

# Python 101: How to Change a Dict Into a Class

🕒 February 14, 2014   📁 Cross-Platform, Python   🐍 Python   👤 Mike

I work with a lot of dictionaries at my job. Sometimes the dictionaries get really complicated with lots of nested data structures embedded within them. Recently I got a little tired of trying to remember all the keys in my dictionaries so I decided to change one of my dictionaries into a class so I could access the keys as instance variables / attributes. If you've ever gotten sick

Here's one simple way to do it:

```

1. #####
2. class Dict2Obj(object):
3.     """
4.     Turns a dictionary into a class
5.     """
6.
7.     #-----
8.     def __init__(self, dictionary):
9.         """Constructor"""
10.        for key in dictionary:
11.            setattr(self, key, dictionary[key])
12.
13.
14.     #-----
15.     if __name__ == "__main__":
16.         ball_dict = {"color":"blue",
17.                      "size":"8 inches",
18.                      "material":"rubber"}
19.         ball = Dict2Obj(ball_dict)

```

This code uses `setattr` to add each of the keys as attributes to the class. The following shows some examples of how it works:

```

1. >>> ball.color
2. 'blue'
3. >>> ball.size
4. '8 inches'
5. >>> print ball
6. <__main__.Dict2Obj object at 0x028CD5B0>

```

When we print the **ball** object, we get a rather unhelpful string back from the class. Let's override the `__repr__` method of our class and make it print out something a little more useful:

```

1. #####
2. class Dict2Obj(object):
3.     """
4.     Turns a dictionary into a class
5.     """
6.
7.     #-----
8.     def __init__(self, dictionary):
9.         """Constructor"""
10.        for key in dictionary:
11.            setattr(self, key, dictionary[key])
12.
13.
14.     #-----
15.     def __repr__(self):
16.         """
17.         """
18.         return "<dict2obj: %s=>" % self.__dict__
19.
20.     #-----
21.     if __name__ == "__main__":
22.         ball_dict = {"color":"blue",
23.                      "size":"8 inches",
24.                      "material":"rubber"}
25.         ball = Dict2Obj(ball_dict)
26.         </dict2obj:>

```

Now if we print out the ball object, we'll get the following:

```

1. >>> print ball
2. <dict2obj: {'color':="" 'blue',="" 'material':="" 'rubber',="" 'size':=""
3. '8="" inches'}="">

```

This is a little unintuitive in that it is printing out a dictionary using the class's internal `__dict__` rather than just the attribute names. This is more a matter of taste than anything, but let's try to get just the method names:

```

1. #####
2. class Dict2Obj(object):
3.     """
4.     Turns a dictionary into a class
5.     """
6.
7.     #-----
8.     def __init__(self, dictionary):
9.         """Constructor"""
10.        for key in dictionary:
11.            setattr(self, key, dictionary[key])
12.
13.    #-----
14.    def __repr__(self):
15.        """
16.        attrs = str([x for x in self.__dict__])
17.        return "<dict2obj: %s>" % attrs
18.
19.    #-----
20.    if __name__ == "__main__":
21.        ball_dict = {"color": "blue",
22.                     "size": "8 inches",
23.                     "material": "rubber"}
24.        ball = Dict2Obj(ball_dict)
25.    </dict2obj:>

```

Here we just loop over the contents of `__dict__` and return a string with just a list of the keys, which match up with the attribute names. You could have also done it like this:

```

1. attrs = str([x for x in dir(self) if "_" not in x])

```

I'm sure there are lots of other ways to accomplish this sort of thing as well. Regardless, I found this little piece of code helpful in some of my work. Hopefully you'll find it useful too.



#### Sponsored

**+ 55ans : Classement des meilleures mutuelles 2019**

Meilleurtaux.com

**Dany Boon : il quitte sa femme pour sa partenaire de cinéma**

News célébrités

**Les Français peuvent obtenir leur pompe à chaleur financée par l'Etat!**

Reduire ses factures

**Panneaux solaires: Pourquoi ne pas profiter des aides de l'Etat?**

Réduisez Vos Factures

**Les 4 sites de rencontres francais qui marchent vraiment**

Le Top 10 Des Sites de Rencontre

**Rencontre des célibataires dans la région de Le Mesnil-amelot !**

Meetic

10 Comments **Mouse Vs. the Python**

**1** Login ▾

♥ Recommend 1    Tweet    Share

Sort by Best ▾








Join the discussion









LOG IN WITH

OR SIGN UP WITH DISQUS 





 **lemuelf** • 6 years ago  
 or if you really just want to quickly convert your dict to an object so you can access the items as attributes and don't care about the repr method:  
`ball = type('D', (object,), ball_dict)()`  
 3   • [Reply](#) • [Share](#)




 **jeffkod** → **lemuelf** • 4 years ago  
 Thanks a lot for this. Very neat trick which I believe could be especially useful in django for creating objects from JSON dicts to display inside a template. Thanks a million for this very neat trick  
 ^   • [Reply](#) • [Share](#)




 **Mike Driscoll** Mod → **lemuelf** • 6 years ago  
 That's a neat trick! Thanks for sharing!  
 ^   • [Reply](#) • [Share](#)


 **Jos Teunissen** • 2 years ago • edited  
 If you add:  
`def __getitem__(self, key):  
 return self.__dict__[key]`  
 you even don't have to remember if it was actually a dict or a Dict2Obj  
 ^   • [Reply](#) • [Share](#)

 **Mike Driscoll** Mod → **Jos Teunissen** • 2 years ago  
 That's a good point. Thanks for pointing that out!  
 ^   • [Reply](#) • [Share](#)

 **Yaser Alraddadi** • 6 years ago  
 great idea. Thanks for sharing  
 ^   • [Reply](#) • [Share](#)

 **javier\_arilos** • 6 years ago  
 Hi there, very interesting! I would change the title of the post... since you are not creating classes, but objects or instances of Dict2Obj class... just a small detail :-)  
 Nicely explained!!  
 ^   • [Reply](#) • [Share](#)

 **Mike Driscoll** Mod → **javier\_arilos** • 6 years ago  
 Yeah, I know I named it oddly, but I didn't really like any of the other titles I came up with.  
 ^   • [Reply](#) • [Share](#)

 **websam max** • 6 years ago  
 If this is really quick and dirty, you can even do:  
`self.__dict__.update(dictionary)`

Instead of the for loop.

But I would actually keep the for loop and add a dependency injection :

```
def __init__(self, dictionary, merge=lambda s, k, v: setattr(s, k, v)):
    for k, v in dictionary.items():
        merge(self, k, v)
```

This would allow to override the attribute setting, this way, if there is an attribute you actually don't want to override if it already exists, you can.

^   • [Reply](#) • [Share](#)

 **Mike Driscoll** Mod → **websam max** • 6 years ago  
 Thanks for the tips. They're quite interesting!  
 ^   • [Reply](#) • [Share](#)

Sponsored

### **N'éteignez pas votre ordinateur avant d'avoir fait cela**

Security Savers

### **Loi de transition Energétique: l'Etat paie le chauffage des Français!**

Reduire ses factures

### **+ 55ans : Classement des meilleures mutuelles 2019**

Meilleurtaux.com

### **Panneaux solaires: Pourquoi ne pas profiter des aides de l'Etat?**