



Python and Salesforce

simple-salesforce

<https://github.com/simple-salesforce/simple-salesforce>



oktana

About Me

K.B. Carte | <https://kbcarte.com>

Role: Developer

Hire Date: February 2020 ~6 months ago

Experience:

- Web Development (PHP, WordPress, CSS, JS/jQuery, Python)
- Game Development (Blender Modeling and BGE, Godot, Adobe Flash)
- Hacking and Security (hackthebox.eu, hackaday.com)
- [3D Printing](#)
- Robotics and Embedded Systems
 - [Parallax BASIC Stamp](#)
 - [Arduino](#)
 - [AVR](#)
 - [RaspberryPi](#)
 - [BeagleBoard](#)
 - [ESP8266](#)



Get to know Python

<https://python.org>

- Interpreted, Interactive, Object-Oriented, Scripting Language built on C
- Includes **1,731** libraries (Standard Library) Batteries Included!
- Package Management is really easy: **PIP** and **PIPEnv**
- Easy to learn and simple syntax
- MANY different ways to use it:
 - Web apps or API's (Django, Flask)
 - Salesforce (simple-salesforce)
 - Data Science (NumPy, SciPy, Pandas)
 - Visuals and Reporting (Matplotlib, Seaborn, Bokeh, Plotly)
 - Automation (Selenium, Robot Framework, Tavern)
 - Machine Learning / AI (TensorFlow, Keras, PyTorch, Scikit-learn)
 - Robotics and IoT (MicroPython, PySerial, PyVISA, PyFirmata)

Syntax Example

```
print("Hello World!")
```

```
x = 10  
y = x+20
```

```
z = ["a", "b", "c"]  
for i in z:  
    print(i)
```

```
def my_function():  
    my_var = "Hello"  
    my_var2 = " World!"  
    return my_var + my_var2
```

```
class my_class:  
    def __init__(self):  
        self.my_var = "Oktana"  
    def my_method(self, my_arg):  
        return my_arg + self.my_var + " class method!"
```

Note the use of indentation. Instead of curly braces “{}” like other languages, Python code blocks and scope is determined by the indentation levels.

You can’t mix tabs and spaces. One or the other, not both. [Pep8](#) recommends 4 **spaces** per indentation.

Uses Dynamic Type Variables. No need to say what type the variable is, Python automatically converts the var to what ever is being assigned to it.

`__init__(self)` is how class constructors are created. The double underscore methods are special. Also called “magic methods” or “dunder methods”

simple-salesforce

<https://github.com/simple-salesforce/simple-salesforce>

“Simple Salesforce is a basic Salesforce.com REST API client built for Python 3.3, 3.4, 3.5, and 3.6. The goal is to provide a very low-level interface to the REST Resource and APEX API, returning a dictionary of the API JSON response.”

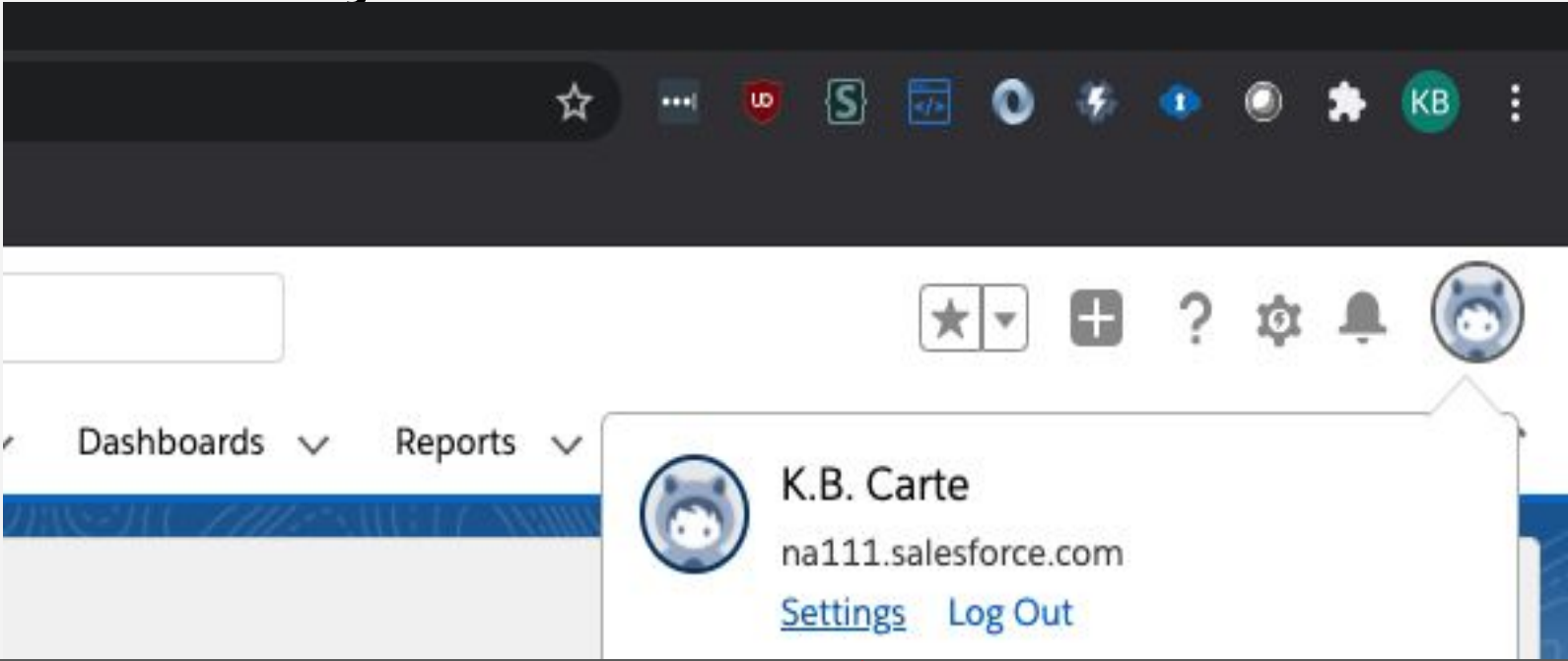
The lib makes it **very** easy to interact with our salesforce org. We only need three things to get started. Our username (email), password, and security token.

```
from simple_salesforce import Salesforce
sf = Salesforce(username='myemail@example.com',
                password='password',
                security_token='token')
```

Now we have a new object called “sf” that we can use to interact with our org. The object has a lot of methods to help with everything from SOQL queries, DML like insert and delete, even bulk API is included!

Login Information

To find your username:

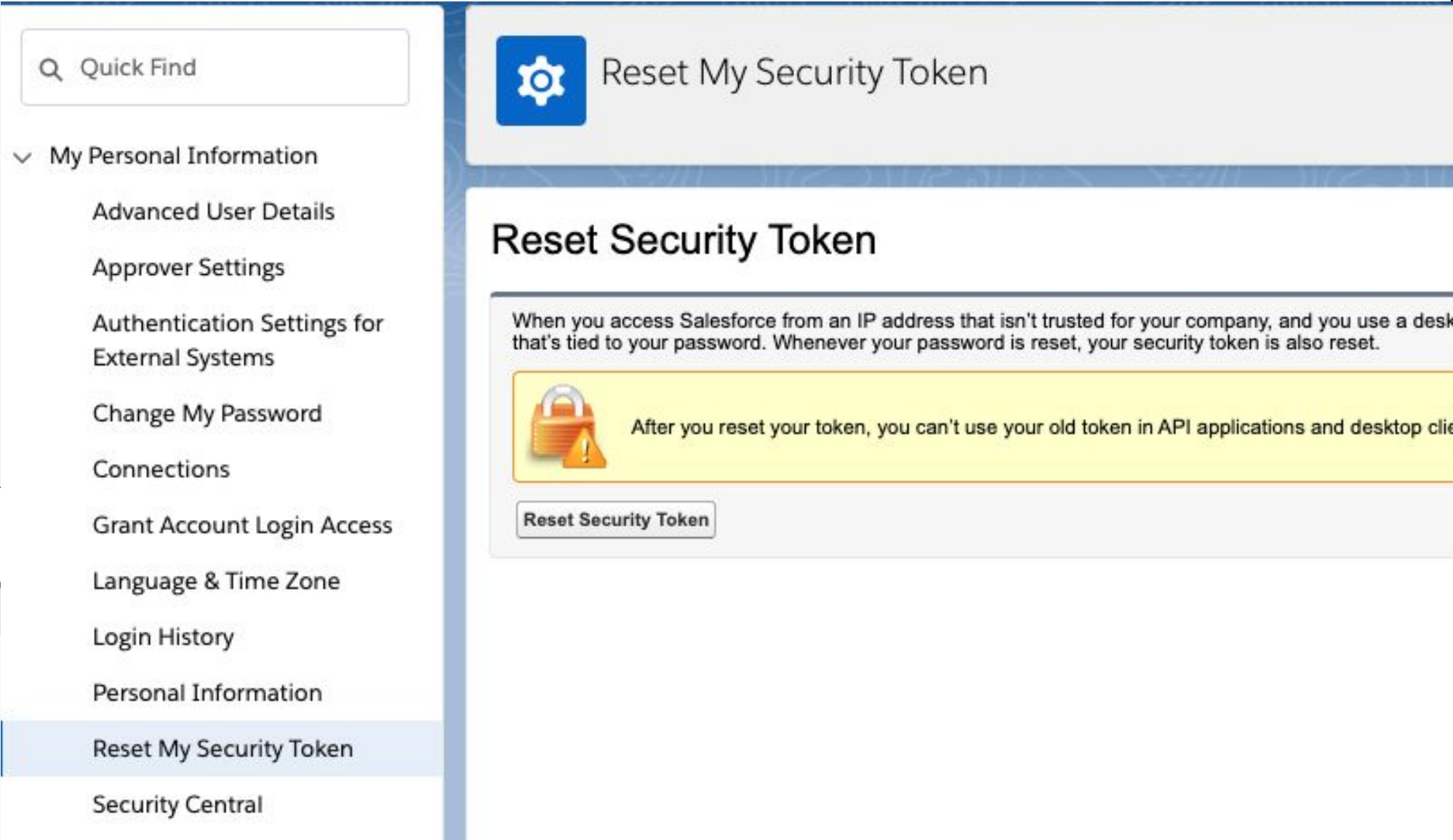


Personal Information

Details

First Name	<input type="text" value="K.B."/>
Last Name	<input type="text" value="Carte"/>
Alias	<input type="text" value="KCart"/>
Email	<input type="text" value="salesforce@kbcarte.com"/>
Username	<input type="text" value="sfdemo@kbcarte.com"/>
Nickname	<input type="text" value="sfdemo"/>
Phone	<input type="text"/>
Extension	<input type="text"/>
Fax	<input type="text"/>
Mobile	<input type="text"/>

Security Token:



Quick word on PIP

Pip is just a package manager that Python uses. It is just like any other manager such as npm, apt, pacman, ruby gems, php composer, etc...

The server/host pip uses is called [PyPi](#). When we run the command `pip install <package_name>` or `pipenv install <package_name>` it is installed from the PyPi servers. This is more than okay for the most part and the vast majority of the time it's fine to have the lib pulled from PyPi. **BUT** there are times, like now, where the version on PyPi isn't the latest version. So there is a way to install libs and packages from Github directly.

While working on this example, there was an update to simple-salesforce lib. Github has the current version and we need the `format_soql` method from it. **BUT** pip (PyPi) hasn't been updated with the new version as of yet, July 2 2020.

So, we'll need to install it via it's repo on Github.

```
pipenv install -e  
git+https://github.com/simple-salesforce/simple-salesforce.git#egg=simple-salesforce
```

If installing this demo from the repo. The `Pipfile` already contains the git repo as the lib source. Thanks to pipenv, it took care of automatically adding the required dependency.

Installing the Demo

```
Python-Salesforce-Examples — unassigned1@unassigned1 — ..orce-Examples — python3.8 /usr/local/Cellar/pipenv/2020.6.2/li...
[% git clone git@github.com:techb/Python-Salesforce-Examples.git ~/Dev/temp ]
Cloning into 'Python-Salesforce-Examples'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 16 (delta 3), reused 14 (delta 1), pack-reused 0
Receiving objects: 100% (16/16), 6.47 KiB | 6.47 MiB/s, done.
Resolving deltas: 100% (3/3), done.
[%
[% cd Python-Salesforce-Examples ~/Dev/temp ]
[(master) % ~/Dev/temp/Python-Salesforce-Examples ]
[(master) % pipenv shell ~/Dev/temp/Python-Salesforce-Examples ]
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instead of creati
ng its own for any project. You can set PIPENV_IGNORE_VIRTUAL_ENVS=1 to force pipenv to ignore that environment and create its own instead.
You can set PIPENV_VERBOSITY=-1 to suppress this warning.
Creating a virtualenv for this project...
Pipfile: /Users/unassigned1/Dev/temp/Python-Salesforce-Examples/Pipfile
Using /usr/local/bin/python3.7.8 (3.7.8) to create virtualenv...
.. Creating virtual environment...created virtual environment CPython3.7.8.final.0-64 in 4430ms
creator CPython3Posix(dest=/Users/unassigned1/.local/share/virtualenvs/Python-Salesforce-Examples-GRQtK0LS, clear=False, global=False)
seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, via=copy, app_data_dir=/Users/unassigned1/Library/Applica
tion Support/virtualenv/seed-app-data/v1.0.1)
activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator

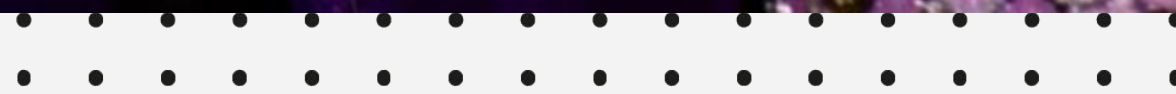
✓ Successfully created virtual environment!
Virtualenv location: /Users/unassigned1/.local/share/virtualenvs/Python-Salesforce-Examples-GRQtK0LS
Launching subshell in virtual environment...
. /Users/unassigned1/.local/share/virtualenvs/Python-Salesforce-Examples-GRQtK0LS/bin/activate
(master) % . /Users/unassigned1/.local/share/virtualenvs/Python-Salesforce-Examples-GRQtK0LS/bin/activate
(Python-Salesforce-Examples) (master) % ~/Dev/temp/Python-Salesforce-Examples
```


Installing the Demo Cont...

```
Python-Salesforce-Examples — unassigned1@unassigned1 — ..orce-Examples — python3.8 /usr/local/Cellar/pipenv/2020.6.2/li...
[(Python-Salesforce-Examples) (master) % pipenv install
Installing dependencies from Pipfile.lock (f060e8)_
  2  ██████████ 11/11 -
[(Python-Salesforce-Examples) (master) % cp EXAMPLE_login.json login.json
[(Python-Salesforce-Examples) (master) %
[(Python-Salesforce-Examples) (master) % vim login.json
[(Python-Salesforce-Examples) (master) %
```

```
Python-Salesforce-Examples — vim login.json — vim — python3.8 /usr/local/Cellar/pipenv/2020.6.2/li...
1 {
2   "login": {
3     "username": "sfdemo@kbcarte.com",
4     "password": "My-REALLY-amazing-password",
5     "token": "kdjfhgdgffGJbDFgd36DFGHDfgh"
6   }
7 }
```

Live Demo Time!





Q&A

Thank you!

