Forum:Hack Forums

Thread: Search, find and extract text from files based on custom search pattern

tcsh - *December 22, 2018, 2:53 am*

snippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppedd
snippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppedd
snippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppedd
snipppppppedd

FriendZone - *December 22, 2018, 6:30 am*

(Dec 22, 2018 03:53 AM)tcsh Wrote: *[CITE]* Hey there Python enthusiasts :)

I've been looking for solutions to a problem I've got and based on googling around I've come under the impression that Python can help me in what I need to achieve.

Unfortunately, since I'm not at all familiar with Python I'm hoping you guys can point me in the right direction in case there is an existing script out there that can do what I'm proposing.

Here we go:

I have a total number of 3248 .doc files.

The doc files contain a standard 'template' which got filled with some information.

The template looks roughly like this:

Its pretty basic, no fancy tables or stuff.
This template is then filled with data, manually by an employee and saved as name_surname.doc in the folder. (ya, an ancient way of patient data storing lol)

an Example on how it would look:

What I'm trying to do:

I need to port all this data into a mysql database. Obviously doing it manually on all the 3k files would take way too much so I'm trying to find a software/script/something that would open each file, extract the data written in the "template" by following a custom condition then save it as a single line in a new file. End goal is to have 3248 lines in a new file (representing the data from all the 3248 files) containing only the values of said template.

So an example line, containing the data from one of the files would be:

Obviously, the tricky bit is the 2nd part of the template, the one containing all the OD/fc/cc/cyl/ax/dp/OS and so on categories.

From what I've read you can write a script in Python and make it so that it would extract any text

present between something and something. In my case, for example, the row thart starts with 1.AV and value would be located between fc and cc (OD= fc; 0.8cc) - 0.8 being the value. And so on for the rest, since they're pretty standard. I could use the fc/cc/cyl/OD/OS/= characters as starting and ending points, getting any text in between these.

I can explain the template better and give an example on the logic i'm looking for, I'd just like an example or tutorial or something on how to do this in Python, if possible. Or any sort of help on the matter as well as any other recommendations or software that you know off with built in advanced search that would meet these needs.

Please feel free to ask If I haven't clarified well or left anything important out.

Is this doable in anyway?

Is the template always the same?

EDIT:

waiting for some more examples (your example is wrong because there are wrong numbers)

are you using linux or windows?

(make a list of every .doc file name and save it in a txt file)

from linux you can easily copy paste the output opening terminal with

Code ls > filenames.txt
replace every space/tab with \n with .replace(" ","\n") or .replace("\t","\n")

so just run

Code
```
f=open(filenames.txt)
for filename in f:
    file=open("/PATH_WHERE_YOU_HAVE_FILES/"+filename)
    text=file.read()
    valuesHextractor(text)
    file.close()
f.close()
```

here the code for the regex (IT COULD BE HALF STRINGS but I had some doubts about your request so I made it longer and explicit for any replacement)

Code
```
##### HEXECUTOR #####
#####  Hackforums.net  #####
#####  txt  #####
#####  data extractor  #####



import re


def DOCtoSTRING(path):
```

```
    return text

def valuesHextractor(text):
    STRING =""
    #let's find pacient personal info
    regex = re.compile('PACIENT DATA(?s).*DIAGNOSTIC:[^\n]*')
    pacientdata=regex.match(text)

    regex = re.compile(':[^\n]*')
    pacientinfo = regex.findall(pacientdata[0])
    for i in pacientinfo:
        info=i.strip(": ")
        STRING=STRING+info+" "

    #values
    regex = re.compile('EXAM VALUES(?s).*TREATMENT')
    pacientvalues=regex.search(text)

    #remove 1.AV 2.RO 3.PIO 4.FO numbers
    replacedvalues=re.sub("[1-4]\.((AV)|(RO)|(PIO)|(FO))","",pacientvalues[0])

    subregex = re.compile('[\-\+0-9\.]+')
    values = subregex.findall(replacedvalues)
    for v in values:
        STRING=STRING+v+" "

    #remove space between a + - and a number
    STRING=re.sub("\-( )+","-",STRING)
    STRING=re.sub("\+( )+ ","+",STRING)

    #treatment
    regex = re.compile('TREATMENT:[^\n]*')
    treatment = regex.search(text)
    treatment = treatment[0].replace('TREATMENT:','')
    STRING=STRING+treatment
    print (STRING)


f=open(input("file PATH: ") ,'r')
text=f.read()
valuesHextractor(text)
f.close
```

test it before with a single .txt containing your data to check regex validity. it should run and work

tcsh - *December 27, 2018, 6:56 am*

(Dec 22, 2018 07:30 AM)Hexecutor Wrote: [CITE] Is the template always the same?

EDIT:

waiting for some more examples (your example is wrong because there are wrong numbers)

are you using linux or windows?

Hey there, thanks for the reply.
I'm using Windows.

Which numbers exactly are you saying are wrong? The ones I highlighted in color in the example are adequate.

Gonna try out your solution and come back with an edit to this reply. ?

FriendZone - *December 27, 2018, 7:03 am*

(Dec 27, 2018 07:56 AM)tcsh Wrote: [CITE] Hey there, thanks for the reply.
I'm using Windows.

Which numbers exactly are you saying are wrong? The ones I highlighted in color in the example are adequate.

Gonna try out your solution and come back with an edit to this reply. ?
Quote:james smith 33 5th avenue 1234567890 05.05.2015 cancer 1 0.8 -2 -1 180 1 0.1 -3.5 +2 33 51 -1 +2 33 -1 +3 22 11 11 3 4 chemo

I guess it’s from 1.AV but then you didn’t pick 2. Or 3. 4.

tcsh - *December 27, 2018, 7:17 am*

snippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppedd
snippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppedd
snippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppeddsnippppppedd
snippppppedd

FriendZone - *December 27, 2018, 8:06 am*

(Dec 27, 2018 08:17 AM)tcsh Wrote: [CITE] Oh, ya, you're right.
Sorry, typo on my part.

Writing an example for you now

LATER EDIT

1st part:

At DATE, there's a small issue.
In most of the files, there is only one date but several other files have multiple dates.

What I will usually find for the DATE area looks like this: 12.05.15

Some files, however, have more at DATE, like this:

Basically, some files have 4-5 or more dates and they're not necessarily on the same row as DATE:

The 1st date is on the same row, others are below. For the DATE: part I'm thinking it should copy any text that starts after the ":" from DATE and ends where DIAGNOSTIC starts.

The 2nd part of the template:

1. 2. 3. 4. are sort of like categories.

AV is the main category that is divided into 2 subcategories, each subcategory with 4 others.

So when it comes to AV I need the value of the first FC, which corresponds to AV(main category) and OD(1st subcategory of AV).
Then for cc, cyl and ax - each corresponding to AV (main category) and OD (1st subcategory of AV)

After that, I need the value of the 2nd FC, which corresponds to AV(main category) and OS(the 2nd subcategory of AV).
Then for cc, cyl and ax - each corresponding to AV (main category) and OS (the 2nd subcategory of AV)

OD values will be 1st always on the AV row, while OS values will follow. Respectively, 1st fc, cc, cyl, and ax followed by 2nd fc, cc, cyl, and ax.

The same logic applies for 2.RO - RO is the main category, OD and OS are the 2 subcategories.

3.PIO has just the 2 subcategories (OD and OS) with their respective values.
4. FO the same.

As for treatment, need to save anything that follows after the ":"

So, the output should be:

name, surname, age, address, phone, date (date, date, date, if more than 1), diagnostic - this is the 1st part of the template.

fc, cc, cyl, ax (from the row starting with 1.AV - right after OD= ending after the 1st ax and its value, at ";")

followed by

fc, cc, cyl, ax (from the row starting with 1.AV - right after OS= ending after the 2nd ax and its value, at ";")

followed by

cyl and ax (from the row starting with 2.RO - right after OD= ending after the 1st ax and its value, at ";")

followed by

cyl and ax (from the row starting with 2.RO - right after OS= ending after the 2nd ax and its value, at ";")

followed by

value immediately after OD= and ending at the 1st ";" (3.PIO row)

followed by

value immediately after OS= and ending at the 2nd ";" (3.PIO row)

followed by

same for 4.FO - like above at 3.PIO

followed by anything after treatment:

Final output string should then be:

name, surname, age, address, phone, date (date, date, date, if more than 1), diagnostic, fc, cc, cyl, ax, fc, cc, cyl, ax, cyl, ax, cyl, ax, PIO-OD, PIO-OS, FO-OD, FO-OS, treatment

One small addition. Notice that 1.AV row, at the end, it has a DP=
If its possible I need that value right after DP= inserted at the end of my string.

So in the end result, right after treament, the DP value. if this is harder then we can drop the DP value and forget about it.

Makes more sense now? @Hexecutor
Ok, I'm reading but I will be probably faster if you can copy paste 2-3 different examples (even in PM)

I will edit here

tcsh - *December 27, 2018, 8:13 am*
(Dec 27, 2018 09:06 AM)Hexecutor Wrote: [CITE] Ok, I'm reading but I will be probably faster if you can copy paste 2-3 different examples (even in PM)

I will edit here
Sent some files via PM. check it out ?

FriendZone - *December 27, 2018, 8:55 am*
(Dec 27, 2018 09:13 AM)tcsh Wrote: [CITE] Sent some files via PM. check it out ?
got it, I will give a look now, not sure how much free time I have this morning. I will try to give a solution late today or tomorrow.

tcsh - *December 27, 2018, 9:04 am*
(Dec 27, 2018 09:55 AM)Hexecutor Wrote: [CITE] got it, I will give a look now, not sure how much free time I have this morning. I will try to give a solution late today or tomorrow.
Whenever you can, it's fine.
Should be straightforward judging by your already made example code above.
I'll try to use/add to that meanwhile

?

FriendZone - *December 27, 2018, 8:55 pm*
(Dec 27, 2018 08:17 AM)tcsh Wrote: [CITE] Oh, ya, you're right.
Sorry, typo on my part.

Writing an example for you now

LATER EDIT

1st part:

At DATE, there's a small issue.
In most of the files, there is only one date but several other files have multiple dates.

What I will usually find for the DATE area looks like this: 12.05.15

Some files, however, have more at DATE, like this:

Basically, some files have 4-5 or more dates and they're not necessarily on the same row as DATE:

The 1st date is on the same row, others are below. For the DATE: part I'm thinking it should copy any text that starts after the ":" from DATE and ends where DIAGNOSTIC starts.

The 2nd part of the template:

1. 2. 3. 4. are sort of like categories.

AV is the main category that is divided into 2 subcategories, each subcategory with 4 others.

So when it comes to AV I need the value of the first FC, which corresponds to AV(main category) and OD(1st subcategory of AV).
Then for cc, cyl and ax - each corresponding to AV (main category) and OD (1st subcategory of AV)

After that, I need the value of the 2nd FC, which corresponds to AV(main category) and OS(the 2nd subcategory of AV).
Then for cc, cyl and ax - each corresponding to AV (main category) and OS (the 2nd subcategory of AV)

OD values will be 1st always on the AV row, while OS values will follow. Respectively, 1st fc, cc, cyl, and ax followed by 2nd fc, cc, cyl, and ax.

The same logic applies for 2.RO - RO is the main category, OD and OS are the 2 subcategories.

3.PIO has just the 2 subcategories (OD and OS) with their respective values.
4. FO the same.

As for treatment, need to save anything that follows after the ":"

So, the output should be:

name, surname, age, address, phone, date (date, date, date, if more than 1), diagnostic - this is the 1st part of the template.

fc, cc, cyl, ax (from the row starting with 1.AV - right after OD= ending after the 1st ax and its value, at ";")

followed by

fc, cc, cyl, ax (from the row starting with 1.AV - right after OS= ending after the 2nd ax and its value, at ";")

followed by

cyl and ax (from the row starting with 2.RO - right after OD= ending after the 1st ax and its value, at ";")

followed by

cyl and ax (from the row starting with 2.RO - right after OS= ending after the 2nd ax and its value, at ";")

followed by

value immediately after OD= and ending at the 1st ";" (3.PIO row)

followed by

value immediately after OS= and ending at the 2nd ";" (3.PIO row)

followed by

same for 4.FO - like above at 3.PIO

followed by anything after treatment:

Final output string should then be:

name, surname, age, address, phone, date (date, date, date, if more than 1), diagnostic, fc, cc, cyl, ax, fc, cc, cyl, ax, cyl, ax, cyl, ax, PIO-OD, PIO-OS, FO-OD, FO-OS, treatment

One small addition. Notice that 1.AV row, at the end, it has a DP=
If its possible I need that value right after DP= inserted at the end of my string.

So in the end result, right after treament, the DP value. if this is harder then we can drop the DP value and forget about it.

Makes more sense now? @Hexecutor

So, to be honest I was afraid about some changes in the template order so I make one of the most universal regex, One for each value (a regex is definitely an heavy operation so I really dislike it) but I guess was the best option to save time.

I used only 2 examples to test anyway.

And translated the whole code in the file language received in PM so you won't need to rewrite it

Code ##### HEXECUTOR #####
Hackforums.net
txt
data extractor
#################################

```
import re


def DOCtoSTRING(path):

    return text

def valuesHextractor(text):
    STRING =""
    #let's find pacient personal info
    regex = re.compile('CONSULTATIE(?s).*EXAMENUL')
    pacientdata=regex.search(text)
    pacientinfo = re.sub('[A-Z]+:',"",pacientdata[0])
    STRING=STRING+pacientinfo.strip("CONSULTATIE").strip("EXAMENUL")+" "

    #values
    regex = re.compile('EXAMENUL OFTALMOLOGIC:(?s).*TRATAMENT')
    pacientvalues=regex.search(text)

    #AV OD
    #fc
    regex=re.compile("1.AV.*OD.*?fc[;\.\+\- 0-9]*")
    AVfc=regex.search(pacientvalues[0])
    AVODfc=re.sub("1.AV.*OD.*?fc[;\. 0-9]*?","",AVfc[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVODfc)==None:
        AVODfc="None"
    STRING=STRING+AVODfc+" "
    #cc
    regex=re.compile("1.AV.*OD.*?cc[;\.\+\- 0-9]*")
    AVcc=regex.search(pacientvalues[0])
    AVODcc=re.sub("1.AV.*OD.*?cc[;\. 0-9]*?","",AVcc[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVODcc)==None:
        AVODcc="None"
    STRING=STRING+AVODcc+" "
    #cyl
    regex=re.compile("1.AV.*OD.*?cyl[;\.\+\- 0-9]*")
    AVcyl=regex.search(pacientvalues[0])
    AVODcyl=re.sub("1.AV.*OD.*?cyl[;\. 0-9]*?","",AVcyl[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVODcyl)==None:
        AVODcyl="None"
    STRING=STRING+AVODcyl+" "
    #ax
    regex=re.compile("1.AV.*OD.*?ax[;\.\+\- 0-9]*")
    AVax=regex.search(pacientvalues[0])
    AVODax=re.sub("1.AV.*OD.*?ax[;\. 0-9]*?","",AVax[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVODax)==None:
        AVODax="None"
    STRING=STRING+AVODax+" "
```

```
    #AV OS
    #fc
    regex=re.compile("1.AV.*OS.*?fc[;\. 0-9]*")
    AVfc=regex.search(pacientvalues[0])
    AVOSfc=re.sub("1.AV.*OS.*?fc[;\. 0-9]*?","",AVfc[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVOSfc)==None:
        AVOSfc="None"
    STRING=STRING+AVOSfc+" "
    #cc
    regex=re.compile("1.AV.*OS.*?cc[;\.\+\- 0-9]*")
    AVcc=regex.search(pacientvalues[0])
    AVOScc=re.sub("1.AV.*OS.*?cc[;\. 0-9]*?","",AVcc[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVOScc)==None:
        AVOScc="None"
    STRING=STRING+AVOScc+" "
    #cyl
    regex=re.compile("1.AV.*OS.*?cyl[;\.\+\- 0-9]*")
    AVcyl=regex.search(pacientvalues[0])
    AVOScyl=re.sub("1.AV.*OS.*?cyl[;\. 0-9]*?","",AVcyl[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVOScyl)==None:
        AVOScyl="None"
    STRING=STRING+AVOScyl+" "
    #ax
    regex=re.compile("1.AV.*OS.*?ax[;\.\+\- 0-9]*")
    AVax=regex.search(pacientvalues[0])
    AVOSax=re.sub("1.AV.*OS.*?ax[;\. 0-9]*?","",AVax[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVOSax)==None:
        AVOSax="None"
    STRING=STRING+AVOSax+" "




############From here ignore names check only the #name

####### DP
    regex=re.compile("1.AV.*OS.*?DP=[;\.\+\- 0-9]*")
    AVax=regex.search(pacientvalues[0])
    AVOSax=re.sub("1.AV.*OS.*?DP=[;\. 0-9]*?","",AVax[0])
    regex=re.compile("[\+\-0-9]+")
    if regex.search(AVOSax)==None:
        AVOSax="None"
    DP=AVOSax


##########
```

```
#RO OD
#cyl
regex=re.compile("2.RO.*OD.*?cyl[;\.\+\- 0-9]*")
AVcyl=regex.search(pacientvalues[0])
AVODcyl=re.sub("2.RO.*OD.*?cyl[;\. 0-9]*?","",AVcyl[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVODcyl)==None:
    AVODcyl="None"
STRING=STRING+AVODcyl+" "
#ax
regex=re.compile("2.RO.*OD.*?ax[;\.\+\- 0-9]*")
AVax=regex.search(pacientvalues[0])
AVODax=re.sub("2.RO.*OD.*?ax[;\. 0-9]*?","",AVax[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVODax)==None:
    AVODax="None"
STRING=STRING+AVODax+" "


#RO OS
#cyl
regex=re.compile("2.RO.*OS.*?cyl[;\.\+\- 0-9]*")
AVcyl=regex.search(pacientvalues[0])
AVOScyl=re.sub("2.RO.*OS.*?cyl[;\. 0-9]*?","",AVcyl[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVOScyl)==None:
    AVOScyl="None"
STRING=STRING+AVOScyl+" "
#ax
regex=re.compile("2.RO.*OS.*?ax[;\.\+\- 0-9]*")
AVax=regex.search(pacientvalues[0])
AVOSax=re.sub("2.RO.*OS.*?ax[;\. 0-9]*?","",AVax[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVOSax)==None:
    AVOSax="None"
STRING=STRING+AVOSax+" "


#3.PIO OD
regex=re.compile("3.PIO.*OD=[;\.\+\- 0-9]*")
AVfc=regex.search(pacientvalues[0])
AVODfc=re.sub("3.PIO.*?OD=[;\. 0-9]*?","",AVfc[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVODfc)==None:
```

```
    AVODfc="None"
STRING=STRING+AVODfc+" "

#3.PIO OS
regex=re.compile("3.PIO.*?OS=[;\. 0-9]*")
AVfc=regex.search(pacientvalues[0])
AVOSfc=re.sub("3.PIO.*?OS=[;\. 0-9]*?","",AVfc[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVOSfc)==None:
    AVOSfc="None"
STRING=STRING+AVOSfc+" "


#4.FO OD
regex=re.compile("4.FO(?s).*OD:[;\.\+\- 0-9]*")
AVfc=regex.search(pacientvalues[0])
AVODfc=re.sub("4.FO(?s).*?OD:[;\. 0-9]*?","",AVfc[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVODfc)==None:
    AVODfc="None"
STRING=STRING+AVODfc+" "

#4.FO OS
regex=re.compile("4.FO(?s).*?OS:[;\. 0-9]*")
AVfc=regex.search(pacientvalues[0])
AVOSfc=re.sub("4.FO(?s).*?OS:[;\. 0-9]*?","",AVfc[0])
regex=re.compile("[\+\-0-9]+")
if regex.search(AVOSfc)==None:
    AVOSfc="None"
STRING=STRING+AVOSfc+" "


##################################################### NORMAL FROM HERE



#remove space between a + - and a number
STRING=re.sub("\-( )+","-",STRING)
STRING=re.sub("\+( )+ ","+",STRING)
STRING=re.sub("(\-\+)","-+ ",STRING)
STRING=re.sub("(\+\-)","+- ",STRING)

#treatment
regex = re.compile('TRATAMENT:(?s).*')
treatment = regex.search(text)
treatment = treatment[0].replace('TRATAMENT:','')
STRING=STRING+treatment
STRING=STRING+DP
regex=re.compile("( )+")
STRING=STRING.replace('\n',' ').replace('\t', ' ').replace(';',")
STRING=re.sub(regex," ",STRING)
print (STRING)
```

MAIN (once you will be ready use the other main I've already shared in the first post this main is just a regex tester)

```
f=open(input("file PATH (must be a .txt for now): ") ,'r')
text=f.read()
valuesHextractor(text)
f.close
```

this function should follow your instructions. I also added "None" in the string when there are Empty Values, I consider "-+" a filled value manwhile "= ;" is an empty value, to avoid mistakes reading the string.

Let me know.

Hexecutor ?