Fitness Tools Documentation

Release 0.1.1

Maverick Coders

Jul 30, 2018

Contents:

1 Introduction **3**

2 Quick Start **5**
- 2.1 Calculating Body Composition . . . 5
- 2.2 Guessing Repetitions . . . 7
- 2.3 Macronutrient Assignments . . . 7

3 How To Contribute **11**
- 3.1 Pull Request Guidelines . . . 11
- 3.2 Code of Conduct . . . 12

4 Road Map **15**

5 Change Log **17**

6 License **19**
- 6.1 Apache 2.0 License . . . 19
- 6.2 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION . . . 19

7 Complete Module Documentation **23**
- 7.1 fitness_tools package . . . 23

8 Indices and tables **29**

Python Module Index **31**

Healthy Lifestyles With Python

CHAPTER 1

Introduction

Thank you for your interest in this project.

For more in depth coverage see the complete documentation.

Fitness Tools is a Python package that facilitates healthy lifestyles. Whether you're a wellness professional, veteran gym rat, or just starting your fitness journey this package will benefit you.

While being healthy requires personal investment, it also requires some complex calculations like:

1. If can lift **x** pounds for **y** repetitions how many pounds can I lift **z** times?
2. If I need to eat **x** calories per day how many calories should I eat at each meal? How do these calories equate to grams of fat, protein, and carbohydrates per day and per meal?
3. What is my bodyfat percentage base on skinfold measurements?

Before calculating everything by hand give this package a try.

CHAPTER 2

Quick Start

Fitness Tools is made with 100% pure, organic Python.

- **There are no third party dependencies.**
- **Only Supports Python 3.**

To get it now type this command:

```
pip install fitness-tools
```

And you will be well on your way solving these problems:

2.1 Calculating Body Composition

This is a collection of the most popular bodyfat percentage equations calculated by measuring various skinfold sites in millimeters.

Here is the typical workflow for calculating bodyfat percentage from skinfold sites:

1. Collect the measurements from the skinfold sites required by the equation you chose. Appropriate skinfold sites can be found in the documentation.
2. Calculate body density.
3. Calculate bodyfat using the above body density value.

Every subclass in this module inherits from the GenericCalculator class which has 5 methods that calculate bodyfat percentage from body density:

- brozek()
- ortiz()
- schutte()
- siri()

- wagner()

These methods are required to convert body density to bodyfat in all but one equation.

> **If you're unsure which calculation to use, chose siri() because it is generically applicable to most populations**.

Here is a hypothetical example.

A 40 year old female whose skinfold measurements in millimeters are:

- triceps = 7
- biceps = 5
- subscapular = 4
- suprailliac = 10

To instantiate classes in this module pass the following arguments in this order:

- Age
- Sex
- A list of skinfold measurements. Order does not matter.

```
>>> from fitness_tools.composition.bodyfat import DurninWomersley
>>> calc = DurninWomersley(40, 'female', (7, 5, 4, 10))
>>> calc.body_density()

# body density value

1.046703631104186

# pass the body density value to a bodyfat equation inherited from GenericCalculator

>>> calc.siri(calc.body_density())
22.9
```

According to the Durnin Womersley equation our hypothetical female's bodyfat is 22.9%.

As noted above, there is one equation that converts your measurements directly into bodyfat. This is the JacksonPollock4Site class.

Lets do another run through.

A 25 year old male skinfold measurements in millimeters are:

- abdominal = 6
- triceps = 6
- thigh = 8
- suprailiac = 6

```
>>> from fitness_tools.composition.bodyfat import JacksonPollock4Site
>>> calc = JacksonPollock4Site(25, 'male', (6, 5, 8, 6))

# Calculates bodyfat directly

>>> calc.body_fat()
5.2
```

Our hypothetical male has a bodyfat percent of 5.2%.

2.2 Guessing Repetitions

Research shows that different repetition ranges yield different results. Generally speaking the following training adaptations occur:

- Endurance between 10 - 15 repetitions
- Hypertrophy (muscle growth) between 8 - 12 repetitions
- Strength <= 6 repetitions
- Power between 1 - 6 repetitions

With that being said, the fitness enthusiast uses repetition ranges congruent with their goals.

Those goals, however, change over time and there is a need to reassign the proper weight and repetition range quickly.

Lets say you can lift 175 lbs. 10 times and now you want increase your strength. Lets set your new rep goal to 6.

When creating a new RM_Estimator object pass the following arguments in this order:

- Current weight used ending in .0 or .5
- Current repetitions you can complete with the above weight
- Desired repetitions

```
>>> from fitness_tools.exercise.rm_estimator import RM_Estimator
>>> new_reps = RM_Estimator(175.0, 10, 6)
>>> new_reps.estimate_weight()
197.5
```

By this calculation if you can lift 175 lbs 10 times you should be able to lift 197.5 lbs. approximately 6 times.

By default the estimate_weight() function rounds the results to the nearest 2.5 lbs. You can alter the rounding behavior by passing the base keyword argument like so:

```
>>> from fitness_tools.exercise.rm_estimator import RM_Estimator
>>> new_reps = RM_Estimator(175.0, 10, 6)
>>> new_reps.estimate_weight(base=5)
200.0
```

If you are trying to estimate your one rep max use the weight from 5 or less repetitions for best results.

Percentages of your one rep max are within $\pm$ 0.5 to 2% depending on your training status.

2.3 Macronutrient Assignments

The idea of proper nutrition is certainly opinionated. While the information one may encounter can vary drastically, calculating your calorie and macronutrient requirements should not be difficult once you have settled on a paradigm that is right for you.

The goal of this package is to automate these calculations so you can spend more time following through with your nutrition plan.

There are two functions of note here:

- daily_requirements() which returns a dictionary of recommended calories and macronutrients for a day based on your input.
- make_meal(int) returns a dictionary of recommended calories and macronutrients for a meal based on your input and passing int through the function.

Please review the documentation for a complete list of parameters and their accepted values.

There is one class in this package, MakeMeal, and four ways to use it. The only positional argument is weight and everything else is dictated by keyword arguments. Here is the usage ordered from most to least opinionated:

2.3.1 Preset Macronutrient Percentages And Calorie Ranges

Your body type dictates your macronutrient percentages. Further, your activity level and goal dictates your calorie range per pound.

```
>>> from fitness_tools.meals.meal_maker import MakeMeal
>>> body_type_activity_level_goal = MakeMeal(180, goal='maintenance',
                                             activity_level='moderate',
                                             body_type='mesomorph')

>>> body_type_activity_level_goal.daily_requirements()

# returns calories and fat, protein, and carbs in grams for one day

{
'min_calories': 2520,
'max_calories': 2880,
'min_fat': 84.0,
'max_fat': 96.0,
'min_protein': 189.0,
'max_protein': 216.0,
'min_carbs': 252.0,
'max_carbs': 288.0
}

# Daily requirements divided by 4 meals

>>> body_type_activity_level_goal.make_meal(4)

{
'min_calories': 630.0,
'max_calories': 720.0,
'min_fat': 21.0,
'max_fat': 24.0,
'min_protein': 47.0,
'max_protein': 54.0,
'min_carbs': 63.0, '
max_carbs': 72.0
}
```

2.3.2 Preset Macronutrient Percentages Custom Calorie Ranges

Your body type sets the macronutrient percentages and you provide min_cal and max_cal per pound.

```
>>> from fitness_tools.meals.meal_maker import MakeMeal
>>> body_type_custom_cal = MakeMeal(180, min_cal=12, max_cal=14, body_type='ectomorph
↪')

# returns calories and fat, protein, and carbs in grams for one day

>>> body_type_custom_cal.daily_requirements()
{
'min_calories': 2160,
'max_calories': 2520,
'min_fat': 48.0,
'max_fat': 56.0,
'min_protein': 135.0,
'max_protein': 158.0,
'min_carbs': 297.0,
'max_carbs': 346.0
}

# Daily requirements divided by 3 meals

>>> body_type_custom_cal.make_meal(3)
{
'min_calories': 720.0,
'max_calories': 840.0,
'min_fat': 16.0,
'max_fat': 19.0,
'min_protein': 45.0,
'max_protein': 53.0,
'min_carbs': 99.0,
'max_carbs': 115.0
}
```

2.3.3 Preset Calorie Ranges Custom Macronutrient Percentages

Your activity level and goal sets the calorie range per pound. You set the percentage of calories from fat, carbs, and protein manually.

```
>>> from fitness_tools.meals.meal_maker import MakeMeal
>>> activity_level_goal_custom_macros = MakeMeal(180, activity_level='sedentary',
                                                 goal='weight_loss', fat_percent=0.2,
                                                 protein_percent=0.2, carb_percent=0.6)

# returns calories and fat, protein, and carbs in grams for one day

>>> activity_level_goal_custom_macros.daily_requirements()
{
'min_calories': 1800,
'max_calories': 2160,
'min_fat': 40.0,
'max_fat': 48.0,
'min_protein': 90.0,
'max_protein': 108.0,
'min_carbs': 270.0,
'max_carbs': 324.0
}
```

(continues on next page)

(continued from previous page)

```
# Daily requirements divided by 6 meals

>>> activity_level_goal_custom_macros.make_meal(6)
{
'min_calories': 300.0,
'max_calories': 360.0,
'min_fat': 7.0,
'max_fat': 8.0,
'min_protein': 15.0,
'max_protein': 18.0,
'min_carbs': 45.0,
'max_carbs': 54.0
}
```

2.3.4 Fully Custom

You are in complete control. Set macronutrient percentages and calorie ranges manually.

```
>>> from fitness_tools.meals.meal_maker import MakeMeal
>>> custom = MakeMeal(180, min_cal=10, max_cal=12, fat_percent=0.2,
                      protein_percent=0.25, carb_percent=0.55)

# returns calories and fat, protein, and carbs in grams for one day

>>> custom.daily_requirements()
{
'min_calories': 1800,
'max_calories': 2160,
'min_fat': 40.0,
'max_fat': 48.0,
'min_protein': 112.0,
'max_protein': 135.0,
'min_carbs': 248.0,
'max_carbs': 297.0
}

 # Daily requirements divided by 8 meals

 >>> custom.make_meal(8)
{
'min_calories': 225.0,
'max_calories': 270.0,
'min_fat': 5.0,
'max_fat': 6.0,
'min_protein': 14.0,
'max_protein': 17.0,
'min_carbs': 31.0,
'max_carbs': 37.0
}
```

CHAPTER 3

How To Contribute

When contributing to this repository, please discuss the change you wish to make first via issue, email, or any other method with the owners of this repository before making a change.

We have pull request guidelines and a code of conduct; please follow these in all your interactions with the project.

3.1 Pull Request Guidelines

Only Edit Relevant Files

- Focus your pull request on a single feature or issue.
- Please do not change files unrelated to that specific issue or feature.

Submit Clean Code

- Style you're code using PEP8 conventions.
- Include docstrings for any new modules, classes, or functions that are recognizable to sphinx-apidoc.

Write Tests

This project uses pytest for unit tests.

- If you're adding a feature please write tests to support it.
- If you're fixing a bug please add tests to reproduce it.

Make Sure Your Tests Pass

All of this project's tests can be ran by typing:

```
pytest
```

in the root directory.

Keep Commit History Short and Clean

Please make one commit per feature or bug. Short histories aid in finding bugs and helping to identify the best fixes.

Be Descriptive

State a convincing case why your PR should be accepted. For tips on writing pull requests see this article

3.2 Code of Conduct

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language.
- Being respectful of differing viewpoints and experiences.
- Gracefully accepting constructive criticism.
- Focusing on what is best for the community.
- Showing empathy towards other community members.

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances.
- Trolling, insulting/derogatory comments, and personal or political attacks.
- Public or private harassment.
- Publishing others' private information, such as a physical or electronic address, without explicit permission.
- Other conduct which could reasonably be considered inappropriate in a professional setting.

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available here.

CHAPTER 4

Road Map

Version	Enhancements
0.1.x	• Refactor all code with unhandled exceptions (pytest.mark.xfail). • Correct errors in docs and add citations.
0.2.0	• Add calculations for theoretical maximal heart rate and cardiac reserve. • Add alternate constructors for weight in kilograms.
0.3.0	• Add metrics for body fat calculations based on gender and weight. • Add metrics for blood sugar measurements. • Add metrics for blood pressure measurements.

CHAPTER 5

Change Log

Version	Date	Changes
0.1.0	07/27/2018	Initial Release
0.1.1	07/28/2018	Links in README

CHAPTER 6

License

6.1 Apache 2.0 License

Version 2.0

Date January 2004

URL http://www.apache.org/licenses/

6.2 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

6.2.1 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means *(i)* the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or *(ii)* ownership of fifty percent (50%) or more of the outstanding shares, or *(iii)* beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

6.2.2 2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

6.2.3 3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

6.2.4 4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- You must give any other recipients of the Work or Derivative Works a copy of this License; and
- You must cause any modified files to carry prominent notices stating that You changed the files; and
- You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
- If the Work includes a `"NOTICE"` text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such `NOTICE` file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a `NOTICE` text file distributed as part of the Derivative Works; within the Source form or documentation,

if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the `NOTICE` file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the `NOTICE` text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

6.2.5 5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6.2.6 6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the `NOTICE` file.

6.2.7 7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an **"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND**, either express or implied, including, without limitation, any warranties or conditions of **TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE**. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

6.2.8 8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

6.2.9 9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

6.2.10 APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

CHAPTER 7

Complete Module Documentation

7.1 fitness_tools package

7.1.1 Subpackages

fitness_tools.composition package

Submodules

fitness_tools.composition.bodyfat module

class fitness_tools.composition.bodyfat.**DurninWomersley**(*age*, *sex*, **args*)
Bases: *fitness_tools.composition.bodyfat.GenericCalculator*

Uses the Durnin Wormersley equation to calculate body density. Use triceps, biceps, subscapular, and suprailliac skinfold measurements.

Parameters

- **age** – Age as a positive, whole number
- **sex** – Sex either 'male' or 'female' case insensative.
- ***args** – A list of positive, whole numbers reflected as skinfold measurements in millimeters.

body_density()
Converts params age, sex, and skinfolds to body density.

Return type float

Returns body_density

class fitness_tools.composition.bodyfat.**GenericCalculator**(*age*, *sex*, **args*)
Bases: object

The base class that all body fat calculations inherit from.

Parameters

- **age** – Age as a positive, whole number
- **sex** – Sex either 'male' or 'female' case insensative.
- ***args** – A list of positive, whole numbers reflected as skinfold measurements in millimeters. See subclass documentation for implementation details.

brozek(*body_density*)

Parameters **body_density** – the results yielded from a body density equation.

Return type float

Returns body_fat

ortiz(*body_density*)

Parameters **body_density** – the results yielded from a body density equation

Return type float

Returns body_fat

schutte(*body_density*)

Parameters **body_density** – the results yielded from a body density equation.

Return type float

Returns body_fat

siri(*body_density*)

Most popular and generic body density to bodyfat conversion equation.

Parameters **body_density** – the results yielded from a body density equation.

Return type float

Returns body_fat

wagner(*body_density*)

Parameters **body_density** – the results yielded from a body density equation.

Return type float

Returns body_fat

class fitness_tools.composition.bodyfat.**JacksonPollock3Site**(*age*, *sex*, **args*)

Bases: *fitness_tools.composition.bodyfat.GenericCalculator*

Uses the Jackson Pollock 3 site equation to calculate body density. Use chest, triceps, and subscapular skinfolds for men and triceps, thigh and suprailiac for women.

Parameters

- **age** – Age as a positive, whole number
- ***args** – A list of positive, whole numbers reflected as skinfold measurements in millimeters.

body_density()

Converts params age, sex, and skinfolds to body density.

Return type float

Returns body_density

class fitness_tools.composition.bodyfat.**JacksonPollock4Site**(*age*, *sex*, **args*)
Bases: *fitness_tools.composition.bodyfat.GenericCalculator*

Uses the Jackson Pollock 4 site equation to calculate body fat. Use abdominal, triceps, thigh, and suprailiac skinfolds.

Parameters

- **age** – Age as a positive, whole number
- ***args** – A list of positive, whole numbers reflected as skinfold measurements in millimeters.

body_fat()
Converts params age, sex, and skinfolds directly to body fat.

Return type float

Returns body_fat

class fitness_tools.composition.bodyfat.**JacksonPollock7Site**(*age*, *sex*, **args*)
Bases: *fitness_tools.composition.bodyfat.GenericCalculator*

Uses the Jackson Pollock 7 site equation to calculate body density. Use chest, axilla, tricep, subscapular, abdominal, suprailiac, and thigh measurements.

Parameters

- **age** – Age as a positive, whole number
- **sex** – Sex either 'male' or 'female' case insensative.
- ***args** – A list of positive, whole numbers reflected as skinfold measurements in millimeters.

body_density()
Converts params age, sex, and skinfolds to body density.

Return type float

Returns body_density

Module contents

fitness_tools.exercise package

Submodules

fitness_tools.exercise.rm_estimator module

class fitness_tools.exercise.rm_estimator.**RM_Estimator**(*current_weight*, *current_reps*, *desired_reps*)
Bases: object

This class is used to estimate correct weight and repetition combinations. Enter the your current weight, current reps, and your desired reps to use this class. NOTES: For best results use the weight from 5 or less reps to estimate your one rep max.

Percentages of the one rep max are within $\pm$ 0.5 to 2% depending on your training status.

Parameters

- **current_weight** – the weight you are currently using as a float ending in 0.0 or 0.5.
- **current_reps** – the reps you are currently completing using the current_weight as a whole number.
- **desired_reps** – the desired repeitions to complete as a whole number.

estimate_weight(*base=2.5*)

Takes params current_weight, current_reps, and desired_reps and returns the estimated weight for your desired reps rounded to the base keyword argment.

Parameters **base** – The value that you wish to round to. Most commonly 2.5 or 5.0

Returns estimated_weight

Return type float

Module contents

fitness_tools.meals package

Submodules

fitness_tools.meals.meal_maker module

class fitness_tools.meals.meal_maker.**MakeMeal**(*weight*, *goal=None*, *body_type=None*, *activity_level=None*, *min_cal=None*, *max_cal=None*, *fat_percent=None*, *protein_percent=None*, *carb_percent=None*)

Bases: object

Use this class to create optimal meals regardless of your body type or fitness goals.

Parameters

- **weight** – Enter your current weight.
- **goal** – Select a goal: 'weight_loss', 'maintenance', 'weight_gain', or None.
- **body_type** – Select a body type: 'endomorph', 'ectomorph', 'mesomorph' or None.
- **activity_level** – Select an activity level, 'sedentary', 'moderate', 'very', or None.
- **min_cal** – Enter the desired minimum calories per pound defaults to None.
- **max_cal** – Enter the desired maximum calories per pound defaults to None.
- **fat_percent** – Enter the desired percent of calories from fat defaults to None.
- **protein_percent** – Enter the desired percent of calories from protein defaults to None.
- **carb_percent** – Enter the desired percent of calories from carbohydrates defaults to None.

Usage: There are four ways to use this class:

1. Fully custom:

Pass the following parameters manually: weight, desired minimum and maximum calories, and fat_percent, protein_percent, carb_percent. This allows for the finest control over all parameters.

2. Preset calorie ranges custom macronutrient percentages:

   Pass a valid combination of goal and activity_level (see above) pass fat_percent, protein_percent, carb_percent manually. Yields ideal min_cal and max_cal values.

3. Preset macronutrient percentages custom calorie ranges:

   Pass a valid body_type (see above) pass min_cal and max_cal manually. Yields ideal fat_percent, protein_percent, and carb_percent values

4. Preset macronitrient percentages and calorie ranges.

   Pass valid body_type, activity_level, and goal (see above). Yields ideal fat_percent, protein_percent, carb_percent min_cal and max_cal.

daily_max_calories()
Returns the total daily maximum calories.

daily_max_carbs()
Returns the total daily maximum protein in grams.

daily_max_fat()
Returns the total daily maximum fat in grams.

daily_max_protein()
Returns the total daily maximum protein in grams.

daily_min_calories()
Returns the total daily minimum calories.

daily_min_carbs()
Returns the total daily minimum carbohydrates in grams.

daily_min_fat()
Returns the total daily minimum fat in grams.

daily_min_protein()
Returns the total daily minimum protein in grams.

daily_requirements()
Returns a dictionary of recommended calories and macronutrients for the day.

Returns daily_requirements

Return type dict

make_meal(*number_meals*)
Returns a dictionary of recommended calories and macronutrients for one meal.

Parameters **number_meals** (*int*) –

Returns meal

Return type dict

Module contents

7.1.2 Module contents

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

fitness_tools, 27
fitness_tools.composition, 25
fitness_tools.composition.bodyfat, 23
fitness_tools.exercise, 26
fitness_tools.exercise.rm_estimator, 25
fitness_tools.meals, 27
fitness_tools.meals.meal_maker, 26

Index

B

body_density() (fitness_tools.composition.bodyfat.DurninWomersley method), 23
body_density() (fitness_tools.composition.bodyfat.JacksonPollock3Site method), 24
body_density() (fitness_tools.composition.bodyfat.JacksonPollock7Site method), 25
body_fat() (fitness_tools.composition.bodyfat.JacksonPollock4Site method), 25
brozek() (fitness_tools.composition.bodyfat.GenericCalculator method), 24

D

daily_max_calories() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_max_carbs() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_max_fat() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_max_protein() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_min_calories() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_min_carbs() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_min_fat() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_min_protein() (fitness_tools.meals.meal_maker.MakeMeal method), 27
daily_requirements() (fitness_tools.meals.meal_maker.MakeMeal method), 27
DurninWomersley (class in fitness_tools.composition.bodyfat), 23

E

estimate_weight() (fitness_tools.exercise.rm_estimator.RM_Estimator method), 26

F

fitness_tools (module), 27
fitness_tools.composition (module), 25
fitness_tools.composition.bodyfat (module), 23
fitness_tools.exercise (module), 26
fitness_tools.exercise.rm_estimator (module), 25
fitness_tools.meals (module), 27
fitness_tools.meals.meal_maker (module), 26

G

GenericCalculator (class in fitness_tools.composition.bodyfat), 23

J

JacksonPollock3Site (class in fitness_tools.composition.bodyfat), 24
JacksonPollock4Site (class in fitness_tools.composition.bodyfat), 25
JacksonPollock7Site (class in fitness_tools.composition.bodyfat), 25

M

make_meal() (fitness_tools.meals.meal_maker.MakeMeal method), 27
MakeMeal (class in fitness_tools.meals.meal_maker), 26

O

ortiz() (fitness_tools.composition.bodyfat.GenericCalculator method), 24

R

RM_Estimator (class in fitness_tools.exercise.rm_estimator), 25

S

schutte() (fitness_tools.composition.bodyfat.GenericCalculator method), 24
siri() (fitness_tools.composition.bodyfat.GenericCalculator method), 24

W

wagner() (fitness_tools.composition.bodyfat.GenericCalculator method), 24