Skip to Content
Toggle navigation After Hours Programming

- After Hours Programming
- Services
- Sign In
- Create Account

Advertisement

Toggle navigation Tutorials

- HTML
- CSS
- JavaScript
- PHP
- ColdFusion
- Python
  - Overview
  - Introduction
  - Comments
  - Variables
  - Operators
  - If Statement
  - Functions
  - For Loop
  - While Loop
  - Strings
  - Lists
  - Tuples
  - Dictionaries
  - Formatting
  - Exceptions
  - Reading Files
  - Writing to Files
  - Classes
  - Django
  - Editors
  - Python Quiz
- SQL
- Graphic Design
- Information Architecture
- Usability
- SEO

You (Level 0)
0%
Complete

Last Badge Earned
None

# Python Classes

The Python Classes tutorial explains classes and object oriented programming in Python.

Well, I'm not going to lie to you. The next few tutorials are going to be rather complicated if you have never had any experience with object oriented programming (OOP). Classes are awesome because of a few reasons. First, they help you reuse code instead of duplicating the code in other places all over your program. Classes will save your life when you realize you want to change a function. You will only change it in one spot instead of 10 different spots with slightly different code. Another important part of Classes is that they allow you to create more flexible functions. First, we need to get our hands dirty and start figuring out how to make a class.

## Creating Classes in Python

Example

```
#ClassOne.py
class Calculator(object):

    #define class to simulate a simple calculator
    def __init__ (self):

        #start with zero
        self.current = 0

    def add(self, amount):

        #add number to current
        self.current += amount

    def getCurrent(self):

        return self.current
```

Now, I'm not going to be like everyone else and start bombing terminology at you. Quite frankly, object oriented programming is rather difficult for a beginner and throwing these abstract concepts at you is just going to make you learn slower and hate OOP. So, a class is kind of like a function. We just establish it using the class keyword and following it up with whatever we want to name our class. In our case, we are making a calculator, so that's what I called it. How original! Next, we throw in the argument of object. This is just simply a class above this class. Don't worry much about it. It involves some serious terminology to explain. So, just type it and we will talk about it later.

Next, we get into this little guy def __init__ (self):. Abstract concept time! Objects are made from classes. So, if we had a class named cake. We could make cake objects. However, whenever you make a cake it's not the same as the previous cake you made. Sure, it may look like it and taste like it, but it's not that identical cake. So, in our class we have instances so that we know that they are two different cakes. Why do we need this you ask? Well, let's make two cakes from the same class. Now, I take a big bite out of one of the cakes. Now, you definitely want to know which cake is which right? That's why instances are so important! So, def __init__ (self): just says let's create an instance of this class. But, why do we pass in a parameter of self, you might ask. Well, it is pretty related to what we just discussed. Classes make objects and the functions in a class become the object's methods. However, we do need to know which class function belongs to which instance of the class, so we just implicitly pass in the objects property of self (discussed further in later example). Finally, we get down into the heart of the initialize function. Where we use self.current to create an instance variable equal to zero. Woo! We are done with the toughest part.

Next stop is the add function. We just pass in self and another parameter called amount. Again, self is so we know which instance. The amount is hopefully some number that was passed in. Using our previous knowledge, we understand that we are just adding whatever the amount is to the current variable.

Last, but certainly not least, we move onto this idea of "getting" variables from a class. To get the value of our self.current variable, we should use the best practice of putting it in a function and then calling the function to get the value so that we don't mix up our instances. We set up the function and pass in the instance, and just tell Python to return the value.

## Using Classes in Python

Example

```
from ClassOne import * #get classes from ClassOne file
myBuddy = Calculator() # make myBuddy into a Calculator object
myBuddy.add(2) #use myBuddy's new add method derived from the Calculator class
print(myBuddy.getCurrent()) #print myBuddy's current instance variable
```

In another file, we put in this code. Once we run it, we see it prints 2 to our screen. All of that work just to get the result of 2! Anyways, let's break this guy down. First, I have assumed that both of your classes are in the Python directory. Next, we use from ClassOne, which basically gives Python a reference to which file we are talking about. Then, we polish the statement off with import *. It just means that we want all of the classes in the file. In our case, we only have one class, the Calculator class. Next, we create an Calculator object called myBuddy by assigning it to the Calculator class with myBuddy = Calculator() . Now, this gives myBuddy all of the functions and variables in the Calculator class. (Note: once myBuddy gets those variables and functions, we call them properties and methods). Since we already initialized myBuddy, it has a property of current, and we see that property as a variable called self.current in the Calculator class that is set to 0. So, now we just call myBuddy.add(2) method, which is the add function in our Calculator class. Put simply, this is just 0 + 2. Finally, we output our instance class variable self.current by using the myBuddy.getCurrent() that returns our variable.

Well done! That was a long one. Take a breather. That's one of the toughest concepts in programming to wrap your mind around. Believe it or not, when you use strings, lists, dictionaries, etc., they all are made from classes. From here, you should take some time and study object oriented programming and attempt to apply it in Python.

For a more tangible and better look into the Python language, consider reading the following book. It's an excellent read.

[Test your code with the code simulator]

Your code will execute in this window.

Your Input:

Please type your code here

Test Code

Let's explore some more tutorials or topics!

Next Tutorial Previous Tutorial
[Django] [Writing to Files]

Advertisement

If you enjoyed this resource, support me by sharing this page with others.

## Comment on

Show Comments

-

  **odaudi29 Dec. 14, 2016, 11:05 a.m.**

  i cannot get this code to work, i keep getting a attribute error saying that Calculator has no attribute add!

-

  **Langevin June 15, 2015, 8:01 p.m.**

  @marko It does work!

- **marko** **Jan. 25, 2015, 8:46 p.m.**

  Excellent explanation... but doesn´t work.

---

- **Neo** **Dec. 11, 2014, 12:22 p.m.**

  This one was a bit tough to understand but was really fun reading the tutorial

---

- **Remondee** **Nov. 3, 2014, 5:18 a.m.**

  From the first topic to here, it was a really, really nice tutorial on python. I feel really attracted to the language already thanks to your tutorial. The way you explained everything was really clear and fun. Just wanted to thank you for this beautiful piece of work. Have a good day...

---

- **crussell191** **June 10, 2014, 3:17 p.m.**

  @Antony Here is how I worked through this tutorial, and finally got it to work exactly like the examples: First I created the first example in IDLE as follows: class Calculator(object): def __init__(self): self.current = 0 def add(self, amount): self.current += amount def getCurrent(self): return self.current ****note on the second line there needs to be a space after def and no space before (self)**** From there I transferred it into the notepad app and saved it as ClassOne.py in my python directory. Then in IDLE, I was able to write this: &gt;&gt;&gt; from ClassOne import* &gt;&gt;&gt; myBuddy = Calculator() &gt;&gt;&gt; myBuddy.add(2) &gt;&gt;&gt; print(myBuddy.getCurrent()) 2 &gt;&gt;&gt; Took several tries, and I almost gave up to work on another area of Python, but finally figured it out. Keep at it!

---

- **d4okeefe** **May 30, 2014, 7:14 a.m.**

  Terrific tutorial. I do wish that you would expand it, however. It seems to end rather abruptly with classes and objects. What can one do, especially online, with python?

---

- **Mike_the_snake** **May 24, 2014, 7:59 a.m.**

  A little tricky had the same problem as Satya, but found that the def _init_(self) needed to be double underscores and a space added like this def __init__ (self). Yet the def getCurrent(self) didn't need spaces.

- **casdidier** **May 15, 2014, 2:35 p.m.**

  Great site for newbies thanks a lot !

- **Keefe2014** **May 7, 2014, 12:30 a.m.**

  I think the parameter &quot;self&quot; is just like the &quot;this&quot; pointer in c++? When we use, add(self, amount), &quot;self&quot; already has its value (might be a address in memory), so add(2) means amount=2?

- **Daniel** **April 8, 2014, 6:57 p.m.**

  Very Good.but one thing i can't understand is that why add(2) mean amount=2 not self=2

- **Andy M** **April 4, 2014, 5:01 p.m.**

  I can't get IDLE to recognize ClassOne. I suspect that it might be a problem with def __init__(self). Where are you supposed to put spaces? The other possibility that I see is that IDLE doesn't recognize the proper path for the file. I've got IDLE for Python 2 and 3. Maybe this is causing the conflict? How would I specify the path to make sure I've go the right directory first?

- **Scottacus** **April 3, 2014, 7:52 a.m.**

  This is a really nice intro to python, I hope that you will make a follow up to it!

---

- **Satya** **March 8, 2014, 4:08 p.m.**

  I get this error. What am I doing wrong? Traceback (most recent call last): File &quot;./classes.py&quot;, line 4, in &lt;module&gt; myBuddy.add(2) #use myBuddy's new add method derived from the Calculator class File &quot;/home/traxaem/Desktop/Python/ClassOne.py&quot;, line 10, in add self.Current += amount AttributeError: 'Calculator' object has no attribute 'Current'

---

- **James** **Jan. 4, 2014, 3:03 p.m.**

  So I'm just a little confused here, for the first example do I right the classOne code into the Python GUI or do we put it into notepad(etc.), for the second example do I also do the same as the first.

---

- **Simon B.** **Dec. 9, 2013, 3:14 p.m.**

  Hardest part of my learing so far... Great tutorial! Any idea when you will have finished new python tutorials?

---

- **Rob MacKay** **July 25, 2013, 9:22 a.m.**

  Typo: &quot;Sure, it make look like it and taste like it&quot; make should be might I'm sure :)

---

-

  **Golvmopp June 2, 2013, 2:05 p.m.**

  Looking forward to diving deeper. :D

-

  **Suri March 22, 2013, 11:22 a.m.**

  Waiting for more tutorials of Python...........

Advertisement

About

Badges

Upcoming Tutorials

Facebook

Twitter

Contact

Privacy Policy

Terms of Use

×Close

**Congratulations!**

You have just earned a new badge: