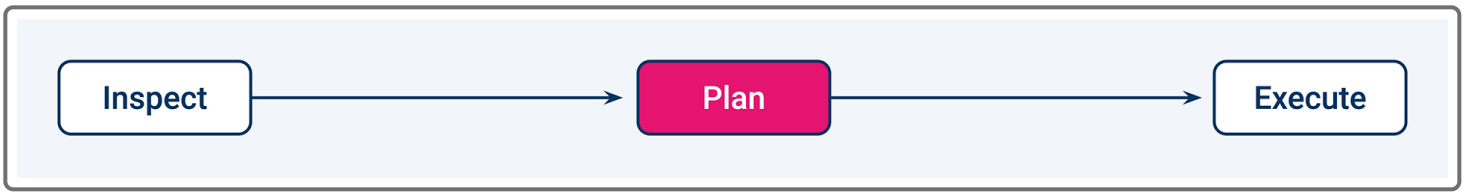
8.4.3

Using Regular Expressions for Data Transformation

Now that you've gotten a quick course about regular expressions, Britta wants you to practice using them for data transformation.

First, we'll plan how to use what we've learned about regular expressions to get the names of all the contacts.

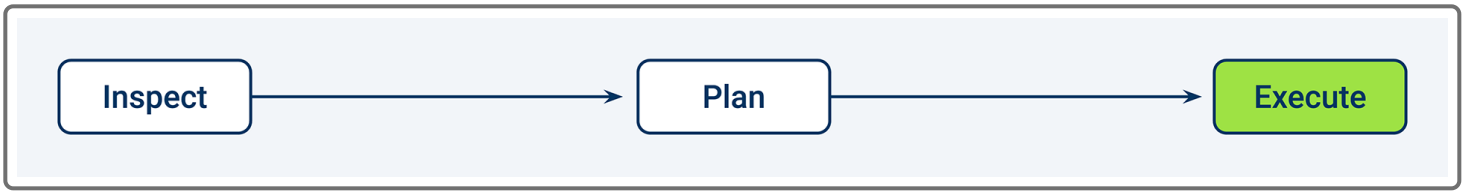


We can parse strings by eliminating characters at the beginning of a string to get the name of each contact in the middle of a string. Let's begin building a regular expression to extract the name from the DataFrame containing the `contacts_string_data.csv` file. Once we find a regular expression that works, we'll use it to extract the name from the "contact_info" column of the `contacts_df` DataFrame..

The regular expression will contain a capture group enclosed in single quotes that contains the following:

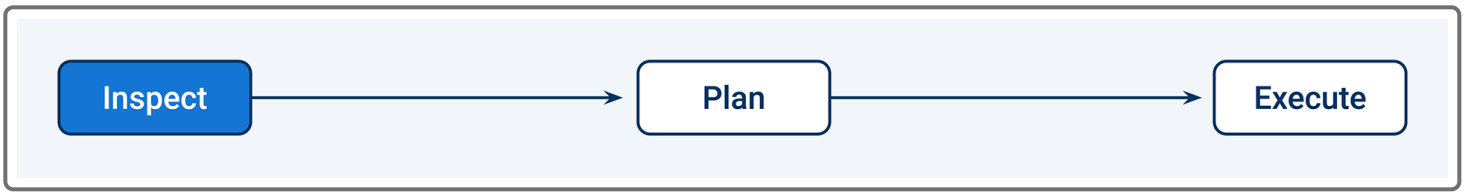
1. A caret, the word "name," a backslash, a lowercase "s," and a plus sign inside brackets (`[^name\s+]`): Ignores the word "name" followed by one or more whitespace characters.
2. The range for uppercase letters, the range for lowercase letters, and a plus sign inside brackets (`[A-Za-z]+`): Captures more than one letter, whether uppercase or lowercase.
3. A backslash, a lowercase "s," and a plus sign (`\s+`): Captures more than one whitespace character.
4. The range for uppercase letters, the range for lowercase letters, and a plus sign inside brackets (`[A-Za-z]+`): Captures more than one letter, whether uppercase or lowercase.

Next, we'll execute the plan.



Test the regular expression by running the following code:

```
# Extract the first and last name after the word "name".
name = re.findall(r'([^name\s+][A-Za-z]+\s+[A-Za-z]+)', string_data)
name
```



The output from running the preceding code is the name and some additional letters from the email address, as follows:

```
['Cecilia Velasco', 'il cecilia']
```

So, we need to modify the regular expression. We can get the name by using list indexing, with `name[0]`. But to extract just the name, we can also add the `il` letters to the existing negative character set, `[^name\s+]`.

Modify your code as follows, and then run it:

```
# Extract the first and last name after the word "name".
name = re.findall(r'([^nameil\s+][A-Za-z]+\s+[A-Za-z]+)', string_data)
name
```

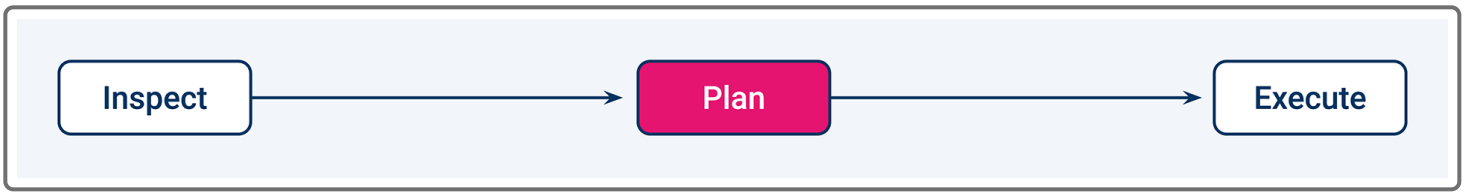
This time, the output consists of only the name, `['Cecilia Velasco']`. The code ignored the letters in the words "name" and "email" and the trailing whitespace characters.

Practice extracting the name from the "contact_info" column of the `contact_df` DataFrame yourself in the following Skill Drill:

SKILL DRILL

Using the Pandas `str.extract` function, extract the name from the "contact_info" column of the `contact_df` DataFrame, and add it to a new column named "name".

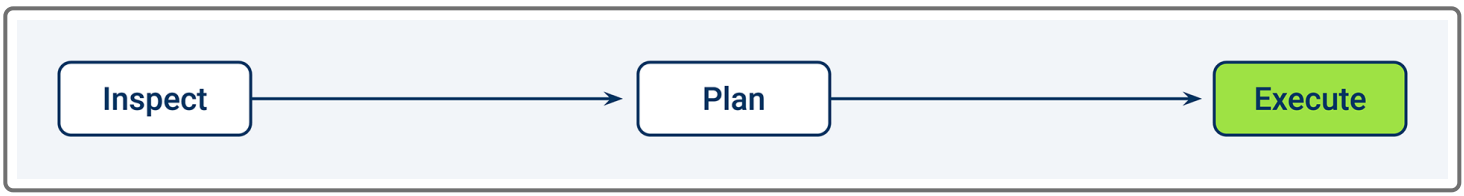
Now, let's go back to planning. We want to create a regular expression that extracts the email address from the `contacts_string_df` DataFrame, and then from the "contact_info" column of the `contacts_df` DataFrame.



Email addresses can contain various characters. So, our regular expression will contain a capture group that consists of the following:

1. A backslash, an uppercase "S", a plus sign, and the at sign (`\S+@`): Captures more than one non-whitespace character before the domain name of an email address.
2. A backslash, a capital "S", and a plus sign (`\S+`): Captures more than one non-whitespace character in the domain name.

Next, we'll execute this plan.



Test the regular expression by running the following code:

```
# Extract the email address using a regular expression pattern.
email_address = re.findall(r'(\S+@\S+)', string_data)
email_address
```

The output from running the preceding code returns the email address, as follows:

```
['cecilia.velasco@rodrigues.fr']
```

Practice extracting the email address from the "contact_info" column of the `contact_df` DataFrame yourself in the following Skill Drill:

SKILL DRILL

Using the Pandas `str.extract` function, extract the email address from the "contact_info" column of the `contact_df` DataFrame, and add it to a new column named "email".

Congratulations on using regular expressions to clean and transform string data!

Next, we'll create the crowdfunding database and load the data.