Dynamic Rule-based Engine in Python

11.11.2020

Kunal Ostwal
Intern
kunaldostwal@gmail.com

Abstract

Developing a Rule-based Engine in Python 3 using the business rule engine library that can dynamically load a dataset and rules from an Excel sheet and generate an output in another Excel file.

Advantages

- Rules can be changed dynamically through the Excel sheet. No technical or computing knowledge is required to modify the code for custom purposes.
- A rule based engine reduces the number of loops used in the program.
- The business rule engine library uses Excel-like functions. Hence, if any part of the preset conditions need modification, it can be done easily.

Goals

1. Make a system that allocates students' scholarships on the basis of their total marks and attendance.
2. Load the student data and scholarship conditions from an Excel and generate an Excel output that lists the scholarship allocation with the students' marks and attendance.

Specifications

Python 3.6+ (Tested in Python 3.8.6)

Libraries used:

Pandas 1.0.5 (https://pypi.org/project/pandas/)

Business rule engine 0.0.2 (https://pypi.org/project/business-rule-engine/)

Documentation

Student Data:

A: Attendance

T: Total Marks

S: Subject

	A	B	C	D	E	F	G	H	I	J
1	Roll	Name	S1	S2	S3	S4	S5	A	T	
2	1	Zorina Abreu	48	41	48	8	37	149	182	
3	2	Zhen Abu-Zahra	32	1	16	0	11	183	60	
4	3	Zhanetta Adeyeye	38	41	25	35	24	80	163	
5	4	Yunzhe Afonso	35	45	34	3	49	32	166	
6	5	Youngjin Ahn	32	8	45	48	31	99	164	
7	6	Yu Ahn	1	37	18	22	25	150	103	
8	7	Yoon Akin-Aderibigbe	31	23	11	26	36	188	127	
9	8	Yookyung Alexander	24	36	48	10	11	42	129	
10	9	Yi-Shiuan Alsamdan	6	41	8	14	18	97	87	
11	10	Yingda Alter	19	15	0	7	28	83	69	
12	11	Ying Altmann	19	27	30	33	33	36	142	
13	12	Yi-Feng Alvarez	16	26	21	17	31	28	111	
14	13	Yi Aramendia	15	42	12	21	2	84	92	
15	14	Yaya Ashkenazi	26	26	38	13	48	91	151	
16	15	Yat-Lun Atri	48	28	23	7	43	9	149	
17	16	Yasuhiro Au	16	1	27	8	44	99	96	
18	17	Yanwen Aurori	40	20	17	42	42	50	161	
19	18	Yan Austin	24	25	14	18	36	34	117	
20	19	Ya-Han Bagdat	19	7	3	2	46	160	77	
21	20	Yael Bala	37	7	6	7	34	114	91	
22	21	Wushen Banovac	42	32	4	18	12	74	108	

Rules defined in Excel:

	A	B	C	D	E	F	G
1	Rule Name	Condition_1	Value_1	Condition_2	Value_2	Action_1	
2	Amount_200(	>	160	>	50	2000	
3	Amount_150(	>	140	>	35	1500	
4	Amount_100(	>	120	>	25	1000	
5	Amount_500	>	75	>	20	500	
6	Amount_400	>	75	<	20	400	
7							
8							
9							

Code:

```
from business_rule_engine import RuleParser
import pandas as pd
```

Imports the required libraries.

```
params = pd.read_excel('Student Dataframe.xls',).to_dict('records')
cond = pd.read_excel('Student Dataframe.xls','rules').to_dict('records')
pd.DataFrame().to_excel('output1.xls')
print(params[0])
print(cond[0])
```

```
{'Roll': 1, 'Name': 'Zorina Abreu', 'S1': 44, 'S2': 12, 'S3': 21, 'S4': 6, 'S5': 40, 'A': 175,
'T': 123}
{'Rule Name': 'Amount_2000', 'Condition_1': '>', 'Value_1': 160, 'Condition_2': '>', 'Value_2': 5
0, 'Action_1': 2000}
```

Importing Excel student data and rules as Pandas Dataframe.

Converting the Pandas Dataframe to dictionary format.

```
def append_df_to_excel(df,excel_path):
    df_excel = pd.read_excel(excel_path)
    result = pd.concat([df_excel, df], ignore_index=True)
    result.to_excel(excel_path, index=False)
```

append_df_to_excel function reads Excel file as a Pandas Dataframe.
df is added to the Pandas Dataframe (*df_excel*) and stored as *result.*
result is converted to an Excel file and stored at *excel_path.*

```
def scholarship(name, roll, total, att, amt):
    sch_result=
pd.DataFrame([roll,name,total,att,amt],index=['Roll','Name','Total','Att','
Amount'])
    sch_result=sch_result.swapaxes('index','columns')
    append_df_to_excel(sch_result,'output1.xls')
```

The parameters provided to *scholarship* function are converted to a Pandas Dataframe and stored as *sch_result*.
append_df_to_excel function is called and *sch_result* is added to the output Excel file.

```
rule_name=[]
condition_1=[]
value_1=[]
condition_2=[]
value_2=[]
action_1=[]

for variable in cond:
    rule_name.append(variable['Rule Name'])
    condition_1.append(variable['Condition_1'])
    value_1.append(variable['Value_1'])
    condition_2.append(variable['Condition_2'])
    value_2.append(variable['Value_2'])
    action_1.append(variable['Action_1'])

print(rule_name)
print(condition_1)
print(value_1)
print(condition_2)
print(value_2)
print(action_1)
```

```
['Amount_2000', 'Amount_1500', 'Amount_1000', 'Amount_500', 'Amount_400']
['>', '>', '>', '>', '>']
[160, 140, 120, 75, 75]
['>', '>', '>', '>', '<']
[50, 35, 25, 20, 20]
[2000, 1500, 1000, 500, 400]
```

Variables of different rules are loaded with a for loop and are stored in different lists.
In this example, *condition_1* and *value_1* correspond to Total marks and *condition_2* and *value_2* correspond to Attendance.

```
rules = ""
i=0
while i<len(cond):
    rules+="""
    rule "{0}"
    when
        AND(T {1} {2} , A {3} {4})
    then
        scholarship(Name, Roll, T, A, {5})
    end

""".format(rule_name[i],condition_1[i],value_1[i],condition_2[i],value_2[i]
,action_1[i])
    i+=1

print(rules)
```

```
rule "Amount_2000"
when
    AND(T > 160 , A > 50)
then
    scholarship(Name, Roll, T, A, 2000)
end

rule "Amount_1500"
when
    AND(T > 140 , A > 35)
then
    scholarship(Name, Roll, T, A, 1500)
end

rule "Amount_1000"
when
    AND(T > 120 , A > 25)
then
    scholarship(Name, Roll, T, A, 1000)
end

rule "Amount_500"
when
    AND(T > 75 , A > 20)
then
    scholarship(Name, Roll, T, A, 500)
end

rule "Amount_400"
when
    AND(T > 75 , A < 20)
then
    scholarship(Name, Roll, T, A, 400)
end
```

The rules are defined as a string and then passed through the RuleParser.
The business rule engine library uses Excel-like functions.
The rules are added with variables from the list created earlier.

```
for x in params:
    parser = RuleParser()
    parser.register_function(scholarship)
    parser.parsestr(rules)
    parser.execute(x)
```

This loops through the student dataset and, according to the rules, allots the scholarships and adds the data to the *"output1.xls"* file.

Output:

	A	B	C	D	E	F
1	**Roll**	**Name**	**Total**	**Att**	**Amount**	
2	1	Zorina Abreu	123	175	1000	
3	2	Zhen Abu-Zahra	148	72	1500	
4	3	Zhanetta Adeyeye	140	37	1000	
5	4	Yunzhe Afonso	153	183	1500	
6	5	Youngjin Ahn	109	156	500	
7	6	Yu Ahn	110	93	500	
8	8	Yookyung Alexander	116	198	500	
9	9	Yi-Shiuan Alsamdan	179	184	2000	
10	10	Yingda Alter	123	50	1000	
11	11	Ying Altmann	182	79	2000	
12	12	Yi-Feng Alvarez	77	64	500	
13	13	Yi Aramendia	158	86	1500	
14	14	Yaya Ashkenazi	114	67	500	
15	15	Yat-Lun Atri	200	69	2000	
16	16	Yasuhiro Au	90	6	400	
17	17	Yanwen Aurori	179	25	500	
18	18	Yan Austin	124	162	1000	
19	19	Ya-Han Bagdat	89	71	500	
20	20	Yael Bala	135	11	400	
21	21	Wushen Banovac	87	97	500	
22	22	Won Barakat	94	21	500	
23	24	William Barrett	86	131	500	
24	26	William Baxter	105	51	500	

All the students who were allotted a scholarship are added to the excel.

Conclusion

The business rule engine is an easy-to-use tool that can build a robust, scalable system that can define multiple rules and conditions. This can be used in several applications like filtered product search, tax bracket calculation, etc. The above mentioned example demonstrates its effectiveness and efficiency. It can be replicated for various other sectors and fields as well.