

# Python @Property Explained – How to Use and When? (Full Examples)

by [Selva Prabhakaran](https://www.machinelearningplus.com/author/selva86/) | Posted on November 5, 2018 | <https://www.machinelearningplus.com/python/python-property/>

A python @property decorator lets a method to be accessed as an attribute instead of as a method with a '()'. Today, you will gain an understanding of when it is really needed, in what situations you can use it and how to actually use it.

## Contents

- [1. Introduction](#)
- [2. What does @property do?](#)
- [3. When to use @property?](#)
- [4. The setter method – When and How to write one?](#)
- [5. The deleter method – When and How to write one?](#)
- [6. Conclusion](#)



Python @property – A Simplified Guide

## 1. Introduction

In well-written python code, you might have noticed a @property decorator just before the method definition.

In this guide, you will understand clearly what exactly the python @property does, when to use it and how to use it. This guide, however, assumes that you have a basic idea about what python classes are. Because the @property is typically used inside one.

## 2. What does @property do?

So, what does the @property do?

The @property lets a method to be accessed as an attribute instead of as a method with a '()'. But why is it really needed and in what situations can you use it?

Search ...

Search

### Recent Posts

[data.table in R – The Complete Beginners Guide](https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/)  
(<https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/>).

[Augmented Dickey Fuller Test \(ADF Test\) – Must Read Guide](https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/)  
(<https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/>).

[KPSS Test for Stationarity](https://www.machinelearningplus.com/time-series/kpss-test-for-stationarity/)  
(<https://www.machinelearningplus.com/time-series/kpss-test-for-stationarity/>).

[101 R data.table Exercises](https://www.machinelearningplus.com/data-manipulation/101-r-data-table-exercises/)  
(<https://www.machinelearningplus.com/data-manipulation/101-r-data-table-exercises/>).

[P-Value – Understanding from Scratch](https://www.machinelearningplus.com/statistics/p-value/)  
(<https://www.machinelearningplus.com/statistics/p-value/>).

[101 Python datatable Exercises \(pydatatable\)](https://www.machinelearningplus.com/data-manipulation/101-python-datatable-exercises-pydatatable/)  
(<https://www.machinelearningplus.com/data-manipulation/101-python-datatable-exercises-pydatatable/>).

[Vector Autoregression \(VAR\) – Comprehensive Guide with Examples in Python](https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/)  
(<https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/>).

[Mahalanobis Distance – Understanding the math with examples \(python\)](https://www.machinelearningplus.com/statistics/mahalanobis-distance/)  
(<https://www.machinelearningplus.com/statistics/mahalanobis-distance/>).

[datetime in Python – Simplified Guide with Clear Examples](https://www.machinelearningplus.com/python/datetime/python-examples/)  
(<https://www.machinelearningplus.com/python/datetime/python-examples/>).

[Principal Component Analysis \(PCA\) – Better Explained](https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/)  
(<https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/>).

[Python Logging – Simplest Guide with Full Code and Examples](https://www.machinelearningplus.com/python/python-logging-guide/)  
(<https://www.machinelearningplus.com/python/python-logging-guide/>).

To understand this, let's create a `Person` class that contains the `first`, `last` and `fullname` of the person as attributes and has an `email()` method that provides the person's email.

```
class Person():

    def __init__(self, firstname, lastname):
        self.first = firstname
        self.last = lastname
        self.fullname = self.first + ' ' + self.last

    def email(self):
        return '{}.{}@email.com_(http://email.com)'.format(self.first, self.last)
```

Let's create an instance of the `Person` 'selva prabhakaran' and print the attributes.

```
# Create a Person object
person = Person('selva', 'prabhakaran')
print(person.first) #> selva
print(person.last) #> prabhakaran
print(person.fullname) #> selva prabhakaran
print(person.email()) #> selva.prabhakaran@email.com
```

### 3. When to use @property?

So far so good.

Now, somehow you decide to change the `last` name of the person.

Here is a fun fact about python classes: If you change the value of an attribute inside a class, the other attributes that are derived from the attribute you just changed don't automatically update.

For example: By changing the `self.last` name you might expect the `self.full` attribute, which is derived from `self.last` to update. But unexpectedly it doesn't. This can provide potentially misleading information about the person.

However, notice the `email()` works as intended, even though it is derived from `self.last`.

```
# Changing the `last` name does not change `self.full` name, but email() works
person.last = 'prasanna'
print(person.last) #> prasanna
print(person.fullname) #> selva prabhakaran
print(person.email()) #> selva.prasanna@email.com
```

So, a probable solution would be to **convert the** `self.fullname` **attribute to a** `fullname()` **method**, so it will provide correct value like the `email()` method did. Let's do it.

[Matplotlib Histogram – How to Visualize Distributions in Python](https://www.machinelearningplus.com/plots/matplotlib-histogram-python-examples/)

(<https://www.machinelearningplus.com/plots/matplotlib-histogram-python-examples/>).

[ARIMA Model – Complete Guide to Time Series Forecasting in Python](https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/)

(<https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>).

[Time Series Analysis in Python – A Comprehensive Guide with Examples](https://www.machinelearningplus.com/time-series/time-series-analysis-python/)

(<https://www.machinelearningplus.com/time-series/time-series-analysis-python/>).

[Matplotlib Tutorial – A Complete Guide to Python Plot w/ Examples](https://www.machinelearningplus.com/plots/matplotlib-tutorial-complete-guide-python-plot-examples/)

(<https://www.machinelearningplus.com/plots/matplotlib-tutorial-complete-guide-python-plot-examples/>).

[Topic modeling visualization – How to present the results of LDA models?](https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/)

(<https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/>).

[Top 50 matplotlib Visualizations – The Master Plots \(with full python code\)](https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/)

(<https://www.machinelearningplus.com/plots/top-50-matplotlib-visualizations-the-master-plots-python/>).

[List Comprehensions in Python – My Simplified Guide](https://www.machinelearningplus.com/python/list-comprehensions-in-python/)

(<https://www.machinelearningplus.com/python/list-comprehensions-in-python/>).

[Python @Property Explained – How to Use and When? \(Full Examples\)](https://www.machinelearningplus.com/python/python-property/)

(<https://www.machinelearningplus.com/python/python-property/>).

[How Naive Bayes Algorithm Works? \(with example and full code\)](https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/)

(<https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/>).

#### Tags

[Bigrams](https://www.machinelearningplus.com/tag/bigrams/) (<https://www.machinelearningplus.com/tag/bigrams/>).

[Classification](https://www.machinelearningplus.com/tag/classification/)

(<https://www.machinelearningplus.com/tag/classification/>).

[Corpus](https://www.machinelearningplus.com/tag/corpus/) (<https://www.machinelearningplus.com/tag/corpus/>).

[Cosine Similarity](https://www.machinelearningplus.com/tag/cosine-similarity/)

(<https://www.machinelearningplus.com/tag/cosine-similarity/>).

[data table](https://www.machinelearningplus.com/tag/data-table/)

(<https://www.machinelearningplus.com/tag/data-table/>).

[Data Manipulation](https://www.machinelearningplus.com/tag/data-manipulation/)

(<https://www.machinelearningplus.com/tag/data-manipulation/>).

[Debugging](https://www.machinelearningplus.com/tag/debugging/)

(<https://www.machinelearningplus.com/tag/debugging/>).

[Doc2Vec](https://www.machinelearningplus.com/tag/doc2vec/)

(<https://www.machinelearningplus.com/tag/doc2vec/>).

Feedback

```

# Converting fullname to a method provides the right fullname
# But it breaks old code that used the fullname attribute without the `()``
class Person():

    def __init__(self, firstname, lastname):
        self.first = firstname
        self.last = lastname

    def fullname(self):
        return self.first + ' ' + self.last

    def email(self):
        return '{}.{}@email.com_(http://email.com)'.format(self.first, self.last)

person = Person('selva', 'prabhakaran')
print(person.fullname()) #> selva prabhakaran

# change Last name to Prasanna
person.last = 'prasanna'

print(person.fullname()) #> selva prasanna

```

Now the *convert to method* solution works.

But there is a problem.

Since we are using `person.fullname()` method with a `()` instead of `person.fullname` as attribute, it will break whatever code that used the `self.fullname` attribute. If you are building a product/tool, the chances are, other developers and users of your module used it at some point and all their code will break as well.

So a better solution (without breaking your user's code) is to convert the method as a property by adding a `@property` decorator before the method's definition. By doing this, the `fullname()` method can be accessed as an attribute instead of as a method with `()`. See example below.

## Evaluation Metrics

(<https://www.machinelearningplus.com/tag/evaluation-metrics/>) FastText

(<https://www.machinelearningplus.com/tag/fasttext/>)

## Feature Selection

(<https://www.machinelearningplus.com/tag/feature-selection/>)

## Gensim

(<https://www.machinelearningplus.com/tag/gensim>) LDA

(<https://www.machinelearningplus.com/tag/lda/>)

## Lemmatization

(<https://www.machinelearningplus.com/tag/lemmatization/>) Linear

Regression (<https://www.machinelearningplus.com/tag/linear-regression/>) Logistic

(<https://www.machinelearningplus.com/tag/logistic/>) LSI

(<https://www.machinelearningplus.com/tag/lsi/>) Matplotlib

(<https://www.machinelearningplus.com/tag/matplo>) Multiprocessing

(<https://www.machinelearningplus.com/tag/multiprocessing/>)

## NLP

(<https://www.machinelearningplus.com/tag/nl>)

NLTK (<https://www.machinelearningplus.com/tag/nltk/>)

## Numpy

(<https://www.machinelearningplus.com/tag/numpy>)

## P-Value

(<https://www.machinelearningplus.com/tag/p-value/>) Pandas

(<https://www.machinelearningplus.com/tag/pandas/>) Parallel

Processing (<https://www.machinelearningplus.com/tag/parallel-processing/>) Phraser

(<https://www.machinelearningplus.com/tag/phraser/>) Practice

Exercise (<https://www.machinelearningplus.com/tag/practice-exercise/>)

# Python

(<https://www.machinelearningplus.com/tag/python/>) R

(<https://www.machinelearningplus.com/tag/r/>) Regex (<https://www.machinelearningplus.com/tag/regex/>)

## Regression

(<https://www.machinelearningplus.com/tag/regression/>)

## Residual Analysis

(<https://www.machinelearningplus.com/tag/residual-analysis/>)

Scikit Learn (<https://www.machinelearningplus.com/tag/scikit-learn/>) Significance Tests

(<https://www.machinelearningplus.com/tag/significance-tests/>) Soft Cosine Similarity

(<https://www.machinelearningplus.com/tag/soft-cosine-similarity/>) spaCy

(<https://www.machinelearningplus.com/tag/spacy/>)

## Stationarity

(<https://www.machinelearningplus.com/tag/stationarity/>)

## Summarization

(<https://www.machinelearningplus.com/tag/summarization/>)

## TaggedDocument

(<https://www.machinelearningplus.com/tag/taggeddocument/>)

TextBlob (<https://www.machinelearningplus.com/tag/textblob/>)

Tfidf (<https://www.machinelearningplus.com/tag/tfidf/>) Time

## Series

(<https://www.machinelearningplus.com/tag/time-series/>) Feedback

```
# Adding @property provides the right fullname and does not break code!
class Person():

    def __init__(self, firstname, lastname):
        self.first = firstname
        self.last = lastname

    @property
    def fullname(self):
        return self.first + ' ' + self.last

    def email(self):
        return '{}.{}@email.com_(http://email.com)'.format(self.first, self.last)

# Init a Person
person = Person('selva', 'prabhakaran')
print(person.fullname) #> selva prabhakaran

# Change Last name to Prasanna
person.last = 'prasanna'

# Print fullname
print(person.fullname) # selva prasanna
```

[series/\). Topic Modeling](https://www.machinelearningplus.com/tag/topic-modeling/)  
[\(https://www.machinelearningplus.com/tag/topic-modeling/\)](https://www.machinelearningplus.com/tag/topic-modeling/), [Visualization](https://www.machinelearningplus.com/tag/visualization/)  
[\(https://www.machinelearningplus.com/tag/visualization/\)](https://www.machinelearningplus.com/tag/visualization/)  
[Word2Vec](https://www.machinelearningplus.com/tag/word2vec/) [\(https://www.machinelearningplus.com/tag/word2vec/\)](https://www.machinelearningplus.com/tag/word2vec/)

## 4. The setter method – When to use it and How to write one?

Now you are able to access the `fullname` like an attribute.

However there is one final problem.

Your users are going to want to change the `fullname` property at some point. And by setting it, they expect it will change the values of the `first` and `last` names from which `fullname` was derived in the first place.

But unfortunately, trying to set the value of `fullname` throws an `AttributeError`.

```
person.fullname = 'raja rajan'

#> -----
#> AttributeError                                Traceback (most recent call last)
#> <ipython-input-36-67cde7461cfc> in <module>
#> ----> 1 person.fullname = 'raja rajan'

#> AttributeError: can't set attribute
```

How to tackle this?

We define an equivalent `setter` method that will be called everytime a user sets a value to this property.

Inside this `setter` method, you can modify the values of variables that should be changed when the value of `fullname` is set/changed.

However, there are a couple of conventions you need to follow when defining a setter method:

1. The setter method should have the same name as the equivalent method that `@property` decorates.
2. It accepts as argument the value that user sets to the property.

Finally you need to add a `@{methodname}.setter` decorator just before the method definition.

Once you add the `@{methodname}.setter` decorator to it, this method will be called everytime the property ( `fullname` in this case) is set or changed. See below.

```
class Person():

    def __init__(self, firstname, lastname):
        self.first = firstname
        self.last = lastname

    @property
    def fullname(self):
        return self.first + ' ' + self.last

    @fullname.setter
    def fullname(self, name):
        firstname, lastname = name.split()
        self.first = firstname
        self.last = lastname

    def email(self):
        return '{}.{}@email.com_(http://email.com)'.format(self.first, self.last)

# Init a Person
person = Person('selva', 'prabhakaran')
print(person.fullname) #> selva prabhakaran
print(person.first) #> selva
print(person.last) #> prabhakaran

# Setting fullname calls the setter method and updates person.first and person.last
person.fullname = 'velu pillai'

# Print the changed values of `first` and `last`
print(person.fullname) #> velu pillai
print(person.first) #> pillai
print(person.last) #> pillai
```

There you go. We set a new value to `person.fullname`, the `person.first` and `person.last` updated as well. Our `Person` class will now automatically update the derived attributes (property) when one of the base attribute changes and vice versa.

## 5. The deleter method

Similar to the setter, the deleter's method defines what happens when a property is deleted.

You can create the deleter method by defining a method of the same name and adding a `@{methodname}.deleter` decorator. See the implementation below.

```
class Person():

    def __init__(self, firstname, lastname):
        self.first = firstname
        self.last = lastname

    @property
    def fullname(self):
        return self.first + ' ' + self.last

    @fullname.setter
    def fullname(self, name):
        firstname, lastname = name.split()
        self.first = firstname
        self.last = lastname

    @fullname.deleter
    def fullname(self):
        self.first = None
        self.last = None

    def email(self):
        return '{}.{}@email.com_(http://email.com)'.format(self.first, self.last)

# Init a Person
person = Person('selva', 'prabhakaran')
print(person.fullname) #> selva prabhakaran

# Deleting fullname calls the deleter method, which erases self.first and self.last
del person.fullname

# Print the changed values of `first` and `last`
print(person.first) #> None
print(person.last) #> None
```

In above case, the `person.first` and `person.last` attribute return `None`, once the `fullname` is deleted.

## 6. Conclusion

So, to summarize:

1. When to use `@property` decorator?  
When an attribute is derived from other attributes in the class, so the derived attribute will update whenever the source attributes is changed.
2. How to make a `@property`?  
Make an attribute as property by defining it as a function and add the `@property` decorator before the fn definition.
3. When to define a setter method for the property?  
Typically, if you want to update the source attributes whenever the property is set. It lets you define any other changes as well.

Hope the purpose of @property is clear and you now know when and how to use it. If you did, congratulations! I will meet you in the next one.

What do you think?

44 Responses



Upvote



Love

2 Comments

machinelearningplus.com

Disqus' Privacy Policy

Login ▾

Recommend 14

Sort by Newest ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

Name



**Hunter** • 7 months ago

Very good explanation

1 ^ | ▾ • Reply •



**Selva Prabhakaran** Mod ➔ Hunter • 3 days ago

Thanks

^ | ▾ • Reply •

Subscribe Add Disqus to your siteAdd DisqusAdd Do Not Sell My Data

Copyright [Machine Learning Plus \(https://www.machinelearningplus.com/\)](https://www.machinelearningplus.com/). All rights reserved.

[Home \(https://www.machinelearningplus.com/\)](https://www.machinelearningplus.com/) [Contact Us \(https://www.machinelearningplus.com/contact-us/\)](https://www.machinelearningplus.com/contact-us/)

[Privacy Policy \(https://www.machinelearningplus.com/privacy-policy/\)](https://www.machinelearningplus.com/privacy-policy/) [About Selva \(https://www.machinelearningplus.com/about/\)](https://www.machinelearningplus.com/about/)

[Web Site Terms and Conditions of Use \(https://www.machinelearningplus.com/terms-of-use/\)](https://www.machinelearningplus.com/terms-of-use/)