CyberCamp at UNK

Grep and Python - Decrypting hidden messages

What you will learn with this tutorial...

This tutorial will give you the basic idea of how to execute grep commands from within a Python script. We will go together through an example to obtain some information from a text file using grep. After that it will be your turn to develop a Python script calling grep as a subprocess. In the exercise you will have to decrypt a hidden message that is present inside the search file that will be provided to you. Let's start!

1 Obtaining the search files

First thing to do is to obtain the search files you will use both in your example script and in your exercise. They are available in an on line repository. To download open a terminal and run the following commands:

```
wget https://raw.githubusercontent.com/igorceridorio/CyberCampUNK
/master/hiddenMessageSample.txt
```

```
wget https://raw.githubusercontent.com/igorceridorio/CyberCampUNK
/master/hiddenMessageExercise.txt
```

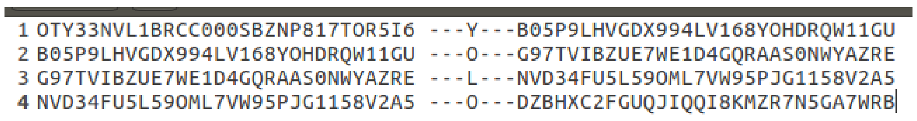
This command will download the files to your current directory. Keep in mind the path where you are, these files should be located at the same directory of your scripts, doing so will make things easier.

2 Sample script

Now that you have the files, let's start by a sample script.

In this script we will use the Python ***subprocess*** module to run the `grep` command from inside our script.

First of all, let's understand how our text files are composed. Down below you have an image of the *hiddenMessageSample.txt* file:



```
OTY33NVL1BRCC000SBZNP817TOR5I6 ---Y---B05P9LHVGDX994LV168YOHDRQW11GU
B05P9LHVGDX994LV168YOHDRQW11GU ---O---G97TVIBZUE7WE1D4GQRAAS0NWYAZRE
G97TVIBZUE7WE1D4GQRAAS0NWYAZRE ---L---NVD34FU5L59OML7VW95PJG1158V2A5
NVD34FU5L59OML7VW95PJG1158V2A5 ---O---DZBHXC2FGUQJIQQI8KMZR7N5GA7WRB
```

Figure 1: Sample search text file

As you can see each line is composed by a **key** of 30 characters (from now on referred as ***currentKey***), a space follow by a delimiter (**- - -**), the character, another delimiter, and finally another key (the ***nextKey***). The ***currentKey*** identifies a character and the ***nextKey*** is used to search for the following one, inside our hidden message context.

The other search file has exactly the same pattern as this one.

In this example, we will build a script that:

- Asks the user for one specific character to execute the search for;
- Execute the `grep` command to search for the desired character in the defined file inside the code;
- If the search succeed prints in the screen the result of it, otherwise prints a message that no matches were found.

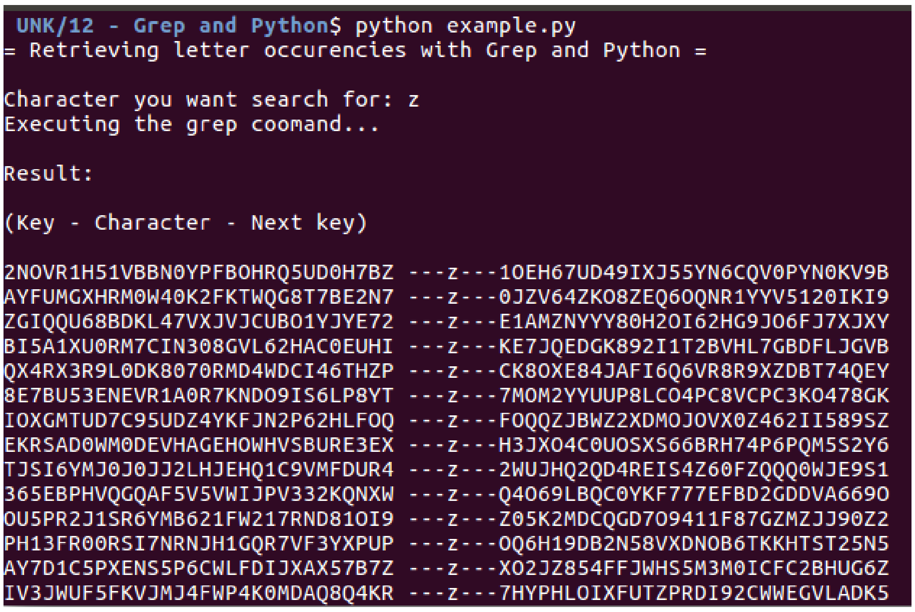
Open your text editor and type the code below. After it, we will briefly discuss in more details what it does.

```python
import subprocess

def main():

  print("= Retrieving letter occurencies with Grep and Python =\n")

  caracter = raw_input("Character you want search for: ")
  fileName = "hiddenMessageExercise.txt"

  print "Executing the grep coomand...\n"

  command = "grep -e \"---" + caracter + "---\" " + fileName

  proc = subprocess.Popen([command], stdout=subprocess.PIPE, shell=True)
  (out, err) = proc.communicate()

  if not out:
    print "No matches found"
  else:
    print "Result:\n"
    print "(Key - Character - Next key)\n"
    print out

main()
```

The most important things we can observe here are:

- **Line 1:** We have to import the module `subprocess`. This module is necessary whenever we want to execute some Linux terminal command from inside our Python code.
- **Lines 7 and 8:** Line 7 receives the character the user want to search inside the file, and line 8 defines in which file we will execute the search. In this case it will be on *hiddenMessageExercise.txt*.
- **Line 12:** Here we build the string that contains our full `grep` command. We concatenate the information provided by the user to the structure of the `grep`.
- **Line 14 and 15:** These lines basically get the command we created, instantiate a new subprocess in the system, and send the command as parameter for execution. The output will be stored at a variable called `out` and eventual errors will be stored at `err`.
- **Line 17 through 22:** Here we have a conditional. If the variable `out` is empty, it means that the search found no results, hence a simple message is printed to the user stating this fact. Otherwise, we print the return of the search, that is stored inside the variable `out`.

Once you're done typing and understanding the code, save and execute it. Down below you can check part of an execution example using the *hiddenMessageExercise.txt* file as search base.



```
UNK/12 - Grep and Python$ python example.py
= Retrieving letter occurencies with Grep and Python =

Character you want search for: z
Executing the grep coomand...

Result:

(Key - Character - Next key)

2NOVR1H51VBBN0YPFBOHRQ5UD0H7BZ ---z---1OEH67UD49IXJ55YN6CQV0PYN0KV9B
AYFUMGXHRM0W40K2FKTWQG8T7BE2N7 ---z---0JZV64ZKO8ZEQ6OQNR1YYV5120IKI9
ZGIQQU68BDKL47VXJVJCUBO1YJYE72 ---z---E1AMZNYYY80H2OI62HG9JO6FJ7XJXY
BI5A1XU0RM7CIN308GVL62HAC0EUHI ---z---KE7JQEDGK892I1T2BVHL7GBDFLJGVB
QX4RX3R9L0DK8070RMD4WDCI46THZP ---z---CK8OXE84JAFI6Q6VR8R9XZDBT74QEY
8E7BU53ENEVR1A0R7KNDO9IS6LP8YT ---z---7MOM2YYUUP8LCO4PC8VCPC3KO478GK
IOXGMTUD7C95UDZ4YKFJN2P62HLFOQ ---z---FOQQZJBWZ2XDMOJOVX0Z462II589SZ
EKRSAD0WM0DEVHAGEHOWHVSBURE3EX ---z---H3JXO4C0UOSXS66BRH74P6PQM5S2Y6
TJSI6YMJ0J0JJ2LHJEHQ1C9VMFDUR4 ---z---2WUJHQ2QD4REIS4Z60FZQQQ0WJE9S1
365EBPHVQGQAF5V5VWIJPV332KQNXW ---z---Q4O69LBQC0YKF777EFBD2GDDVA669O
OU5PR2J1SR6YMB621FW217RND81OI9 ---z---Z05K2MDCQGD7O9411F87GZMZJJ90Z2
PH13FR00RSI7NRNJH1GQR7VF3YXPUP ---z---OQ6H19DB2N58VXDNOB6TKKHTST25N5
AY7D1C5PXENS5P6CWLFDIJXAX57B7Z ---z---XO2JZ854FFJWHS5M3M0ICFC2BHUG6Z
IV3JWUF5FKVJMJ4FWP4K0MDAQ8Q4KR ---z---7HYPHLOIXFUTZPRDI92CWWEGVLADK5
```

Figure 2: Sample script execution

Now try for yourself!

Now it's our turn to create a script and decrypt a hidden message. The idea of this script is pretty simple. You will be given a key. This key represents the first character of the message. With this information and the name of the file you have to reveal the secret message.
Here are some hints that you may find helpful:

- You will be given the initial keys for the *hiddenMessageSample.txt* nad *hiddenMessageExercise.txt* files. This means that you already know the first `grep` command that you should execute. In the case of the *hiddenMessageSample.txt* file, the line of code to build the command inside a Python and store it on a variable would be:

  ```
  command = "grep -e \"OTY33NVL1BRCC000SBZNP817TOR5I6 \" " + fileName
  ```

  Note the backslashes here (\). They are used because we want grep to search for our key followed by a space, because that's how it is stored in our file.
- After executing the first `grep` you will obtain the character for this key and the key of the following character. Now you can create a new `grep` call with the new key below the previous one.
- Keep repeating this process until obtain the final hidden message!

If you are feeling adventurous try thinking how you could execute the `grep` commands inside a loop instead of copying and pasting the same code for each one of the characters :-)

Here are the initial key you need for each of the files:

- **hiddenMessageSample.txt:** OTY33NVL1BRCC000SBZNP817TOR5I6
- **hiddenMessageExercise.txt:** 71P0DSN0V5L00GG3LVKB405HPP0G9C

Good luck!