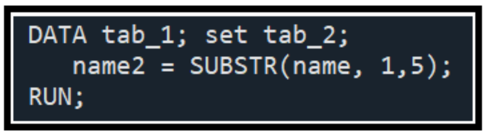
ADDING SAS FUNCTIONS: INSTRUCTIONS FOR USE

This document should allow anyone wishing to participate in the development of this translator to add a SAS function so that they can translate it into Python code.

To understand how to add your codes to the translator, let's take the example of the SAS ***substr*** function. Let's imagine that you want to introduce the ***substr*** function into the translator so that it identifies this function and translates the following SAS code *(cf. figure 1)*.

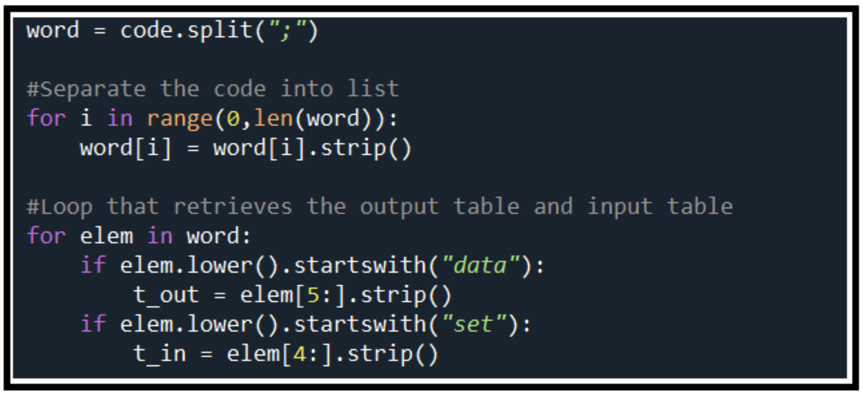


```
DATA tab_1; set tab_2;
   name2 = SUBSTR(name, 1,5);
RUN;
```

Figure 1: SAS code containing the SAS substr function

But before we explain how to introduce your codes, we will show you how our Python function, called ***translator***, identifies the SAS code so that it recognises the input and output table but also the SAS functions to be translated.

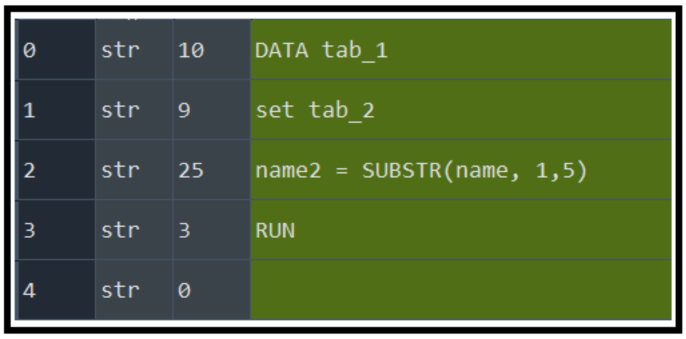
In Figure 2 below, part of the Python code from the ***translator*** function identifies the SAS code by dis differentiating the input and output table.



```
word = code.split(";")

#Separate the code into list
for i in range(0,len(word)):
    word[i] = word[i].strip()

#Loop that retrieves the output table and input table
for elem in word:
    if elem.lower().startswith("data"):
        t_out = elem[5:].strip()
    if elem.lower().startswith("set"):
        t_in = elem[4:].strip()
```

Figure 2: Python code to identify the SAS code

First, the SAS code is presented in a list, which is stored in the variable word, which contains each line of SAS code *(cf. figure 3)*.

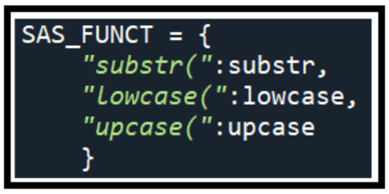


0	str	10	DATA tab_1
1	str	9	set tab_2
2	str	25	name2 = SUBSTR(name, 1,5)
3	str	3	RUN
4	str	0	

Figure 3: Content of the word variable

This method will then identify through a loop each SAS function (or possibly SAS statement) contained in each line of the list.

Before entering your codes, you must add to the ***SAS_FUNCT*** dictionary, in inverted commas, in lower case and followed by an open parenthesis "(", the name of the SAS function you want to integrate *(cf. figure 4)*, which will be the key of the dictionary. You will then need to create the Python function that will have the same name as the value in the dictionary. This function will process and translate the code

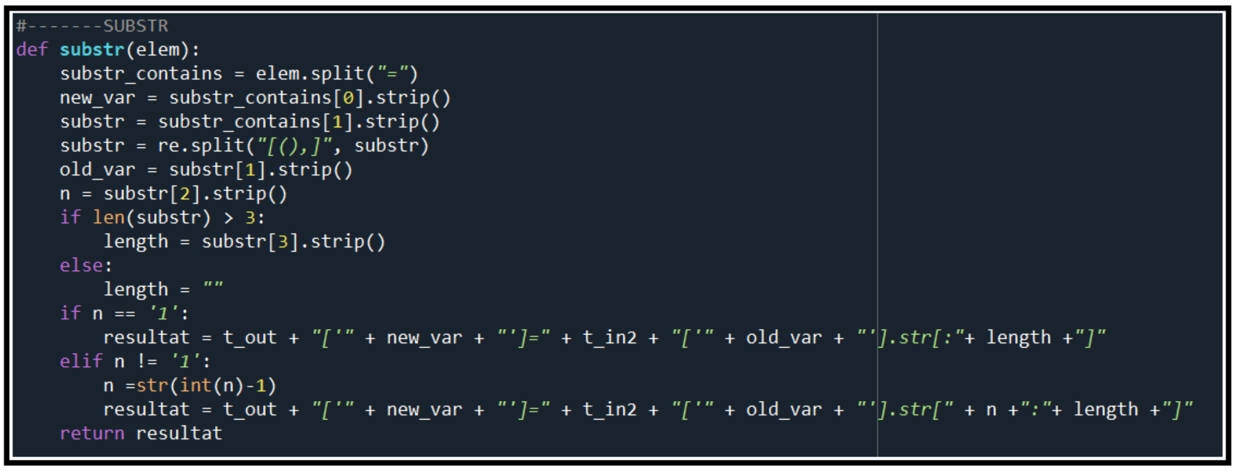


```
SAS_FUNCT = {
    "substr(":substr,
    "lowcase(":lowcase,
    "upcase(":upcase
    }
```

Figure 4: Python dictionary under the name SAS_FUNCT

In our example, the key is "***substr(***" and the value is the Python function we created: ***substr*** *(cf. figure 5).*

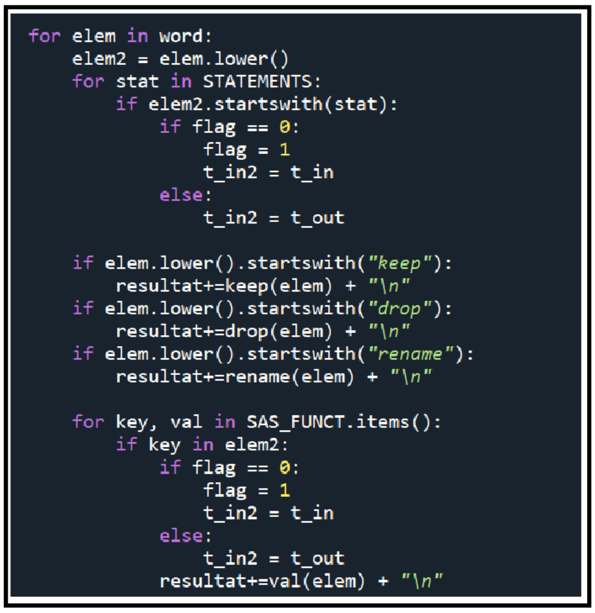
Next, write your Python code into this Python ***substr*** function, which will identify the SAS code containing the SAS function to be translated.



```
#-------SUBSTR
def substr(elem):
    substr_contains = elem.split("=")
    new_var = substr_contains[0].strip()
    substr = substr_contains[1].strip()
    substr = re.split("[(),]", substr)
    old_var = substr[1].strip()
    n = substr[2].strip()
    if len(substr) > 3:
        length = substr[3].strip()
    else:
        length = ""
    if n == '1':
        resultat = t_out + "['" + new_var + "']=" + t_in2 + "['" + old_var + "'].str[:"+ length +"]"
    elif n != '1':
        n =str(int(n)-1)
        resultat = t_out + "['" + new_var + "']=" + t_in2 + "['" + old_var + "'].str[" + n +":"+ length +"]"
    return resultat
```

Figure 5: Creating the Python function substr

You can see that the Python function called ***substr*** *(cf. figure 5)* that we created takes as parameter ***elem,*** which corresponds to the rows of the variable ***word*** *(cf. figure 3)*. The variable ***t_out*** corresponds to the output table, and ***t_in2*** to the input table of the SAS code. In the function, it is essential to define the variable ***resultat*** and to add the variable ***t_out*** and ***t_in2*** to ***resultat***, otherwise the translator will not work. This function will then be called in the translator function in a loop *(cf. figure 6)* which will identify if the string "***substr(***" is contained in one of the lines (***elem***) of the variable ***word***.

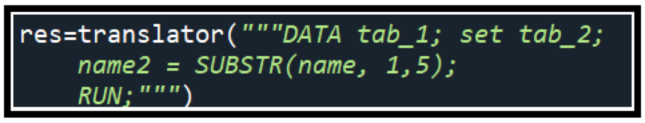


```python
for elem in word:
    elem2 = elem.lower()
    for stat in STATEMENTS:
        if elem2.startswith(stat):
            if flag == 0:
                flag = 1
                t_in2 = t_in
            else:
                t_in2 = t_out

    if elem.lower().startswith("keep"):
        resultat+=keep(elem) + "\n"
    if elem.lower().startswith("drop"):
        resultat+=drop(elem) + "\n"
    if elem.lower().startswith("rename"):
        resultat+=rename(elem) + "\n"

    for key, val in SAS_FUNCT.items():
        if key in elem2:
            if flag == 0:
                flag = 1
                t_in2 = t_in
            else:
                t_in2 = t_out
            resultat+=val(elem) + "\n"
```

Figure 6: Loop to identify SAS functions in the word variable

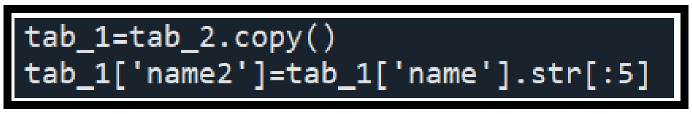
If the string "***substr(***", which is the key we put in the ***SAS_FUNCT*** dictionary, has been identified, then the Python function ***substr*** we created will be called in the loop that calls the Python ***SAS_FUNCT*** dictionary. In this way, the processing and identification will be done by the Python function ***substr*** which will determine the variable names.

Thus, when the translator function is called *(cf. figure 7)*, the variable ***res*** will take as its value a character string which will be the translation of the SAS code *(cf. figure 8)*.



```python
res=translator("""DATA tab_1; set tab_2;
    name2 = SUBSTR(name, 1,5);
    RUN;""")
```

Figure 7: Calling the translator function to translate SAS code



```python
tab_1=tab_2.copy()
tab_1['name2']=tab_1['name'].str[:5]
```

Figure 8: Result of the SAS code translation

You can see that the translator function has successfully translated the SAS code containing the SAS substr function.

So, by following these steps, you can participate in our translator and, if you wish, add other SAS functions to make it even more powerful.