from jinja2 import Environment as \_Environment  
from jinja2 import FileSystemLoader as \_FileSystemLoader  
  
import os as \_os  
  
class Child:  
 def \_\_init\_\_(self, name\_):  
 self.name = name\_  
  
class Dictlike:  
 def \_\_init\_\_(self):  
 self.a = Child("ola")  
 self.b = Child("ela")  
  
 def items(self):  
 return self.\_\_dict\_\_.items()  
  
  
VAR = 3  
DICT\_KTORY\_BEDZIEMY\_WIDZIEC = {  
 "KEY\_KTOREMU\_TRZEBA\_DAC\_VALUE\_ROWNE\_VAR"  
}  
  
template = rf"C:\Users\BryschP\Desktop\testus\_top\TESTUS-Automation\_Framework\PLAYGROUND\my\_template.jinja2"  
output\_file = rf"C:\Users\BryschP\Desktop\testus\_top\TESTUS-Automation\_Framework\PLAYGROUND\output.py"  
context = {  
 'input': "\"JAKIS\_INPUT\_KTORY\_MUSI\_BYC\_STRINGIEM\_BY\_SKOMPILOWALO\"",  
 'if\_var1': 5,  
 'if\_var2': 6,  
 # 'bad\_path': "C:\Users\BryschP\Desktop\testus\_test\_data\tmp\_dbc.dbc",  
 'good\_path': rf"C:\Users\BryschP\Desktop\testus\_test\_data\tmp\_dbc.dbc",  
 "list\_": [4, 5, 6],  
 "dictlike": Dictlike()  
}  
tmpl\_path, tmpl\_file = \_os.path.split(template)  
env = \_Environment(loader=\_FileSystemLoader(tmpl\_path or './'))  
rendered\_content = env.get\_template(tmpl\_file).render(context)  
with open(output\_file, 'w', encoding='utf-8') as fibex:  
 fibex.writelines(rendered\_content)

byle\_jaki\_text = 0  
r"""  
Byle jaki text w komentarzu.  
"""  
  
wstaw\_cos = {{input}} #<- tylko podwojna klamra dziala  
  
from PLAYGROUND.draft1 import DICT\_KTORY\_BEDZIEMY\_WIDZIEC, VAR  
DICT\_KTORY\_BEDZIEMY\_WIDZIEC["KEY\_KTOREMU\_TRZEBA\_DAC\_VALUE\_ROWNE\_VAR"] = VAR  
  
  
### If  
# Ponizej bedzie przeklestwo, jesli context[if\_var1] bedzie rowne 5, inaczej kucyk  
{%- if if\_var1 == 5 %}  
\_ = "Przeklenstwo"  
{% endif -%}  
{%- if if\_var1 != 5 %}  
\_ = "Kucyk"  
{% endif -%}  
# Ponizej bedzie przeklestwo, jesli context[if\_var2] bedzie rowne 5, inaczej kucyk <- wersja z else  
{%- if if\_var2 == 5 %}  
\_ = "Przeklenstwo"  
{%- else %}  
\_ = "Kucyk"  
{% endif -%}  
  
  
### Sciezki  
try:  
 good\_path = {{good\_path}}  
 good\_path\_safe = {{good\_path|safe}}  
 o\_dziwo\_trzeba\_w\_stringu = "{{good\_path}}"  
 a\_najlepiej\_dodac\_jeszcze\_rf = rf"{{good\_path}}"  
except:  
 pass  
# komentarz: sciezka bez rf nawet sie nie kompiluje (Python 3.7?)  
  
  
### For  
{%- for elem in list\_ %}  
"Kolejny element to {{elem}}."  
{% endfor -%}  
  
  
arg1 = "sth"  
arg2 = "sth"  
### Makro  
{%- macro wypisz\_trzy\_razy(argument) -%}  
{%- for i in range(3) %}  
\_ = "{{ argument }}"  
{% endfor -%}  
{%- endmacro %}  
  
# Pierwsze uzycie dla argument = arg1  
{{ wypisz\_trzy\_razy("arg1") }}  
# Drugie uzycie dla argument = arg1 <- argument przekazujemy w (" ") zamiast ( )  
{{ wypisz\_trzy\_razy('arg2') }}  
  
# dictlike.items()|map('last')|selectattr('name', 'equalto', 'ola')| list  
# -> {{ dictlike.items()|map('last')|selectattr('name', 'equalto', 'ola')| list }}  
# dictlike.items()|map('last')|selectattr('name', 'equalto', 'ela')| list  
# ->{{ dictlike.items()|map('last')|selectattr('name', 'equalto', 'ela')| list }}  
# dictlike.items()|map('last')|selectattr('name', 'equalto', 'maksio')| list  
# ->{{ dictlike.items()|map('last')|selectattr('name', 'equalto', 'maksio')| list }}  
  
# Wyjasnienie:  
# map(func, to\_co\_po\_lewej\_od\_|) -> generator  
# def last(iterable) return iterable[-1]  
# selectattr('attrname', 'equalto', 'value', to\_co\_po\_lewej) <=> filter(<zwraca true, jak atrybut elemtu jest rowny value>, to\_co\_po\_lewej) -> generator  
# list(to\_co\_po\_lewej) -> list  
# z listy potrzebna nam wartosc logiczna

byle\_jaki\_text = 0  
r"""  
Byle jaki text w komentarzu.  
"""  
  
wstaw\_cos = "JAKIS\_INPUT\_KTORY\_MUSI\_BYC\_STRINGIEM\_BY\_SKOMPILOWALO" #<- tylko podwojna klamra dziala  
  
from PLAYGROUND.draft1 import DICT\_KTORY\_BEDZIEMY\_WIDZIEC, VAR  
DICT\_KTORY\_BEDZIEMY\_WIDZIEC["KEY\_KTOREMU\_TRZEBA\_DAC\_VALUE\_ROWNE\_VAR"] = VAR  
  
  
### If  
# Ponizej bedzie przeklestwo, jesli context[if\_var1] bedzie rowne 5, inaczej kucyk  
\_ = "Przeklenstwo"  
# Ponizej bedzie przeklestwo, jesli context[if\_var2] bedzie rowne 5, inaczej kucyk <- wersja z else  
\_ = "Kucyk"  
### Sciezki  
try:  
 good\_path = C:\Users\BryschP\Desktop\testus\_test\_data\tmp\_dbc.dbc  
 good\_path\_safe = C:\Users\BryschP\Desktop\testus\_test\_data\tmp\_dbc.dbc  
 o\_dziwo\_trzeba\_w\_stringu = "C:\Users\BryschP\Desktop\testus\_test\_data\tmp\_dbc.dbc"  
 a\_najlepiej\_dodac\_jeszcze\_rf = rf"C:\Users\BryschP\Desktop\testus\_test\_data\tmp\_dbc.dbc"  
except:  
 pass  
# komentarz: sciezka bez rf nawet sie nie kompiluje (Python 3.7?)  
  
  
### For  
"Kolejny element to 4."  
  
"Kolejny element to 5."  
  
"Kolejny element to 6."  
arg1 = "sth"  
arg2 = "sth"  
### Makro  
  
# Pierwsze uzycie dla argument = arg1  
  
\_ = "arg1"  
  
\_ = "arg1"  
  
\_ = "arg1"  
  
# Drugie uzycie dla argument = arg1 <- argument przekazujemy w (" ") zamiast ( )  
  
\_ = "arg2"  
  
\_ = "arg2"  
  
\_ = "arg2"  
  
  
# dictlike.items()|map('last')|selectattr('name', 'equalto', 'ola')| list  
# -> [<\_\_main\_\_.Child object at 0x000002BE845DA588>]  
# dictlike.items()|map('last')|selectattr('name', 'equalto', 'ela')| list  
# ->[<\_\_main\_\_.Child object at 0x000002BE845CE160>]  
# dictlike.items()|map('last')|selectattr('name', 'equalto', 'maksio')| list  
# ->[]  
  
# Wyjasnienie:  
# map(func, to\_co\_po\_lewej\_od\_|) -> generator  
# def last(iterable) return iterable[-1]  
# selectattr('attrname', 'equalto', 'value', to\_co\_po\_lewej) <=> filter(<zwraca true, jak atrybut elemtu jest rowny value>, to\_co\_po\_lewej) -> generator  
# list(to\_co\_po\_lewej) -> list  
# z listy potrzebna nam wartosc logiczna