Address Linkage Key with AnalyticsIQ
Milestone 2 Presentation

Project Owners :

Warren Smith

Jin Wang

Project Team :

Kat McElveen, Team Lead

Saritha Gudala, Technical Writer

Kat Greer, Technical Specialist

Byron Smith, Technical Specialist

IT7993 Capstone, Spring 2021

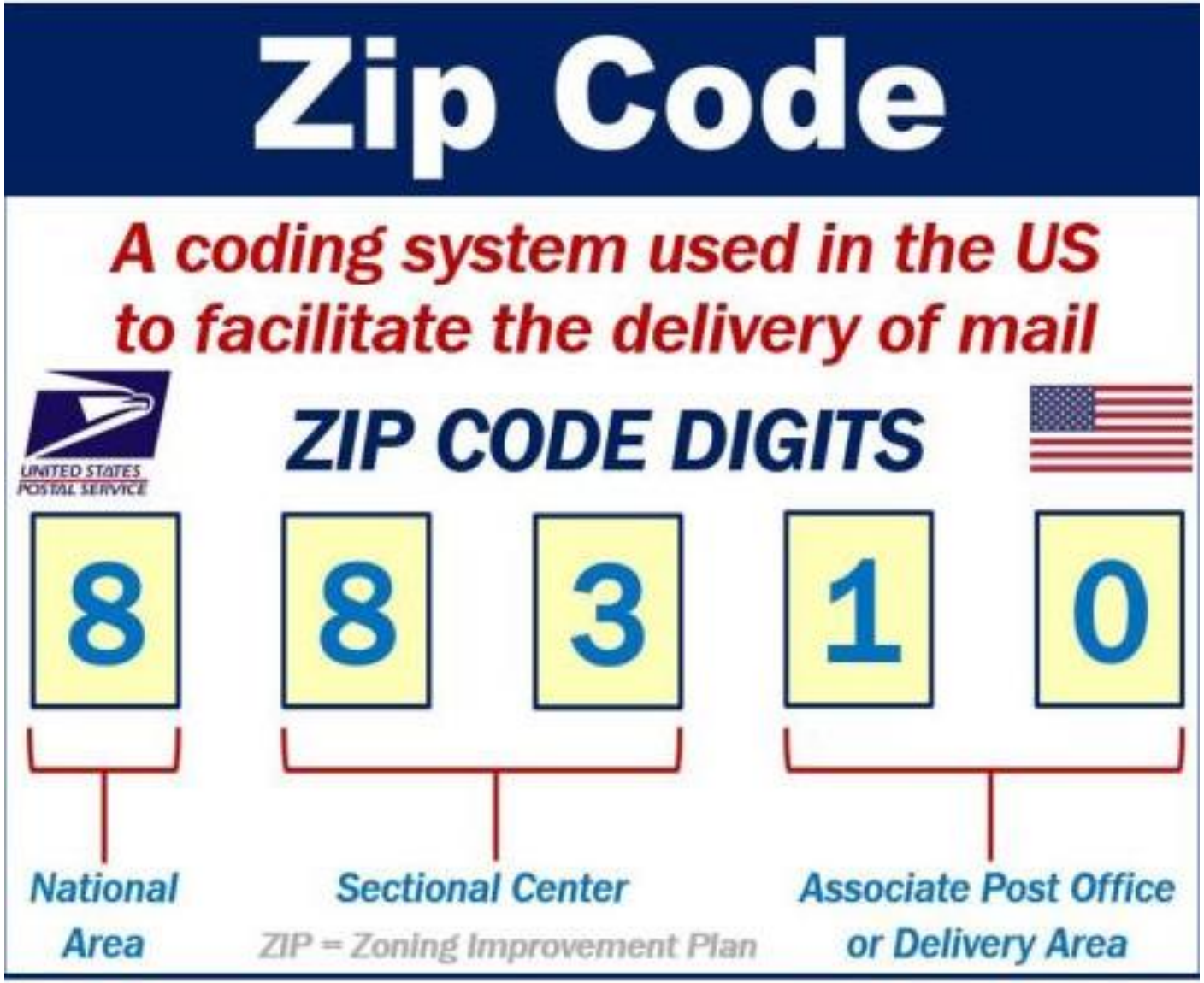
March 26, 2021

Project Overview

- Develop a tool/set of tools to compute the delivery point (DPBC) of a given address following the guidelines of USPS.
- The tool will be initially developed and tested for properly formatted addresses.
- As part of further improvements, it will be tested for improper address format and/or for the presence of certain types of delivery points such as high-rise buildings.

What is Minimally Viable Product?

- A tool or set of tools which allow me to compute the delivery point code of a given address given:
  1) A “clean” address (as defined by publication 28)
  2) ZIP and ZIP+4 appended
  3) A dataset which contains Highrise addresses



Project Progress Summary

- Project team continued weekly meetings (Wednesday and Friday) to assess the progress and discuss further plans.
- As a major part of project deals with regular expressions, team members studied and applied regular expressions for implementation.
- Completed the development part of minimally viable product.
  - Team initially self-assigned tasks according to DPBC rules
  - After team discussions, some tasks were combined or discarded because they were redundant, unnecessary, or overly burdensome
  - Team utilized Jupyter Notebooks for local testing
  - Functions and Test Cases were created for the applicable rules
  - Team used Git (via TortoiseGit and command line) to commit and push local edits, which were then merged into the master branch upon approval of Project Owner
- For next milestone, we plan to extend the implementation of tool for secondary address rules (addresses with apartment numbers, PO Boxes, rural route addresses)

Milestone 2 Deliverables Status Update

Complete:

- Finished the implementation of minimally viable product.
- Currently we can generate a DBPC for all scenarios stated in slide 6 and 7 (excluding 4 and 11)

In Progress:

- CI/CD Integration

Changed:

- Eliminated rules 4 and 11
- Combined rules 2, 7, and 13

Delivery Point Barcode Rules (Primary)

1. General Rule

Address: 1234 MAIN ST (PO BOX 44, RR 1 BOX 154, HC 1 BOX 1264)
DPBC: 34 (44, 54, 64)

Use last two digits. Print code characters in DPBC representing last two digits of primary street number (or post office box, rural route box, or highway contract route number).

2. No Numbers

Address: MAIN St (RR 1, HC 1)
DPBC: 99 (99, 99)

Use 99. Print code characters in DPBC representing last two digits of primary street number (or PO Box, rural route, or highway contract route number).

3. Single Digits

Address: 8 MIAN St (PO BOX 1, RR 1 BOX 2, HC 1 BOX 3)
DPBC: 08 (01, 02, 03)

Add leading zero. Print code characters in DPBC representing leading zero and single digit.

4. Fractional Number

Address: 1234 ½ MAIN ST (PO BOX 1 ½. RR 1 BOX 2 ¾, HC 1 BOX 10 ¼)
DPBC: 34 (01, 02, 10)

Ignore fraction. Print code characters in DPBC representing two digits to left of fraction. If single digit to left of fraction, add leading zero.

8. Leading/Embedded Alphas

Address: 23S41 MAIN ST (23S4 MAIN ST, 2W3S1 MAIN ST, MAINS ST, C8INT)
DPBC: 11 (04, 01, 01)

Print code characters in DPBC representing last two digits to right of alphas. If single digit to right of alphas, add leading zero.

9. Slashes (/)

Address: 123/4 MAIN ST (PO BOX ¼, RR 1 BOX 123/124/125, H 3 BOX 11/13)
DPBC: 23 (03, 23, 07)

Print code characters in DPBC representing 99 whenever a slash appears directly next to numeric in the primary street number.

10. Other Embedded Symbols

Address: 1.23 MAIN ST (PO BOX 1-3, RR 1 BOX 1.23, HC 3 BOX 11*7)
DPBC: 23 (03, 23, 07)

Use last two digits to right of the symbol. Print code characters in DPBC representing last two digits to the right of all symbols (except slashes), such as periods and hyphens appearing in primary street numbers. If single digit to right, add leading zero.

11. Embedded Spaces

Address: 1 23 MAIN ST (PO BOX 1 3, RR 1 BOX 1 7, HC 1 BOX 12 34)
DPBC: 23 (03, 07, 34)

Treat embedded spaces like other symbols (Rule 10). Print code characters in DPBC representing last two digits to right of space. If single digit to right, add leading zero.

5. Trailing Alphas Address: 1234A MAIN ST (PO BOX 4A, RR 1 BOX 154A, HC 1 BOX 12644AA) DPBC: 34 (04, 54, 44) Ignoring trailing alphas. Print code characters in DPBC representing last two digits to left of space and alphas. If single digit to left of space and alphas, add leading zero.	**12. Numeric Street Names** Address: 8 33 ST (123 7th ST) DPBC: 08 (23) Ignore numeric street name. Print code characters in DPBC representing last two digits of primary street number (Rule 1).
6. Spaces and Alphas Address: 1234 A MAIN ST (PO BOX 4A, RR 1 BOX 154A, HC1 BOX 12644AA) DPBC: 34 (04, 54, 44) Ignoring space and alphas. Print code characters in DPBC representing two digits to left of space and alphas. If single digit to left of space and alphas, add leading zero.	**13. All other Anomalies** Use 99. Print code characters in DPBC representing 99 for conditions not covered by Rules 1 – 12.
7. Alphas Only Address : A Main St (PO Box AA, RR 1 Box X, HC 1 Box AB) DPBC: 99 (99, 99, 99) Ignore alphas and se 99. Print code characters in DPBC representing 99 when alphas appear as the only primary street number.	

Task Completion Summary

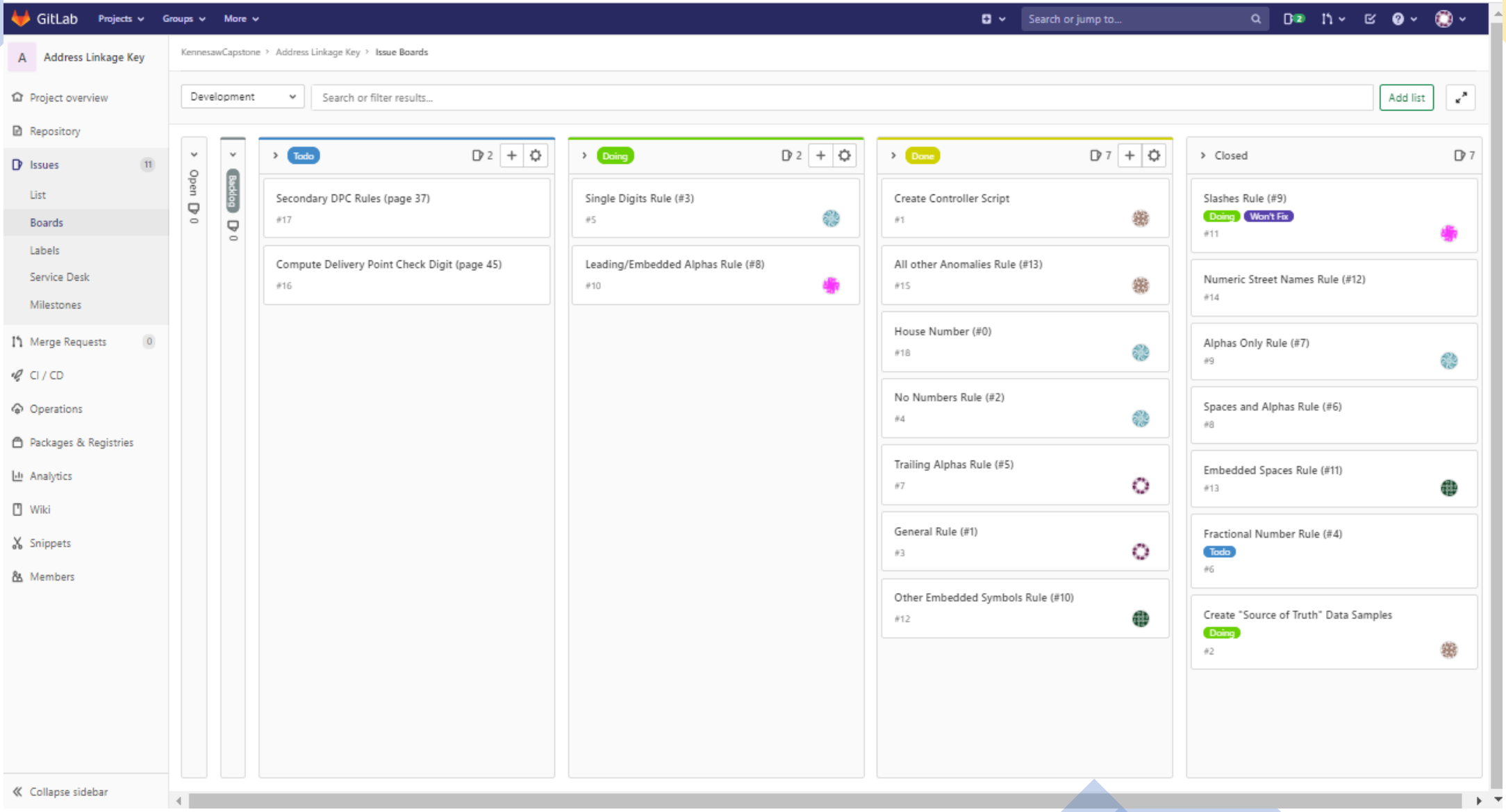
Completed

- **General Rule**
- **No Numbers Rule**
- **Single Digits Rule**
- **Trailing Alphas Rule**
- **Spaces and Alphas Rule**
- **Alphas Only Rule**
- **Leading/Embedded Alphas**
- **All other Anomalies**

In Progress

- **Other Embedded Symbols Rule- Testing Phase**
- **Slashes Rule (/)-Testing Phase**

Task Status Update from GitLab



Handler Function

```
15
16 def handler(df):
17     df = house_number_extract(df)
18     df = apply_rule_1(df)
19     df = apply_rule_5(df)
20     df = apply_rule_8(df)
21     df = apply_rule_10(df)
22     df = apply_rule_13(df)
23     #apply rule 3 (single digit rule) last
24     df = apply_rule_3(df)
25
26     return df
27
```

Rule Functions

```
32 def apply_rule_1(df):
33     # Use last two digits. Print code characters in DPBC representing last two digits of primary street number
34     df = df.withColumn('dpc',
35         f.when(
36             f.col('address_line_2').isNull() &
37             f.col('housenumber').isNotNull() &
38             f.col('housenumber').rlike('^[0-9]*$'),
39             f.col('housenumber').substr(-2,2)))
40     return df
41
42 def apply_rule_2(df):
43     # Use 99 when address contains no house number
44     df = df.withColumn('dpc',f.when((f.col('dpc').isNull()) & (f.col('housenumber').isNull()),f.lit('99'))
45                         .otherwise(f.col('dpc')))
46     return df
47
48 def apply_rule_3(df):
49     # Add a leading 0 when address contains a single digit house number
50
51     # left pads dpc column with 0 up to length 2
52     df = df.withColumn('dpc', f.lpad('dpc', 2, '0'))
53     return df
54
55 def apply_rule_5(df):
56     #Ignoring trailing alphas. Print code characters in DPBC representing last two digits to left of space and alphas
57     df = df.withColumn('dpc',
58                f.when(
59                     f.col('address_line_2').isNull() &
60                     f.col('housenumber').isNotNull() &
61                     f.col('dpc').isNull() &
62                     #f.col('housenumber').rlike('^[a-zA-Z0-9]*$'),
63                     f.col('housenumber').rlike('^[0-9]+[A-Z]+$'),
64                     f.regexp_extract(f.col('housenumber'),'(\d+)',1).substr(-2,2)).otherwise(f.col('dpc')))
```

Test Case Classes

```
126 class TestRule1(SparkTestCase):
127     def test(self):
128         # Test Case: Rule 1 (General Rule) should provide the last two digits of a primary street number
129         # run a query that gives me the input that the function expects and the expected value from that
130         # parse operation
131         testdf = self.spark.sql("SELECT '1234 MAIN ST' AS address_line_1, CAST(NULL AS string) AS address_line_2, '1234' AS
    housenumber, CAST(NULL AS string) AS dpc, '34' AS expected")
132         #run the function
133         testdf = apply_rule_1(testdf)
134         #gather the results [row 0 only] and compare the returned dpc to the expected dpc
135         row = testdf.collect()[0]
136         self.assertEquals(row.expected, row.dpc, "rule 1 created match")
137 
138 class TestRule2(SparkTestCase):
139     def test(self):
140         # Test Case: Rule 2 should use 99 when the address contains no house number.
141         # run a query that gives me the input that the function expects and the expected value from that
142         # parse operation
143         testdf = self.spark.sql("SELECT 'MAIN ST' AS address_line_1, CAST(NULL AS string) AS housenumber, CAST(NULL AS string)
    AS dpc, '99' AS expected")
144         #run the function
145         testdf = apply_rule_2(testdf)
146         #gather the results [row 0 only] and compare the returned dpc to the expected dpc
147         row = testdf.collect()[0]
148         self.assertEquals(row.expected, row.dpc, "rule 2 fail; expected does not match dpc")
149 
150 class TestRule3(SparkTestCase):
151     def test(self):
152         # Test Case: Rule 3 should add a leading 0 when dpc is a single digit
153         # run a query that gives me the input that the function expects and the expected value from that
154         # parse operation
155         testdf = self.spark.sql("SELECT '8 MAIN ST' AS address_line_1, '8' AS dpc, '08' AS expected")
156         #run the function
157         testdf = apply_rule_3(testdf)
158         #gather the results [row 0 only] and compare the returned dpc to the expected dpc
159         row = testdf.collect()[0]
160         self.assertEquals(row.expected, row.dpc, "rule 3 fail; expected does not match dpc")
```

Running Test Cases

To run test cases, run the module containing the test classes (address_link/__init__.py file) in a terminal. The script will connect to a local spark instance and run the test cases.

- Open a terminal in Jupyter Notebook
- cd to work/address-linkage-key directory
- Run all tests with either of these commands
  - *python3 address_link/__init__.py*
  - *make test (looks for all .py files and runs unit tests)*
- Run an individual test with this command
  - *python3 address_link/__init__.py TestRule1.test*

Running Test Cases

```
(base) jovyan@f49c99065829:~$ cd work/address-linkage-key
(base) jovyan@f49c99065829:~/work/address-linkage-key$ python3 address_link/__init__.py
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/spark-3.0.1-bin-hadoop3.2/jars/
-unsafe_2.12-3.0.1.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/03/24 01:17:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java c
s where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
......F
======================================================================
FAIL: test (__main__.TestRule8)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "address_link/__init__.py", line 184, in test
    self.assertEquals(row.expected, row.dpc, "Rule 8 fails; expected does not match dpc")
AssertionError: '41' != None : Rule 8 fails; expected does not match dpc

----------------------------------------------------------------------
Ran 7 tests in 12.576s

FAILED (failures=1)
(base) jovyan@f49c99065829:~/work/address-linkage-key$
```

Learnings

- As our whole project deals with generating DPBC (delivery point barcode) in multiple scenarios for different format of addresses. Regex functions (regular expressions) are best way to deal with different address formats of and generate a DPBC.
- As a team we all got chance to learn and implement regular expressions.
- Exposure to the wonderful world of DevOps.
  - Example of Continuous Integration (CI): configure GitLab to run unit tests upon commit.
  - Example of Continuous Delivery (CD): if all tests are okay, ship to production.

Regular Expressions

- A regular is a special text string for describing a search pattern. Regular expressions commonly referred to as regex, regexp, or re.
- The re module is used to write regular expressions (regex) in Python. To load this module, we need to use the import statement. The following line of code is necessary to include at the top of your code

  import re

- Example Regex: ('^[0-9]+[A-Z]+$').The sample regular expression extracts only numbers from the specified column.

  Example Address: 1234A MAIN ST

  Extracts New Address: 1234 (Ignores Alpha)

Common Syntax(Regex)

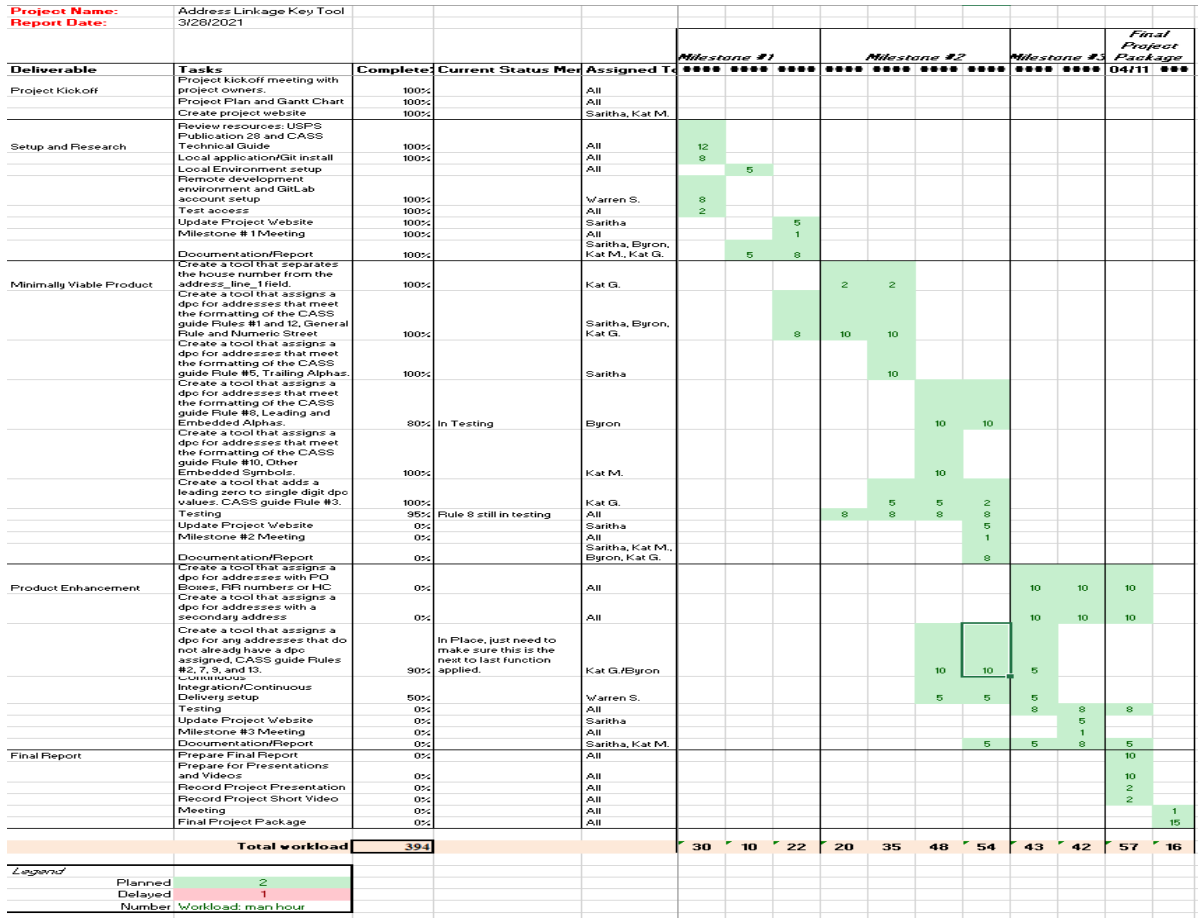
Table 1. **Common Regular Expression Syntax**

Syntax	Description
.	Matches any one character
^	Anchor; matches from the start of a string
$	Anchor; matches at the end of a string
\	Escape character
|	Pipe Character OR; C|T will match C or T
*	Matches zero or more repetitions of the previous character
+	Matches one or more repetitions of the previous character
?	Matches zero or one repetitions of the previous character
{n}	Quantifier; matches n repetitions of the previous character
{n, x}	Quantifier; matches from n to x repetitions of the previous character
[ ]	Character group; e.g. [AGCT] will match the characters AGCT
[^]	Negated character group e.g. [^AGCT] will match any characters not in this group
( )	Matches the pattern specified in the parentheses exactly

Challenges

- As part of initial project kick off, we have aimed to develop a tool for all 13 rules (more details about rules are included in slide 6 & 7).
- The rule numbers 4 (Fractional number) and 11(Embedded Spaces) are excluded from the scope of project development due to limitations in real time scenarios and current pyspark framework as advised by project owners.

Gantt Chart(Screenshot)



Project Name:	Address Linkage Key Tool				Milestone #1			Milestone #2				Milestone #3		Final Project Package	
Report Date:	3/28/2021														
Deliverable	Tasks	Complete:	Current Status Me	Assigned T										04/11	
Project Kickoff	Project kickoff meeting with project owners.	100%		All											
	Project Plan and Gantt Chart	100%		All											
	Create project website	100%		Saritha, Kat M.											
Setup and Research	Review resources: USPS Publication 28 and CASS Technical Guide	100%		All	12										
	Local application/Git install	100%		All	8										
	Local Environment setup			All		5									
	Remote development environment and GitLab account setup	100%		Warren S.	8										
	Test access	100%		All	2										
	Update Project Website	100%		Saritha			5								
	Milestone # 1 Meeting	100%		All			1								
	Documentation/Report	100%		Saritha, Byron, Kat M., Kat G.		5	8								
Minimally Viable Product	Create a tool that separates the house number from the address_line_1 field.	100%		Kat G.				2	2						
	Create a tool that assigns a dpc for addresses that meet the formatting of the CASS guide Rules #1 and 12, General Rule and Numeric Street	100%		Saritha, Byron, Kat G.			8	10	10						
	Create a tool that assigns a dpc for addresses that meet the formatting of the CASS guide Rule #5, Trailing Alphas.	100%		Saritha					10						
	Create a tool that assigns a dpc for addresses that meet the formatting of the CASS guide Rule #8, Leading and Embedded Alphas.	80%	In Testing	Byron						10	10				
	Create a tool that assigns a dpc for addresses that meet the formatting of the CASS guide Rule #10, Other Embedded Symbols.	100%		Kat M.						10					
	Create a tool that adds a leading zero to single digit dpc values. CASS guide Rule #3.	100%		Kat G.					5	5	2				
	Testing	95%	Rule 8 still in testing	All				8	8	8	8				
	Update Project Website	0%		Saritha							5				
	Milestone #2 Meeting	0%		All							1				
	Documentation/Report	0%		Saritha, Kat M., Byron, Kat G.							8				
Product Enhancement	Create a tool that assigns a dpc for addresses with PO Boxes, RR numbers or HC	0%		All								10	10	10	
	Create a tool that assigns a dpc for addresses with a secondary address	0%		All								10	10	10	
	Create a tool that assigns a dpc for any addresses that do not already have a dpc assigned, CASS guide Rules #2, 7, 9, and 13.	90%	In Place, just need to make sure this is the next to last function applied.	Kat G./Byron						10	10	5			
	Continuous Integration/Continuous Delivery setup	50%		Warren S.						5	5	5			
	Testing	0%		All								8	8	8	
	Update Project Website	0%		Saritha									5		
	Milestone #3 Meeting	0%		All									1		
	Documentation/Report	0%		Saritha, Kat M.							5	5	8	5	
Final Report	Prepare Final Report	0%		All										10	
	Prepare for Presentations and Videos	0%		All										10	
	Record Project Presentation	0%		All										2	
	Record Project Short Video	0%		All										2	
	Meeting	0%		All											1
	Final Project Package	0%		All											15
	Total workload	394			30	10	22	20	35	48	54	43	42	57	16

Legend	
Planned	2
Delayed	1
Number	Workload: man hour

Milestone 3 Plans

- Design an enhanced and optimized product.
- Planning to work on secondary address rule, where the tool or tools will determine the delivery point for secondary address lines (e.g. apartment or suite numbers) as well as PO Boxes and Rural Route addresses.
- The tool or tools will determine if specific types of delivery points, such as high-rises, are included in the dataset.

REFERENCES

1. USPS Publication 28 (Postal Addressing Standards)
2. USPS CASS Technical Guide
3. https://regexr.com/

THANK YOU