The 9522: A python project

Heidi Beezub

December 4, 2017

Figure: Beezub

Outline

Install and Load Libraries

Using PythonInR to access the python code through R studio

The 9522 and Rejects

- The DCM 9522 report is one of the reports I use in order to correct 'rejects' in my job.DCM stand for Department Cost Manager. It includes multiple departments and all the products (both supplies services) that each department provides.

The 9522 and Rejects

- The DCM 9522 report is one of the reports I use in order to correct 'rejects' in my job.DCM stand for Department Cost Manager. It includes multiple departments and all the products (both supplies services) that each department provides.
- **What is a reject?**
  A reject is data that the system collects but then cannot process or put into the correct category. Rejects are typically a 'new' item that the simply needs the interface created or built in the system. Sometimes this is an existing item that is either coming into the system in a new way.

The 9522 and Rejects

- The DCM 9522 report is one of the reports I use in order to correct 'rejects' in my job.DCM stand for Department Cost Manager. It includes multiple departments and all the products (both supplies services) that each department provides.
- **What is a reject?**
  A reject is data that the system collects but then cannot process or put into the correct category. Rejects are typically a 'new' item that the simply needs the interface created or built in the system. Sometimes this is an existing item that is either coming into the system in a new way.
- The 9522 helps me to determine if this is an existing item for which I need to create an additional interface. Or if it is a new product, which means I need to create the new product (including the interface).

Background continued

- The report is unwieldy (6,600 pages). I have to save the text file as a word file then search the word file for the feeder key information. Since this is typically numerical it can result in multiple 'hits' many for the wrong field. I also cannot easily compare the multiple hits in the word file.

Background continued

- The report is unwieldy (6,600 pages). I have to save the text file as a word file then search the word file for the feeder key information. Since this is typically numerical it can result in multiple 'hits' many for the wrong field. I also cannot easily compare the multiple hits in the word file.
- As an excel file I would be able to search on the columns and view any similar items (all at the same time). It helps in identifying items to see what department(s) similar items are used in.

Methodology

- **Source data** My first step was to take a VERY critical look at the structure of the report. The report includes headers on each page which can bisect the data at any point. In addition to the information I want, the data also includes data I don't want that is part of the Bill of Materials.

Methodology

- **Source data** My first step was to take a VERY critical look at the structure of the report. The report includes headers on each page which can bisect the data at any point. In addition to the information I want, the data also includes data I don't want that is part of the Bill of Materials.
- **The desired end result** A csv (excel file) that I will be able to filter to check items, sort to identify duplicate products and otherwise manipulate the data.

Methodology

- **Source data** My first step was to take a VERY critical look at the structure of the report. The report includes headers on each page which can bisect the data at any point. In addition to the information I want, the data also includes data I don't want that is part of the Bill of Materials.
- **The desired end result** A csv (excel file) that I will be able to filter to check items, sort to identify duplicate products and otherwise manipulate the data.
- **Pseudocode** First, I wrote down (pen paper) what I needed to do. Insert picture

Code for IP Number

```
1  def process_file(reader):
2      result_line= ''
3      result=''
4      slide="IPNUM" +'\n'
5      #first we need to add headers
6      with open('9522_slide_new.csv', 'a') as output_file
       :
7          output_file.write('"IPNUM"' +'\n')
8      for line in reader:
9          line=line.strip()    #removes leading/trailing
       whitespace
10         field = line.split()
```

Code for IP Number continued

```
1
2  if len(field)>3:
3          for i in range(0,2):
4              #find IPNUM
5              if field[i] == 'IP' and field[i+1] =='NUM':
6                  #save IPNUM
7                  ipnum=field[i+2].strip(':')
8                  #(last item before write to file)
9                  #when result has no data it is a blank line
10                 if result_line != None:
11                     with open('9522_slide_new.csv', 'a') as
       output_file:
12                         list_2_line=ipnum
13                         output_file.write(list_2_line+'\n')
14                         result=''
15                         list_2_line=''
16
17 if __name__ == '__main__':
18     with open('9522_slide.txt', 'r') as input_file:
19         process_file(input_file)
```

Seq num, feeder system and feeder key

blank slide

Product description

blank slide

Debugging

blank slide

Additional Debugging

Blank slide

Conclusion

Blank slide