Grand Challenge

Z06394 Chen Boyan
Hosei University(Japan)

- # Work Flow

  - Python COBOL Generator.
  - How to use
  - Result

Python COBOL Generator

- In this section, I built a simple Cobol code generator with python.
- Use python String Template to achieve the goal.

How to use

1. First import the COBOL and define the identification_divison and environment_division.

```
from COBOL import COBOL


c=COBOL()
c.identification_division.update(PROGRAM_ID="123",AUTHOR="chen" )
c.envirnoment_division.update(InputFile="PRTLINE",
                              InputName="PRT-LINE",
                              OutputFile="PRTDONE",
                              OutputName="PRT-DONE")
```

2. The function toPrint() can display the current file at the console.

```
c.toPrint()
```

- # How to use

3. Define the data_division fields, files[0] is the InputName we defined in environment_division; files[1] is the OutputName.
4. For files that have 05 level fields we can use addChild() function to add the field .

```
c.data_division.files[0].addData_division_F01_a(fieldName="PRT-REC",
                      stype="X",
                      length="80")
#


c.data_division.files[1].addData_division_F01_b(fieldName="PRT-REC-DONE") # add file 01
c.data_division.files[1].data_division_F01_b.addChild(fieldName2="PRT-DATE",
                      stype2="X",
                      length2="8")# add 05 child fields
c.data_division.files[1].data_division_F01_b.addChild(fieldName2="FILLER",
                      stype2="X",
                      length2="1")
```

- # How to use

5. Define the work station variables, alike the last step

```
c.data_division.addWorkStationFileA(fieldName4="PGM-VARIABLES")# add working storage variables
c.data_division.workstation[0].addWorkChild(fieldName3="PGM-COUNT",
                            stype3="9",
                            length3="05")# add 05 child variables
#
c.data_division.addWorkStationFileB(fieldName6="YYYYMMDD",
                                    stype6="9",
                                    length6="8")#add working storage variables with out 05 level
```

- # How to use

6. Add a function at the procedure_division.
7. First give a functionName, the function are stored in a dictionary.
8. Then use the functionName and addStep with keyworks(up to 4).

```
4 ################################################################
5 c.procedure_division.addFunction(functionName="A000-START")
6 c.procedure_division.funcs["A000-START"].addStep("DISPLAY","'HELLO WORLD'")
7 c.procedure_division.funcs["A000-START"].addStep("OPEN","OUTPUT","PRT-LINE")
8 ################################################################
```

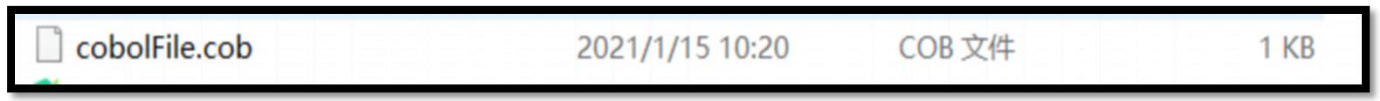
- # How to use

9. Save the file.

```
c.save2File(fileName="cobolFile")
```

- # Result

The generated code are save in a .cob file.




```
IDENTIFICATION DIVISION.
PROGRAM-ID.     123.
AUTHOR.         chen.
ENVIRONMENT DIVISION.
*
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT PRT-LINE   ASSIGN TO PRTLINE.
    SELECT PRT-DONE   ASSIGN TO PRTDONE.

DATA DIVISION.
FILE SECTION.

FD  InputName  RECORD CONTAINS 80 CHARACTERS RECORDING MODE F.

01  PRT-REC         PIC X(80) VALUE SPACES.

FD  OutputName RECORD CONTAINS 80 CHARACTERS RECORDING MODE F.

01  PRT-REC-DONE.
    05  PRT-DATE        PIC X(8)  VALUE SPACES.
    05  FILLER          PIC X(1)  VALUE SPACES.

WORKING-STORAGE SECTION.


01  fieldName4.
    PGM-COUNT          PIC 9(05).

01  YYYYMMDD          PIC 9(8).


PROCEDURE DIVISION.


A000-START.
    DISPLAY  'HELLO WORLD'
    OPEN     OUTPUT   PRT-LINE
```

- Thank you