DESKTOP NOTIFIER APP

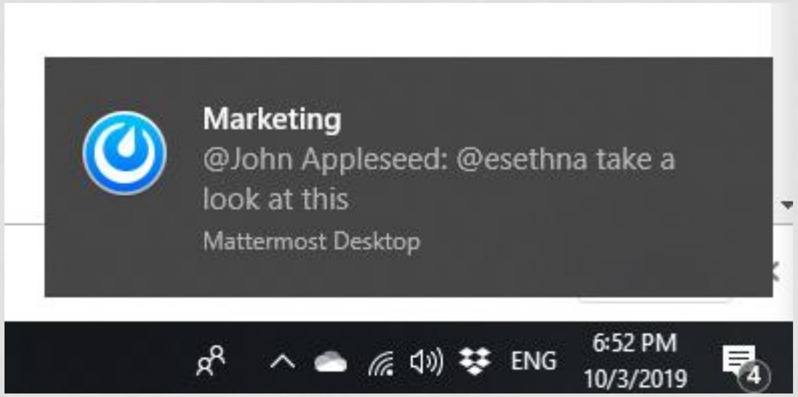
WHAT WILL BE COVERED IN THIS

1. Desktop Notification Introduction
2. What is Plyer?
3. What is Request?
4. What is BeautifulSoup?
5. Create a Desktop Notifier

DESKTOP NOTIFICATION

A desktop notifier is a simple application which produces a notification message in form of a pop-up message on desktop

The action of notifying someone or something.


Marketing
@John Appleseed: @esethna take a look at this
Mattermost Desktop

PURPOSE

The purpose behind a notifications help people to remember things. It is a small piece of text which appears on the desktop or mobile screen to inform the user about the updates or any other important pieces of information.

PLYER

Plyer is an open source library to access features commonly found in various platforms via **python**. Plyer module is used to access the features of the hardware.

To install this module type the below command in the terminal.

```
pip install plyer
```

REQUESTS

The **requests** module allows you to send HTTP **requests** using **Python**. The HTTP **request** returns a Response Object with all the response data.

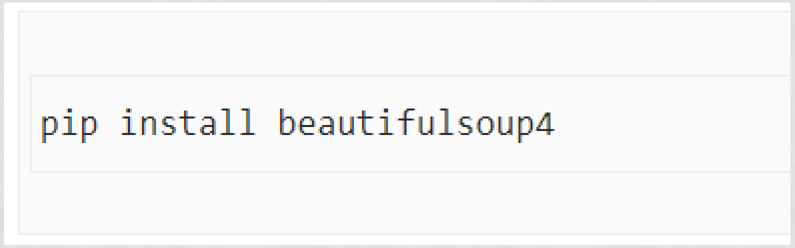
Requests officially supports Python 2.7 & 3.5+,

BEAUTIFULSOUP

Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages.

It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web.

The following command in the terminal to install Beautiful Soup:



```
pip install beautifulsoup4
```

SCOPE

- Set daily tracker for COVID stats
- Daily notification to take medicine.
- Hourly notification to drink water.
- Top news
- Important updates

WAYS TO DESIGN THE APPLICATION

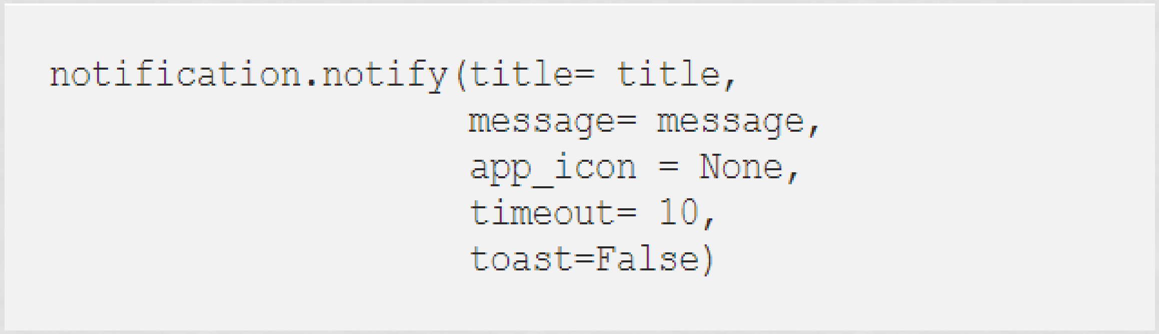
1. Customize desktop notifier.

i.e. For personal reminder.

2. Notifier that notify the user about the updates and important informations.

CUSTOMIZED DESKTOP NOTIFIER

In this method I simply use notification from plyer and pass the parameters using notify method.



```
notification.notify(title= title,
                    message= message,
                    app_icon = None,
                    timeout= 10,
                    toast=False)
```

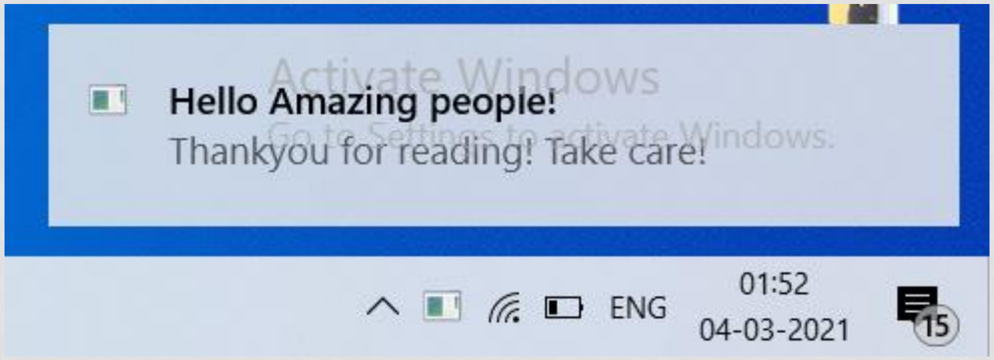
PARAMETERS OF NOTIFY METHOD

- title:
- message:
- app_icon:
- timeout:
- toast:

OUTPUT

I am passing:

Title as Hello Amazing people!
Message as 'Thankyou for reading! Take care!'
App_icon as None
Timeout as 10 secs
Toast as False.



Hello Amazing people!
Thankyou for reading! Take care!

COVID DESKTOP NOTIFIER

It is a simple application in form of a pop-up message to check active Covid cases with the help of requests and beautifulsoup module.

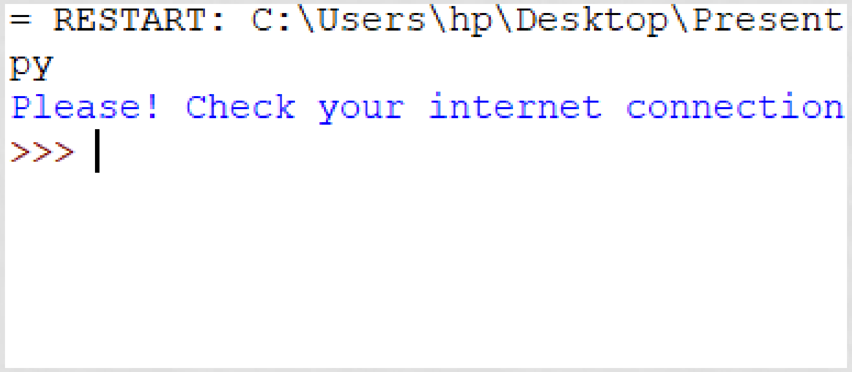
USING REQUESTS AND BEAUTIFULSOUP

- In this method I use notification from plyer and pass the parameters using notify method.
- Requests allows you to send HTTP/requests extremely easily.
- Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information.

- For this I request for govt. site to check the active Covid cases.ie. https://www.mohfw.gov.in/
- With the help of BeautifulSoup I parse the html pages from the requested url and find the required result.

OUTPUT

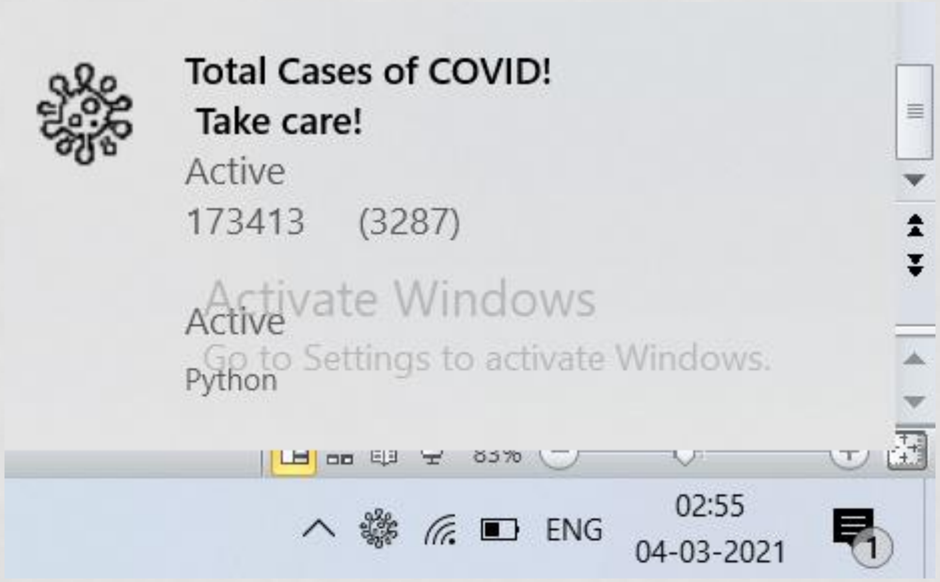
- To get output from this method first make sure the internet connection is enable otherwise it will not generate any output because the data is fetched from the requested URL.



```
= RESTART: C:\Users\hp\Desktop\Present
py
Please! Check your internet connection
>>> |
```

OUTPUT

- Output when the requests method is sucessfully get the data from the requested URL.



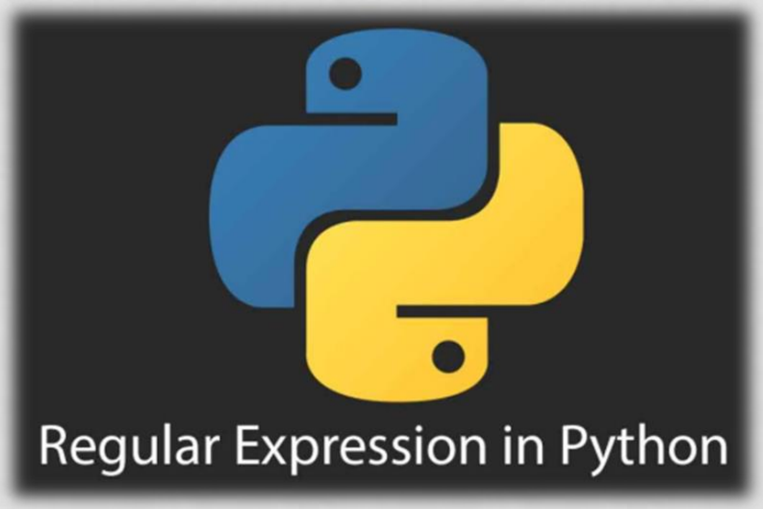
```
Total Cases of COVID!
 Take care!
Active
173413     (3287)

Active
Python
```

REGEX QUERY TOOL

REGEX MODULE

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern. python has a built-in package called re.



WORKING WITH REGEX

To work with regular expression first,we have to import re module in our file.

Ie. **import re**

REGEX FUNCTIONS

- The re module offers a set of functions that allows us to search a string for a match:

Function	Description
Findall()	Returns a list containing all matches
Search()	Returns a Match object if there is a match anywhere in the string
Split()	Returns a list where the string has been split at each match.
Sub()	Replaces one or many matches with a string

METACHARACTERS

Chracter	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence	"\+"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
$	Ends with	"world$"
?	Occure once or not	s? = _ or s
*	Zero or more occurrences	s* =_,s,ss,sss,…
+	One or more occurrences	s+ = s,ss,sss,s
{}	Exactly the specified number of occurrences	"a{2}"

SPECIAL SEQUENCES

Character	Description	Example
\A	if the specified characters are at the beginning of the string	
\d	string contains digits (numbers from 0-9)	\d = 7, \d\d=77
\D	string DOES NOT contain digits	\D=
\s	string contains a white space character	\s = “hello”
\S	string DOES NOT contain a white space character	\S = “Beautiful Day”
\w	the string contains any word characters(a-z A-z _)	\w= seek
\W	DOES NOT contain any word characters	\W = %
\Z	specified characters are at the end of the string	\Z= "The rain in Spain"

SETS

Set	Description
[arn]	Returns a match where one of the specified characters (a, r, or n) are present
[a-n]	Returns a match for any lower case character, alphabetically between a and n
[^arn]	Returns a match for any character EXCEPT a, r, and n
[0123]	Returns a match where any of the specified digits (0, 1, 2, or 3) are present
[0-9]	Returns a match for any digit between 0 and 9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-z][A-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case.
[+]	In sets, +, *, ., |, (), $,{} has no special meaning, so [+] means: return a match for any + character in the string

HOW TO WRITE PATTERN

Example1 :

Write pattern for Mobile no?

Condition: Mobile no. starts with 8 or 9 and total number of digits are10.

Pattern: **[8-9][0-9]{9}**

Example2 :

Write pattern for **Sim1ran**

Condition: First character Upper case,contains lower case alphabet only 1 digit is allowed.

Pattern: **[A-Z][a-z]*[0-9][a-z]***

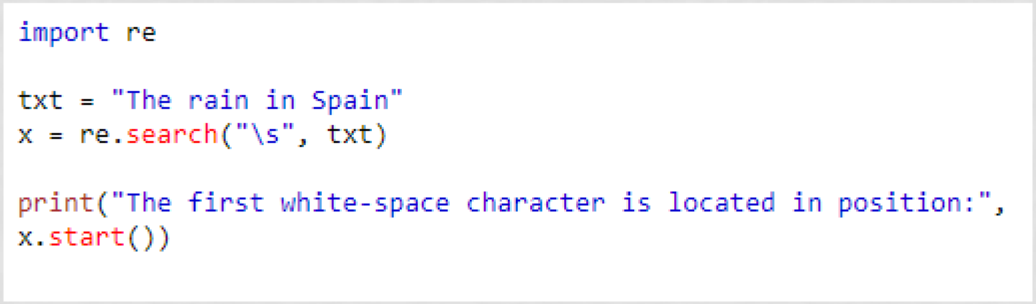
RE.FINDALL()

findall() module is used to search for "all" occurrences that match a given pattern. In contrast, search() module will only return the first occurrence that matches the specified pattern.

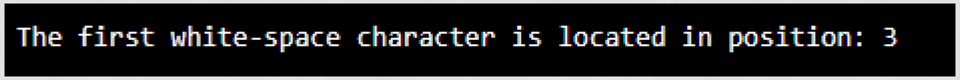
findall() will iterate over all the lines of the file and will return all non-overlapping matches of pattern in a single step.

RE.SEARCH()

re.search() function will search the regular expression pattern and **return the first occurrence**. Python re.search() function returns a match object when the pattern is found and "null" if the pattern is not found.



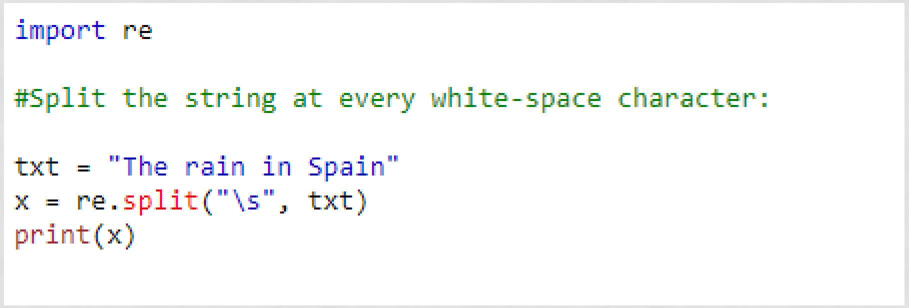
```
import re

txt = "The rain in Spain"
x = re.search("\s", txt)

print("The first white-space character is located in position:",
x.start())
```



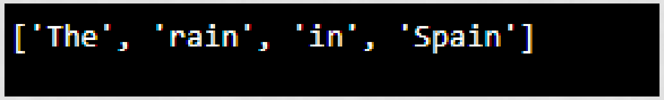
```
The first white-space character is located in position: 3
```

RE.SPLIT()

The split() function returns a list where the string has been split at each match:



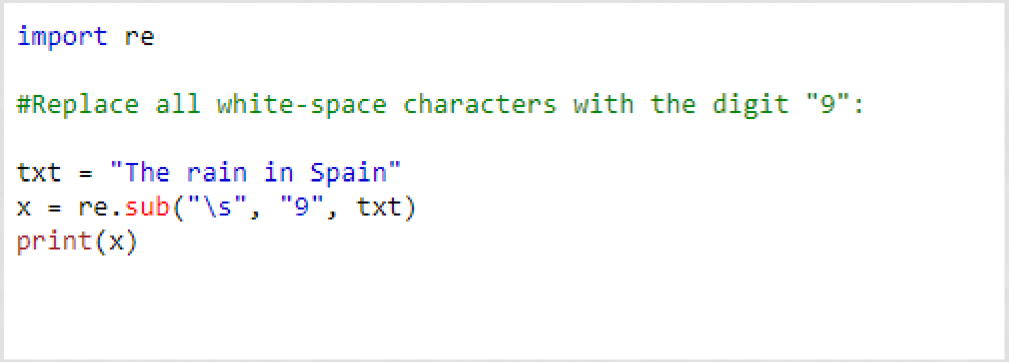
```
import re

#Split the string at every white-space character:

txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```



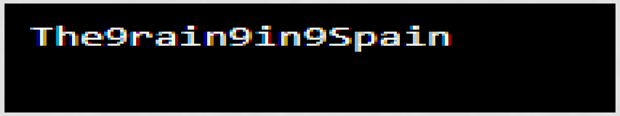
```
['The', 'rain', 'in', 'Spain']
```

RE.SUB()

The sub() function replaces the matches with the text of your choice:



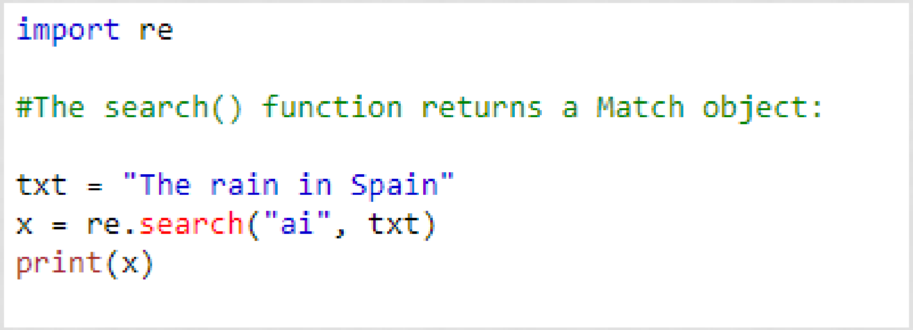
```
import re

#Replace all white-space characters with the digit "9":

txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
print(x)
```



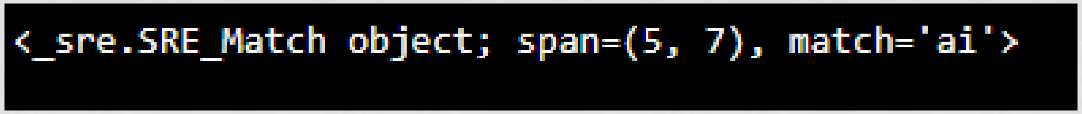
```
The9rain9in9Spain
```

MATCH OBJECT

A Match Object is an object containing information about the search and the result.



```
import re

#The search() function returns a Match object:

txt = "The rain in Spain"
x = re.search("ai", txt)
print(x)
```



```
<_sre.SRE_Match object; span=(5, 7), match='ai'>
```

MATCH OBJECT METHODS

The Match object has properties and methods used to retrieve information about the search, and the result:

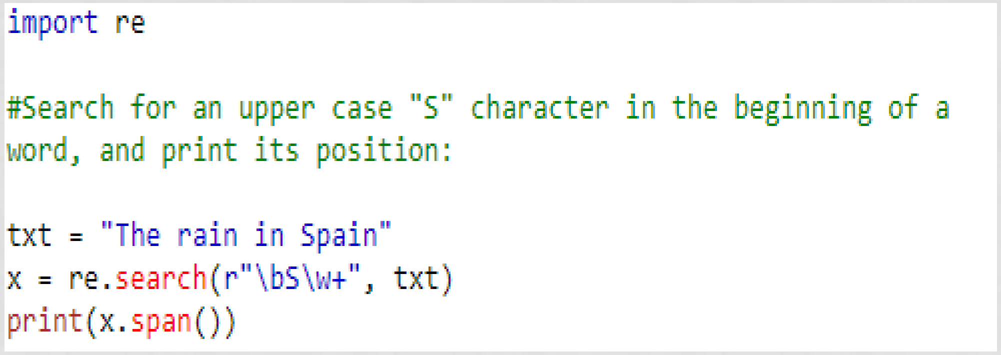
- .span()
- .string
- .group()

.SPAN()

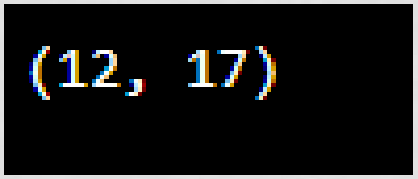
Returns a tuple containing the start-, and end positions of the match.

- Print the position (start- and end-position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S":



```
import re

#Search for an upper case "S" character in the beginning of a
word, and print its position:

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.span())
```

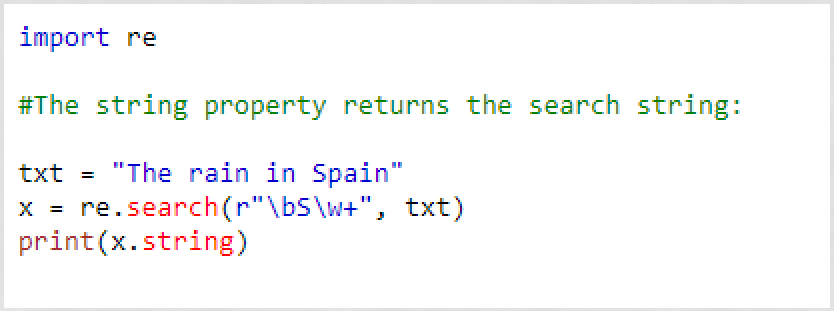


```
(12, 17)
```

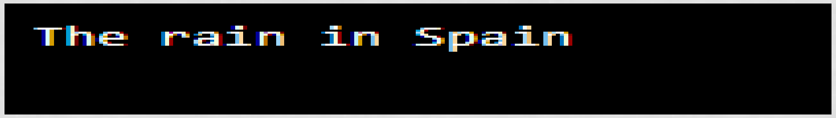
.STRING

Returns the string passed into the function.

Print the string passed into the function:



```python
import re

#The string property returns the search string:

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.string)
```

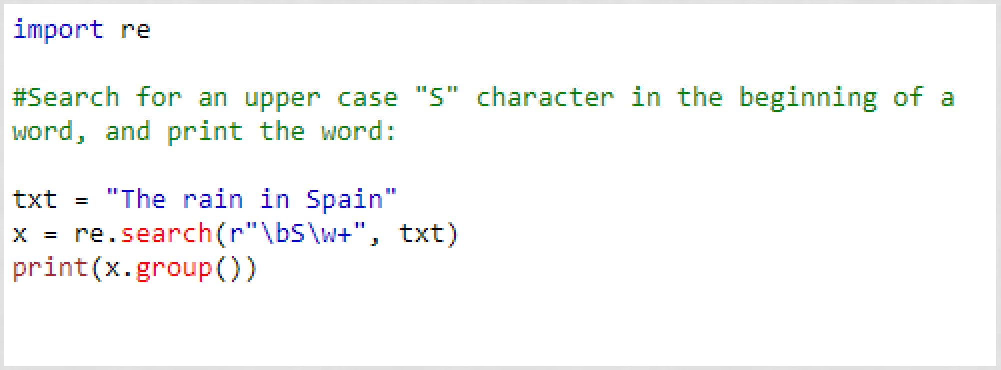


```
The rain in Spain
```

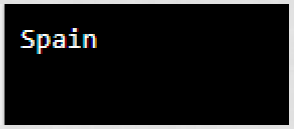
.GROUP()

Returns the part of the string where there was a match.

- Print the part of the string where there was a match.
- The regular expression looks for any words that starts with an upper case "S":



```
import re

#Search for an upper case "S" character in the beginning of a
word, and print the word:

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.group())
```



```
Spain
```

APPLICATIONS

- Email validator
- Password validator
- Parsing
- Web Scraping (Data Collection)

THANK YOU.