
AltTabPy Documentation

Release 0.1a

innobi

Apr 15, 2020

Contents

1	Getting Started	3
1.1	Installing	3
1.2	Connecting a Tableau Workbook	3
2	Sample Usage	7
2.1	Reversing the Case of All Text	7
2.2	Replacing Characters via Regex	8
3	Comparison to TabPy	11
3.1	Maintenance and Ideology	11
3.2	Use Cases	11
3.3	Security	11

AltTabPy is a alternative to [TabPy](#) with a focus on being lightweight, minimal and easy to use.

CHAPTER 1

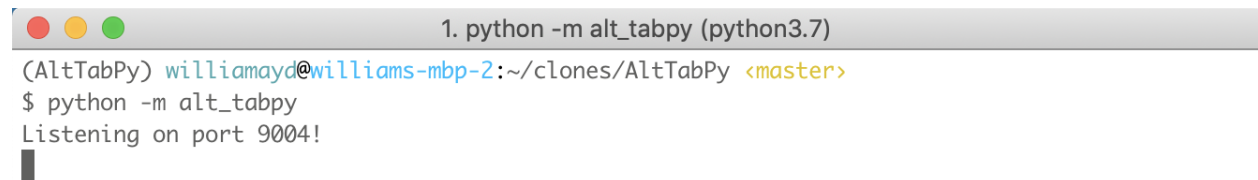
Getting Started

1.1 Installing

AltTabPy is available on [PyPy](#) and can be installed via pip:

```
pip install alt-tabpy
```

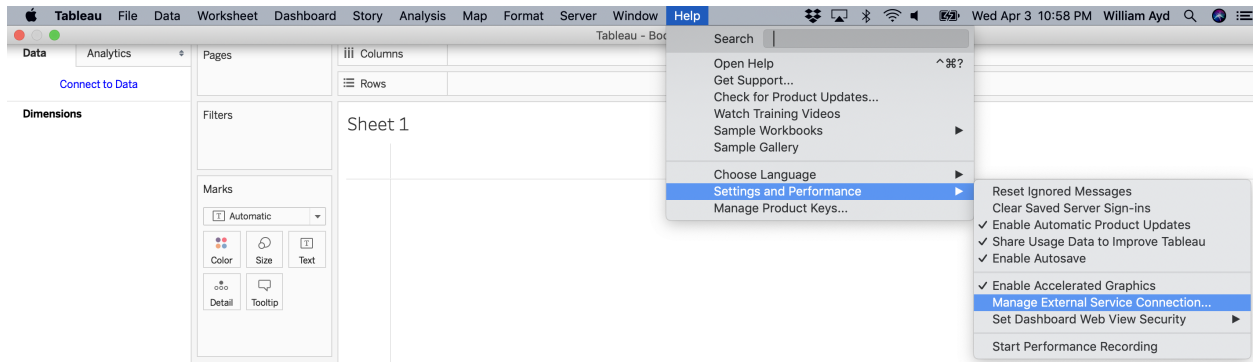
Once installed, you can launch the application via the command line, optionally specifying a port:

A terminal window with a title bar containing three colored circles (red, yellow, green) and the text "1. python -m alt_tabpy (python3.7)". The terminal content shows the prompt "(AltTabPy) williamayd@williams-mbp-2:~/clones/AltTabPy <master>" followed by the command "\$ python -m alt_tabpy" and the output "Listening on port 9004!". A cursor is visible on the line following the output.

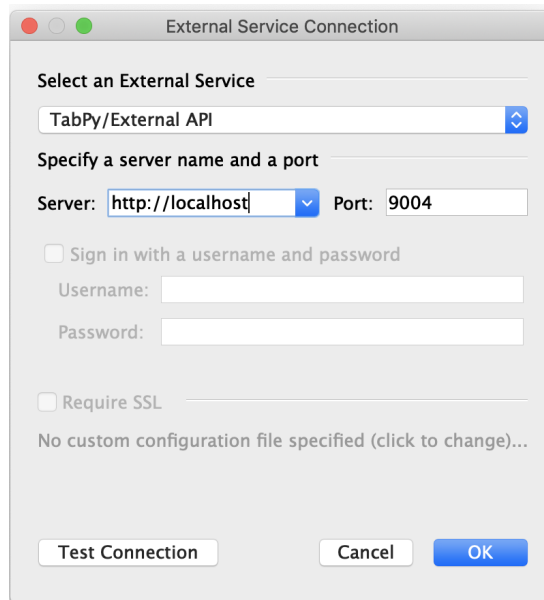
```
1. python -m alt_tabpy (python3.7)
(AltTabPy) williamayd@williams-mbp-2:~/clones/AltTabPy <master>
$ python -m alt_tabpy
Listening on port 9004!
```

1.2 Connecting a Tableau Workbook

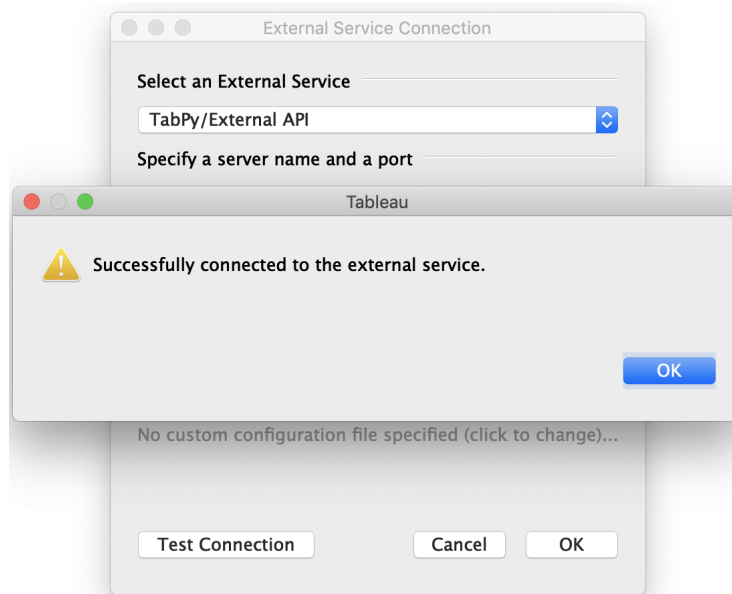
With AltTabPy running you'll need to tell Tableau Desktop where it can communicate with the service. To do so go to Help > Settings and Performance > Manage External Service Connection



On the subsequent screen specify the URL and port where the service is running:



Test the connection and if all is working well you should get a success message!



If you've gotten this far then congratulations - you are ready to start leveraging the powers of Python and Tableau together! If you haven't already be sure to check out the [Samples](#) for ideas on how to use these together.

Sample Usage

For all of these examples we will use the *Sample - Superstore* datasource that comes with Tableau Desktop.

2.1 Reversing the Case of All Text

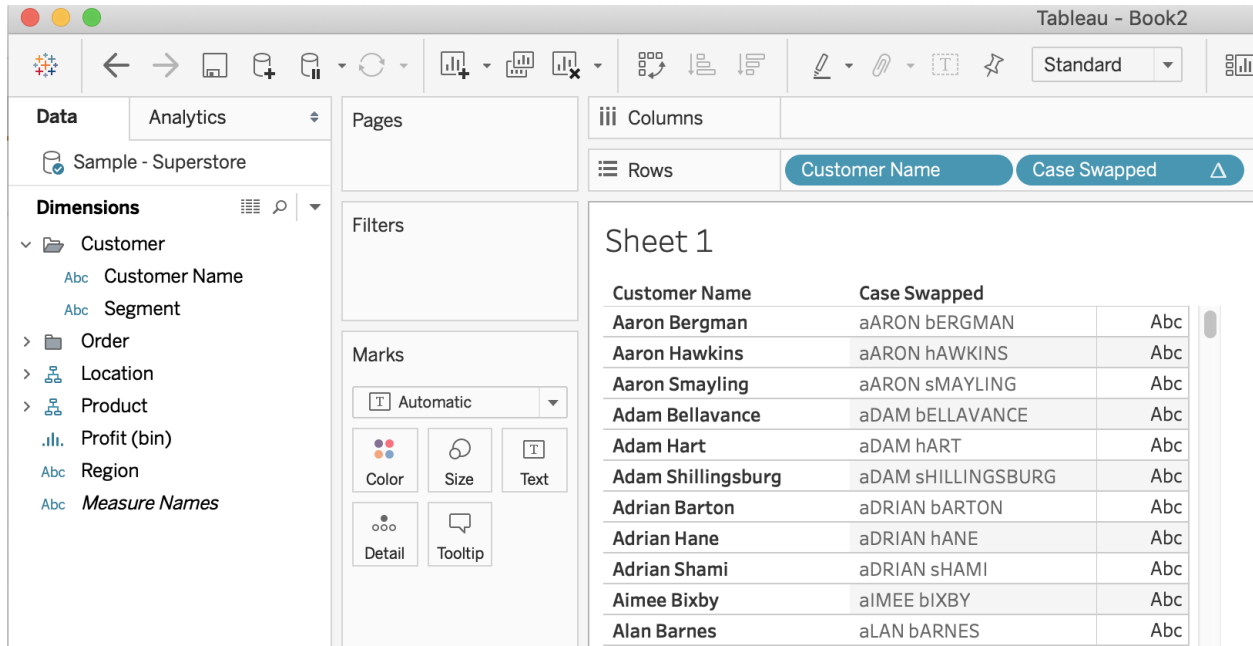
Let's assume we wanted to reformat the *Customer Name* dimension in the sample dataset to make anything that is upper case lower and vice-versa. In Python you can achieve this using the `swapcase` string method as such:

```
print(x.swapcase() for x in ["John Smith", "Jane Doe"])
```

The equivalent function in AltTabPy looks something like this:



Note the semantics required by Tableau to achieve this. Because the result of this expression is a string we need to use `SCRIPT_STR` (as opposed to `SCRIPT_BOOL`, `SCRIPT_INT` or `SCRIPT_REAL`). The `Customer Name` dimension in this case gets passed to the function as `_arg1`. Subsequent dimensions or measures would repeat this pattern of `_arg2`, `_arg3`.. `_argn`. The code block being passed then iterates over the values, returning back a list of values for the Tableau engine to subsequently processed. The result of this appears as follows:



2.2 Replacing Characters via Regex

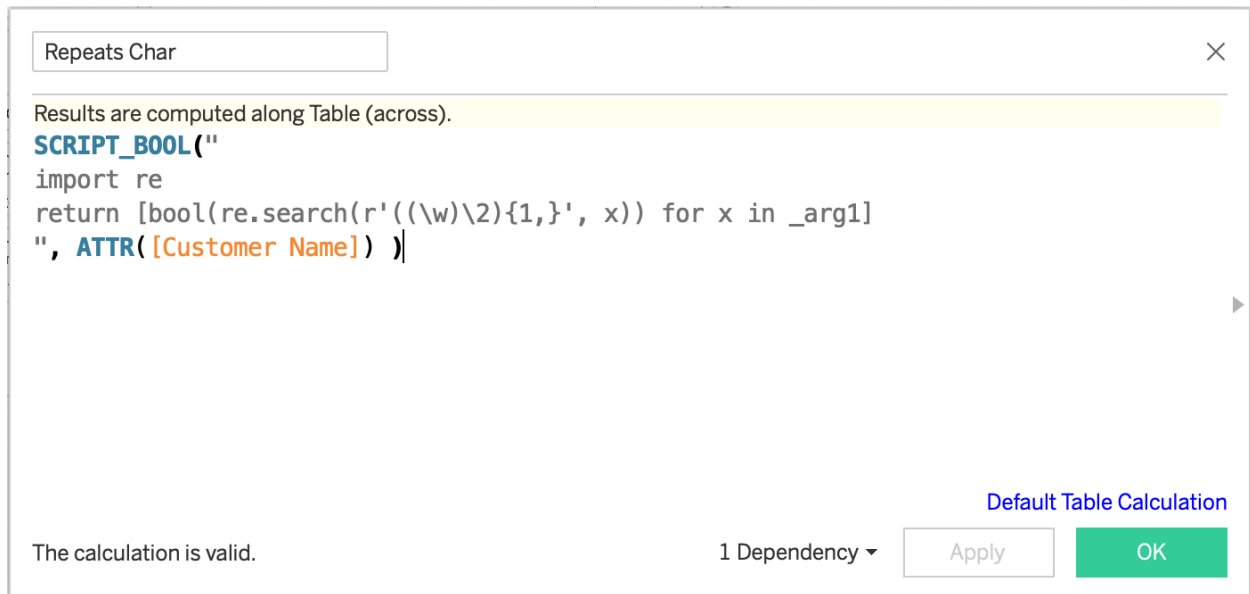
You aren't limited to builtin commands and methods - you can also import standard and third party libraries for use in your analyses!

To illustrate, let's imagine a scenario where we wanted to identify names containing at least one letter which was repeated twice. To put this in other words, we'd like some way to flag that the names Allen, Benny and Connor differ from Al, Benjamin and Conor given the former have repeating characters whereas the latter do not.

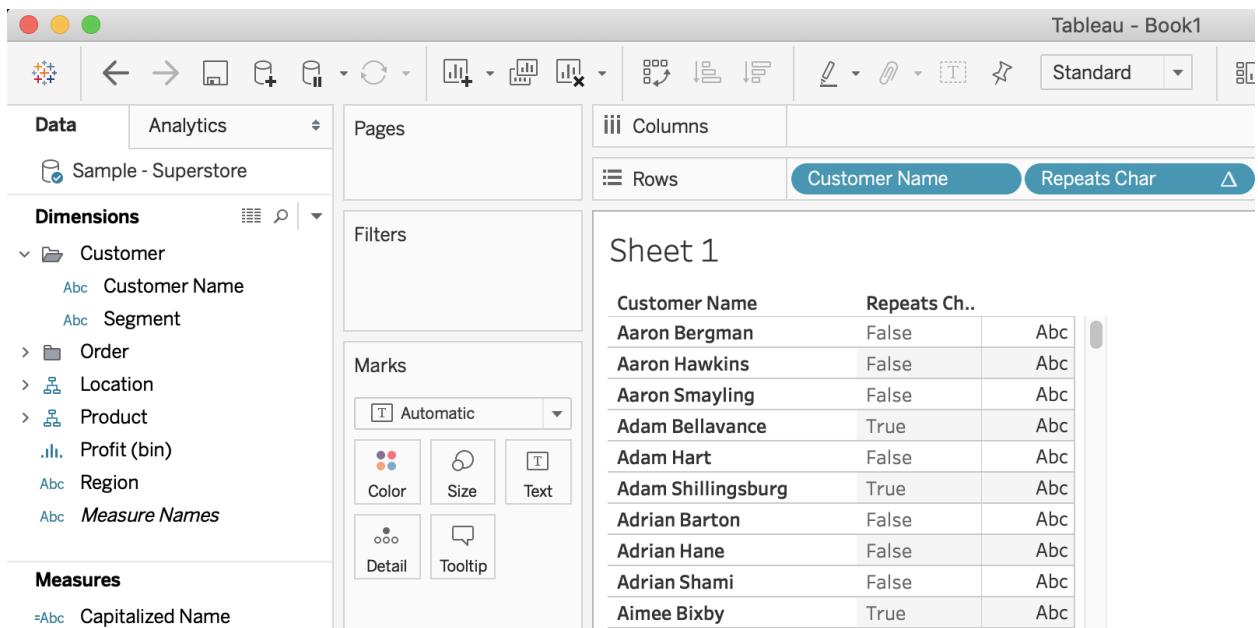
Again in python you could do something like this:

```
import re
bool(re.search(r'((\w)\2){1,}', 'Connor')) # True
bool(re.search(r'((\w)\2){1,}', 'Conor')) # True
```

Here's the subsequent calculation in Tableau:



And our result confirming the calculation:



3.1 Maintenance and Ideology

AltTabPy focuses on a small subset of the functionality offered by TabPy. Whereas TabPy offers a component to publish, create and manage endpoints containing code samples, AltTabPy does no such thing. Instead, AltTabPy leaves it to the user to configure their Python environment with everything they need and simply execute code within Tableau, rather than trying to configure it through any extra means.

The advantage here is that the code base for AltTabPy is significantly smaller than that of TabPy; as of authorship TabPy was close to 6000 lines of code whereas AltTabPy was less than 300, or less than 5% of the size of TabPy.

AltTabPy also had the advantage of starting with a clean code base with no compatability concerns. TabPy still manages a large amount of code from Python2 and has not yet evolved to some of the more recent Python3 offerings like type hinting.

3.2 Use Cases

AltTabPy aims to get a single user up and running as fast as possible using Tableau and Python in tandem. This can be done on virtually any platform in a few minutes at most.

By contrast, TabPy aims to be a larger enterprise solution which as of writing requires managing dependencies, targeting specific platforms and going through a rather complex installation process.

This does not mean that AltTabPy can't extend beyond single-use applications and that TabPy couldn't streamline it's installation processes, but in any case both projects are tackling this area from different angles.

3.3 Security

AltTabPy offers no extra security configurations. Because it's initial goal is for single-use computing, it simply would be overkill to offer an authentication system on top of that.

TabPy has a new authentication system under development which may be more suitable when trying to manage a large team of users.