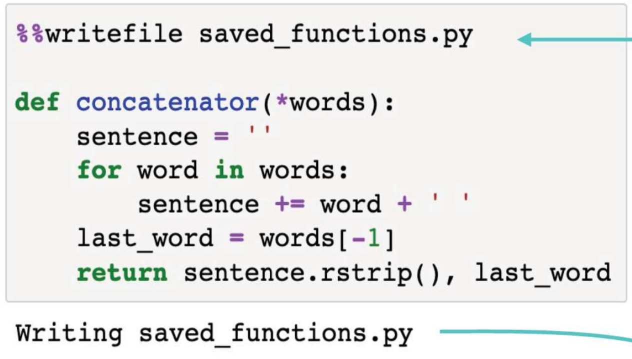
CREATING MODULES

To save your functions, **create a module** in Jupyter by using the %%writefile magic command and the .py extension



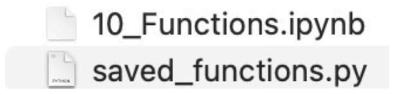
```
%%writefile saved_functions.py

def concatenator(*words):
    sentence = ''
    for word in words:
        sentence += word + ' '
    last_word = words[-1]
    return sentence.rstrip(), last_word
```

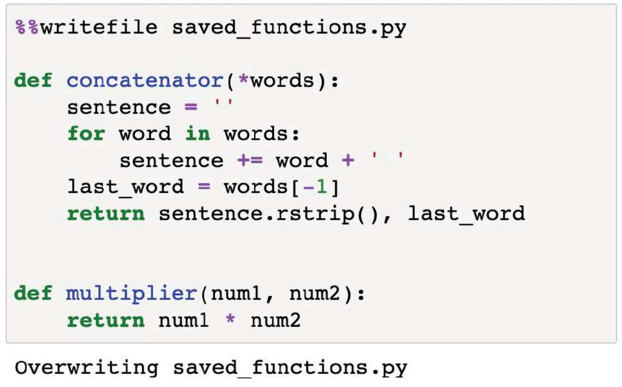
```
Writing saved_functions.py
```

This creates a Python module that you can import functions from

- *Follow **%%writefile** with the name of the file and the **.py** extension*
- *By default, the .py file is stored in the same folder as the notebook*
- *You can share functions easily by sending this file to a friend or colleague!*



10_Functions.ipynb
saved_functions.py



```
%%writefile saved_functions.py

def concatenator(*words):
    sentence = ''
    for word in words:
        sentence += word + ' '
    last_word = words[-1]
    return sentence.rstrip(), last_word


def multiplier(num1, num2):
    return num1 * num2
```

```
Overwriting saved_functions.py
```

Multiple functions can be saved to the same module

IMPORTING MODULES

- Function Components
- Defining Functions
- Variable Scope
- **Modules**
- Packages
- Lambda Functions
- Comprehensions

To **import saved functions**, you can either import the entire module or import specific functions from the module

```
import saved_functions

saved_functions.concatenator('Hello', 'world!')
```

```
('Hello world! ', 'world!')
```

```
saved_functions.multiplier(5, 10)
```

```
50
```

import module
reads in external Python modules

If you import the entire module, you need to reference it when calling its functions, in the form of ***module.function()***

```
from saved_functions import concatenator, multiplier

concatenator('Hello', 'world!')
```

```
('Hello world! ', 'world!')
```

```
multiplier(5, 10)
```

```
50
```

from module import function
imports specific functions from modules

By importing specific functions, you don't need to reference the entire module name when calling a function

⚠ This method can lead to **naming conflicts** if another object has the same name