date(2015, 5, 10),
↪1722.4), (datetime.date(2015, 5, 9), 1720.2),
#             (datetime.date(2015, 5, 8), 1271.0), (datetime.date(2015, 5, 7), 1721.
↪2), (datetime.date(2015, 5, 6), 1720.8),
#             (datetime.date(2015, 5, 5), 1721.8), (datetime.date(2015, 5, 4), 1274.
↪2)])

```

Report data is returned as ordered dictionaries. The first argument specifies the report name, the second argument specifies the category name - both of which can be anything listed in the MyFitnessPal [Reports](#) page. When specifying a date range, the order of the date arguments does not matter.

3.6 Use on Windows Subsystem for Linux

When using python-myfitnesspal on Windows via Windows Subsystem for Linux you will encounter one extra complication when logging in due to the mechanism python-myfitnesspal uses for gathering authentication credentials.

You must log in to MyFitnessPal via a browser installed on your guest (Linux) installation instead of via one installed the conventional way.

CONTRIBUTING

4.1 I think I've found a bug

We'd love to hear about it – create a bug report [on github](#) and be sure to include:

- A description of what you were doing when the error occurred. Were you trying to fetch your diary, exercises, or measurements? Were you trying to look up food? The more we know about what you were doing, the easier it will be to find the problem.
- A traceback, if at all possible. Without a traceback, it's really hard to know what kind of problem you're running into, and given that a lot of what this library does is interact with individuals' accounts, reproducing the bug might not be possible, and a traceback will at least give hints about what's going wrong.

4.2 I want to add a new feature

You're a hero. New feature submissions are almost always welcome, but we recommend [starting a discussion](#) to make sure that this feature isn't already under development by somebody else, and to possibly talk through how you're planning to implement a feature to make sure it'll sail through the Pull Request Review process as smoothly as possible.

There are a few things to keep in mind for your submission in order to help it sail through the review process smoothly including:

- The build process will most likely inform you if you've missed something, but your submission should follow the project's style and standards including:
 - Proper typings for your methods and variables.
 - Following of the standard `black` style.
- Your Pull Request description should include a clear description of what problem your submission solves and why.
- Your submission should include a little documentation if at all possible including:
 - API Documentation covering your newly-added public methods and properties. This is mostly automated, luckily; just make sure you've added a docstring and proper types to your methods.
 - Maybe a "How to" article in the docs to show folks how to use the feature your new submission adds.
- If you want extra credit, some tests. We recognize that this is mostly an integration tool, and that testing such things is extremely tricky, but if you can think of a way of making your work testable, we'd all appreciate it.

Once you've developed your feature, post a Pull Request, and the community and collaborators will have a look at what you've put together and possibly post comments and/or suggestions for your consideration. You need at least one official maintainer's approval before the Pull Request can be merged into the codebase.

4.3 I have a feature idea

Start a [discussion](#) about it. Maybe you can find somebody to help you bring it to fruition.

5.1 Client

```
class myfitnesspal.Client(cookiejar: Optional[CookieJar] = None, unit_aware: bool = False)
    Provides access to MyFitnessPal APIs

    property user_id: Optional[str]
        The user_id of the logged-in account.

    property user_metadata: UserMetadata
        Metadata about of the logged-in account.

    property access_token: Optional[str]
        The access token for the logged-in account.

    property effective_username: str
        One's actual username may be different from the one used for login

        This method will return the actual username if it is available, but will fall back to the one provided if it is
        not.

    get_date(year: int, month: int, day: int) → Day
    get_date(date: date) → Day
        Returns your meal diary for a particular date

    get_measurements(measurement='Weight', lower_bound: Optional[date] = None, upper_bound:
        Optional[date] = None) → Dict[date, float]
        Returns measurements of a given name between two dates.

    set_measurements(measurement='Weight', value: Optional[float] = None, date: Optional[date] = None)
        → None
        Sets measurement for today's date.

    get_report(report_name: str = 'Net Calories', report_category: str = 'Nutrition', lower_bound:
        Optional[date] = None, upper_bound: Optional[date] = None) → Dict[date, float]
        Returns report data of a given name and category between two dates.

    get_food_search_results(query: str) → List[FoodItem]
        Search for foods matching a specified query.

    get_food_item_details(mfp_id: int) → FoodItem
        Get details about a specific food using its ID.
```

set_new_food(*brand: str, description: str, calories: int, fat: float, carbs: float, protein: float, sodium: Optional[float] = None, potassium: Optional[float] = None, saturated_fat: Optional[float] = None, polyunsaturated_fat: Optional[float] = None, fiber: Optional[float] = None, monounsaturated_fat: Optional[float] = None, sugar: Optional[float] = None, trans_fat: Optional[float] = None, cholesterol: Optional[float] = None, vitamin_a: Optional[float] = None, calcium: Optional[float] = None, vitamin_c: Optional[float] = None, iron: Optional[float] = None, serving_size: str = '1 Serving', servingspercontainer: float = 1.0, sharepublic: bool = False*) → None

Function to submit new foods / groceries to the MyFitnessPal database. Function will return True if successful.

set_new_goal(*energy: float, energy_unit: str = 'calories', carbohydrates: Optional[float] = None, protein: Optional[float] = None, fat: Optional[float] = None, percent_carbohydrates: Optional[float] = None, percent_protein: Optional[float] = None, percent_fat: Optional[float] = None*) → None

Updates your nutrition goals.

This Function will update your nutrition goals and is able to deal with multiple situations based on the passed arguments. First matching situation will be applied and used to update the nutrition goals.

Passed arguments - Hints: energy and all absolute macro values - Energy value will be adjusted/calculated if energy from macro values is higher than provided energy value. energy and all percentage macro values - Energy will be adjusted and split into macros by provided percentage. energy - Energy will be adjusted and split into macros by percentage as before.

Optional arguments: energy_unit - Function is able to deal with calories and kilojoules. If not provided user preferences will be used.

Additional hints: Values will be adjusted and rounded by MFP if no premium subscription is applied!

get_recipes() → Dict[int, str]

Returns a dictionary with all saved recipes.

Recipe ID will be used as dictionary key, recipe title as dictionary value.

get_recipe(*recipeid: int*) → *Recipe*

Returns recipe details in a dictionary.

See <https://schema.org/Recipe> for details regarding this schema.

get_meals() → Dict[int, str]

Returns a dictionary with all saved meals.

Key: Meal ID Value: Meal Name

get_meal(*meal_id: int, meal_title: str*) → *Recipe*

Returns meal details.

See <https://schema.org/Recipe> for details regarding this schema.

5.2 Day

```
class myfitnesspal.day.Day(date: date, meals: Optional[List[Meal]] = None, goals: Optional[Dict[str, float]] = None, notes: Optional[Callable[[], str]] = None, water: Optional[Callable[[], float]] = None, exercises: Optional[Callable[[], List[Exercise]]] = None, complete: bool = False)
```

Stores meal entries for a particular day.

```
__getitem__(value: str) → Meal
```

Returns a meal matching the provided name.

```
keys() → List[str]
```

Returns a list of meal names.

```
property meals: List[Meal]
```

Returns a list of meals.

```
property entries: Generator[Entry, None, None]
```

Yields all entries from all meals.

```
property goals: Dict[str, float]
```

Returns goals.

```
property notes: str
```

Returns notes.

```
property water: float
```

Returns water.

```
property exercises: List[Exercise]
```

Returns list of exercises.

```
get_as_dict() → Dict[str, List[MealEntry]]
```

Returns a mapping of meal names to the list of entries for that meal.

5.3 Meal

```
class myfitnesspal.meal.Meal(name: str, entries: List[Entry])
```

Stores information about a particular meal.

```
__getitem__(value: int) → Entry
```

Returns a particular entry for this meal.

```
property entries: List[Entry]
```

Entries for this meal.

```
property name: str
```

Name of this meal.

```
property totals: Dict[str, float]
```

Nutrition totals for all entries for this meal.

5.4 Entry

```
class myfitnesspal.entry.Entry(name: str, nutrition: Dict[str, float])
```

Stores information about a single entry.

```
property name: str
```

Name of the item.

```
property totals: Dict[str, float]
```

Totals for the item.

```
get_as_dict() → MealEntry
```

Returns totals per-item as a dictionary.

```
property short_name: Optional[str]
```

Short name.

```
property unit: Optional[str]
```

Unit.

```
property quantity: Optional[str]
```

Quantity.

5.5 Exercise

```
class myfitnesspal.exercise.Exercise(name: str, entries: List[Entry])
```

Shows information about your exercise.

```
__getitem__(value: int) → Entry
```

Returns a particular entry.

```
property entries: List[Entry]
```

List of entries.

```
property name: str
```

Name of exercise.

```
get_as_list() → List[MealEntry]
```

Returns exercises as a list of dictionaries.

5.6 Food Item

```
class myfitnesspal.fooditem.FoodItem(mfp_id: int, name: str, brand: Optional[str], verified: bool,  
                                     calories: float, details: Optional[FoodItemNutritionDict] = None,  
                                     confirmations: Optional[int] = None, serving_sizes:  
                                     Optional[List[ServingSizeDict]] = None, client: Optional[Client] =  
                                     None)
```

Stores information about a particular food item.

```
property details: FoodItemNutritionDict
```

Nutritional details.

```
property mfp_id: int
    Myfitnesspal ID for this item.
property name: str
    Name
property brand: Optional[str]
    Brand
property verified: bool
    Verified?
property serving: Optional[str]
    Serving
property calories: float
    Calories
property calcium: float
    Calcium
property carbohydrates: float
    Carbohydrates
property cholesterol: float
    Cholesterol
property fat: float
    Fat
property fiber: float
    Fiber
property iron: float
    Iron
property monounsaturated_fat: float
    Monounsaturated Fat
property polyunsaturated_fat: float
    Polyunsaturated Fat
property potassium: float
    Potassium
property protein: float
    Protein
property saturated_fat: float
    Saturated Fat
property sodium: float
    Sodium
property sugar: float
    Sugar
property trans_fat: float
    Trans Fat
```

```
property vitamin_a: float
    Vitamin A

property vitamin_c: float
    Vitamin C

property confirmations: int
    Confirmations

property servings: List[FoodItemServing]
    Servings
```

5.7 Serving

```
class myfitnesspal.fooditemserving.FoodItemServing(serving_id: str, nutrition_multiplier: float, value: float, unit: str, index: int)
```

```
property serving_id: str
    Serving ID

property nutrition_multiplier: float
    Nutrition Multiplier

property value: float
    Value

property unit: str
    Unit

property index: int
    Index
```

5.8 Note

```
class myfitnesspal.note.Note(note_data: NoteDataDict)
```

```
    Stores information about a note

property type: Optional[str]
    Type

property date: Optional[date]
    Date

as_dict()
    Returns data as a dictionary.
```


5.9 Exceptions

exception myfitnesspal.exceptions.MyfitnesspalError

exception myfitnesspal.exceptions.MyfitnesspalRequestFailed

exception myfitnesspal.exceptions.MyfitnesspalLoginError

5.10 Types

class myfitnesspal.types.CommandDefinition(**kwargs)

function: Callable

description: str

is_alias: bool

aliases: List[str]

class myfitnesspal.types.GoalDisplayDict(**kwargs)

id: str

display_type: str

nutrients: List[str]

class myfitnesspal.types.UnitPreferenceDict(**kwargs)

energy: str

weight: str

distance: str

height: str

water: str

class myfitnesspal.types.DiaryPreferencesDict(**kwargs)

default_foot_view: str

meal_names: List[str]

tracked_nutrients: List[str]

class myfitnesspal.types.UnitValueContainer(**kwargs)

unit: str

value: float

class myfitnesspal.types.GoalPreferencesDict(**kwargs)

workouts_per_week: int

```
    weekly_workout_duration: int
    weekly_exercise_energy: UnitValueContainer
    weight_change_goal: UnitValueContainer
    weight_goal: UnitValueContainer
    diary_goal_display: str
    home_goal_display: str
    macro_goal_format: str

class myfitnesspal.types.LocationPreferencesDict(**kwargs)
    time_zone: str
    country_code: str
    locale: str
    postal_code: str
    state: str
    city: str

class myfitnesspal.types.AdminFlagDict(**kwargs)
    status: str
    has_changed_username: bool
    forgot_password_or_username: bool
    warnings: int
    strikes: int
    revoked_privileges: List

class myfitnesspal.types.AccountDict(**kwargs)
    created_at: str
    updated_at: str
    last_login: str
    valid_email: bool
    registration_source: str
    roles: List[str]
    admin_flags: AdminFlagDict

class myfitnesspal.types.SystemDataDict(**kwargs)
    login_streak: int
```

```
    unseen_notifications: int

class myfitnesspal.types.UserProfile(**kwargs)

    type: str

    starting_weight_date: str

    starting_weight: UnitValueContainer

    main_image_url: str

    main_image_id: Optional[Any]

    birthdate: str

    height: UnitValueContainer

    first_name: Optional[str]

    last_name: Optional[str]

    sex: typing_extensions.Literal[M, F]

    activity_factor: str

    headline: Optional[str]

    about: Optional[str]

    why: Optional[str]

    inspirations: List

class myfitnesspal.types.UserMetadata(**kwargs)

    id: str

    username: str

    email: str

    goal_displays: List[GoalDisplayDict]

    unit_preferences: UnitPreferenceDict

    diary_preferences: DiaryPreferencesDict

    goal_preferences: GoalPreferencesDict

    location_preferences: LocationPreferencesDict

    account: AccountDict

    system_data: SystemDataDict

    step_sources: List

    profiles: List[UserProfile]

class myfitnesspal.types.AuthData(**kwargs)
```

```
    token_type: str
    access_token: str
    expires_in: int
    refresh_token: str
    user_id: str

class myfitnesspal.types.MealEntry(**kwargs)
    name: str
    nutrition_information: Dict[str, float]

class myfitnesspal.types.NoteDataDict(**kwargs)
    body: str
    type: str
    date: str

class myfitnesspal.types.FoodItemNutritionDict(**kwargs)
    calcium: float
    carbohydrates: float
    cholesterol: float
    fat: float
    fiber: float
    iron: float
    monounsaturated_fat: float
    polyunsaturated_fat: float
    potassium: float
    protein: float
    saturated_fat: float
    sodium: float
    sugar: float
    trans_fat: float
    vitamin_a: float
    vitamin_c: float

class myfitnesspal.types.ServingSizeDict(**kwargs)
    id: str
```

```
    nutrition_multiplier: float
    value: float
    unit: str
    index: int

class myfitnesspal.types.FoodItemDetailsResponse(**kwargs)
    description: str
    brand_name: Optional[str]
    verified: bool
    nutrition: FoodItemNutritionDict
    calories: float
    confirmations: int
    serving_sizes: List[ServingSizeDict]

class myfitnesspal.types.NutritionInformation(**kwargs)
    calories: str
    carbohydrateContent: str
    fiberContent: str
    sugarContent: str
    sodiumContent: str
    proteinContent: str
    fatContent: str
    saturatedFatContent: str
    monounsaturatedFatContent: str
    polyunsaturatedFatContent: str
    unsaturatedFatContent: str
    transFatContent: str

class myfitnesspal.types.Recipe(**kwargs)
    author: str
    org_url: str
    name: str
    recipeYield: str
    recipeIngredient: List[str]
```

```
nutrition: NutritionInformation  
recipeInstructions: str  
tags: List[str]
```

UPGRADING FROM 1.X TO 2.X

Between the 1.x and 2.x versions of this library, the mechanism used for authentication changed, and this change has an impact on how you instantiate the `myfitnesspal.Client` object.

For more information about why this change was necessary, see [Issue #144](#).

6.1 Version 1.x (Obsolete)

Before getting started, you would store a password in your system keyring for the user account you would like to use (in this example: 'myusername').

In your code, you would then instantiate your `myfitnesspal.Client` like this:

```
import myfitnesspal

client = myfitnesspal.Client('myusername')
```

6.2 Version 2.x (Current)

Before getting started, now you should open a web browser on the same computer you will be using this library from, go to <https://myfitnesspal.com/>, and log in to MyFitnessPal using the user account you would like to use.

In your code, you can then instantiate your `myfitnesspal.Client` like this:

```
import myfitnesspal

client = myfitnesspal.Client()
```

Note that the instantiation no longer accepts a username, and instead reads the log in information directly from your browser.

Do you track your eating habits on [MyFitnessPal](#)? Have you ever wanted to analyze the information you're entering into MyFitnessPal programmatically?

Although MyFitnessPal [does have an API](#), it is private-access only; this creates an unnecessary barrier between you and your data that can be overcome using this library.

Having problems? [Issues live on github](#). Have questions? [Ask your questions in our gitter room](#) or [start a discussion](#).

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

m

`myfitnesspal.exceptions`, [21](#)

`myfitnesspal.types`, [21](#)

Symbols

`__getitem__()` (*myfitnesspal.day.Day* method), 17
`__getitem__()` (*myfitnesspal.exercise.Exercise* method), 18
`__getitem__()` (*myfitnesspal.meal.Meal* method), 17

A

`about` (*myfitnesspal.types.UserProfile* attribute), 23
`access_token` (*myfitnesspal.Client* property), 15
`access_token` (*myfitnesspal.types.AuthData* attribute), 24
`account` (*myfitnesspal.types.UserMetadata* attribute), 23
`AccountDict` (class in *myfitnesspal.types*), 22
`activity_factor` (*myfitnesspal.types.UserProfile* attribute), 23
`admin_flags` (*myfitnesspal.types.AccountDict* attribute), 22
`AdminFlagDict` (class in *myfitnesspal.types*), 22
`aliases` (*myfitnesspal.types.CommandDefinition* attribute), 21
`as_dict()` (*myfitnesspal.note.Note* method), 20
`AuthData` (class in *myfitnesspal.types*), 23
`author` (*myfitnesspal.types.Recipe* attribute), 25

B

`birthdate` (*myfitnesspal.types.UserProfile* attribute), 23
`body` (*myfitnesspal.types.NoteDataDict* attribute), 24
`brand` (*myfitnesspal.fooditem.FoodItem* property), 19
`brand_name` (*myfitnesspal.types.FoodItemDetailsResponse* attribute), 25

C

`calcium` (*myfitnesspal.fooditem.FoodItem* property), 19
`calcium` (*myfitnesspal.types.FoodItemNutritionDict* attribute), 24
`calories` (*myfitnesspal.fooditem.FoodItem* property), 19
`calories` (*myfitnesspal.types.FoodItemDetailsResponse* attribute), 25
`calories` (*myfitnesspal.types.NutritionInformation* attribute), 25

`carbohydrateContent` (*myfitnesspal.types.NutritionInformation* attribute), 25
`carbohydrates` (*myfitnesspal.fooditem.FoodItem* property), 19
`carbohydrates` (*myfitnesspal.types.FoodItemNutritionDict* attribute), 24
`cholesterol` (*myfitnesspal.fooditem.FoodItem* property), 19
`cholesterol` (*myfitnesspal.types.FoodItemNutritionDict* attribute), 24
`city` (*myfitnesspal.types.LocationPreferencesDict* attribute), 22
`Client` (class in *myfitnesspal*), 15
`CommandDefinition` (class in *myfitnesspal.types*), 21
`confirmations` (*myfitnesspal.fooditem.FoodItem* property), 20
`confirmations` (*myfitnesspal.types.FoodItemDetailsResponse* attribute), 25
`country_code` (*myfitnesspal.types.LocationPreferencesDict* attribute), 22
`created_at` (*myfitnesspal.types.AccountDict* attribute), 22

D

`date` (*myfitnesspal.note.Note* property), 20
`date` (*myfitnesspal.types.NoteDataDict* attribute), 24
`Day` (class in *myfitnesspal.day*), 17
`default_foot_view` (*myfitnesspal.types.DiaryPreferencesDict* attribute), 21
`description` (*myfitnesspal.types.CommandDefinition* attribute), 21
`description` (*myfitnesspal.types.FoodItemDetailsResponse* attribute), 25
`details` (*myfitnesspal.fooditem.FoodItem* property), 18
`diary_goal_display` (*myfitnesspal.types.GoalPreferencesDict* attribute),

22
 diary_preferences (*myfitnesspal.types.UserMetadata attribute*), 23
 DiaryPreferencesDict (*class in myfitnesspal.types*), 21
 display_type (*myfitnesspal.types.GoalDisplayDict attribute*), 21
 distance (*myfitnesspal.types.UnitPreferenceDict attribute*), 21

E

effective_username (*myfitnesspal.Client property*), 15
 email (*myfitnesspal.types.UserMetadata attribute*), 23
 energy (*myfitnesspal.types.UnitPreferenceDict attribute*), 21
 entries (*myfitnesspal.day.Day property*), 17
 entries (*myfitnesspal.exercise.Exercise property*), 18
 entries (*myfitnesspal.meal.Meal property*), 17
 Entry (*class in myfitnesspal.entry*), 18
 Exercise (*class in myfitnesspal.exercise*), 18
 exercises (*myfitnesspal.day.Day property*), 17
 expires_in (*myfitnesspal.types.AuthData attribute*), 24

F

fat (*myfitnesspal.fooditem.FoodItem property*), 19
 fat (*myfitnesspal.types.FoodItemNutritionDict attribute*), 24
 fatContent (*myfitnesspal.types.NutritionInformation attribute*), 25
 fiber (*myfitnesspal.fooditem.FoodItem property*), 19
 fiber (*myfitnesspal.types.FoodItemNutritionDict attribute*), 24
 fiberContent (*myfitnesspal.types.NutritionInformation attribute*), 25
 first_name (*myfitnesspal.types.UserProfile attribute*), 23
 FoodItem (*class in myfitnesspal.fooditem*), 18
 FoodItemDetailsResponse (*class in myfitnesspal.types*), 25
 FoodItemNutritionDict (*class in myfitnesspal.types*), 24
 FoodItemServing (*class in myfitnesspal.fooditem.serving*), 20
 forgot_password_or_username (*myfitnesspal.types.AdminFlagDict attribute*), 22
 function (*myfitnesspal.types.CommandDefinition attribute*), 21

G

get_as_dict() (*myfitnesspal.day.Day method*), 17
 get_as_dict() (*myfitnesspal.entry.Entry method*), 18
 get_as_list() (*myfitnesspal.exercise.Exercise method*), 18
 get_date() (*myfitnesspal.Client method*), 15

get_food_item_details() (*myfitnesspal.Client method*), 15
 get_food_search_results() (*myfitnesspal.Client method*), 15
 get_meal() (*myfitnesspal.Client method*), 16
 get_meals() (*myfitnesspal.Client method*), 16
 get_measurements() (*myfitnesspal.Client method*), 15
 get_recipe() (*myfitnesspal.Client method*), 16
 get_recipes() (*myfitnesspal.Client method*), 16
 get_report() (*myfitnesspal.Client method*), 15
 goal_displays (*myfitnesspal.types.UserMetadata attribute*), 23
 goal_preferences (*myfitnesspal.types.UserMetadata attribute*), 23
 GoalDisplayDict (*class in myfitnesspal.types*), 21
 GoalPreferencesDict (*class in myfitnesspal.types*), 21
 goals (*myfitnesspal.day.Day property*), 17

H

has_changed_username (*myfitnesspal.types.AdminFlagDict attribute*), 22
 headline (*myfitnesspal.types.UserProfile attribute*), 23
 height (*myfitnesspal.types.UnitPreferenceDict attribute*), 21
 height (*myfitnesspal.types.UserProfile attribute*), 23
 home_goal_display (*myfitnesspal.types.GoalPreferencesDict attribute*), 22

I

id (*myfitnesspal.types.GoalDisplayDict attribute*), 21
 id (*myfitnesspal.types.ServingSizeDict attribute*), 24
 id (*myfitnesspal.types.UserMetadata attribute*), 23
 index (*myfitnesspal.fooditem.serving.FoodItemServing property*), 20
 index (*myfitnesspal.types.ServingSizeDict attribute*), 25
 inspirations (*myfitnesspal.types.UserProfile attribute*), 23
 iron (*myfitnesspal.fooditem.FoodItem property*), 19
 iron (*myfitnesspal.types.FoodItemNutritionDict attribute*), 24
 is_alias (*myfitnesspal.types.CommandDefinition attribute*), 21

K

keys() (*myfitnesspal.day.Day method*), 17

L

last_login (*myfitnesspal.types.AccountDict attribute*), 22
 last_name (*myfitnesspal.types.UserProfile attribute*), 23
 locale (*myfitnesspal.types.LocationPreferencesDict attribute*), 22

location_preferences (myfitnesspal.types.UserMetadata attribute), 23
 LocationPreferencesDict (class in myfitnesspal.types), 22
 login_streak (myfitnesspal.types.SystemDataDict attribute), 22

M

macro_goal_format (myfitnesspal.types.GoalPreferencesDict attribute), 22
 main_image_id (myfitnesspal.types.UserProfile attribute), 23
 main_image_url (myfitnesspal.types.UserProfile attribute), 23
 Meal (class in myfitnesspal.meal), 17
 meal_names (myfitnesspal.types.DiaryPreferencesDict attribute), 21
 MealEntry (class in myfitnesspal.types), 24
 meals (myfitnesspal.day.Day property), 17
 mfp_id (myfitnesspal.fooditem.FoodItem property), 18
 module
 myfitnesspal.exceptions, 21
 myfitnesspal.types, 21
 monounsaturated_fat (myfitnesspal.fooditem.FoodItem property), 19
 monounsaturated_fat (myfitnesspal.types.FoodItemNutritionDict attribute), 24
 monounsaturatedFatContent (myfitnesspal.types.NutritionInformation attribute), 25
 myfitnesspal.exceptions
 module, 21
 myfitnesspal.types
 module, 21
 MyfitnesspalError, 21
 MyfitnesspalLoginError, 21
 MyfitnesspalRequestFailed, 21

N

name (myfitnesspal.entry.Entry property), 18
 name (myfitnesspal.exercise.Exercise property), 18
 name (myfitnesspal.fooditem.FoodItem property), 19
 name (myfitnesspal.meal.Meal property), 17
 name (myfitnesspal.types.MealEntry attribute), 24
 name (myfitnesspal.types.Recipe attribute), 25
 Note (class in myfitnesspal.note), 20
 NoteDataDict (class in myfitnesspal.types), 24
 notes (myfitnesspal.day.Day property), 17
 nutrients (myfitnesspal.types.GoalDisplayDict attribute), 21
 nutrition (myfitnesspal.types.FoodItemDetailsResponse attribute), 25

nutrition (myfitnesspal.types.Recipe attribute), 25
 nutrition_information (myfitnesspal.types.MealEntry attribute), 24
 nutrition_multiplier (myfitnesspal.fooditem-serving.FoodItemServing property), 20
 nutrition_multiplier (myfitnesspal.types.ServingSizeDict attribute), 24
 NutritionInformation (class in myfitnesspal.types), 25

O

org_url (myfitnesspal.types.Recipe attribute), 25

P

polyunsaturated_fat (myfitnesspal.fooditem.FoodItem property), 19
 polyunsaturated_fat (myfitnesspal.types.FoodItemNutritionDict attribute), 24
 polyunsaturatedFatContent (myfitnesspal.types.NutritionInformation attribute), 25
 postal_code (myfitnesspal.types.LocationPreferencesDict attribute), 22
 potassium (myfitnesspal.fooditem.FoodItem property), 19
 potassium (myfitnesspal.types.FoodItemNutritionDict attribute), 24
 profiles (myfitnesspal.types.UserMetadata attribute), 23
 protein (myfitnesspal.fooditem.FoodItem property), 19
 protein (myfitnesspal.types.FoodItemNutritionDict attribute), 24
 proteinContent (myfitnesspal.types.NutritionInformation attribute), 25

Q

quantity (myfitnesspal.entry.Entry property), 18

R

Recipe (class in myfitnesspal.types), 25
 recipeIngredient (myfitnesspal.types.Recipe attribute), 25
 recipeInstructions (myfitnesspal.types.Recipe attribute), 26
 recipeYield (myfitnesspal.types.Recipe attribute), 25
 refresh_token (myfitnesspal.types.AuthData attribute), 24
 registration_source (myfitnesspal.types.AccountDict attribute), 22

revoked_privileges (myfitnesspal.types.AdminFlagDict attribute), 22

roles (myfitnesspal.types.AccountDict attribute), 22

S

saturated_fat (myfitnesspal.fooditem.FoodItem property), 19

saturated_fat (myfitnesspal.types.FoodItemNutritionDict attribute), 24

saturatedFatContent (myfitnesspal.types.NutritionInformation attribute), 25

serving (myfitnesspal.fooditem.FoodItem property), 19

serving_id (myfitnesspal.fooditem.serving.FoodItemServing property), 20

serving_sizes (myfitnesspal.types.FoodItemDetailsResponse attribute), 25

servings (myfitnesspal.fooditem.FoodItem property), 20

ServingSizeDict (class in myfitnesspal.types), 24

set_measurements() (myfitnesspal.Client method), 15

set_new_food() (myfitnesspal.Client method), 15

set_new_goal() (myfitnesspal.Client method), 16

sex (myfitnesspal.types.UserProfile attribute), 23

short_name (myfitnesspal.entry.Entry property), 18

sodium (myfitnesspal.fooditem.FoodItem property), 19

sodium (myfitnesspal.types.FoodItemNutritionDict attribute), 24

sodiumContent (myfitnesspal.types.NutritionInformation attribute), 25

starting_weight (myfitnesspal.types.UserProfile attribute), 23

starting_weight_date (myfitnesspal.types.UserProfile attribute), 23

state (myfitnesspal.types.LocationPreferencesDict attribute), 22

status (myfitnesspal.types.AdminFlagDict attribute), 22

step_sources (myfitnesspal.types.UserMetadata attribute), 23

strikes (myfitnesspal.types.AdminFlagDict attribute), 22

sugar (myfitnesspal.fooditem.FoodItem property), 19

sugar (myfitnesspal.types.FoodItemNutritionDict attribute), 24

sugarContent (myfitnesspal.types.NutritionInformation attribute), 25

system_data (myfitnesspal.types.UserMetadata attribute), 23

SystemDataDict (class in myfitnesspal.types), 22

T

tags (myfitnesspal.types.Recipe attribute), 26

time_zone (myfitnesspal.types.LocationPreferencesDict attribute), 22

token_type (myfitnesspal.types.AuthData attribute), 23

totals (myfitnesspal.entry.Entry property), 18

totals (myfitnesspal.meal.Meal property), 17

tracked_nutrients (myfitnesspal.types.DiaryPreferencesDict attribute), 21

trans_fat (myfitnesspal.fooditem.FoodItem property), 19

trans_fat (myfitnesspal.types.FoodItemNutritionDict attribute), 24

transFatContent (myfitnesspal.types.NutritionInformation attribute), 25

type (myfitnesspal.note.Note property), 20

type (myfitnesspal.types.NoteDataDict attribute), 24

type (myfitnesspal.types.UserProfile attribute), 23

U

unit (myfitnesspal.entry.Entry property), 18

unit (myfitnesspal.fooditem.serving.FoodItemServing property), 20

unit (myfitnesspal.types.ServingSizeDict attribute), 25

unit (myfitnesspal.types.UnitValueContainer attribute), 21

unit_preferences (myfitnesspal.types.UserMetadata attribute), 23

UnitPreferenceDict (class in myfitnesspal.types), 21

UnitValueContainer (class in myfitnesspal.types), 21

unsaturatedFatContent (myfitnesspal.types.NutritionInformation attribute), 25

unseen_notifications (myfitnesspal.types.SystemDataDict attribute), 22

updated_at (myfitnesspal.types.AccountDict attribute), 22

user_id (myfitnesspal.Client property), 15

user_id (myfitnesspal.types.AuthData attribute), 24

user_metadata (myfitnesspal.Client property), 15

UserMetadata (class in myfitnesspal.types), 23

username (myfitnesspal.types.UserMetadata attribute), 23

UserProfile (class in myfitnesspal.types), 23

V

valid_email (myfitnesspal.types.AccountDict attribute), 22

value (myfitnesspal.fooditem.serving.FoodItemServing property), 20

value (myfitnesspal.types.ServingSizeDict attribute), 25

`value` (*myfitnesspal.types.UnitValueContainer* attribute),
21
`verified` (*myfitnesspal.fooditem.FoodItem* property), 19
`verified` (*myfitnesspal.types.FoodItemDetailsResponse*
attribute), 25
`vitamin_a` (*myfitnesspal.fooditem.FoodItem* property),
19
`vitamin_a` (*myfitnesspal.types.FoodItemNutritionDict*
attribute), 24
`vitamin_c` (*myfitnesspal.fooditem.FoodItem* property),
20
`vitamin_c` (*myfitnesspal.types.FoodItemNutritionDict*
attribute), 24

W

`warnings` (*myfitnesspal.types.AdminFlagDict* attribute),
22
`water` (*myfitnesspal.day.Day* property), 17
`water` (*myfitnesspal.types.UnitPreferenceDict* attribute),
21
`weekly_exercise_energy` (*myfitness-*
pal.types.GoalPreferencesDict attribute),
22
`weekly_workout_duration` (*myfitness-*
pal.types.GoalPreferencesDict attribute),
21
`weight` (*myfitnesspal.types.UnitPreferenceDict* at-
tribute), 21
`weight_change_goal` (*myfitness-*
pal.types.GoalPreferencesDict attribute),
22
`weight_goal` (*myfitnesspal.types.GoalPreferencesDict*
attribute), 22
`why` (*myfitnesspal.types.UserProfile* attribute), 23
`workouts_per_week` (*myfitness-*
pal.types.GoalPreferencesDict attribute),
21