Kumaravelu Sarabojimahalingam (Cognizant)

COGNIZANT TECHNOLOGY SOLUTIONS LTD

**Alternatives to DFsort/Syncsort features in Python - A comparative study**

Table of Contents

[Objective 3](#_Toc479933760)

[Target audience 3](#_Toc479933761)

[Approach 3](#_Toc479933762)

[Assumptions/Constraints 3](#_Toc479933763)

[Feature 1: Filtering using DFsort/Syncsort and Python 4](#_Toc479933764)

[Applicable DFsort/Syncsort statements 4](#_Toc479933765)

[Python way 4](#_Toc479933766)

[How/where/why Python scores over DFsort/Syncsort 5](#_Toc479933767)

[Feature 2: Reformatting using DFsort/Syncsort and Python 5](#_Toc479933768)

[Applicable DFsort/Syncsort statements 5](#_Toc479933769)

[Python way 6](#_Toc479933770)

[How/where/why Python scores over DFsort/Syncsort 7](#_Toc479933771)

[Feature 3: Sorting using DFsort/Syncsort and Python 8](#_Toc479933772)

[Applicable DFsort/Syncsort statements 8](#_Toc479933773)

[Python way 8](#_Toc479933774)

[How/where/why Python scores over DFsort/Syncsort 8](#_Toc479933775)

[Feature 4: Copying using DFsort/Syncsort and Python 9](#_Toc479933776)

[Applicable DFsort/Syncsort statements 9](#_Toc479933777)

[Python way 9](#_Toc479933778)

[How/where/why Python scores over DFsort/Syncsort 9](#_Toc479933779)

[Feature 5: Summarizing using DFsort/Syncsort and Python 10](#_Toc479933780)

[Applicable DFsort/Syncsort statements 10](#_Toc479933781)

[Python way 10](#_Toc479933782)

[How/where/why Python scores over DFsort/Syncsort 11](#_Toc479933783)

[Feature 6: Selecting using DFsort/Syncsort and Python 12](#_Toc479933784)

[Applicable DFsort/Syncsort statements 12](#_Toc479933785)

[Python way 12](#_Toc479933786)

[How/where/why Python scores over DFsort/Syncsort 13](#_Toc479933787)

[Feature 7: Joining using DFsort/Syncsort and Python 14](#_Toc479933788)

[Applicable DFsort/Syncsort statements 14](#_Toc479933789)

[Python way 14](#_Toc479933790)

[How/where/why Python scores over DFsort/Syncsort 14](#_Toc479933791)

[Feature 8: Reporting using DFsort/Syncsort and Python 15](#_Toc479933792)

[Applicable DFsort/Syncsort statements 15](#_Toc479933793)

[Python way 15](#_Toc479933794)

[How/where/why Python scores over DFsort/Syncsort 16](#_Toc479933795)

[Findings and conclusion 17](#_Toc479933796)

## Objective

The objective of this paper is to delineate the probable alternatives to various DFsort/Syncsort features available in Python. It also lists down the areas where Python scores over the limitations/constraints found in the quoted mainframe utilities. Pandas (data analysis and manipulation toolkit) is the Python library used for this comparison study

## Target audience

* Any technology specialist who extensively uses these sort utilities within z-series for data analysis and reporting.
* Any technology specialist without a mainframe background but interested to understand the nitty-gritties of the features available in the 2 most widely used sort utilities across industries.

## Approach

Text files serve as inputs to Python scripts alike datasets/PS files to DFsort/Syncsort utilities. Data in the text file is initially loaded into Pandas ‘panels’ data structures and then the functionalities are compared.

## Assumptions/Constraints

* It is assumed that any packed/binary (COMP) data in the datasets/PS files is transformed to either SFF(signed free form)/ UFF(unsigned free form) or ZD(zoned decimal) before the comparison is done.
* The comparison tests were done against files <=250MB in size. Although pandas support large sized files through chunking concept, it was deliberately kept outside of this study’s scope.
* Only most widely used DFsort/Syncsort statements are considered in this study.

## Feature 1: Filtering using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

INCLUDE COND, OMIT COND, OUTFIL INCLUDE/OMIT (included numeric/alphanumeric tests, comparison tests)



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. As column names or headers can be easily attributed to the data read from a fixed width file, it is easier for a programmer to understand the filter conditions applied
2. If the data in the text file is comma separated and squeezed rather than a fixed width file, it is easier in python to adapt to such a layout change. i.e., read statement alone should be replaced. There are no additional statements needed alike DFsort/Syncsort (INREC PARSE statement is needed to parse the data to fixed width first before applying the filters).



1. It is easier to code and maintain in Python if the need is to filter records based on specific patterns than compared to mainframe sort utilities. This is made possible with the regex module



## Feature 2: Reformatting using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

INREC FIELDS, OUTREC FIELDS, OUTFIL OUTREC, OUTREC/INREC OVERLAY, OUTREC BUILD, OUTFIL BUILD, SQZ, TRAN=ALTSEQ, FINDREP, JFY







### Python way





### How/where/why Python scores over DFsort/Syncsort

1. As names can be associated with the fixed width columns in Python it is easier to maintain the code compared to DFsort/Syncsort utilities.
2. Squeezing columns having more than one word in DFsort/Syncsort requires additional handling (using quotes and pairing) and moreover the length of the column fields is kept intact without realizing any savings in bytes. In contrast, Iterrows Python generator keeps the squeezing simpler and maintainable
3. Removal of non-printable characters or replacing a target pattern is made easier (more lines of code required in DFsort/Syncsort) using the Python regex module

## Feature 3: Sorting using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

SORT, SORT FIELDS, SUM FIELDS



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. As names can be associated with the fixed width columns in Python it is easier to maintain the code compared to DFsort/Syncsort utilities.
2. Summing on the sorted/grouped field is easily achieved using ‘groupby’ method in python. The output is more readable compared to DFsort/Syncsort’s output
3. No need to explicitly mention the data types of the fields considered in the sort/group. Python automatically infers based on the data spread

## Feature 4: Copying using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

COPY, SORT FIELDS=COPY, OUTFIL FNAMES, IFTHEN WHEN…BUILD



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. When the need is to copy the data as comma separated Python comes handy over DFsort/Syncsort. Also, directing the output to an excel is easier using Python



1. Complex conditions can be easily coded and maintained in Python than DFsort/Syncsort

## Feature 5: Summarizing using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

COUNT, OCCURS HIGHER/LOWER/ALLDUPS/NODUPS, RANGE



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. Any Boolean operations are possible in Python which is not possible using DFsort/Syncsort’s OCCURS statement like the below one. RANGE statement can be used but it’s output is quite different from that of OCCURS and moreover RANGE can be used only on numeric fields.



1. Describe, info, dtypes methods can be used on a dataframe to get basic and quick insights on the data spread in the file. Although DFsort/Syncsort have STATS statement it doesn’t reveal extensive info like Python methods.





1. Only a maximum of 10 ON fields can be used in as single OCCUR/RANGE statement of DFsort/Syncsort. Any # of columns can be used get the value counts in Python.

## Feature 6: Selecting using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

SELECT FIRST/FIRSTDUP/LAST/LASTDUP/HIGHER/LOWER



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. Only counts between 0-99 can be specified in HIGHER/LOWER/EQUAL operands of DFsort/Syncsort’s SELECT statement which is not the case with Python programming.
2. Alike OCCURS, Boolean operations are not possible in SELECT statement
3. In python, using head and tail method top or bottom n rows can be easily selected as follows



## Feature 7: Joining using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

JOIN KEYS, SPLICE



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. As the dataframe joins are like RDMS joins (left/right/inner/outer) the code is readable and maintainable than compared to the DFsort/Syncsort statements
2. With the concat() method in Python files with different layouts can be combined and processed. When the rows are combined row-wise Python automatically applies NULL values and aligns the data based on the column headers. DFsort/Syncsort lacks this feature.



## Feature 8: Reporting using DFsort/Syncsort and Python

### Applicable DFsort/Syncsort statements

DISPLAY HEADER/TITLE/TOTAL/BTOTAL



### Python way



### How/where/why Python scores over DFsort/Syncsort

1. Dataframes can be easily exported to excel or CSV file which is not possible using the mainframe utilities.
2. Most of the excel formatting techniques can be easily made possible using openpyxl Python package
3. Grand total and subtotals calculations are also made easier using the pivot table methods and numpy package as shown below



## Findings and conclusion

It is found that most of the sort, merge, copy, analysis, and reporting done by DFsort/Syncsort mainframe utilities can be quickly done in Python using pandas package. Python’s affinity to easily export/import the analyzed/sorted/merged/copied data to/from CSV/excel files makes it preferable than these mainframe utilities. Python being dynamically-type language alike REXX combined with the versatility of pandas makes it a more compelling alternative to DFsort/Syncsort utilities.